TOR VERGATA
UNIVERSITÀ DEGLI STUDI DI ROMA

Faculty of Engineering

Master's Degree in Mechanical Engineering

Multiphysics optimization of an exhaust port using mesh morphing

Advisor:

Prof. Marco Evangelos Biancolini

Co-advisors:

Prof. Lorenzo Bartolucci

Prof. Corrado Groth

Candidate:

Enrico Benso

A.A. 2022/2023

"... Not all those who wander are lost ..."

J.R.R. Tolkien

Acknowledgments

This work is the culmination of a long path that began several years ago, many more than I would like to admit. For this reason I feel that it would be rather naive of me to believe that it would have been possible to reach the end without the support of so many people.

The biggest thanks goes to my family, especially my father Stefano and Carolina, on which I have always been able to count; secondly to all those with which I have shared this years of hard fought battles, in particular the master's degree projects companions.

Lastly I would like to express my appreciation and gratitude to Professors Marco Evangelos Biancolini, Lorenzo Bartolucci and Corrado Groth whose knowledge and help has been invaluable in this thesis work.

Table of contents

List of Figures

List of Tables

Introduction

One of the cornerstones of modern engineering is the optimization field. This type of problem is present in almost every sector under various guises: in energy production for the reduction of CO_2 emission, in aerospace applications for the reduction of weight, in computer science for the improvements of algorithms and the list may go on forever.

In particular, this master's thesis main effort has been the optimization of exhaust ports for Internal Combustion Engines (ICE), both in terms of fluid dynamics and structural performance. In order to achieve this a workflow concerning a set of software regarding Computational Fluid Dynamics (CFD), Computational Structural Mechanics (CSM) and mesh morphing has been developed. The program *RBF viewer*, heavily utilized in this work, allows to morph a surface geometry taking as input and output a great variety of files which are then imported in *CONVERGE CFD* and *ANSYS Mechanical* for the numerical simulations.

The analysis of two similar problems is presented in this thesis: the first geometry is an exhaust port for motor sport applications of which only the CFD side has been considered, the second is an engine head of which both the fluid dynamics and structural performance of the exhaust have been studied. Through the use of a Python code it has been possible to automate the workflow and interaction of the different programs, allowing for a series of Design Of Experiment (DOE).

The workflow is thus composed: a setup for the CFD and CSM simulations is prepared, then the geometry variants are created starting from a DOE table containing the values of the morphing transformations, the CFD analyses are performed and a map of the temperature and pressure for the solid portion is realized which will then be used inside the CSM simulations. Once the needed models are created, together with the morphing parameters, the DOE is executed by simply running the Python script.

The desired outcome of this work is to help in the development of a solid foundation for the implementation of shape optimization in every engineering field concerning fluid-structure interaction. The workflow presented can be easily adapted to multiple problems, not only to the automotive applications explored here.

The structure of the thesis consists in a series of chapter underlying the theoretical aspects of the tools employed in this work, followed by a detailed explanation of the workflow, automation and preparation of the simulations. More specifically, for the ini-

tial theoretical dissertation, the first chapter introduces the concepts of mesh morphing through the use of radial basis functions, the second one concerns the flow of fluids through exhaust ports in the internal combustion engine field and the third explores the governing principles of fluid dynamics and the finite volume method for CFD. The explanation for the numerical part is detailed in chapter four, which shows how the workflow and the simulation setups have been prepared, followed by a discussion of the achieved results in chapter five. In the appendix the final version of the developed Python code is presented as well; Appendix A contains the main code that interacts with the above mentioned software, Appendix B lists the utility functions that have been created to take care of specific necessities that have arisen in the completion of the work.

Formula 1 engines as a modern example of optimization

To understand the need for optimization in the ICE field one needs to look no further than the extremely regulated and exceedingly competitive Formula 1 championship. It would be reasonable to think that given the modern automotive industry tendency to move away from fossil fuels further developments in the traditional ICEs sector would have stopped altogether, looking at what has been done recently by the Formula 1 teams this could not be further from the truth.

Since the FIA (Federation Internationale de l'Automobile) rules regarding the weight, dimensional constraints, amount of fuel consumed are most stringent, not only for the entire car both for the individual components as well, a series of extremely ingenious solutions have been developed by the engineers.

The modern power units are capable of delivering up to around 1000 hp, taking into account the contribution of both the electrical and thermal components. The internal combustion engine is comprised of a 1.6 liters, turbocharged V6; for this component the optimization resides in making the best use of the limited fuel, which is regulated both in terms of maximum flow and stored amount; thus, combustion control is one of the key factors in achieving maximum performance and most modern designs rely on pre-combustion chambers and in some cases also on pre-ignition. Considering the electrical components working inside the propulsion system the main focus is the energy recovery from the heat generated by the brakes and from the shaft of the turbo-compressor assembly, adding additional complexity not only in terms of achievable performance but in space managing and construction as well.

Having considered the above, it is easy to se how the field may greatly benefit from innovative shape optimization techniques; this is not limited only to the power unit assembly but to the aerodynamics and structural departments as well.

Chapter 1

Radial basis functions for mesh morphing

The aim of this chapter is to provide a brief theoretical introduction to the mathematical concepts and engineering implementations of radial basis functions (or RBFs). For a more in-depth overview the book *"Fast Radial Basis Functions for Engineering Applications"* [1] by Professor Marco Evangelos Biancolini may be referenced.

1.1 Basis of RBF theory

RBFs were originally developed as a method for data interpolation as they are capable of interpolating a scalar function defined at discrete points throughout space [2][3]. However, their utility extends beyond interpolation to regression and the development of neural networks [4][5].

For a given number of source points, it is necessary to solve a linear system with the same number of equations and to compute the values of the unknown coefficients. The scalar interpolation function (1.1) is thus composed of the sum of the contribution of each source point and an optional polynomial term:

$$s(\mathbf{x}) = \sum_{i=1}^{N} \gamma_i \varphi(\|\mathbf{x} - \mathbf{x_{si}}\|) + h(\mathbf{x}) \tag{1.1}$$

where:

- $s(\mathbf{x})$ = Scalar function at evaluation point
- γ_i = Weight of the source
- φ = Radial function
- $h(\mathbf{x})$ = Polynomial function

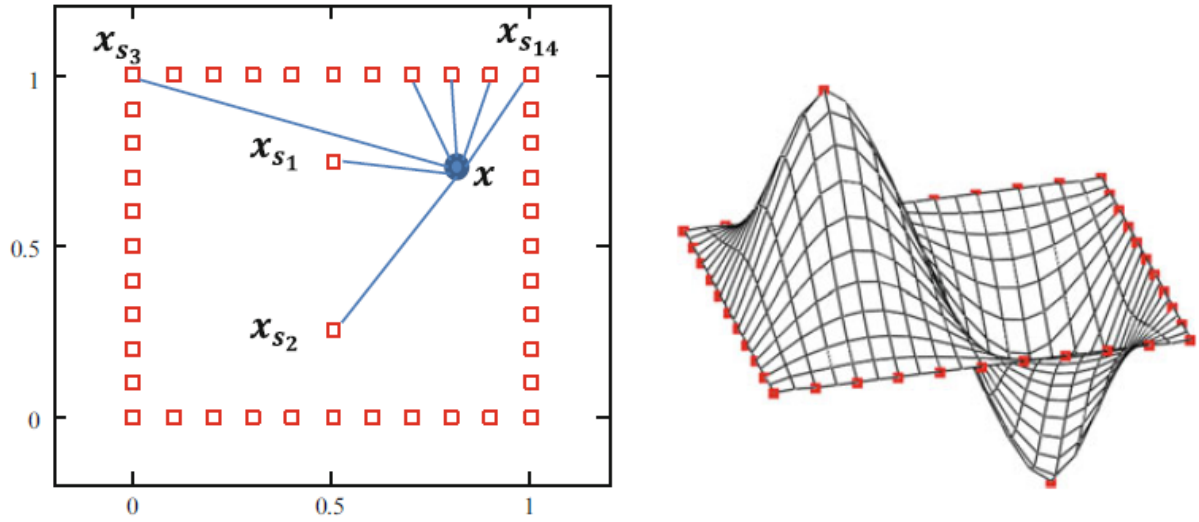


Figure 1.1: Example of RBF interaction between points: in red the source points, in blue the evaluation point. *Left*: two dimensional disposition, *right*: three dimensional view of the interpolation [1]

- $\mathbf{x_{si}}$ = Source point
- $\mathbf{x}$ = Evaluation point

Interpolation involves transitioning from an n-dimensional space to a single dimension, with interactions between the point $\mathbf{x}$ and the sources $\mathbf{x_s}$ computed by evaluating their radial distance. The weight γ_i can be considered as the intensity of the respective source. The concept of radial interaction may better understood by observing Fig. 1.1: a planar arrangement of source points is depicted, with coefficients calculated so that the function equals ± 1 at the center sources and 0 along the edges; the scalar value of the function can be graphically represented as the height in a three-dimensional plot. It is necessary to compute the weights γ and the coefficients of the polynomial term to ensure that the function takes the desired values g_{si} at the source points, as indicated in (1.2):

$$s(\mathbf{x_{si}}) = g_{si} \tag{1.2}$$

The degree of the radial function φ influences the degree of the polynomial h that can be used. However, it is possible to assume [1] that if the base functions are conditionally positive and of order equal or less than 2, a linear one is sufficient. A collection of possible polynomials that might be employed is reported in [6]. For this case, the interpolation function is as follows:

$$h(\mathbf{x}) = \beta_1 + \beta_2 x + \beta_3 y + \beta_4 z \tag{1.3}$$

As it can be easily verified, the number of unknowns (weights and polynomial terms) exceeds the number of equations, so it is necessary to use the orthogonality conditions

(1.4), which are true for all polynomials p with a degree less than or equal to the one currently in use.

$$\sum_{i=1}^{N} \gamma_i p(\mathbf{x_{si}}) = 0 \tag{1.4}$$

For relation (1.3) the orthogonality conditions are as follows:

$$\begin{cases} \sum_{i=1}^{N} \gamma_i = 0 \\ \sum_{i=1}^{N} \gamma_i x_{si} = 0 \\ \sum_{i=1}^{N} \gamma_i y_{si} = 0 \\ \sum_{i=1}^{N} \gamma_i z_{si} = 0 \end{cases} \tag{1.5}$$

Thus, by combining (1.2) and (1.5), the linear system (1.6) is obtained, where the vector of unknowns consists of the weights γ and coefficients β of the linear polynomial, g_s are the known terms of the function at the source points, $\mathbf{M}$ is the interpolation matrix defined by computing all radial interactions between the source points, and $\mathbf{P_s}$ is the constraint matrix used to balance the contribution of the polynomial function.

$$\begin{bmatrix} \mathbf{M} & \mathbf{P_s} \\ \mathbf{P_s^T} & 0 \end{bmatrix} \begin{pmatrix} \gamma \\ \beta \end{pmatrix} = \begin{pmatrix} \mathbf{g_s} \\ 0 \end{pmatrix} \tag{1.6}$$

Arrays $\mathbf{M}$ and $\mathbf{P_s}$ are defined as:

$$M_{ij} = \varphi \|\mathbf{x_{si}} - \mathbf{x_{sj}}\| \tag{1.7}$$

$$\mathbf{P_s} = \begin{bmatrix} 1 & x_{s1} & y_{s1} & z_{s1} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_{sN} & y_{sN} & z_{sN} \end{bmatrix} \tag{1.8}$$

In a three dimensional domain the displacement components are as follows:

$$\begin{cases} s_x(\mathbf{x}) = \sum_{i=1}^{N} \gamma_i^x \varphi \|\mathbf{x} - \mathbf{x_{si}}\| + b_1^x + b_2^x x + b_3^x y + b_4^x z \\ s_y(\mathbf{x}) = \sum_{i=1}^{N} \gamma_i^y \varphi \|\mathbf{x} - \mathbf{x_{si}}\| + b_1^y + b_2^y x + b_3^y y + b_4^y z \\ s_z(\mathbf{x}) = \sum_{i=1}^{N} \gamma_i^z \varphi \|\mathbf{x} - \mathbf{x_{si}}\| + b_1^z + b_2^z x + b_3^z y + b_4^z z \end{cases} \tag{1.9}$$

If the scalar quantities need to be evaluated at a specific set of values $\mathbf{x_e}$, a direct mapping matrix (1.10) can be used to interpolate as a linear transformation of the values $\mathbf{g_s}$ at the evaluation points. If the polynomial term is not present, the system can be

	$\varphi(r)$
Spline	r^n, n odd
Thin plate spline	$r^n \log(r)$, n even
Multiquadratic	$\sqrt{1+r^2}$
Inverse multiquadratic	$\frac{1}{\sqrt{1+r^2}}$
Inverse quadratic	$\frac{1}{1+r^2}$
Gaussian	e^{-r^2}

Table 1.1: Global support radial functions [1]

	$\varphi(r) = f(\xi)$
Wendland C^0	$(1-\xi)^2$
Wendland C^2	$(1-\xi)^(4\xi+1)$
Wendland C^4	$(1-\xi)^6(\frac{35}{3}\xi^2+6\xi+1)$

Table 1.2: Compact support radial functions [1]

written as (1.11), where $\mathbf{g_e}$ represents the aforementioned scalar terms.

$$A_{ij} = \varphi(\mathbf{x_{ei}} - \mathbf{x_{sj}}) \tag{1.10}$$

$$\mathbf{g_e} = \mathbf{A}\mathbf{M^{-1}}\mathbf{g_s} \tag{1.11}$$

To characterize the behavior of the function outside the original set of points, interpolation and extrapolation operations are required, that depends on the radial functions. These can be classified into two main groups: global support and compact support, respectively listed in Tables 1.1 and 1.2. Global support functions involve complete interaction among all points, leading to high accuracy but also potential numerical issues due to densely populated matrices. Compact support functions, on the other hand, utilize a preset radius to limit the interaction distance, resulting in null values outside of it, simplifying the calculation. Typical examples of this category are Wendland functions [7]. These functions are expressed in terms of a dimensionless quantity ξ, whose maximum value corresponds to the limiting radius.

1.2 RBF use example

This section outlines the procedure followed in Chapter 2.3 of [1], where the use of RBFs for a "sculpting" application of two-dimensional surfaces in 3D structures is explained.

The task involves introducing an additional coordinate, height, controlled using RBF centers on the boundary and in the middle of the domain. For instance, consider a rectangle with sides a and b on which exists a certain number of equally spaced points n_e with zero height and a central point with a non-zero value. These are the source points $\mathbf{x_s}$ in two-dimensional form, whose graphical representation is provided in Fig. 1.2.

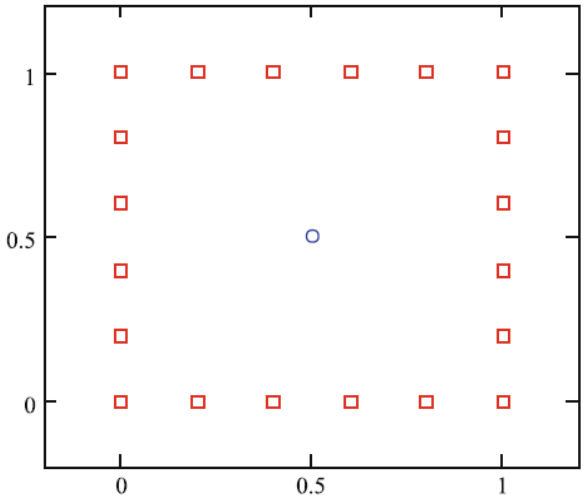


Figure 1.2: Two dimensional disposition of control points [1]

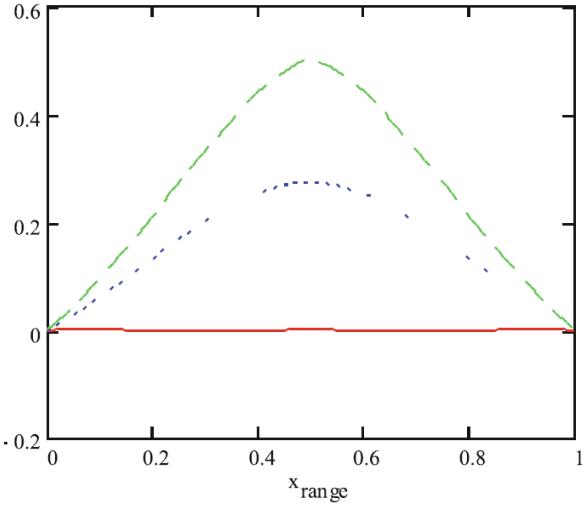


Figure 1.3: Interpolation error: the three curves represent different segments for y coordinate equal to 0 (red), 0.25 (blue) and 0.5 (green) [1]

The vector of known terms $\mathbf{g_s}$ has a value of 0 at the boundary points and 0.5 at the central point. As for the radial function, a biharmonic spline shape in two dimensions is considered, expressed as:

$$\varphi(r) = r^2 \log(r) \tag{1.12}$$

It is then possible to compose the matrices $\mathbf{M}$ and $\mathbf{P}$ and solve the third equation of the system (1.9) by imposing $s_z(x) = 0$ for the external points and $s_z(x) = 0.5$ for the central point. In Fig. 1.3, the quality of the interpolation can be observed: the exact values are obtained only at the extreme points and at the center, while in the rest of the domain the function is approximated; the three curves represent different values of the y coordinate. One aspect to consider is that along the domain's boundaries, the function presents a value of exactly zero only at the source points and, for the rest, possesses an approximation error, the magnitude of which depends on the number of source points (a

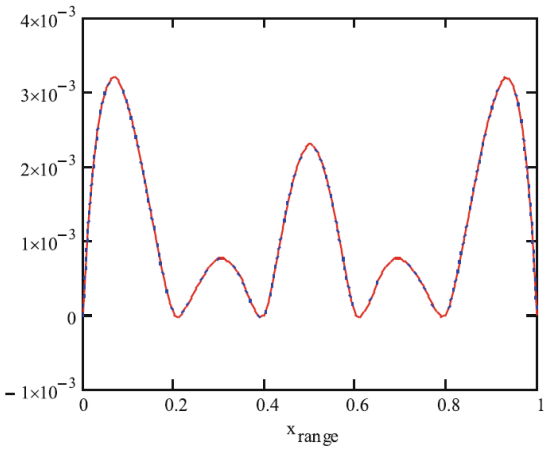


Figure 1.4: Interpolation error at boundaries edges [1]

finer grid minimizes this error). This phenomenon can be observed in Fig. 1.4; the trend is a detail of the red curve in Fig. 1.3. By creating a two dimensional grid that intercepts the points on the domain's borders, as in Fig. 1.5, the effect of the height imposed by the sources is observable. At the center of the grid, there is a maximum value of 0.5, exactly as prescribed.

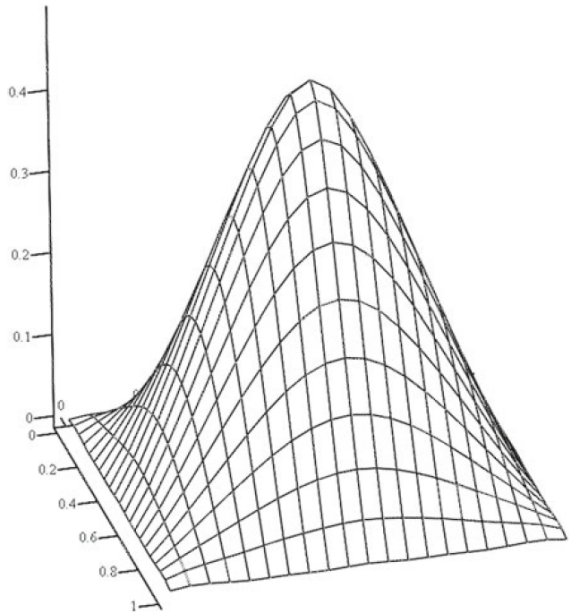


Figure 1.5: 3D view of the interpolation function: the height represent the value [1]

It is worth considering two additional aspects: the first concerns the use of radial functions of different degrees, while the latter regards the effect of modifying the source points. In Fig. 1.6, it can be viewed how a linear function, $\varphi(r) = r$, and a cubic one, $\varphi(r) = r^3$, interpolate the domain differently, while in Fig. 1.7, the effect of an additional source with a value of -0.5 in the middle of the domain is shown.

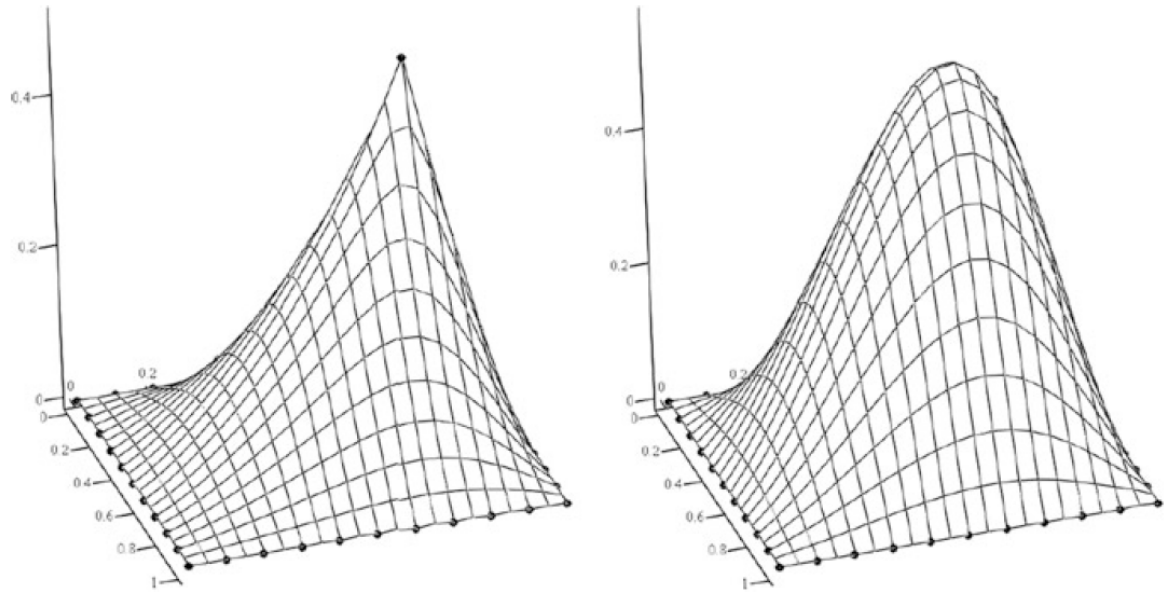


Figure 1.6: Effect of the RBF degree on interpolation: linear on the left, cubic on the right [1]

1.3 Differentiation

RBFs can be differentiated like any other function. Assuming a three-dimensional space and a linear polynomial, the function (1.1) becomes:

$$s(\mathbf{x}) = \sum_{i=1}^{N} \gamma_i \varphi \sqrt{(x - x_{si})^2 - (y - y_{si})^2 - (z - z_{si})^2} + \beta_1 + \beta_2 x + \beta_3 y + \beta_4 z \tag{1.13}$$

The derivatives can be obtained by applying the differentiation rule for nested functions. The first partial derivative along x is expressed compactly in (1.14), while the gradient of the function (1.13) is reported in (1.15).

$$\frac{\partial s(\mathbf{x})}{\partial x} = \sum_{i=1}^{N} \gamma_i \frac{\partial \varphi \|\mathbf{x} - \mathbf{x_{si}}\|}{\partial r} \frac{x - x_{si}}{\|\mathbf{x} - \mathbf{x_{si}}\|} + \beta_2 \tag{1.14}$$

$$\nabla s(\mathbf{x}) = \begin{pmatrix} \frac{\partial s(\mathbf{x})}{\partial x} \\ \frac{\partial s(\mathbf{x})}{\partial y} \\ \frac{\partial s(\mathbf{x})}{\partial z} \end{pmatrix} = \sum_{i=1}^{N} \gamma_i \frac{\frac{\partial \varphi \|\mathbf{x} - \mathbf{x_{si}}\|}{\partial \mathbf{r}}}{\|\mathbf{x} - \mathbf{x_{si}}\|} \begin{pmatrix} x - x_{si} \\ y - y_{si} \\ z - z_{si} \end{pmatrix} + \begin{pmatrix} \beta_2 \\ \beta_3 \\ \beta_4 \end{pmatrix} \tag{1.15}$$

1.4 Interpolation from known arbitrary values

Until now, the value of the function has been imposed precisely on the source points, a more general method is to define it at arbitrary positions. In this case, equation (1.2) becomes (1.16), where $\mathbf{x_f}$ represents the positions of the known points. The solving system becomes (1.17), and in turn, the matrix $\mathbf{P}$ transforms into (1.18).

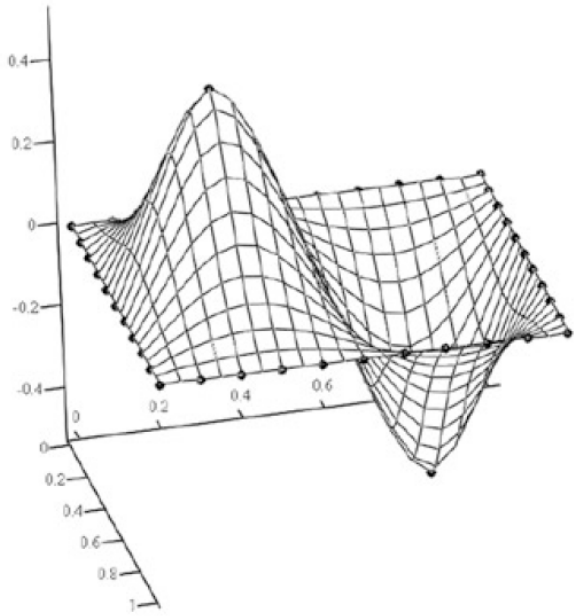


Figure 1.7: Effect of adding another source of opposite value in the domain [1]

$$s(\mathbf{x_{fi}}) = g_{fi} \tag{1.16}$$

$$\begin{bmatrix} \mathbf{M} & \mathbf{P_f} \\ \mathbf{P_f^T} & 0 \end{bmatrix} \begin{pmatrix} \gamma \\ \beta \end{pmatrix} = \begin{pmatrix} \mathbf{g_f} \\ 0 \end{pmatrix} \tag{1.17}$$

$$\mathbf{P_f} = \begin{bmatrix} 1 & x_{f1} & y_{f1} & z_{f1} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_{fN} & y_{fN} & z_{fN} \end{bmatrix} \tag{1.18}$$

As an example, one can consider modifying the previously used two-dimensional case by creating another set of interpolating points inside the domain. Additionally, a central point overlapping the source point is added. An example of this is visible in Fig. 1.8.

1.5 Regression

Unlike interpolation, regression implies that the number of source points is less than the those where the scalar function is known; consequently, it is necessary to admit a certain error. Once the system (1.6) is solved, one has to search for coefficients that provide the best approximation of the points where $\mathbf{g}$ is known. In this case, the problem retains the form expressed in (1.17), but since the number of equations and unknowns is not equal, the matrix is rectangular. Since a direct solution is impossible, it is necessary to resort to the method of least squares to find the optimal values of the coefficients γ and β. For this purpose, equation (1.19) is used, followed by the system (1.20).

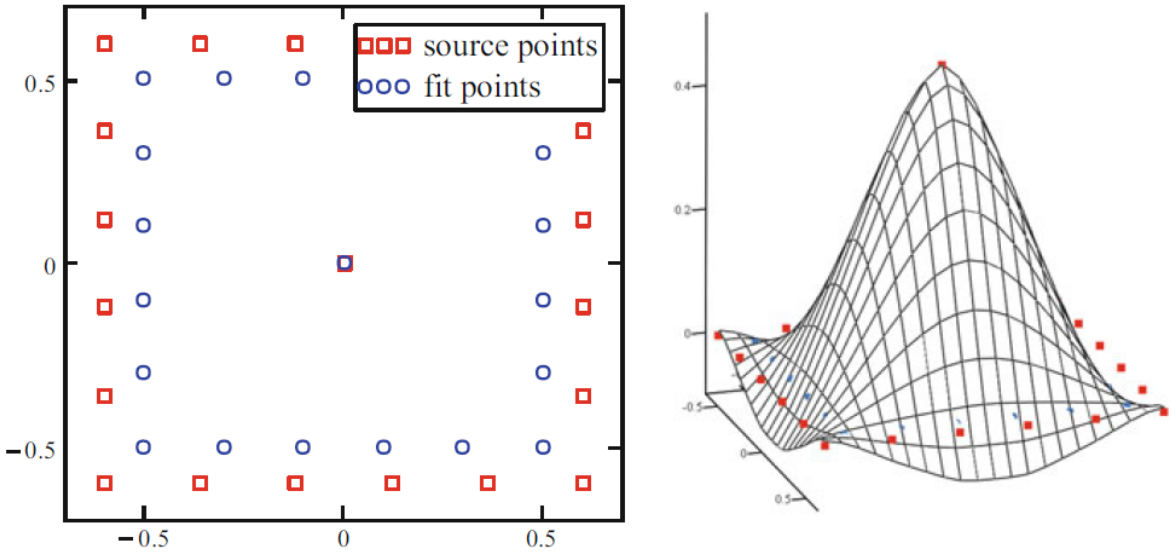


Figure 1.8: Effect of known arbitrary points: on the left the 2D placement, o the right the function behavior [1]

$$S = \frac{1}{N}\sum_{j=1}^{N}(s(\mathbf{x_{f}}_j) - g_{fj})^2 \tag{1.19}$$

$$\begin{bmatrix} \mathbf{M} & \mathbf{P_f} \\ \mathbf{P_s^T} & 0 \end{bmatrix}^T \begin{bmatrix} \mathbf{M} & \mathbf{P_f} \\ \mathbf{P_s^T} & 0 \end{bmatrix} \begin{pmatrix} \gamma \\ \beta \end{pmatrix} = \begin{bmatrix} \mathbf{M} & \mathbf{P_f} \\ \mathbf{P_s^T} & 0 \end{bmatrix}^T \begin{pmatrix} \mathbf{g_f} \\ 0 \end{pmatrix} \tag{1.20}$$

Considering the 2D case already discussed, it is possible to imagine having two sets of entities: the sources and the fitting points, respectively squares and circles in Fig. 1.9. Both are arranged in the same way, but the latter are denser and more numerous. In Fig. 1.10, it is possible to observe the behavior of the function on the edges of the domain: the source points impose their value, and there is a certain error in trying to respect it.

1.6 RBF mesh morphing principles

When radial basis functions are used to deform a grid of points, or mesh, the three components of the displacement vector, defined on a set of source points (also called RBF sources or control points), are interpolated in space and allow for the modification of the position of the nodes. Therefore, equation (1.9) is utilized in (1.21) to define the new position of all the nodes in the starting mesh.

$$\mathbf{x_{new}} = \mathbf{x_0} + \begin{Bmatrix} s_x(\mathbf{x_0}) \\ s_y(\mathbf{x_0}) \\ s_z(\mathbf{x_0}) \end{Bmatrix} \tag{1.21}$$

Thanks to the flexibility of the mathematics behind RBFs, there are numerous solu-

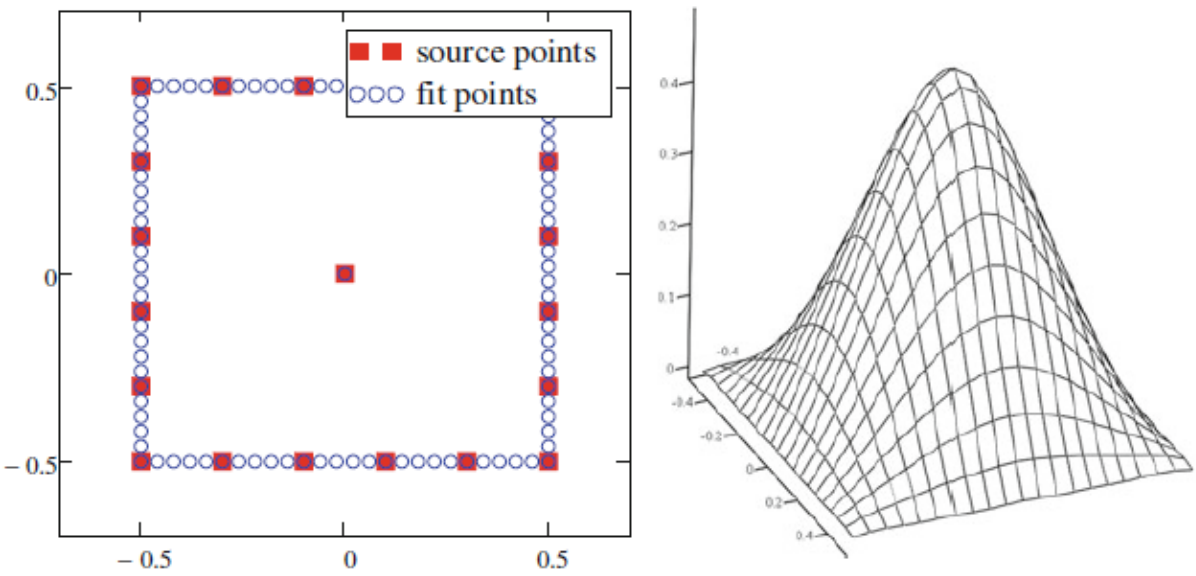


Figure 1.9: Regression using RBF: on the left the starting domain, on the right the effect on the function [1]

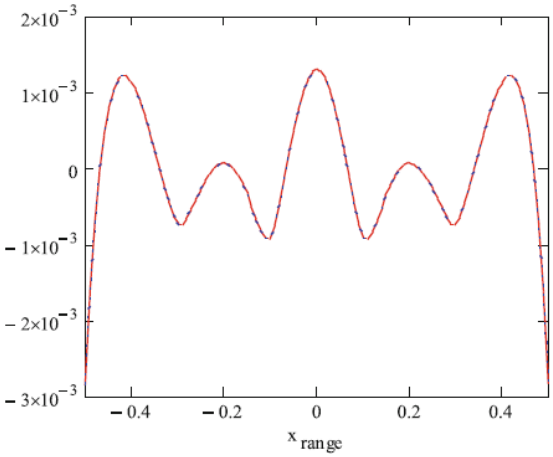


Figure 1.10: Regression error at boundary edges [1]

tions available for solving mesh morphing problems. For a more comprehensive collection of fundamental strategies and industrial applications, one can refer to [8] and [9].

As for the advantages of implementing RBF methods, it is important to first consider that they can handle any type of mesh, regardless of the number of dimensions in space and the type of application. Elements can be tetrahedrons, hexahedrons, squares, triangles, and so on. Therefore, their use is well-suited for both surface and volume grids, with the essential characteristic of maintaining the topology unchanged. This aspect, extensively exploited in this work, allows for easy automation of simulations by modifying the shape while keeping the boundary conditions of the problem unchanged. Being purely mathematical and not mesh dependent, the method's implementation is extremely powerful and versatile, suitable for integration into any type of numerical solver that allows for the alteration of node positions in the mesh. Thus, possible applications include fields such as Computational Fluid Dynamics (CFD) and Computational Structural Mechanics

(CSM), both explored in this thesis.

However, there are also some limitations. The most significant is undoubtedly the high computational cost associated with handling a large number of RBF points. Additionally, there is the possibility of a deterioration in mesh quality and the need to create a Computer-Aided Design (CAD) model for the modified mesh. The former can be partially mitigated by using "fast RBF methods" (see [1]), limiting the interpolation domain using auxiliary entities (discussed later) and utilizing techniques for parallelization or partitioning of numerical computation. The final quality of the model may be improved, if permitted, by selecting appropriate radial basis functions $\varphi(r)$. Finally, regarding the generation of the CAD model of the new geometry, there are two mature possibilities: one needs the morphing of the same geometry, provided that the RBF field remains unchanged, the other involves maintaining the association between the original and modified surfaces (thanks to the same topology) but requires the use of advanced reconstruction tools for implementation.

As mentioned earlier, there are two families of radial basis functions: those with global support and those with compact support. The former requires the user to define the areas where morphing occurs; this can be done by defining a series of source points with zero value on the edges of the area of interest to prevent long-range interactions. The latter, instead, include in their definition an extinction distance beyond which the interaction between nodes ceases. Since one of the simplest radial basis functions to implement, with relatively low degradation effect on the starting mesh, is the linear spline $\varphi(r)$ with global support, some paradigms for using this type of RBF will be illustrated in the following paragraphs.

For a given initial domain of nodes, of which only a part is to be modified, it is necessary to implement functions, geometries, or auxiliary domains. For example, it may be necessary to set a zero displacement for a series of nodes beyond which the effect of interpolation should not propagate. For those coming from the Finite Element Method (FEM), it can be useful to imagine these as constraints similar to fixed supports. Thus, three main zones can be distinguished:

- A set of nodes with null prescribed displacement
- A set of nodes left free to move
- A set of nodes with prescribed displacement and directly controlled

Another solution could be the construction of containment geometries within which morphing might occur. For example, as shown in Fig. 1.11 [10], it is possible to define a rectangle on whose surfaces there is a series of fixed control points. This implies that any action defined within such a shape cannot propagate beyond the boundaries. The size of this type of auxiliary domain has implications on computational complexity and final

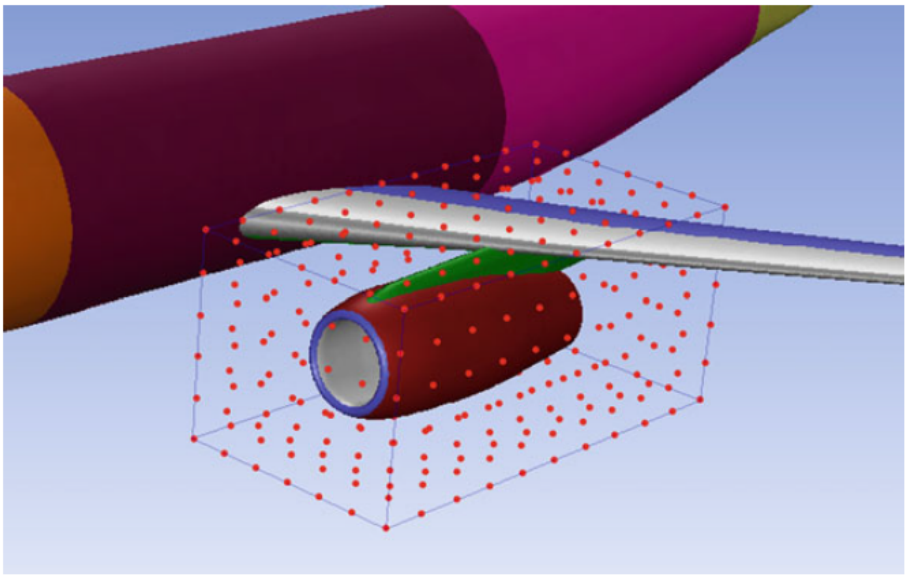


Figure 1.11: Domain for local morphing [1][10]

quality: larger volumes allow for a greater number of better distributed source points, thus improving the quality of the mesh obtained, albeit at the expense of computational cost and time.

To impart motion to the nodes of the grid, it is possible to use mobile auxiliary domains: in Fig. 1.12 [11], two cylinders of points on which a displacement is imposed are shown. Due to proximity, there is a transfer of motion to the nodes of the mesh, resulting in a final sculpting effect on the model.

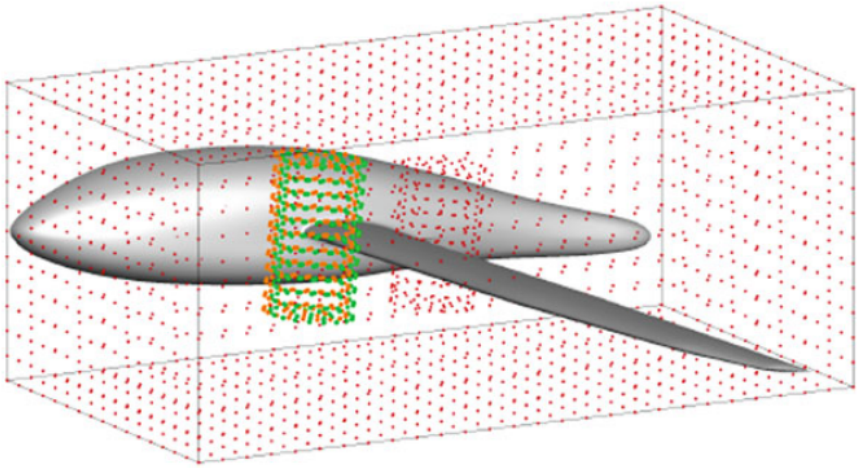


Figure 1.12: Mobile auxiliary domains for local morphing [1][11]

A complete example is shown in Fig. 1.13: the morphing on the tube is delimited by the rectangle, and the five disks of points prescribe local displacements. In particular, the two on the straight sections impose the maintenance of the shape, while the other three perform the transformation shown in the right figure.

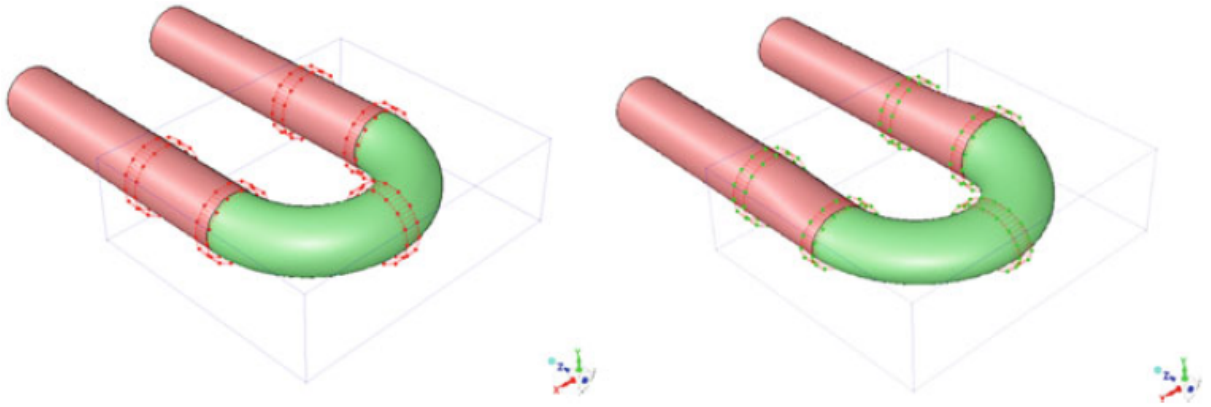


Figure 1.13: Example of use of both fixed and mobile domains: on the left the original shape, on the right the morphed one [1]

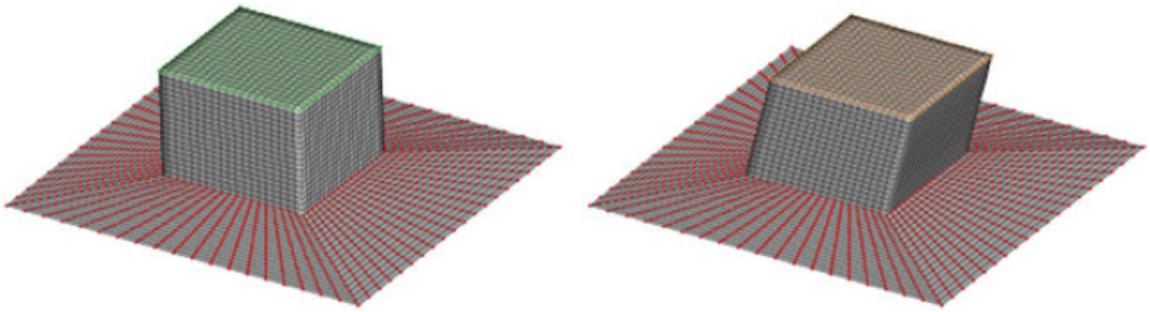


Figure 1.14: Example of translation, on the left the red points are the constrained ones and the green those moving of a prescribed value [1]

1.7 RBF basic transformations

The possible transformations of the nodes of a mesh are numerous. In this section, only the simplest ones are reported, namely those allowed by the software used in the thesis work.

1.7.1 Translation

Translation is the simplest type of modification, allowing the displacement of a prescribed value for each point of interest in a specific direction in space. Additionally, it is a linear transformation, meaning that the effect of multiple translations on a set of points is the same regardless of the order with which they occur. In Fig. 1.14, the effect of two translation transformations imposed on a set of points is observed: the first with a zero value on the bottom face, equivalent to a constraint, and the second with a non-zero value in a specific direction on the top face. To specify the magnitude of the displacement, the components in the three directions of the reference system are necessary, which can be either global or local.

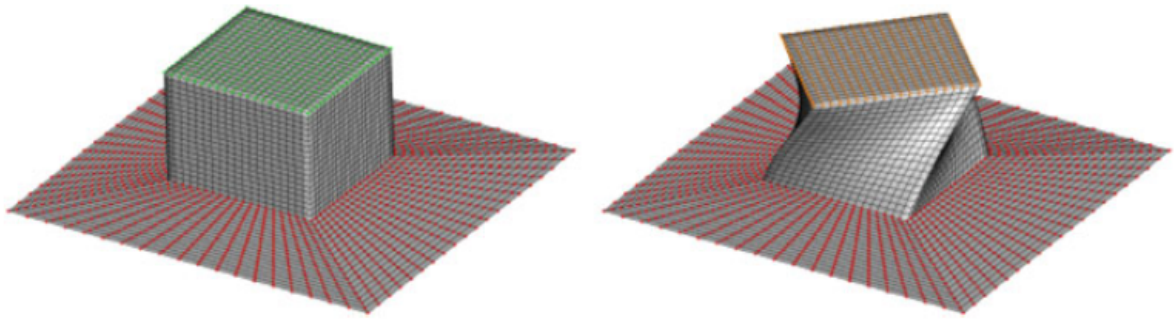


Figure 1.15: Example of rotation, on the left the red points are the fixed ones and the green ones those moving of a prescribed value [1]

1.7.2 Rotation

Rotation is a type of transformation that guarantees at least one fixed point in space; the other points will move around it, the entity being directly proportional to their distance. In three dimensions, this type of transformation becomes nonlinear and requires two defining parameters: an axis around which the motion occurs and an angle representing the magnitude of the transformation. The rotation matrix for a three-dimensional space is as follows:

$$R = \begin{bmatrix} \cos\theta + u_x^2(1-\cos\theta) & u_x u_y(1-\cos\theta) - u_z\sin\theta & u_x u_z(1-\cos\theta) - u_y\sin\theta \\ u_x u_y(1-\cos\theta) - u_z\sin\theta & \cos\theta + u_y^2(1-\cos\theta) & u_y u_z(1-\cos\theta) - u_x\sin\theta \\ u_x u_z(1-\cos\theta) - u_y\sin\theta & u_y u_z(1-\cos\theta) - u_x\sin\theta & \cos\theta + u_z^2(1-\cos\theta) \end{bmatrix} \tag{1.22}$$

where:

- u_x, u_y, u_z represent the components of the axis of rotation $\mathbf{u}$
- θ is the angle of rotation

In Fig. 1.15, the effect of this type of transformation applied to the top face is observed, while the bottom face remains constrained by a translation with a zero value.

1.7.3 Scaling

Like translation, this transformation is also linear and allows for the movement in a radial direction of a set of points relative to their center. By using a scale factor, it's possible to enlarge or shrink a shape along the orthogonal axes. For example, a figure lying on an x, y plane will change its shape by different amounts from the unit value in the x and y directions (in particular, if the value is greater than one, it will expand, while if the value is greater than zero but less than one, it will decrease in size), but it will remain

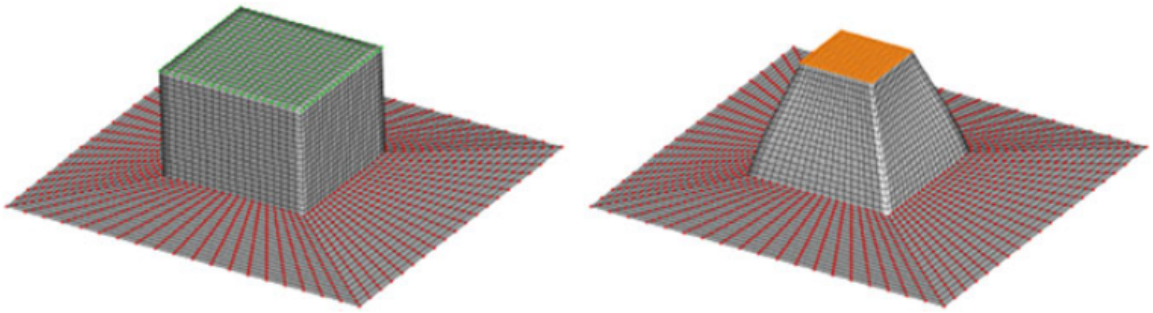


Figure 1.16: Example of scaling, on the left the red points are the fixed ones and the green ones those moving of a prescribed value [1]

unaffected by any value in the z direction. For a transformation with respect to the global system, it's sufficient to use the matrix:

$$S = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & S_z \end{bmatrix} \tag{1.23}$$

$$S_0 = \begin{pmatrix} S_{0x} \\ S_{0y} \\ S_{0z} \end{pmatrix} \tag{1.24}$$

To apply a scaling with respect to a different reference system, at first there is the need to apply the translation transformation using equation (1.24) to align the origins of the two systems, perform the scaling using (1.23), and then move the system again using the opposite of (1.24). In Figure 1.16, an example of this transformation is illustrated.

Chapter 2

Flow through exhaust valves and ports

In this chapter, a brief description of the geometry of valves and exhaust ducts is provided. The aspects related to the intake component have been omitted since this work focuses solely on the steady flow exiting from a cylinder. For a comprehensive treatment of fluid flow through ducts and valves, refer to chapter 6.3 of [12] and chapter 2.5 of [13].

2.1 Geometry considerations

Usually, in the motion of fluid inside a four-stroke internal combustion engine, the valves and their seats on the ducts define the areas with the smallest passage section. Most of the times they are circular in shape, and nowadays, the configuration with four valves per cylinder is the most common, although variants with three and five valves exist (where, respectively, one and two are for exhaust). Unlike the intake case, the exhaust must also consider problems related to heat exchange with the exiting gases. Therefore, during

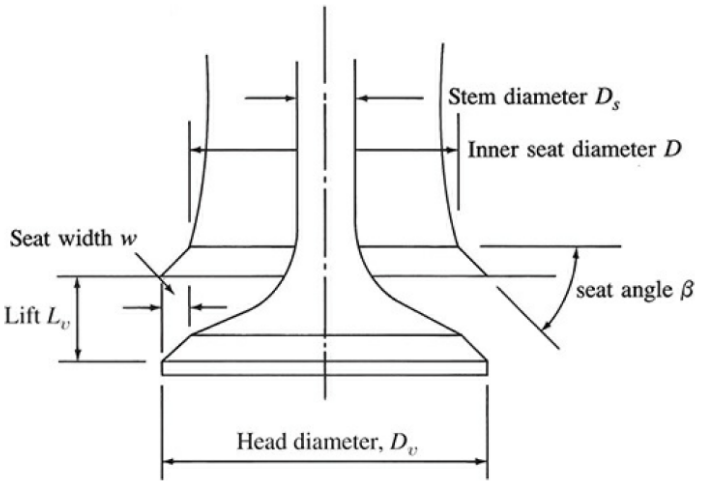


Figure 2.1: Geometry parameters of a poppet valve [12]

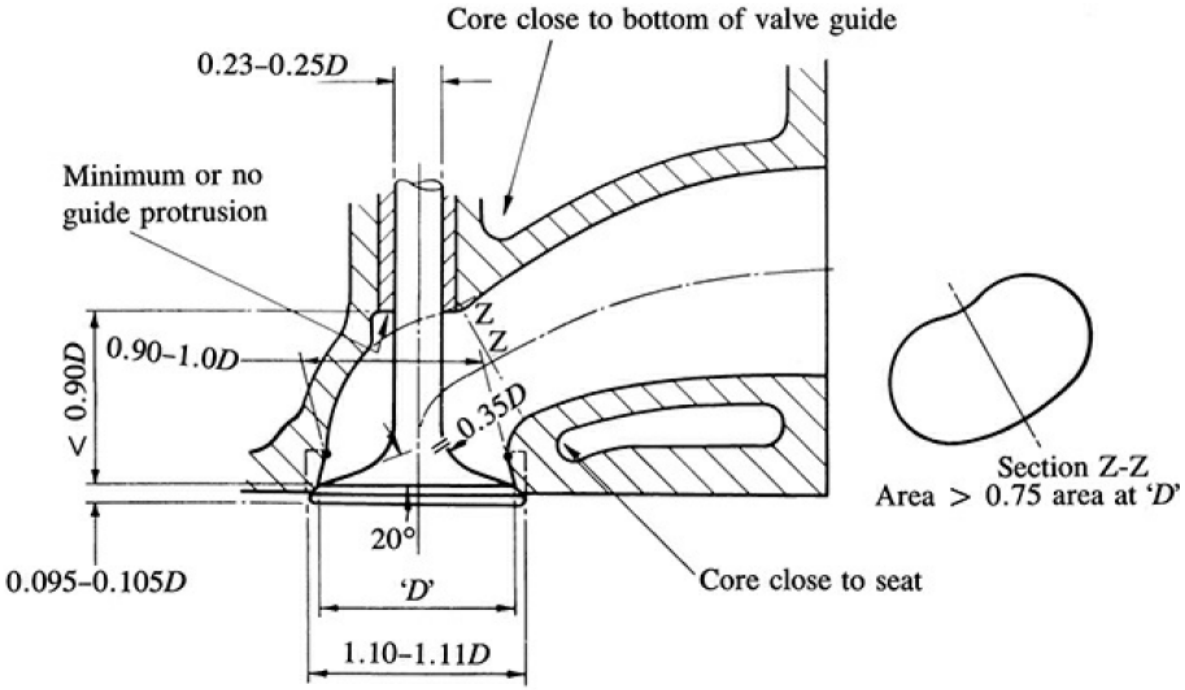


Figure 2.2: Typical shape and proportions of exhaust valves and ducts [12]

Combustion chamber shape	Intake	Exhaust
Two-valve wedge or bathtub	$(0.43 - 0.46)B$	$(0.35 - 0.37)B$
Two-valve hemispherical	$(0.48 - 0.50)B$	$(0.41 - 0.43)B$
Three-valve open	$(0.38 - 0.40)B$	$(0.41 - 0.43)B$
Four-valve pent-roof	$(0.38 - 0.41)B$	$(0.31 - 0.35)B$

Table 2.1: Valve head diameter in terms of cylinder bore B [12]

construction and design phases, particular attention must be paid to cooling issues. Fig. 2.1 illustrates the main geometric parameters of a poppet valve and its seat, while Fig. 2.2 shows common construction proportions of the valve-duct assembly.

The movement of the valves is controlled by the camshaft, and it is possible to distinguish between mechanisms with variable timing control and fixed timing. The former are replacing the latter due to better performance and flexibility. The geometry and dimensions of the valve heads depend on their number, the shape of the cylinder head and bore. Some typical dimensions are listed in Tab. 2.1. Another notable characteristic is the smaller size of the exhaust components compared to their intake counterparts. This is due to the need for the exhaust valves to fill the available volume inside the cylinder with replacement fluid as quickly as possible.

Generic profiles for lift and flow area computed as a function of the cam angle are shown in Fig. 2.3. Regarding the available area, a plateau can be observed near the lift values for which the curtain area (the cylindrical surface with dimensions equal to the valve diameter and lift) exceeds the area of maximum narrowing of the duct.

As already stated the geometry of the valve head, stem, seat and lift itself have a great

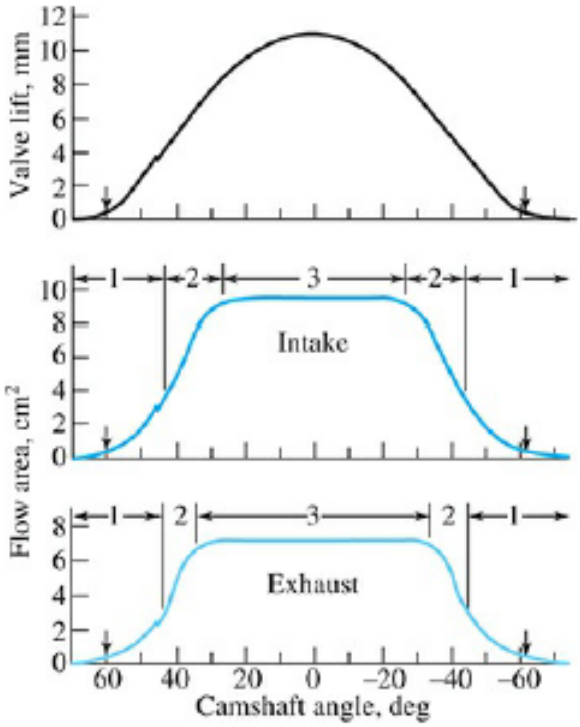


Figure 2.3: Lift and flow area in respect to the cam angle [12]

impact upon the instantaneous flow area. The latter in particular can be divided in three stages: two of low lift and a higher one. For the least amount of lift the section of flow corresponds to the slant face of frustum of a right circular cone that exists between the valve and seat faces. So, for this stage, the lift and flow area are as follows:

$$0 < L_v < \frac{w}{\sin\beta\cos\beta} \tag{2.1}$$

$$A_{min} = \pi L_v \cos\beta D_v - 2_w + \frac{L_v}{2}\sin(2\beta) \tag{2.2}$$

The geometry definitions are the ones in Fig. 2.1: L_v is the valve lift, D_v is the valve head diameter, w is the seat width and β the valve seat angle.

For the second stage, as the valve moves away from the seat, the slant surface ceases to be perpendicular to the other faces and it's base angle increases from (90-β) toward 90. So:

$$\frac{w}{\sin\beta\cos\beta} \leq L_v \leq \sqrt{\frac{D_p^2 - D_s^2}{4D_m} - w^2} + w\tan\beta \tag{2.3}$$

$$A_{min} = \pi D_m \sqrt{(L_v - w\tan\beta)^2 + w^2} \tag{2.4}$$

where D_p and D_s are the port and seat diameters respectively and $D_m = (D_v - w)$ is the seat mean diameter.

When lift is high enough the minimum flow area becomes fixed and corresponds to

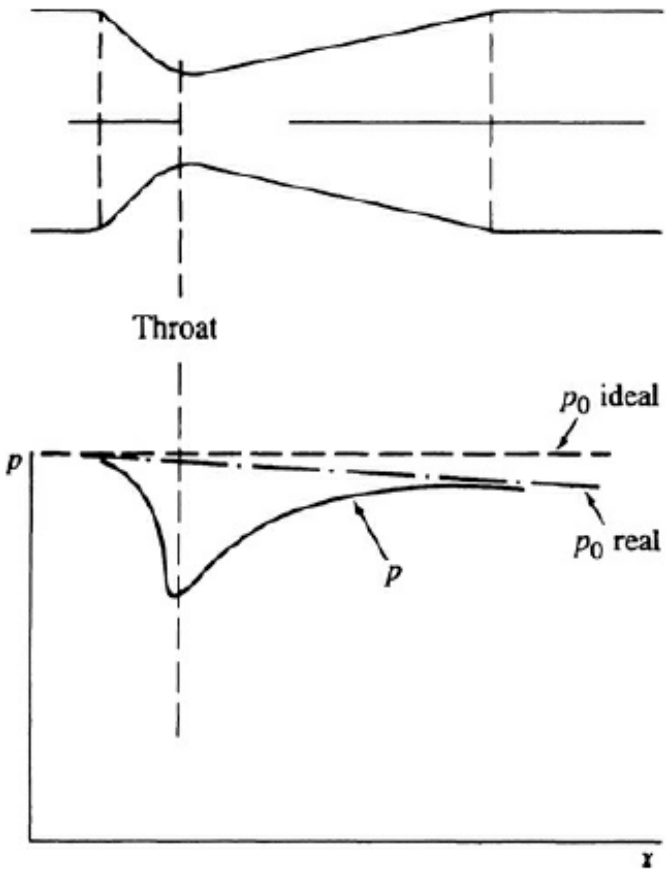


Figure 2.4: Pressure distribution for gas flow through a restriction [12]

the port flow of minimum area minus the cross section of valve stem:

$$L_v > \sqrt{\frac{D_p^2 - D_s^2}{4D_m} - w^2} + w \tan \beta \tag{2.5}$$

$$A_{min} = \frac{\pi}{4}(D_p^2 - D_s^2) \tag{2.6}$$

2.2 Flow through a restriction and discharge coefficient

In this section the procedure to obtain the equation of compressible flow through a restriction and the definition of the discharge coefficient is given.

Considering an ideal flow, T_0 and p_0 are the reference values for temperature and pressure, γ is the specific heat ratio, u is the velocity of the fluid and (2.7), (2.8) are the steady flow energy equation and the isoentropic relation:

$$T_0 = T + \frac{u^2}{2c_p} \tag{2.7}$$

$$\frac{T}{T_0} = \frac{p}{p_0}^{\frac{\gamma-1}{\gamma}} \tag{2.8}$$

If $a = \sqrt{\gamma RT}$ is the speed of sound then the Mach number is: $M = \frac{u}{a}$ and the following equations are obtained:

$$\frac{T_o}{T} = 1 + \frac{\gamma - 1}{2} M^2 \tag{2.9}$$

$$\frac{p_0}{p} = \left(1 + \frac{\gamma - 1}{2} M^2\right)^{\frac{\gamma}{\gamma-1}} \tag{2.10}$$

The mass flow rate is $\dot{m} = \rho A V$, for an ideal gas the above equations can be arranged as:

$$\frac{\dot{m}_{id}\sqrt{\gamma RT_0}}{AP_0} = \gamma M \left(1 + \frac{\gamma - 1}{2} M^2\right)^{-\frac{\gamma+1}{2(\gamma-1)}} = \gamma \left(\frac{p}{p_0}\right)^{\frac{1}{\gamma}} \left\{\frac{2}{\gamma - 1}\left[1 - \left(\frac{p}{p_0}\right)^{\frac{\gamma-1}{\gamma}}\right]\right\}^{\frac{1}{2}} \tag{2.11}$$

The flow rate is maximum when the velocity at the choke point equals the speed of sound, the ratio between the throat pressure (at the choke point) and the stagnation pressure is called "critical pressure ratio" and is defined as follows:

$$\frac{p_T}{p_0} = \left(\frac{2}{\gamma + 1}\right)^{\frac{\gamma}{\gamma-1}} \tag{2.12}$$

When $\frac{p}{p_0} \leq \frac{p_T}{p_0}$ equation (2.11) can be simplified as:

$$\frac{\dot{m}_{id}\sqrt{\gamma RT_0}}{AP_0} = \gamma \left(\frac{2}{\gamma + 1}\right)^{\frac{\gamma+1}{2(\gamma-1)}} \tag{2.13}$$

Discharge coefficient for a real gas flow can be introduced as:

$$C_D = \frac{\text{actual mass flow}}{\text{ideal mass flow}} = \frac{A_E}{A_R} \tag{2.14}$$

this value represent the departure from an ideal steady adiabatic reversible flow, and in can be measured as the ratio between the effective area of flow restriction A_E and the reference area A_R, usually the minimum cross-section area. More precisely A_E is the cross-sectional area at the throat of an ideal (no friction) duct which would pass the measured mass flow between two large reservoirs, considering the stagnation pressure for the upstream one and the static pressure for the other.

So, for a sub-critical flow ($M < 1$) the real mass flow rate at the minimum cross section, A_r, is:

Valve head area	$\frac{\pi D_v^2}{4}$
Port area at valve seat	$\frac{\pi D_p^2}{4}$
Minimum flow area	$\frac{\pi}{4}(D_p^2 - D_s^2)$
Curtain area	$\pi D_v L_v$

Table 2.2: Typical reference areas A_R for restricted flow

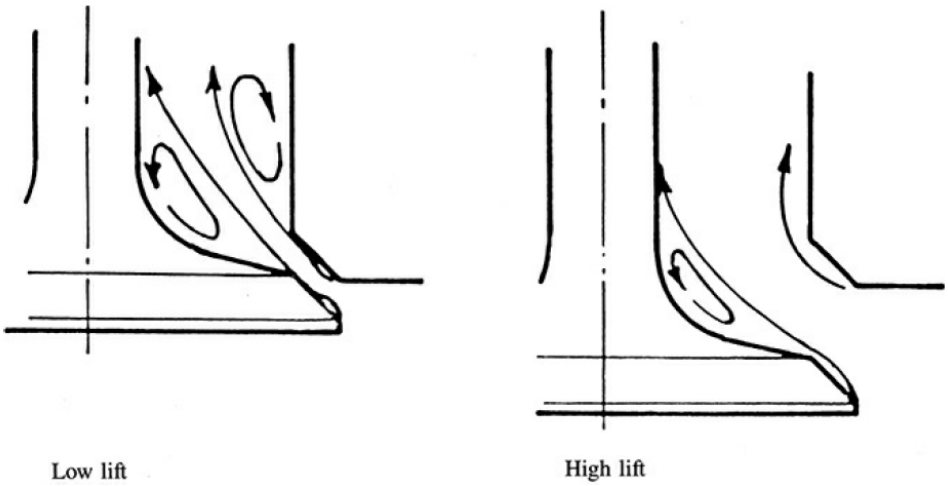


Figure 2.5: Influence of lift on exhaust flow pattern [12]

$$\dot{m}_r = \frac{C_D A_r P_0}{\sqrt{RT_0}} \left(\frac{p_T}{p_0}\right)^{\frac{1}{\gamma}} \left\{\frac{2\gamma}{\gamma - 1}\left[1 - \left(\frac{p_T}{p_0}\right)^{\frac{\gamma-1}{\gamma}}\right]\right\}^{\frac{1}{2}} \tag{2.15}$$

whereas for a chocked flow ($M = 1$):

$$\dot{m}_r = \frac{C_D A_r P_0}{\sqrt{RT_0}} \gamma^{\frac{1}{2}} \left(\frac{2}{\gamma + 1}\right)^{\frac{\gamma+1}{2(\gamma-1)}} \tag{2.16}$$

In the case of the flow through an exhaust p_0 is the cylinder pressure and p_T is the downstream system pressure. The combination of C_D and reference area gives the effective flow area of the port, and so the choice of A_R influences the measured performance of the assembly. There are multiple possible choices: the port area at the valve seat, the minimum flow area, the valve head area and the curtain area. Of those the latter is the most used, as it is relatively simple to calculate since it varies linearly with the valve diameter and lift. Tab. 2.2 shows some typical definitions of A_R.

The value of L_v has an effect on the flow through the restriction: at low lift the fluid incurs in more separation, as it can be seen in Fig. 2.5; moreover the seat angle has a key role on the behaviour of the fluid. Some different values of C_D based on curtain area shown in Fig. 2.6 for different shape of ports and valves. At high lifts the shape of the port influences heavily the value of C_D, optimization of the geometry can approach the performance of isolated valves with a straight downstream pipe.

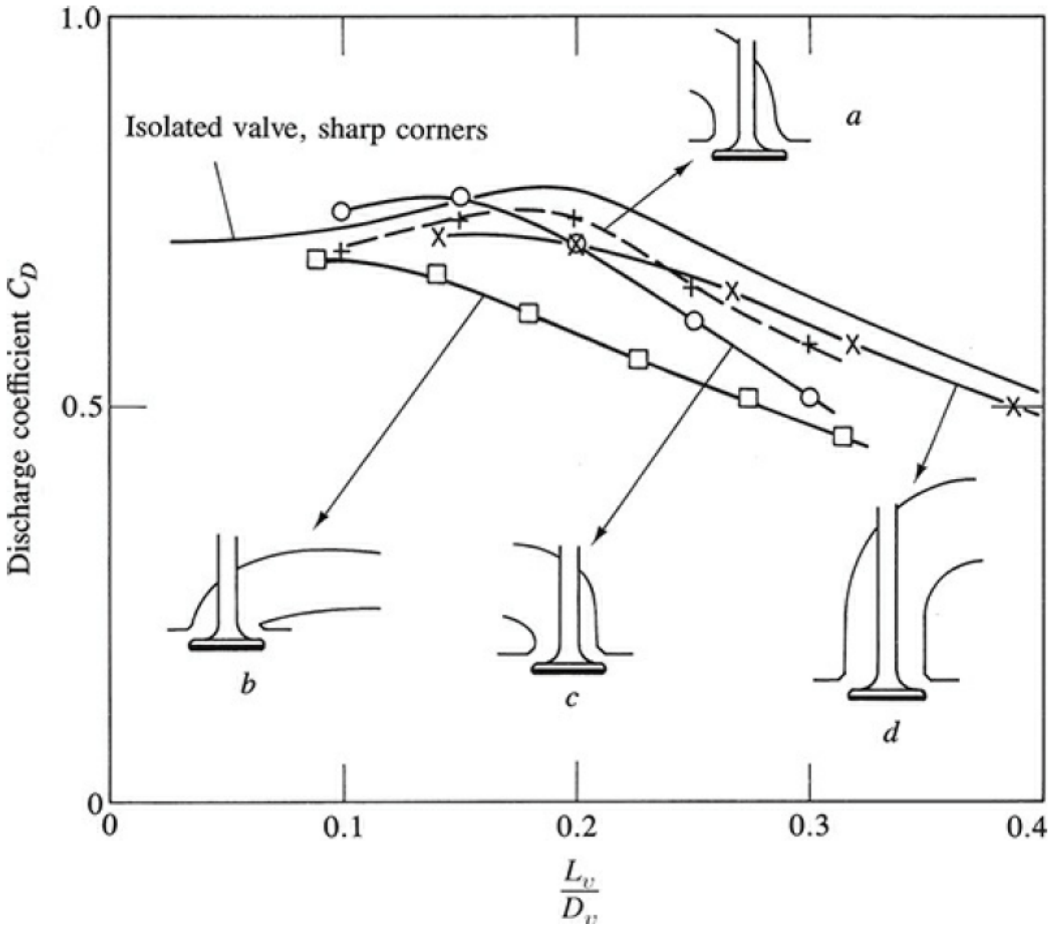


Figure 2.6: Discharge coefficient for different lift values and designs [12]

Chapter 3

Fluid dynamics and CFD principles

Since the majority of this work involves the study and optimization of the flow of air in an internal combustion engine exhaust port this chapter will give a brief overview of the governing equations of fluid dynamics, the basic principles of Computational Fluid Dynamics (CFD), an introduction to turbulence modeling, near wall treatment and some aspects of heat exchange.

As it will be explained in the following section, a fluid in motion can be described by a system of five equations: the conservation of mass, the conservation of momentum (for x,y and z) and the conservation of energy. Assuming thermodynamic equilibrium one can describe the state of a fluid by writing a relation between three variables; this is called **equation of state** and it's generic form is: $f(p, \rho, T) = 0$. The *equation of state for a perfect gas* is one of the most famous examples:

$$\frac{p}{\rho} = RT \tag{3.1}$$

where R is the gas constant and it's defined as: $R = \frac{R_u}{M}$, where R_u is the universal perfect gas constant, and M the molecular weight.

3.1 Governing equations

The following passages are covered in a more exhaustive manner in [14] and [15], in particular the contents of this section are borrowed heavily from the second chapter of [14].

The governing equations of fluid flow follow the principles of the conservation laws of physics, namely:

- Conservation of the total mass of fluid,
- Momentum rate of change equals the sum of the acting forces (Newton's second law),

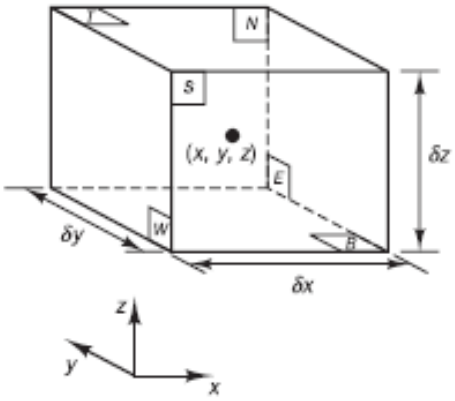


Figure 3.1: Element of fluid [14]

- Energy rate of change is equal to the sum of heat and work entering or exiting the system (first principle of thermodynamics).

A fluid can be regarded as a continuum, as it is done for a body in solid mechanics, thus molecular structure and motion on the microscopic scale can be ignored. Its behaviour can be explained by the variation in time and space of it's macroscopic properties (velocity, pressure, temperature and so on). An element (or volume) of fluid, the smallest possible portion that can be selected in such a way that the aforementioned properties are not affected by changes at the molecular level, is represented in Fig. 3.1, it's sides are defined along the axes of the global coordinates system and it is fixed in space. This type subdivision of a domain is called eulerian approach and it is the preferred one in CFD, the other type is the lagrangian approach and implies the definition of moving particles and the study their evolution in space and time. To describe the change of the properties four variables must be defined: x, y, z and t. It's important to consider that the element is so small that a generic value on a face can be expressed in respect of the one at the center by only considering the first two terms of a Taylor series expansion; as an example for the pressure on the two faces normal to the x axis: $p \pm \frac{\partial p}{\partial x}\frac{1}{2}\partial x$.

3.1.1 Mass conservation

For the element of fluid the mass balance is as follows: *the rate of increase of mass equals the net rate of flow into or out of it.* From a mathematical point of view the rate of mass increase and the flow rate across the are defined as follows:

$$\frac{\partial}{\partial t}(\rho \delta x \delta y \delta z) = \frac{\partial \rho}{\partial t}\delta x \delta y \delta z \tag{3.2}$$

Considering the entering flow along the x axis, the flow of mass across an element face becomes :

$$\left(\rho u - \frac{\partial(\rho u)}{\partial x}\frac{1}{2}\delta x\right)\delta y \delta z \tag{3.3}$$

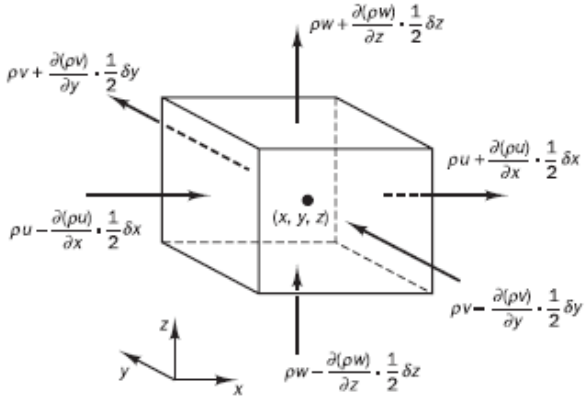


Figure 3.2: Mass flow in and out of an element [14]

Fig. 3.2 shows the total mass flows in and out an element, considering (3.2) and (3.3) for all the faces, after dividing for $\delta x \delta y \delta z$ one can write the *unsteady mass conservation* or *continuity equation*, both in extended and compact form:

$$\frac{\partial \rho}{\partial t} + \frac{\partial(\rho u)}{\partial x} + \frac{\partial(\rho v)}{\partial y} + \frac{\partial(\rho w)}{\partial z} = \frac{\partial \rho}{\partial t} + div(\rho \mathbf{u}) = 0 \tag{3.4}$$

The first term describes the rate of change in time inside the element and the second one is called the convective term and quantifies the net flow across the boundaries. For an incompressible fluid the density remains constant and the equation is thus simplified:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = div(\mathbf{u}) = 0 \tag{3.5}$$

3.1.2 Substantive derivative

In the lagrangian approach, considering that a particle changes it's properties with x, y, z and t, the substantive derivative with respect to time of a given quantity ϕ can be written as:

$$\frac{D\phi}{Dt} = \frac{\partial \phi}{\partial t} + \frac{\partial \phi}{\partial x}\frac{dx}{dt} + \frac{\partial \phi}{\partial y}\frac{dy}{dt} + \frac{\partial \phi}{\partial z}\frac{dz}{dt} = \frac{\partial \phi}{\partial t} + u\frac{\partial \phi}{\partial x} + v\frac{\partial \phi}{\partial y} + w\frac{\partial \phi}{\partial z} \tag{3.6}$$

In the eulerian approach, instead, for the rate of change in an element of fluid one can write:

$$\rho \frac{D\phi}{Dt} = \rho \left(\frac{\partial \phi}{\partial t} + \mathbf{u}\nabla\phi \right) \tag{3.7}$$

The generalization of the continuity equation for an arbitrary property ϕ is $\frac{\partial \rho\phi}{\partial t} + div(\rho\phi\mathbf{u})$, and it's relation with the substantive derivative is as follows:

$$\frac{\partial \rho\phi}{\partial t} + div(\rho\phi\mathbf{u}) = \rho \left(\frac{\partial \phi}{\partial t} + \mathbf{u}\nabla\phi \right) + \phi \left(\frac{\partial \rho}{\partial t} + div(\rho\mathbf{u}) \right) = \rho \frac{D\phi}{Dt} \tag{3.8}$$

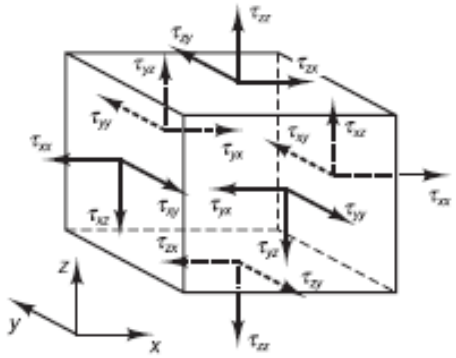


Figure 3.3: Stress components on a fluid element faces [14]

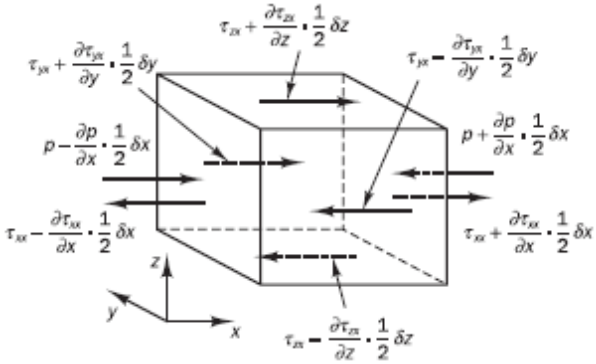


Figure 3.4: Stress components in the x direction [14]

where $\phi\left(\frac{\partial \rho}{\partial t} + div(\rho \mathbf{u})\right) = 0$. Considering momentum and energy, the respective conservation equations are as follows:

$$\begin{cases} \rho \frac{Du}{Dt} = \frac{\partial(\rho u)}{\partial t} + div(\rho u \mathbf{u}) \\ \rho \frac{Dv}{Dt} = \frac{\partial(\rho v)}{\partial t} + div(\rho v \mathbf{u}) \\ \rho \frac{Dw}{Dt} = \frac{\partial(\rho w)}{\partial t} + div(\rho w \mathbf{u}) \\ \rho \frac{DE}{Dt} = \frac{\partial(\rho E)}{\partial t} + div(\rho E \mathbf{u}) \end{cases} \tag{3.9}$$

3.1.3 Momentum equations

For a fluid particle the second principle ofon dynamics states that *the rate of increase in momentum is equal to the sum of the forces*, the former being the left term of the first three equations of (3.9). The forces can be divided in two categories: body (or volume) forces and surface forces. Considering the same fluid element as before Fig. 3.3 shows the stresses acting on its faces, each side of the element is subjected to a normal and transversal component. The pressure acts as the normal component and the viscous stresses as the transverse ones. The components of stress acting on the element along the x direction are identified in Fig. 3.4

For the pair of faces normal to x, y and z respectively:

$$\left[\left(p-\frac{\partial p}{\partial x}\frac{1}{2}\delta x\right)-\left(\tau_{xx}-\frac{\partial \tau_{xx}}{\partial x}\frac{1}{2}\delta x\right)\right]\delta y\delta z+$$
$$\left[-\left(p+\frac{\partial p}{\partial x}\frac{1}{2}\delta x\right)+\left(\tau_{xx}-\frac{\partial \tau_{xx}}{\partial x}\frac{1}{2}\delta x\right)\right]\delta y\delta z= \tag{3.10}$$
$$\left(-\frac{\partial p}{\partial x}+\frac{\partial \tau_{xx}}{\partial x}\right)\delta x\delta y\delta z$$

$$-\left(\tau_{yx}-\frac{\partial \tau_{yx}}{\partial y}\frac{1}{2}\delta y\right)\delta x\delta z+\left(\tau_{yx}-\frac{\partial \tau_{yx}}{\partial y}\frac{1}{2}\delta y\right)\delta x\delta z=\frac{\partial \tau_{yx}}{\partial y}\delta x\delta y\delta z \tag{3.11}$$

$$-\left(\tau_{zx}-\frac{\partial \tau_{zx}}{\partial z}\frac{1}{2}\delta z\right)\delta x\delta y+\left(\tau_{zx}-\frac{\partial \tau_{zx}}{\partial z}\frac{1}{2}\delta z\right)\delta x\delta y=\frac{\partial \tau_{zx}}{\partial z}\delta x\delta y\delta z \tag{3.12}$$

The sum of the previous equations, divided by $\delta x\delta y\delta z$ gives the total surface force per unit volume acting on the fluid in the x-direction, a generic term S_{Mx} of source of x-momentum for body forces can be added:

$$F_{Totx}=\frac{\partial(-p+\tau_{xx})}{\partial x}+\frac{\partial \tau_{yx}}{\partial y}+\frac{\partial \tau_{zx}}{\partial z}+S_{Mx} \tag{3.13}$$

So, following the same procedure for the y and z faces, the three equations of momentum can be written as follow:

$$\begin{cases}\rho\frac{Du}{Dt}=\frac{\partial(-p+\tau_{xx})}{\partial x}+\frac{\partial \tau_{xy}}{\partial y}+\frac{\partial \tau_{xz}}{\partial z}+S_{Mx}\\ \rho\frac{Dv}{Dt}=\frac{\partial(-p+\tau_{yy})}{\partial y}+\frac{\partial \tau_{yx}}{\partial x}+\frac{\partial \tau_{yz}}{\partial z}+S_{My}\\ \rho\frac{Dw}{Dt}=\frac{\partial(-p+\tau_{zz})}{\partial z}+\frac{\partial \tau_{zx}}{\partial x}+\frac{\partial \tau_{zy}}{\partial y}+S_{Mz}\end{cases} \tag{3.14}$$

If we were to write the stress tensor of the element it would be possible to distinguish between an hydrostatic and deviatoric part: the first considers only the pressure which is the same in all directions, the second takes in to account the viscous stress.

$$\sigma=\begin{pmatrix}-p & 0 & 0\\ 0 & -p & 0\\ 0 & 0 & -p\end{pmatrix}+\begin{pmatrix}\tau_{xx} & \tau_{xy} & \tau_{xz}\\ \tau_{xy} & \tau_{yy} & \tau_{yz}\\ \tau_{zx} & \tau_{zy} & \tau_{zz}\end{pmatrix} \tag{3.15}$$

3.1.4 Energy equation

As we have already seen, the first law of thermodynamics governs the rate of change in the element of fluid: *the rate of energy in the fluid particle is equal to the sum of heat added and of work done.* The rate of increase of energy of a fluid particle per unit volume is given by: $\rho\frac{DE}{Dt}$.

Considering the **rate of work done** in the x-direction one can compute the net value done by the forces (3.10), (3.11), (3.12) as:

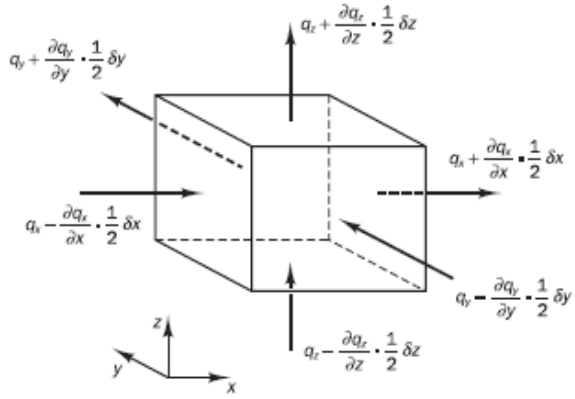


Figure 3.5: Heat flux in the fluid element [14]

$$\left[\frac{\partial(u(-p+\tau_{xx}))}{\partial x} + \frac{\partial(u\tau_{xy})}{\partial y} + \frac{\partial(u\tau_{xz})}{\partial z}\right]\delta x\delta y\delta z \tag{3.16}$$

similarly, along the y and z axes:

$$\left[\frac{\partial(v(-p+\tau_{yy}))}{\partial y} + \frac{\partial(v\tau_{yx})}{\partial x} + \frac{\partial(u\tau_{yz})}{\partial z}\right]\delta x\delta y\delta z \tag{3.17}$$

$$\left[\frac{\partial(w(-p+\tau_{zz}))}{\partial z} + \frac{\partial(w\tau_{zx})}{\partial x} + \frac{\partial(w\tau_{zy})}{\partial y}\right]\delta x\delta y\delta z \tag{3.18}$$

The sum of the last three divided by $\delta x\delta y\delta z$ gives the **total rate of work done per unit volume on the fluid particle by surface stresses**:

$$\begin{aligned} &-div\ (p\mathbf{u}) + \frac{\partial(u\tau_{xx})}{\partial x} + \frac{\partial(u\tau_{yx})}{\partial y} + \frac{\partial(u\tau_{zx})}{\partial z} + \\ &\frac{\partial(v\tau_{xy})}{\partial x} + \frac{\partial(v\tau_{yy})}{\partial y} + \frac{\partial(v\tau_{zy})}{\partial z} + \frac{\partial(w\tau_{xz})}{\partial x} + \frac{\partial(w\tau_{yz})}{\partial y} + \frac{\partial(w\tau_{zz})}{\partial z} \end{aligned} \tag{3.19}$$

Taking in to account the **exchange of heat**, Fig 3.5 shows the flux through the boundaries of the fluid element. The sum of the values on the opposite faces gives the **net value of heat transferred** along the three directions:

$$\begin{cases} -\frac{\partial q_x}{\partial x}\delta x\delta y\delta z \\ -\frac{\partial q_y}{\partial y}\delta x\delta y\delta z \\ -\frac{\partial q_z}{\partial z}\delta x\delta y\delta z \end{cases} \tag{3.20}$$

Recalling Fourier's law of heat transfer $q_i = -k\frac{\partial T}{\partial i}$, where k is the thermal conductivity of the fluid, and considering the sum of (3.20), the **exchange of heat through conduction for the fluid particle** equation is reached:

$$-div\mathbf{q} = div\ (k\nabla T) \tag{3.21}$$

Thus, combining the effects of both work and heat, the **equation of energy** can be written as follows:

$$E = E_{in} + \frac{u^2 + v^2 + w^2}{2} \tag{3.22}$$

where E_{in} represent the internal energy. It's rate of change is defined as:

$$\begin{aligned} \rho\frac{DE}{Dt} &= -div(p\mathbf{u}) + \frac{\partial(u\tau_{xx})}{\partial x} + \frac{\partial(u\tau_{yx})}{\partial y} + \frac{\partial(u\tau_{zx})}{\partial z} + \\ &\frac{\partial(v\tau_{xy})}{\partial x} + \frac{\partial(v\tau_{yy})}{\partial y} + \frac{\partial(v\tau_{zy})}{\partial z} + \frac{\partial(w\tau_{xz})}{\partial x} + \frac{\partial(w\tau_{yz})}{\partial y} + \frac{\partial(w\tau_{zz})}{\partial z} + \\ &div\ (k\nabla T) + S_{Epot} \end{aligned} \tag{3.23}$$

the term S_{Epot} indicates that the effect of potential energy are taken in to account. Equation (3.23) can be also written in terms of: kinetic energy, internal energy, temperature and enthalpy [14].

The energy equation has to be solved only if the heat transfer is part of the problem. For a **compressible flow** mass and momentum conservation are linked together by the fluid's equation of state $f(p, \rho, T) = 0$, thus all the equations described have to be solved. If the fluid is a liquid or a low speed gas the assumption of **incompressible flow** ($\rho = cost$) may be made, this implies that only the mass and momentum conservation equations need to be solved.

3.2 Navier-Stokes equations

As of right now the viscous stress components τ_{ij} are still unknown terms and need to be resolved. Moving forward the assumption that the fluid is isotropic will be made, which is true for all gasses. Isotropy means that a property of a given material is the same in all directions. A Newtonian fluid follows this principle: "*viscous stresses are proportional to the rates of deformation*"[14], for the element of fluid in Fig 3.1 there are 9 terms of deformation components, three that define linear elongation deformation (3.24) and six of shearing linear deformation (3.25):

$$\epsilon_{xx} = \frac{\partial u}{\partial x} \quad \epsilon_{yy} = \frac{\partial v}{\partial y} \quad \epsilon_{zz} = \frac{\partial w}{\partial z} \tag{3.24}$$

$$\epsilon_{xy} = \epsilon_{yx} = \frac{1}{2}\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right), \quad \epsilon_{xz} = \epsilon_{zx} = \frac{1}{2}\left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}\right), \quad \epsilon_{yz} = \epsilon_{zy} = \frac{1}{2}\left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}\right) \tag{3.25}$$

thus, the corresponding nine viscous stresses can be computed as follows:

$$\tau_{xx} = 2\mu\frac{\partial u}{\partial x} + \lambda\ div\ \mathbf{u}, \quad \tau_{yy} = 2\mu\frac{\partial v}{\partial y} + \lambda\ div\ \mathbf{u}, \quad \tau_{zz} = 2\mu\frac{\partial w}{\partial y} + \lambda\ div\ \mathbf{u} \tag{3.26}$$

$$\tau_{xy} = \tau_{yx} = \mu\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right), \quad \tau_{xz} = \tau_{zx} = \mu\left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}\right), \quad \tau_{yz} = \tau_{zy} = \mu\left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}\right) \tag{3.27}$$

where μ is the kinematic viscosity of the fluid and λ represent the bulk viscosity coefficient, for gases it ha been estimated: $\lambda = -\frac{2}{3}\mu$ [14],[15]. By using these stresses in the equations of conservation of momentum (3.14) the **Navier-Stokes equations** are obtained:

$$\begin{cases} \rho\frac{Du}{Dt} = -\frac{\partial p}{\partial x} + \frac{\partial}{\partial x}\left[2\mu\frac{\partial u}{\partial x} + \lambda\ div\ \mathbf{u}\right] + \frac{\partial}{\partial y}\left[\mu\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)\right] + \frac{\partial}{\partial z}\left[\mu\left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}\right)\right] + S_{Mx} \\ \rho\frac{Dv}{Dt} = -\frac{\partial p}{\partial y} + \frac{\partial}{\partial y}\left[2\mu\frac{\partial v}{\partial y} + \lambda\ div\ \mathbf{u}\right] + \frac{\partial}{\partial x}\left[\mu\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)\right] + \frac{\partial}{\partial z}\left[\mu\left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}\right)\right] + S_{My} \\ \rho\frac{Dv}{Dt} = -\frac{\partial p}{\partial z} + \frac{\partial}{\partial z}\left[2\mu\frac{\partial w}{\partial z} + \lambda\ div\ \mathbf{u}\right] + \frac{\partial}{\partial x}\left[\mu\left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}\right)\right] + \frac{\partial}{\partial y}\left[\mu\left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}\right)\right] + S_{Mz} \end{cases} \tag{3.28}$$

and can by rearranging the viscous stress can be rewritten in a more compact form:

$$\begin{cases} \rho\frac{Du}{Dt} = -\frac{\partial p}{\partial x} + div(\mu\nabla u) + S_{Mx} \\ \rho\frac{Dv}{Dt} = -\frac{\partial p}{\partial y} + div(\mu\nabla v) + S_{My} \\ \rho\frac{Dw}{Dt} = -\frac{\partial p}{\partial z} + div(\mu\nabla w) + S_{Mz} \end{cases} \tag{3.29}$$

This three, combined with the continuity equation (3.4), conservation of energy (3.23) and the equation of state of the fluid allow for a closed system of equations that can be solved defining initial and boundary conditions. Another form of the Navier-Stokes equations, for a incompressible flow, is presented [16]:

$$\rho\frac{\partial \mathbf{u}}{\partial t} + \rho\mathbf{u}\nabla\mathbf{u} = -\nabla p + \nabla\mathbf{f} + \mu\nabla^2\mathbf{u} \tag{3.30}$$

this form allows for an easy comparison with Newton's law ($m\cdot\mathbf{a} = \mathbf{F}$), the left term is equal to the product between mass and acceleration of the fluid whereas the terms on the right represent respectively: the pressure gradient, the external forces and the internal forces (such as stresses).

3.3 Turbulence

Above a certain threshold fluids cease to be laminar and become more chaotic and unpredictable, their properties of pressure, velocity and temperature change continuously both in space and time. This phenomenon is called turbulence and it is one of the most complex topic in fluid dynamics. A measure of fluid turbulence is given by the Reynolds number:

$$Re = \frac{\rho U L}{\mu} = \frac{U L}{\nu} \tag{3.31}$$

where U is the fluid mean velocity, D the characteristic length or dimension, μ the dynamic viscosity and ν the kinematic viscosity. Re represent the ratio between the inertial and viscous effect acting on a fluid, for sufficiently high Reynolds number (depending on the application) the flow moves from an ordered laminar structure to a more chaotic one. Conducting laboratory experiments on internal flows Osborne Reynolds reached (3.31) by varying the diameter of a pipe and the velocity of the fluid within, Fig. 3.6 shows a representation of the setup and the effects of turbulence on a streakline of colorant released on the fluid. There are three possible states:

- **Laminar flow** ($Re < 2100$): the fluid posses a stationary motion and it's structure can be thought as if a series of thin sheets are stacked upon each other. This foils move with different velocities (the one close to the pipe wall are fixed whereas those in the center have the most speed) and interact with each other through tangential stresses alone. The colorant remains a line or it diffuses to other layers only after a long time.

- **Transition flow** ($2100 \leq Re \leq 4000$): the colored line starts to show an unstable motion profile but it continues to retain a defined structure.

- **Turbulent flow** ($Re > 4000$): the dye immediately begins to spread to other portions of fluid until it's concentration in the downstream pipe becomes homogeneous. This behaviour is three dimensional, time dependent and non deterministic.

It has been said by Hinze (1975): *"Turbulent fluid motion is an irregular condition of flow in which the various quantities show a random variation with time and space coordinates, so that statistically distinct average values can be discerned"*[17], this means that instantaneous measured value for the same type of experiment but in distinct iterations can be extremely different from one another; mean values and statistic parameters, on the other hand, stay the same. This behaviour is mathematically conceptualized by the *Reynolds decomposition* and, for a generic property u, can be expressed as follows:

$$u(t) = U + u'(t) \tag{3.32}$$

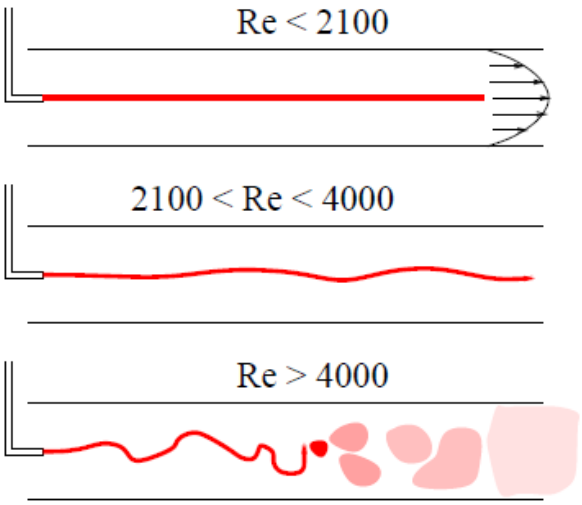


Figure 3.6: Reynolds experiment [16]

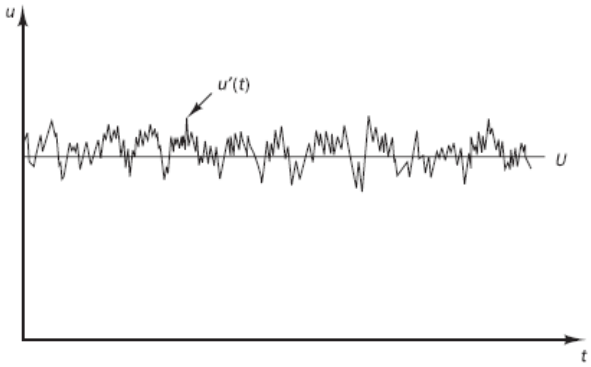


Figure 3.7: Mean and fluctuating component in turbulent flow [14]

where U is the average value and u' is variable component, see also Fig. 3.7.

3.3.1 Energy cascade and dissipation

Turbulence is always a three dimensional phenomenon and involves a wide range of scales of motion. It's vortices, also called eddies, are rotational structures that span various lengths and dissipate energy moving from the larger to the smaller ones, this characteristic of a turbulent flow is called *energy cascade*. Since it's smaller scales are still bigger than a molecule of fluid, turbulence can still be studied under the assumption of being in a continuum [16]. Eddies are intrinsically unstable and thus they degrade and fragment over time and in so doing they transfer energy, as shown in Fig. 3.8; this process takes place until a sufficiently small scale is reached for which there is not enough time for the instability to propagate before the structure (the small vortex) dissipation.

The vortex fragmentation allows for great diffusion and mixing inside the fluid, as it has been observed in the Reynolds experiment (Fig. 3.6). The phenomenon by which an eddy extracts energy from the flow is called *vortex stretching*, the distortion of the structures is a consequence of the different velocity gradients. It has been observed that

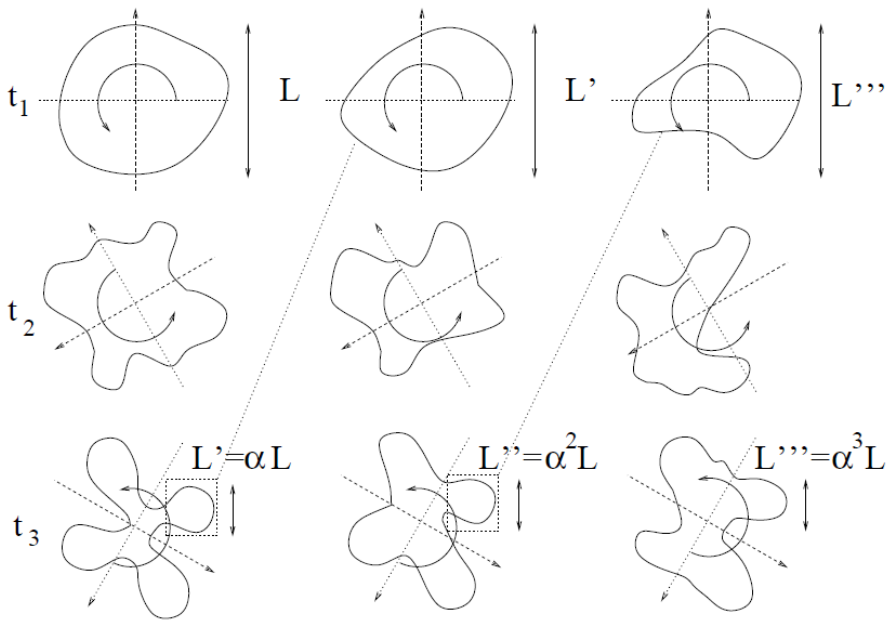


Figure 3.8: Vortex instability and creation of smaller scales [16]

the characteristic length and velocity of larger vortices is comparable to those of the mean flow [14], this leads to the assumption that inertial effects are still dominant at this dimensions. Thus, there is an increase in rotation rate and decrease in radial dimension that leads to smaller length and time scales. Moving down it can be seen that small vortices are stretched by the larger ones, and in so doing they absorb energy and allow its cascade. The smallest scales, also called *Kolmogorov micro scales*, posses a Reynolds number completely different (comparable to 1) and are dominated by viscous effects, by defining η and υ as the characteristic length and characteristic velocity respectively the micro scale Reynolds number becomes:

$$Re_\eta = \frac{\upsilon\eta}{\nu} = 1 \tag{3.33}$$

which underlines how inertial and viscous forces are comparable. The work performed by the fluid opposes the viscous effects and so there is a conversion from kinetic to thermal energy, thus the dissipation associated with the cascade. Since turbulence is not an unlimited phenomenon there has to be a balance between dissipation and production of turbulent energy, thus the Kolmogorov micro scales can also be defined by the viscosity of the flow and the rate at which the energy is dissipated. From dimensional analysis the following ratios between the larger and smaller scales are obtained [18], [14]:

$$\begin{cases} \frac{\eta}{L} = Re_L^{-\frac{3}{4}} \\ \frac{\tau}{t} = Re_L^{-\frac{1}{2}} \\ \frac{\upsilon}{U} = Re_L^{-\frac{1}{4}} \end{cases} \tag{3.34}$$

where L, t and U are large scales characteristic values of length, time and velocity respectively whereas η, τ and υ are the corresponding ones for the micro scales. Considering that typical value of Reynolds number can be order of magnitude greater than 10^3 it is easy see to how the values of the properties associated with the small scales can be extremely low. Large vortices are anisotropic and their behaviour strongly depends on the type of problem, by contrast smaller eddies can be considered as isotropic since their evolution depends only on the viscosity and rate of turbulent energy dissipation ϵ, as it has been proposed by Kolmogorov.

The previously descripted phenomena can be also viewed by considering the Navier-Stokes equations [16], given (3.30) for a viscous incompressible flow the kinetic energy is as follows:

$$K = \frac{1}{2}\int \rho \mathbf{u}\cdot\mathbf{u}dV = \frac{1}{2}\rho\int |\mathbf{u}|^2 dV \tag{3.35}$$

where V is the volume of fluid. Considering that the derivative with respect to time is:

$$\frac{dK}{dt} = \rho\int \mathbf{u}\frac{\partial \mathbf{u}}{\partial t}dV \tag{3.36}$$

the $\frac{\partial u}{\partial t}$ term can be obtained from (3.30) and by recalling the conservation of mass for incompressible fluids (3.5):

$$\frac{\partial K}{\partial t} = \int \left(-\rho\mathbf{u}\nabla(\mathbf{u}\mathbf{u}) - \mathbf{u}\nabla p + \mu\mathbf{u}\nabla^2\mathbf{u} + \rho\mathbf{u}\mathbf{f}\right) dV \tag{3.37}$$

some of the terms in the integral can be rewritten as:

$$\begin{cases} \mathbf{u}\nabla(\mathbf{u}\mathbf{u}) = \frac{1}{2}\nabla(\mathbf{u}|\mathbf{u}|^2) \\ \mathbf{u}\nabla p = \nabla(p\mathbf{u}) \\ \mathbf{u}\nabla^2\mathbf{u} = \nabla(\nabla\mathbf{u}\mathbf{u}) - |\nabla\mathbf{u}|^2 \end{cases} \tag{3.38}$$

by making use of Gauss's divergence theorem one eventually comes to:

$$\frac{dK}{dt} = \int \rho\mathbf{f}\cdot\mathbf{u}dV - \int \rho\nu|\nabla\mathbf{u}|^2 dV \tag{3.39}$$

the last term of (3.39) is the already seen *rate of dissipation for kinetic energy*:

$$\epsilon = \nu|\nabla\mathbf{u}|^2 \tag{3.40}$$

since this value is always positive it implies a continuous degradation of kinetic energy. Considering (3.39) if there were no external forces $\mathbf{f}$ acting on the fluid a state of no motion

would be eventually reached; instead, for a stationary flow, the two terms are equal and so $\frac{dK}{dt} = 0$. Another important aspect is that since the terms regarding the volume forces and the viscous effects are the only ones present in (3.39) then the effect of the pressure forces and the convective terms do not participate in varying the kinetic energy of the fluid but only in it's transferring, both in space and length scales. Finally it has to be noted that ϵ cannot be ignored, regardless of the Reynolds number: one could be inclined to believe that for great values of Re the viscosity would tend to zero and the kinetic energy would remain constant but, following the decrease on ν, the length scales become smaller and to that corresponds an increase in $\nabla \mathbf{u}$.

Another important assumption by Kolmogorov is that for sufficiently high Reynolds numbers, the small scales characteristics are universal for all turbulent flows and can be expressed only in therm of viscosity ν and dissipation ϵ [16]; thus it is possible to define the length, time and velocity for the micro scales used in (3.34) by also considering that, for a stationary flow, the dissipation rate of energy equals its production and that $\epsilon = \frac{U^3}{L}$ from dimensional analysis:

$$\begin{cases} \eta = \left(\frac{\nu^3}{\epsilon}\right)^{\frac{1}{4}} \\ \tau = \left(\frac{\nu}{\eta}\right)^{\frac{1}{2}} \\ \upsilon = \left(\frac{\nu}{\epsilon}\right)^{\frac{1}{4}} \end{cases} \tag{3.41}$$

3.3.2 Law of the Wall

Turbulence possess different behaviors that depend on the distance from a solid boundary, this has been empirically demonstrated and for a more accurate explanation one can refer to Wilcox's *"Turbulence Modeling for CFD"* [17].

It has been measured [17] that a fluid velocity varies at a logarithmic rate with respect to its distance to the nearest wall and that inertia and pressure variance effects tend to lower as well. Assuming δ as the boundary layer thickness the ratio $\frac{y}{\delta}$ can be defined and the above mentioned properties increase in accuracy as $\frac{y}{\delta}$ decreases, this is due to the fact that the ratio between vortices far and near the wall increases as $\frac{y}{\delta}$ becomes smaller, owing to the more varied range of length scales at high Re. Considering stresses it can also be noted that near wall variations tend to be negligible and one can assume a more or less constant value τ_w. The parameters $\frac{\tau_w}{\rho}\left[\frac{L^2}{t^2}\right]$ and $\nu = \frac{\mu}{\rho}\left[\frac{L^2}{t}\right]$ are used to begin dimensional analysis (where L stands for length and t for time). A term describing a length scale and one for velocity can be defined as follows:

$$u_\tau = \sqrt{\frac{\tau_w}{\rho}}, \qquad \textit{length scale} = \frac{\nu}{u_\tau} \tag{3.42}$$

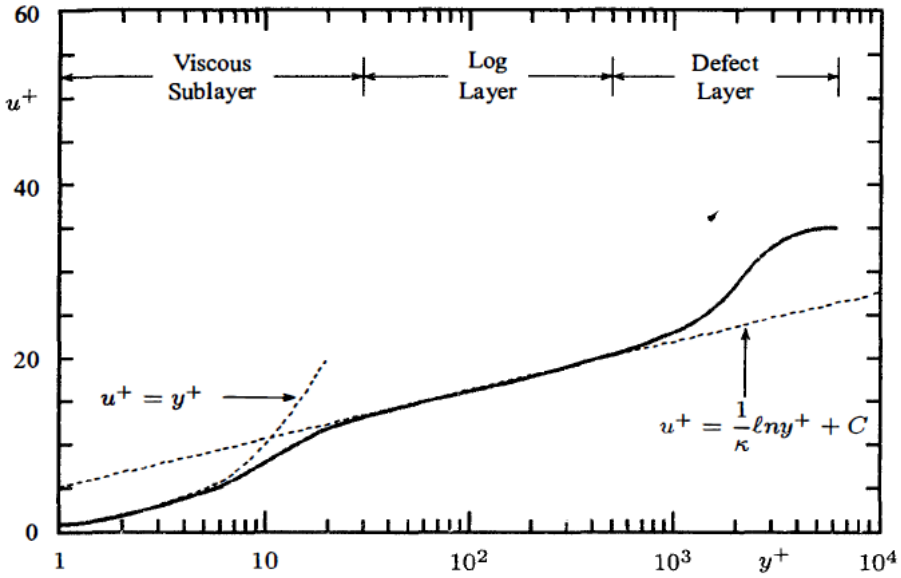


Figure 3.9: Dimensionless values near wall relation: velocity profile of the boundary layer [16]

The first of (3.42) is also known as friction velocity and it describes near wall speeds. By hypothesis, the rate of variation of stream-wise velocity with respect to boundary distance $\frac{\partial U}{\partial y}$ can be expressed in terms of the previous defined quantities $u_\tau, \frac{\nu}{u_\tau}$ and y, which brings:

$$\frac{\partial U}{\partial y} = \frac{u_\tau}{y} F\left(\frac{u_\tau y}{\nu}\right) \tag{3.43}$$

Results from experiments allow for the following approximation (where k is the von Karman's constant): $F\left(\frac{u_\tau y}{\nu}\right) \approx \frac{1}{k}$ if $\frac{u_\tau y}{\nu} \approx \infty$ [17], this is consistent with the assumption that viscous effects are more relevant near the wall. By integrating (3.43) one reaches the *Law of the Wall*:

$$\frac{U}{u_\tau} = \frac{1}{k} \log \frac{u_\tau y}{\nu} + C \tag{3.44}$$

it has been demonstrated that typical values for smooth walls are: $C \approx 5$ and $k \approx 0.41$ [17]. Bearing in mind this relation it is possible to define two dimensionless values for velocity and wall distance which are used for measurements in the near boundary region:

$$\begin{cases} u^+ = \frac{U}{u_\tau} \\ y^+ = \frac{u_\tau y}{\nu} \end{cases} \tag{3.45}$$

and so equation (3.44) can be rewritten as:

$$u^+ = \frac{1}{k} \log y^+ + C \tag{3.46}$$

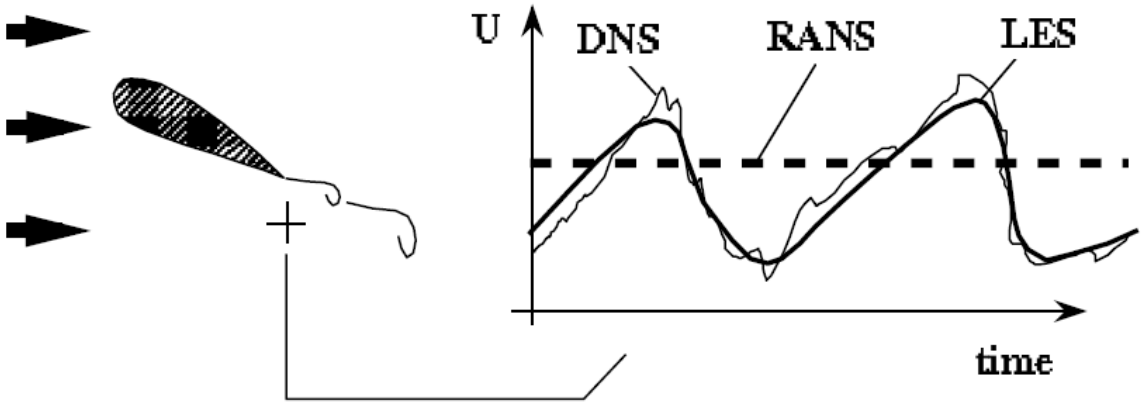


Figure 3.10: Example of different turbulence modeling approaches for the study of an airfoil wake [19]

Moreover, the above defined quantities and their relation are represented in Fig. 3.9, the continuous black line shows the measured values whereas the dotted lines are the theoretical results. It is possible to identify three distinct regions where the fluid behaves differently:

- **Viscous sub layer**: the near wall region of fluid, where the viscous stresses are predominant and the state of motion is laminar. The value of y^+ is typically smaller than 30 and the u^+ increases linearly.
- **Log layer**: the portion of boundary layer where the theoretical results match almost perfectly the experimental ones. It starts for values of y^+ above 30, u^+ follows a logarithmic law.
- **Defect layer**: the outermost part of the boundary layer, where the u^+ ceases to increase logarithmically. A model for velocity in this region has been proposed by Clauser [17].

3.3.3 Modeling of Turbulence

As it has already been observed all turbulence phenomena are characterized by energy cascade from large to small scales. There is no dissipation until the Kolmogorov micro scales and most of the fluid energy is present in the large vortices. Following equation (3.34) the distance between the characteristic lengths of big and small structures rapidly increases with the Reynolds number, thus arises the need to establish a threshold below which turbulence is no longer solved but modeled, in order to keep a reasonable computational time. There are three main approaches to tackle turbulence:

- **Direct Numerical Simulation (DNS)**: the governing equations are solved without any modeling. This brings great accuracy in results but the price to pay is the

need to resolve even the micro scales η. This implies an exorbitant computational cost. Their use is limited in academic research and for moderate Reynolds numbers $(10^3 - 10^4)$ [16]. In Fig. 3.10 the thin continuous line represent the complete velocity profile with respect to time, considering all the scales of motion.

- **Reynolds-Averaged Navier-Stokes equations (RANS)**: the opposite approach compared to the DNS method, only the mean part is solved and all the fluctuations are modeled (following 3.32). In Fig. 3.10 they are depicted by the dotted straight line. Many models have been proposed over the years and they are characterized by the number of equations that needs to be solved, the most famous examples are:
  - *Spalart-Allmaras*: a one equation model that solves for a kinematic viscosity like variable $\overline{\nu}$, developed primarily for aerodynamic applications. Since it only needs for an equation to be solved its computational cost is relatively low.
  - $K - \epsilon$: a family of two equations models, firstly developed by Launders and Spalding. The turbulent kinetic energy K and it's rate of dissipation ϵ (3.40) are used to create length and time scales $l_m = \frac{K^{\frac{3}{2}}}{\epsilon}$, $T = \frac{K}{\epsilon}$ and a turbulent viscosity $\nu_T = \frac{C_v l_m^2}{T}$ (where C_v is a constant to be determined). It is one of the most used models in engineering but it is not useful in conditions with fluid separations or adverse pressure gradients.
  - $K - \omega$: another family of two equations models, initially proposed by Kolmogorov. It still considers the kinetic turbulent energy but uses the rate of specific dissipation ω, defined as $\omega = \frac{\epsilon}{K}$, instead of ϵ. ω can be considered as characteristic frequency of turbulence. Unlike $K - \epsilon$ models, $K - \omega$ allows for a better resolution in near wall and adverse pressure gradient regions, however it suffers from an amplified sensitivity to boundary conditions.
- **Large Eddy Simulations (LES)**: an intermediate solution between DNS and RANS. The resolution scale is sufficiently small to capture the biggest turbulent structures, the smaller scales continue to be modeled. A considerable advantage of this procedure is the universal nature of the smaller scales, in fact only a turbulence model is needed, regardless of the geometry of the problem. Their computational cost, however, continues to be considerable. The thick continuous line in Fig. 3.10 describes well the model behaviour: it is sufficiently sensible to simulate the largest oscillation but filters out the smaller ones.

Fig. 3.11 shows the difference in turbulent structures obtained from RANS and a LES simulation of the same problem, it is possible to see that although the great scales and mean values are the same in both cases the RANS model loses details for the smaller structures and vortices. It's up to the analyst to evaluate if the lesser precision of a RANS simulation is still enough for the solution of a given fluid dynamic problem.

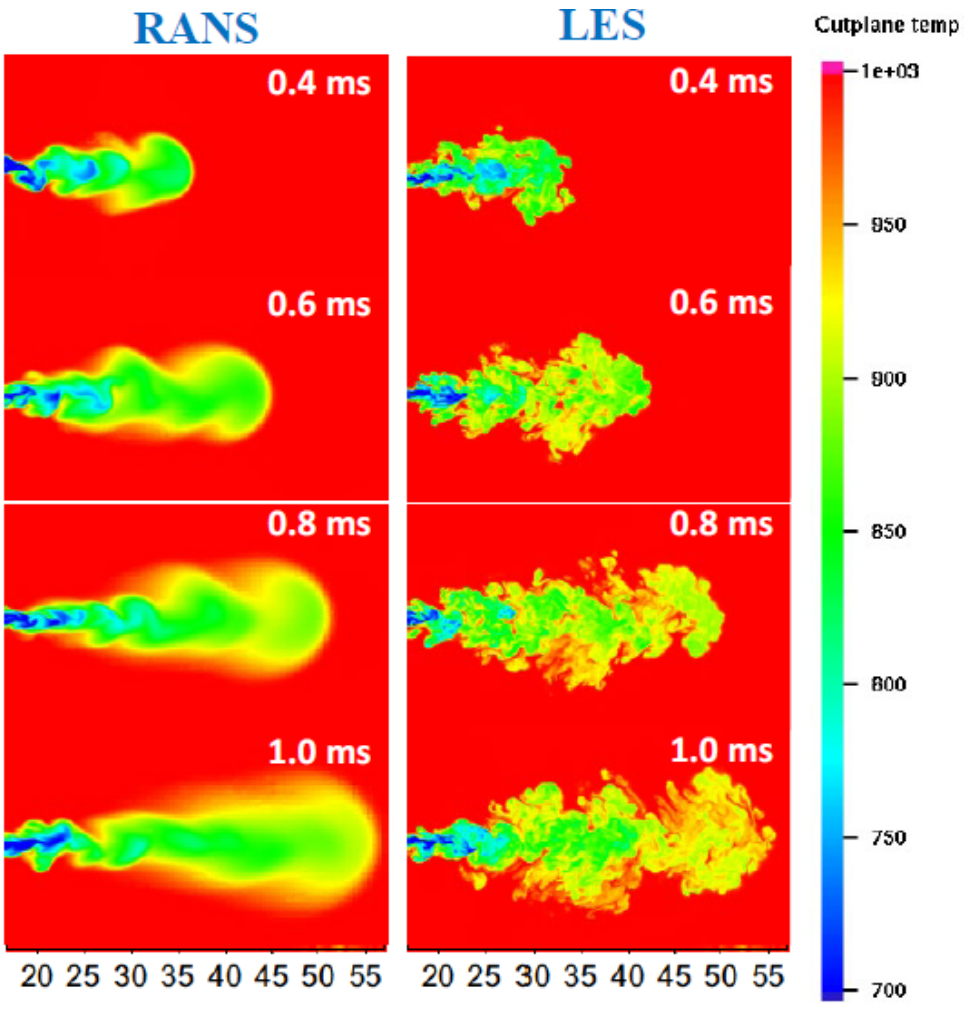


Figure 3.11: Difference between RANS and LES contours (Convergent Science lecture slides on turbulence modeling [20])

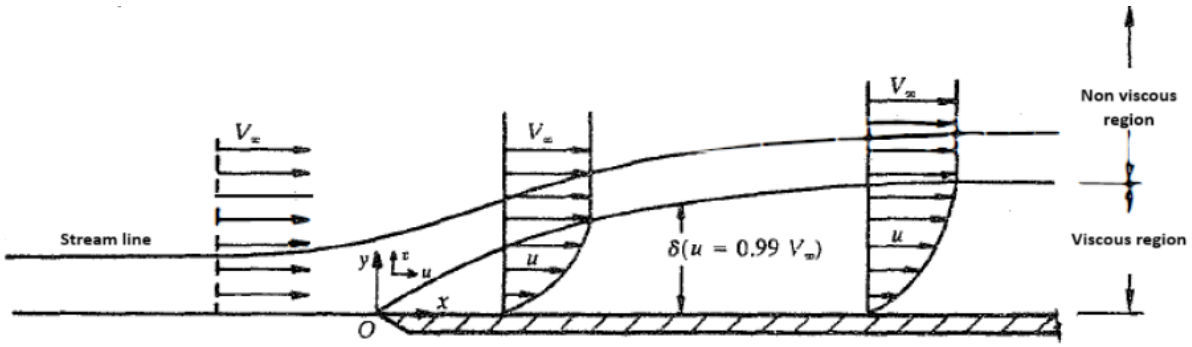


Figure 3.12: Hydrodynamic boundary layer [22]

3.4 Boundary layer analytical solution

The following section contains the theoretical solutions for hydrodynamic and thermal boundary layers of a incompressible flow over a thin plate. This models are limited in scope, but they are the foundation for near wall region solutions and wall functions. Heat exchange theoretical aspects are thoroughly explained in [21].

3.4.1 Hydrodynamic boundary layer

Considering a fluid in laminar motion, where the free flow velocity is u_∞, it is possible to reach *Blasius exact solution* [22] of the boundary layer by making the following assumptions:

- constant viscosity ($\mu = cost$)
- negligible shear stresses in the y-direction ($\tau_y = 0$)
- steady state flow ($\frac{\partial u}{\partial t} = 0$)
- incompressible fluid ($\rho = cost$)

it is then possible to simplify the conservation of mass and momentum along the x and y directions as:

$$\begin{cases} \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \\ u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial x} + \frac{\mu}{\rho}\frac{\partial^2 u}{\partial y^2} \\ \frac{\partial p}{\partial y} = 0 \end{cases} \tag{3.47}$$

For a thin plate the hydrodynamic boundary layer of a laminar flow is depicted in Fig. 3.12. Friction with the solid causes a non negligible pressure drop in the x-direction as noted by the $\frac{\partial p}{\partial x}$, this can be easily taken account of by applying the Darcy-Weissbach equation (3.48), where ξ is the coefficient for distributed pressure losses.

$$\frac{\Delta p}{L} = \frac{1}{2}\rho\xi u^2 \tag{3.48}$$

It is possible to simplify further by considering the order of magnitude of the acting terms. While $\frac{\partial p}{\partial x}$ is not zero it can be omitted as the pressure losses for a free stream are extremely small, furthermore for the values inside the boundary layer, except for the region closest to the plate, it is plausible to assume:

$$y \approx \delta, \quad u \approx u_\infty \tag{3.49}$$

where δ is the boundary layer thickness, so the continuity equation brings:

$$\frac{u_\infty}{x} + \frac{v}{\delta} \rightarrow v \approx \frac{u_\infty \delta}{x} \tag{3.50}$$

By replacing the terms in the momentum conservation equation with (3.49) and (3.50) and assuming $\frac{\partial p}{\partial x} = 0$ one reaches:

$$u_\infty \frac{u_\infty}{x} + u_\infty \frac{\delta u_\infty}{x\delta} \approx \nu \frac{u_\infty}{\delta^2} \tag{3.51}$$

after some algebraic operations it is possible to write the relation between the local boundary layer thickness and the distance from the front of the plate:

$$\delta(x) = \frac{x}{\sqrt{Re_x}} \tag{3.52}$$

where $Re_x = \frac{u_\infty x}{\nu}$ is the local Reynolds number with respect to distance x, the thickness of the boundary layer is defined as the value of y for which $u = 0,99u_\infty$. Experimental data show that $u(y)$ has the same profile for all x values, this means that dimensionless velocity $\frac{u}{u_\infty}$ can be expressed in terms of the dimensionless wall distance $\frac{y}{\delta}$, thus simplifying the problem into one total derivative equation. Assuming:

$$\frac{u}{u_\infty} = f\left(\frac{y}{\delta}\right) = f\left(\frac{y}{x}\frac{x}{\delta}\right) = f\left(\frac{y}{x}\sqrt{\frac{u_\infty x}{\nu}}\right) = g(\eta) \tag{3.53}$$

where η is a dimensionless value:

$$\eta = y\sqrt{\frac{u_\infty}{\nu x}} \tag{3.54}$$

It is possible to define a *stream function* (potential) $\psi(x, y)$ so that the continuity equation remains balanced:

$$u = \frac{\partial \psi}{\partial y} \quad v = -\frac{\partial \psi}{\partial x} \tag{3.55}$$

considering this definition, at a distance x from the front of the plate it is possible to write:

$$\begin{aligned}\psi &= \int u_x dy + C(x) = u_\infty \int g(\eta)\frac{dy}{d\eta}d\eta + C(x) = \\ &= u_\infty \int g(\eta)\frac{1}{\sqrt{\frac{u_\infty}{\nu x}}}d\eta + C(x) = \sqrt{u_\infty \nu x}\int g(\eta)d\eta + C(x)\end{aligned} \tag{3.56}$$

by assuming $\int g(\eta)d\eta = f(\eta)$ and $C(x) = 0$ one can write:

$$\begin{cases}\psi = f(\eta)\sqrt{u_\infty \nu x} \\ f'(\eta) = g(\eta) = \frac{u}{u_\infty}\end{cases} \tag{3.57}$$

through algebra it is possible to compute $u_x, u_y, \frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}, \frac{\partial^2 u}{\partial y^2}$ and to rewrite the terms of x-momentum equation:

$$\begin{cases}u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{1}{2}\frac{u_\infty^2}{x}f(\eta)\frac{\partial^2 f}{d\eta^2} \\ \nu\frac{\partial^2 u_x}{\partial y^2} = \nu u_\infty \frac{d^3 f}{d\eta^3}\left(\frac{u_\infty}{\nu x}\right)\end{cases} \tag{3.58}$$

from which an third order non linear differential equation is obtained:

$$2\frac{d^3 f}{d\eta^3} + f(\eta)\frac{d^2 f}{d\eta^2} \tag{3.59}$$

with the following the boundary conditions:

$$\begin{cases}y = 0 \Rightarrow u = 0 \\ y = 0 \Rightarrow v = 0 \\ y \to \infty \Rightarrow \frac{\partial u}{\partial y} = 0\end{cases} \tag{3.60}$$

that can be rewritten according to equation (3.54):

$$\begin{cases}\eta = 0 \Rightarrow f'(\eta) = 0 \\ \eta = 0 \Rightarrow f(\eta) = 0 \\ \eta \to \infty \Rightarrow f'(\eta) = 1\end{cases} \tag{3.61}$$

The boundary layer is usually completely developed for $\eta \approx 5$, this means:

$$\frac{\delta}{x} = 5\sqrt{\frac{1}{Re_x}} \tag{3.62}$$

Defining the shear stresses acting on the wall as (3.63), it is possible to reach the local

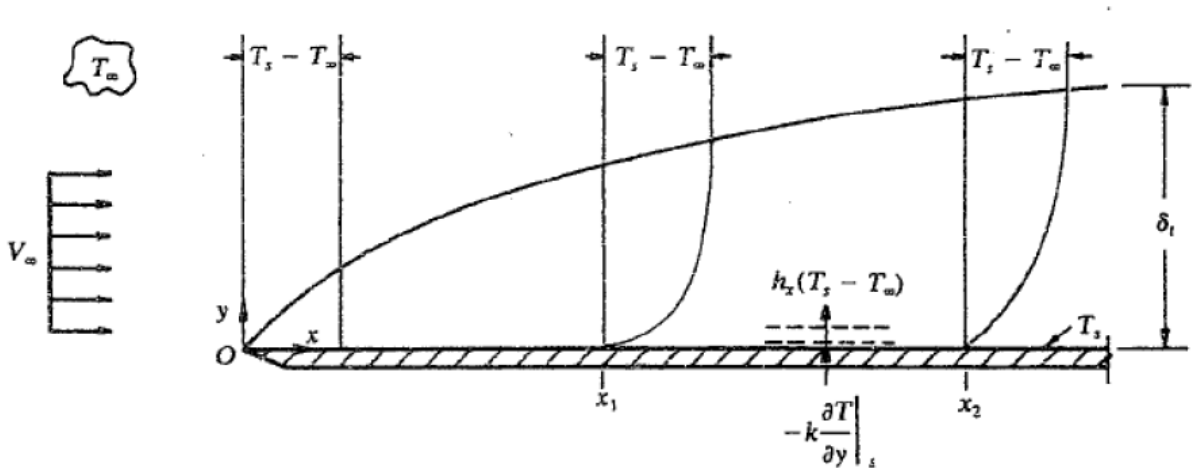


Figure 3.13: Thermal boundary layer [22]

and global friction factors, C_x and C_L respectively:

$$\tau_s = \mu \left.\frac{\partial u}{\partial y}\right|_{y=0} = \ ... = 0.332\mu u_\infty \sqrt{\frac{u_\infty}{\nu x}} \tag{3.63}$$

$$C_x = \frac{\tau_s}{\rho\frac{u_\infty^2}{2}} = 2\frac{0.332\mu u_\infty \sqrt{\frac{u_\infty}{\nu x}}}{\rho u_\infty^2} = 0.664\sqrt{\frac{\nu}{u_\infty x}} = \frac{0.664}{\sqrt{Re_x}} \tag{3.64}$$

$$C_L = \frac{1}{L}\int_0^L C_x dx = \frac{1}{L} 0.664\sqrt{\frac{\nu}{u_\infty}} \int_0^L x^{\frac{1}{2}} dx = 1.328\sqrt{\frac{1}{Re_L}} \tag{3.65}$$

3.4.2 Thermal boundary layer

Considering now the effects on temperature for a laminar flow, the thermal boundary layer is depicted in Fig. 3.13, assuming different values for the free flow and plate temperatures, T_∞ and T_s respectively. The objective of this section is to obtain the convective heat transfer coefficient, following a procedure called *Pohlhausen solution* [22]. For a given volume inside the boundary layer the energy equation (3.23) can be rewritten as:

$$u\frac{\partial T}{\partial x} + v\frac{\partial T}{\partial y} = \alpha\frac{\partial^2 T}{\partial y^2} \tag{3.66}$$

the shape is analogous to the x-momentum equation. Having assumed both the fluid and plate temperature, defined the boundary layer thickness as the value of y for which $T = 0.99\ T_\infty$ and hypothesized the absence of heat generation it is possible to write the boundary conditions:

$$\begin{cases} y = 0 \rightarrow T = T_s \\ y = 0 \rightarrow \frac{\partial^2 T}{\partial y^2} = 0 \\ y = \delta_T \rightarrow T = 0.99\ T_\infty \\ y = \delta_T \rightarrow \frac{\partial T}{\partial y} = 0 \end{cases} \tag{3.67}$$

The second equation derives from *Fourier's law*: $\nabla^2 T = \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2} = 0$, the flow is supposed incompressible and laminar. The *dimensionless temperature* can be defined:

$$\theta = \frac{T - T_S}{T_\infty - T_S} \tag{3.68}$$

Velocity and θ within the boundary layer are analogous if the thermal diffusivity α and kinetic viscosity ν are equal; the relation between this two properties is expressed by the *Prandtl number*:

$$Pr = \frac{\nu}{\alpha} \tag{3.69}$$

By employing a similar approach to what has been done in the previous section it is possible to transform equation (3.66) into a total differential, and rewrite the boundary conditions:

$$\frac{d^2\theta}{d\eta^2} + f\frac{Pr}{2}\frac{d\theta}{d\eta} = 0 \tag{3.70}$$

$$\begin{cases} \eta = 0 \rightarrow \theta = 0 \\ \eta = \infty \rightarrow \theta = 1 \end{cases} \tag{3.71}$$

The thermal gradient on the plate surface is:

$$\left.\frac{\partial T}{\partial y}\right|_{y=0} = (T_\infty - T_S)\sqrt{\frac{u_\infty}{\nu x}}\left.\frac{\partial \theta}{\partial \eta}\right|_{\eta=0} = \ \ldots\ = (T_\infty - T_S)\sqrt{\frac{u_\infty}{\nu x}}0.332Pr^{\frac{1}{3}} \tag{3.72}$$

since the flow is laminar the heat transfer is achieved by pure conduction, introducing λ and h as the conductive and convective heat transfer coefficients respectively it is possible to determine the local heat transfer coefficient as follows:

$$\dot{q} = -\lambda \left.\frac{\partial T}{\partial y}\right|_{y=0} = h_x(T_s - T_\infty) \tag{3.73}$$

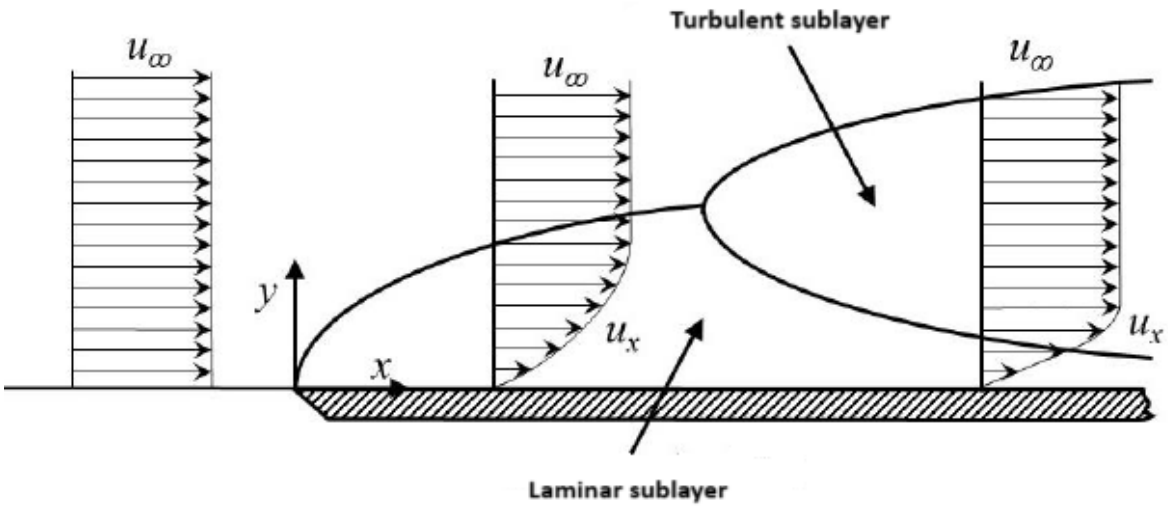


Figure 3.14: Division in turbulent and laminar sub-layers [22]

$$h_x = \frac{\lambda}{T_\infty - T_S} \left.\frac{\partial T}{\partial y}\right|_{y=0} = \lambda \sqrt{\frac{u_\infty}{\nu x}} 0.332 Pr^{\frac{1}{3}} \tag{3.74}$$

which leads to the local *Nusselt number*, that measures the ratio between total and convective heat transferred, the former composed by the sum of both conduction and convection:

$$Nu_x = \frac{h_x x}{\lambda} = \sqrt{\frac{u_\infty x}{\nu}} 0.332 Pr^{\frac{1}{3}} = 0.332 \ Re_x^{\frac{1}{2}} \ Pr^{\frac{1}{3}} \tag{3.75}$$

Global values are computed as:

$$h_L = \frac{1}{L} \int_0^L h_x dx = \lambda \sqrt{\frac{u_\infty L}{\nu}} 0.664 Pr^{\frac{1}{3}} \tag{3.76}$$

$$Nu_L = 0.664 \ Re_L^{\frac{1}{2}} \ Pr^{\frac{1}{3}} \tag{3.77}$$

3.4.3 Momentum and heat transfer in a turbulent flow

For sufficiently high Reynolds numbers after a specific distance from the front of the plate the flow becomes turbulent, this behaviour is also observable inside the boundary layer: a division between two sub-layers happen, the one nearest to the surface remains laminar whereas the furthest becomes turbulent, Fig. 3.14 shows this phenomenon for the velocity.

Near the wall the absence of turbulence hampers the transfer of heat and momentum, thus the laminar sub-layer can be described by only considering conduction and viscous effects:

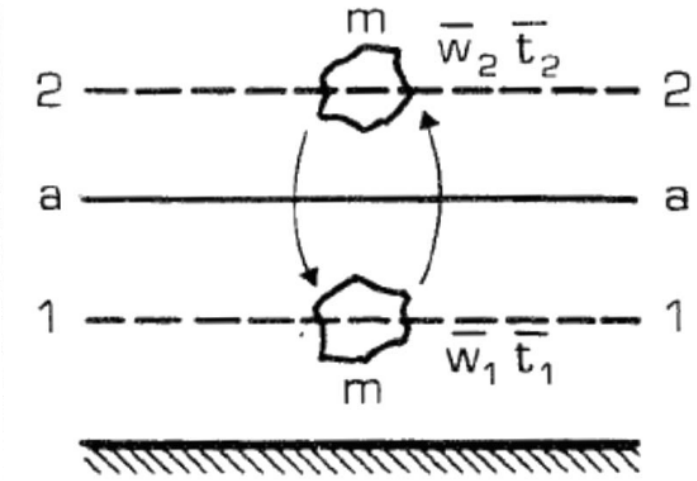


Figure 3.15: Transport of heat and momentum between two region possessing different velocities and temperatures [22]

$$\begin{cases} \tau_s = \mu \left.\frac{\partial u}{\partial y}\right|_{y=0} \\ \dot{q}_l = -\lambda \left.\frac{\partial T}{\partial y}\right|_{y=0} \end{cases} \tag{3.78}$$

after some math operations the following relation can be written the exchanged heat and viscous stress:

$$\dot{q}_l = -\frac{\lambda}{\mu}\tau_s \frac{\partial T}{\partial u} \tag{3.79}$$

In the turbulent sub-layer the fluid particles motion is no longer completely parallel to the plate but possesses a vertical velocity component, rendering more chaotic the overall behaviour of the fluid. Considering Fig. 3.15 it is possible to explain this phenomenon by creating a virtual plane $a-a$ separating two regions of the fluid, characterized by different values of temperature and velocity. If $\bar{u}_i$ and $\bar{T}_i$ are the mean values in the regions, a characteristic length l in the direction of the vertical axis can be hypothesized for which, as the particles travel along it, the quantities $\bar{u}_i$ and $\bar{T}_i$ are conserved; after having traveled the entire distance the particles assume the values of the destination region, transferring momentum and heat in the process. The quantities of *mean mass flow rate* and *rate of change of momentum with respect to area* are expressed in the equations (3.80), the latter in particular is representative of the apparent shear stresses acting on the $a-a$ plane in a perpendicular direction to length l and their action can accelerate or slow down the fluid, they are defined as *Reynolds stresses* [22] and are independent from the viscosity of the fluid (3.81).

$$\begin{cases} \frac{\dot{m}}{A} = \rho u \\ \rho u(\bar{u}_i - \bar{u}_j) \end{cases} \tag{3.80}$$

$$\tau_{i\to j} = \rho u(\bar{u}_i - \bar{u}_j) \tag{3.81}$$

The transversal motion of the particles is also causes heat exchange through the $a-a$ plane as well:

$$\dot{q}_t = \rho u_t c_p(\bar{T}_i - \bar{T}_j) \tag{3.82}$$

since both momentum and heat exchange are governed by the same in principle it is possible to write a relation between the two, in similar manner to the laminar flow:

$$\dot{q}_t = c_p \tau_{j\to i} \frac{\bar{T}_i - \bar{T}_j}{\bar{u}_j - \bar{u}_i} \tag{3.83}$$

taking in to account the temperature and velocity of plane $a-a$, the differences between the mean values of the regions can be rewritten (3.84) and a version of the *Reynolds analogy* [22] is reached:

$$\begin{cases} \bar{u}_j - \bar{u}_i = \bar{u}_2 - u_a - (\bar{u}_1 - u_a) = \left.\frac{d\bar{u}}{dy}\right|_{y=a} l \\ \bar{T}_i - \bar{T}_j = \bar{T}_1 - T_a - (\bar{T}_2 - T_a) = -\left.\frac{d\bar{T}}{dy}\right|_{y=a} l \end{cases} \tag{3.84}$$

$$\dot{q}_t = -c_p \tau_{j\to i} \frac{d\bar{T}}{d\bar{u}} \tag{3.85}$$

equation (3.85) correlates the change of local apparent stresses and heat exchange, due to transversal motion of fluid in turbulent flow. By observing the relations for both laminar (3.79) and turbulent (3.85) conditions it can be concluded that:

$$\frac{\lambda}{\mu} = c_p \to \frac{\lambda}{\mu c_p} = 1 \to Pr = 1 \quad \Rightarrow \quad \dot{q}_t \approx \dot{q}_l \tag{3.86}$$

Having defined the values for the transition between turbulent and laminar sub-layers as u_L and T_L, the integration along the thickness of the boundary layer brings:

$$\int_0^{u_L} \frac{\dot{q}}{\tau}\frac{\mu}{\lambda} du = \int_{T_s}^{T_L} dT \to ... \to \frac{\dot{q}}{\tau}\frac{\mu}{\lambda} u_L = T_s - T_L \tag{3.87}$$

$$\int_{u_L}^{u_\infty} \frac{\dot{q}}{\tau}\frac{1}{c_p} du = \int_{T_L}^{T_\infty} dT \to ... \to \frac{\dot{q}}{\tau}\frac{1}{c_p}(u_\infty - u_L) = T_L - T_\infty \tag{3.88}$$

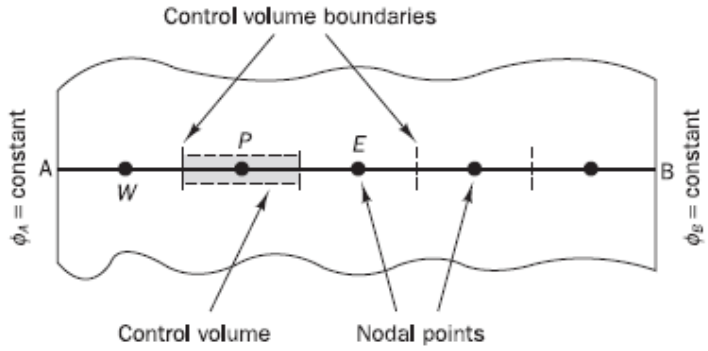


Figure 3.16: One dimensional domain [14]

after some passages one obtains the *Prandtl-Taylor analogy* [22], which allows for a rapid computation of the convective heat transfer coefficients, provided that the values of the local friction coefficient C_x and the dimensionless velocity $\frac{u_L}{u_\infty}$ can be obtained from experimental data:

$$\begin{cases} Nu_x = \frac{C_x}{2} Re_x \frac{Pr}{1+\frac{u_L}{u_\infty}(Pr-1)} & \leftarrow \quad Pr \neq 1 \\ Nu_x = Re_x \frac{C_x}{2} & \leftarrow \quad Pr = 1 \end{cases} \tag{3.89}$$

3.5 Finite volume method

This section outlines the principles of the finite volume method for CFD, a more complete and straightforward approach is presented in [14].

In most fluid dynamics problems the effect of convection and diffusion must be taken into account, for a given property ϕ it is possible to define the *transport equation* (3.90) where the effects of rate of change, convection, diffusion (Γ = diffusion coefficient) and possible sources are written left to right.

$$\frac{\partial(\rho\phi)}{\partial t} + div(\rho\phi\mathbf{u}) = div(\Gamma\nabla\phi) + S_\phi \tag{3.90}$$

the key aspect of the the finite volume method is the integration over a number of control volumes of the previous equation, which leads to:

$$\int \frac{\partial(\rho\phi)}{\partial t} dV + \int div(\rho\phi\mathbf{u}) dV = \int div(\Gamma\nabla\phi) dV + \int S_\phi dV \tag{3.91}$$

For a one dimensional steady-state problem shown in Fig. 3.16 the rate of change therm can be ignored and it is possible to identify three steps:

- **1. Grid generation**

  The points A and B are characterized by boundary conditions ϕ_a and ϕ_b respectively. The domain is divided in discrete regions called "finite (or control) volumes". Each

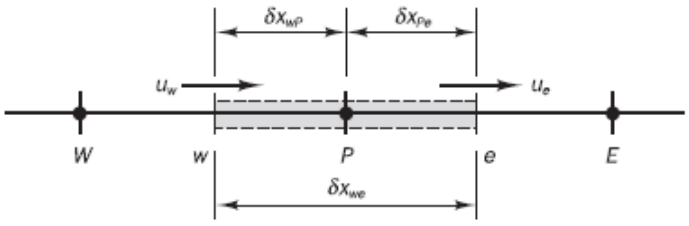


Figure 3.17: One dimensional finite volume notation [14]

volume possesses a center node such as P in Fig. 3.16. In a one dimensional geometry it is possible to follow the notation given in Fig. 3.17: P is the current central node and W and E are the neighboring volume's center nodes, West and East respectively, whereas w and e identify the faces. Distances between the nodes are identified by δx_{WP} and δx_{PE}. The linear dimension of a single volume is $\Delta x = \delta x_{we} = \delta x_{wP} + \delta x_{Pe}$.

- **2. Discretisation**

Transport and continuity equations, assuming no sources and a steady flow, are expressed in (3.92), their integration over a generic control volume brings (3.93)

$$\begin{cases} \frac{d(\rho u \phi)}{dx} = \frac{d}{dx}\left(\Gamma \frac{d\phi}{dx}\right) \\ \frac{d(\rho u)}{dx} = 0 \end{cases} \tag{3.92}$$

$$\begin{cases} (\rho u A \phi)_e - (\rho u A \phi)_w = \left(\Gamma \frac{d\phi}{dx}\right)_e - \left(\Gamma \frac{d\phi}{dx}\right)_w \\ (\rho u A)_e - (\rho u A)_w = 0 \end{cases} \tag{3.93}$$

By assuming that the area of the volume is constant throughout the length equations (3.93) may be simplified and divided by A, thus it is possible to define F and D, which represent *convective mass flux per unit area* and *diffusion conductance at cell faces* respectively:

$$F_i = (\rho u)_i \qquad D_i = \frac{\Gamma_i}{\delta x_{ij}} \tag{3.94}$$

which give a discrete form of the problem, in an approach called *central differencing*:

$$\begin{cases} F_e \phi_e - F_w \phi_w = D_e(\phi_E - \phi_P) - D_w(\phi_P - \phi_W) \\ F_e - F_w = 0 \end{cases} \tag{3.95}$$

The first of the two requires the definition of additional schemes to calculate the values of ϕ at the volume faces, the second one requires the knowledge of the velocity field.

- **3. Solution**

  The discrete equations (3.95) are written for each center node, in order to use the boundary conditions at the domain extremes the corresponding equations of the control volumes are modified accordingly. The result is a linear equations system that can be solved by applying a multitude of techniques [14].

3.5.1 Discretisation schemes

In a numerical model, given that the number of volumes (or elements, or cells) is finite, the results obtained do not match the exact solution of the problem but are an approximation, the goodness of which depends on properties such as *conservativeness, boundedness, transportiveness* and *accuracy.*

- **Conservativeness**

  This property describes the consistency with which the arbitrary value ϕ is represented in the domain; in other words the flux of ϕ exiting a given volume from a face must be the same entering the adjacent control volume from the same face. Considering an entire domain of finite volumes the conservation of ϕ must be respected and so it has to be that the entering flux at one extreme matches the exiting one at the other: $\phi_B - \phi_A = 0$.

- **Boundedness**

  Boundedness implies that in the absence of sources the value of a given node property ϕ must be contained within the values of the nodes in the vicinity. Not only that, but the coefficients of the nodal equations have to be all positive or negative. This can be synthesized in the *Scarborough condition* [14]:

$$\frac{\sum |a_{nb}|}{a'_P} \leq 1 \quad \text{at all nodes} \quad \textbf{or} \quad < 1 \quad \text{at one node at least} \tag{3.96}$$

  where a'_p is the coefficient for the center node and the numerator takes in to account all the near ones. So, if the value of ϕ varies for one node, the neighboring ones should change in a similar fashion.

- **Transportiveness**

  Considering three nodes as those in Fig. 3.18, the value of ϕ at point P is dependent from the neighboring nodes W and E. It is possible to define the *Peclet number* as a ratio between convection and diffusion:

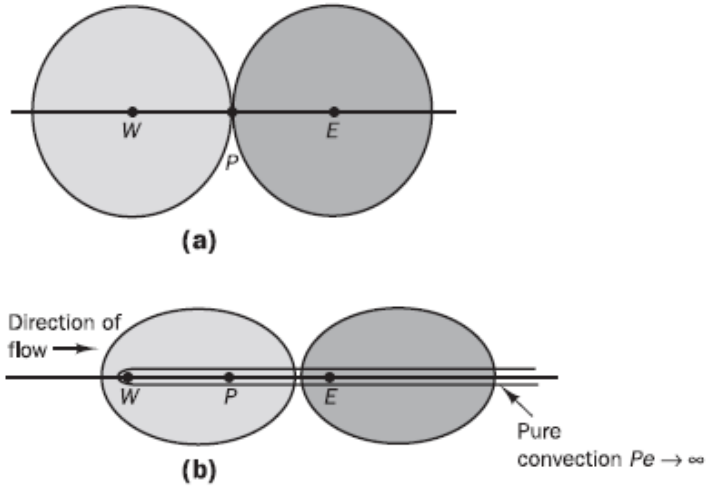


Figure 3.18: Distribution of ϕ from two sources for different Peclet numbers: a) pure diffusion, b) high and pure convection [14]

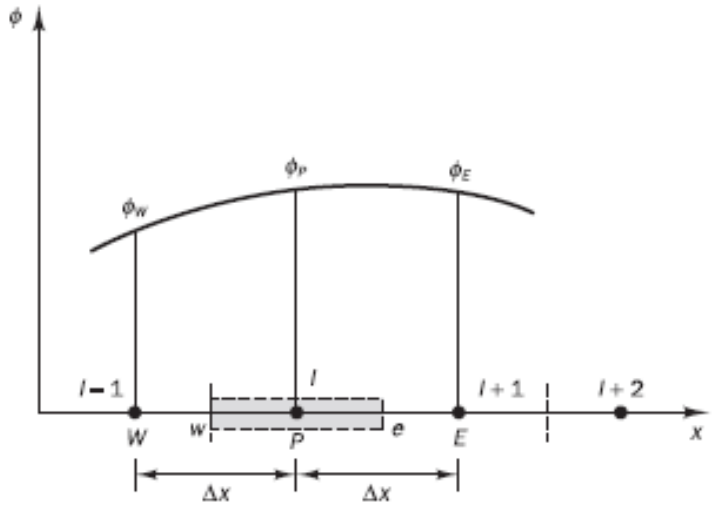


Figure 3.19: Uniform one dimension grid for truncation error of ϕ [14]

$$Pe = \frac{F}{D} = \frac{\rho u}{\Gamma/\delta x} \tag{3.97}$$

The two colored shapes in Fig. 3.18 represent the values of ϕ: the upper figure shows a pure diffusion case ($Pe \to 0$), for greater values of Pe the shapes become ellipsis and for the upper end of the spectrum ($Pe \to \infty$) they collapse into a thin line. This implies that, for the central node P, the effect of the westward node increases with the Peclet number whereas the eastward one diminishes, the node is thus only effected by the upstream ones. Transportiveness measures how well the relationship between adjacent nodes reflects the direction of the flow.

- **Accuracy**

  The accuracy of a scheme is dependent on the truncation error of its Taylor series. For example, considering an uniform spaced grid as the one in Fig. 3.19. The

development along the x direction of ϕ can be expressed as a Taylor series, both in forward and backward manner:

$$\begin{cases} \phi(x+\Delta x) = \phi(x) + \left(\frac{\partial \phi}{\partial x}\right)_x \Delta x + \left(\frac{\partial^2 \phi}{\partial x^2}\right)_x \frac{\Delta x^2}{2} + ... \\ \phi(x-\Delta x) = \phi(x) - \left(\frac{\partial \phi}{\partial x}\right)_x \Delta x + \left(\frac{\partial^2 \phi}{\partial x^2}\right)_x \frac{\Delta x^2}{2} - ... \end{cases} \tag{3.98}$$

for example, the value at node E is computed as follows:

$$\phi_E = \phi_P + \left(\frac{\partial \phi}{\partial x}\right)_P \Delta x + \left(\frac{\partial^2 \phi}{\partial x^2}\right)_P \frac{\Delta x^2}{2} + ... \tag{3.99}$$

by rearranging the terms and not considering the higher order ones, one reaches a *first order error equation in* Δx:

$$\left(\frac{\partial \phi}{\partial x}\right)_P = \frac{\phi_E - \phi_P}{\Delta x} + O(\Delta x) \tag{3.100}$$

and it is possible to also write one for the backward case. In order to achieve a higher order multiple points must be taken into account, by subtracting the two equations in (3.98) one reaches:

$$\phi(x+\Delta x) - \phi(x-\Delta x) = 2\left(\frac{\partial \phi}{\partial x}\right)_P \Delta_x + \left(\frac{\partial^3 \phi}{\partial x^3}\right)_P \frac{\Delta x^3}{3!} + ... \tag{3.101}$$

rearranging this last formula brings *second order error equation in* Δx:

$$\left(\frac{\partial \phi}{\partial x}\right)_P = \frac{\phi_E - \phi_W}{2\Delta x} + O(\Delta x^2) \tag{3.102}$$

Quadratic error means that it reduces more quickly across the grid, bringing better results. Higher order equations can be achieved as well, this implies more accurate results but at the same time a more complicated calculations.

Since multiple discretisation exist they can be compared with each other on how well they respect the properties that have been previously exposed, here are explained some of the most used schemes but the possible choices are much more numerous [14].

– **Central differencing scheme**

This scheme follows the example pictured in Fig. 3.17. Assuming an uniform grid it is possible to write the value of ϕ at the cell surfaces: $\phi_e = (\phi_P + \phi_E)/2$ and $\phi_w = (\phi_W + \phi_P)/2$. Starting from the first equation of (3.95), after some rearrangements the expression for both diffusion and convection is reached:

$$a_P\phi_P = a_W\phi_W + a_E\phi_E \tag{3.103}$$

where:

$$a_W = D_w + \frac{F_w}{2}, \quad a_E = D_e - \frac{F_e}{2}, \quad a_P = a_W + a_E + (F_e - F_w) \tag{3.104}$$

This scheme is intrinsically conservative but it has been demonstrated that at high Peclet numbers it loses the properties of transportiveness and boundedness. Thus, this scheme is suitable for problems where diffusion dominates but becomes less reliable as convection increases. Considering equation (3.103) it is apparent the face values are always equally influenced by the westward and eastward nodes, this hinders the ability of the scheme to recognize the flow direction. Central differencing possesses a truncation error of the second order, the same one expressed in the previous paragraph.

– **Upwind differencing scheme (or donor cell)**

This scheme has been created to overcome the inability of the central differences to recognize the directionality of the flow. In Fig. 3.20 is shown how now neighboring nodes share the ϕ value with the central element faces in two flow directions. Assuming a positive flow (west to east) the faces assume $\phi_w = \phi_W$ and $\phi_e = \phi_P$, whereas for negative direction (east to west) they become $\phi_w = \phi_P$ and $\phi_e = \phi_E$. The scheme can then be written in the same form of equation (3.103) but with different coefficients, for the central one there is independence from the flow directionality whereas the other two are not

$$a_P = a_W + a_E + (F_e - F_w) \tag{3.105}$$

$$\begin{cases} a_W = D_w + F_w, \quad a_E = D_e & \leftarrow \quad \text{Positive flow} \\ a_W = D_w, \quad a_E = D_e - F_e & \leftarrow \quad \text{Negative flow} \end{cases} \tag{3.106}$$

This representation is consistent across all nodes for the calculation of ϕ, therefore it assures conservativeness; boundedness and transportiveness are achieved as well. However, when the flow is not aligned with the lines of the grid, a problem of *false diffusion* of ϕ between adjacent cells may arise. Furthermore its Taylor series truncation error is only of the first order.

– **Hybrid differencing scheme**

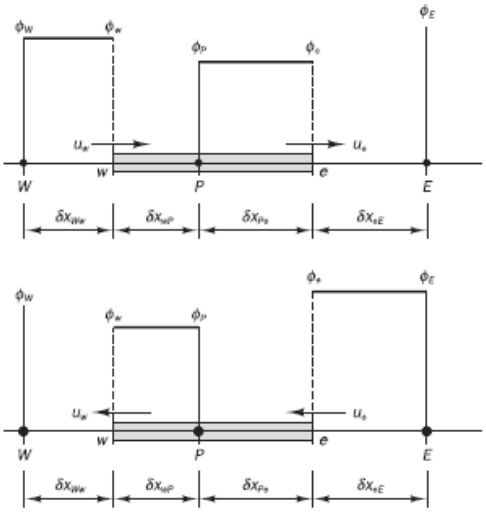


Figure 3.20: Upwind differencing scheme nodal values; positive flow on the top, negative flow on the bottom [14]

This is a combination of the previous two schemes. The Peclet number is the decisive factor for the behaviour: if $Pe < 2$ the central scheme is employed, whereas for $Pe > 2$ the upwind one is preferred. In order to achieve this solution the Peclet number is computed at the front element, for the west face:

$$Pe_w = \frac{F_w}{D_w} = \frac{\rho_w u_w}{\Gamma/\delta x_{WP}} \tag{3.107}$$

The value of the flux of ϕ is computed according to the value of (3.107):

$$\begin{cases} q_w = F_w \phi_P & Pe_w \leq -2 \\ q_w = F_w \left[\frac{1}{2} \left(1 + \frac{2}{Pe_w} \right) \phi_W + \frac{1}{2} \left(1 - \frac{2}{Pe_w} \right) \phi_P \right] & -2 < Pe_w < 2 \\ q_w = F_w \phi_W & Pe_w \geq 2 \end{cases} \tag{3.108}$$

The discretised equation is the same as the other two and the coefficients vary accordingly:

$$\begin{cases} a_P = a_W + a_E + (F_e - F_w) \\ a_W = max \left[F_w, \left(D_w + \frac{F_w}{2} \right), 0 \right] \\ a_E = max \left[-F_e, \left(D_e - \frac{F_e}{2} \right), 0 \right] \end{cases} \tag{3.109}$$

This schemes satisfies all the properties previously discussed, it has been proved to be highly stable and it is able to successfully switch between the central and upwind schemes exploiting their favorable characteristic in both diffusion and convection problems. The drawback of this method is its lower accuracy compared to other

schemes, due to the first order truncation error.

– **Quadratic upwind differencing scheme (QUICK)**

In order to limit the error diffusion of the previous scheme it has proven necessary to discretise the grid with a higher order method, ensuring that a greater number of neighbouring nodes dampens the negative effects. The QUICK (Quadratic Upwind Interpolation for Convection Kinetics) scheme emploies a quadratic fit of three nodes to obtain the ϕ face value as shown in Fig. 3.21, in particular two nodes are the surrounding ones and the other is the second one found in the upstream direction. For example, if the flow is positive, to compute ϕ_w nodes WW, W and E are used, whereas W, P and E are utilized for ϕ_e. The following equations are for positive and negative flow respectively.

$$\begin{cases} \phi_w = \frac{6}{8}\phi_W + \frac{3}{8}\phi_P - \frac{1}{8}\phi_{WW} \\ \phi_e = \frac{6}{8}\phi_P + \frac{3}{8}\phi_E - \frac{1}{8}\phi_W \end{cases} \tag{3.110}$$

$$\begin{cases} \phi_w = \frac{6}{8}\phi_P + \frac{3}{8}\phi_W - \frac{1}{8}\phi_E \\ \phi_e = \frac{6}{8}\phi_E + \frac{3}{8}\phi_P - \frac{1}{8}\phi_{EE} \end{cases} \tag{3.111}$$

Considering both the effects of positive and negative flow the discretised equation is as follows:

$$a_P\phi_P = a_W\phi_W + a_E\phi_E + a_{WW}\phi_{WW} + a_{EE}\phi_{EE} \tag{3.112}$$

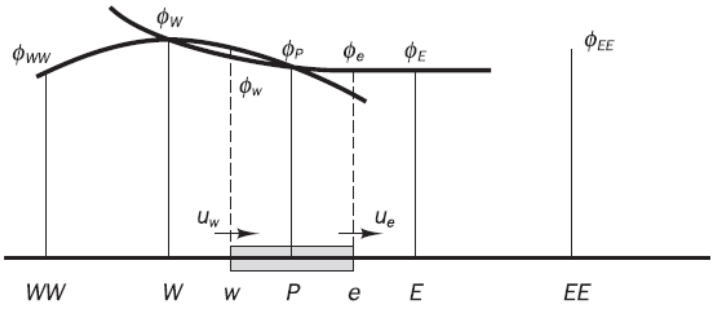


Figure 3.21: QUICK scheme notation, three nodes are used to compute ϕ at the faces of a cell [14]

$$\begin{cases} a_W = D_w + \frac{6}{8}\alpha_w F_w + \frac{1}{8}\alpha_e F_e + \frac{3}{8}(1-\alpha_w)F_w \\ a_{WW} = -\frac{1}{8}\alpha_w F_w \\ a_E = D_e - \frac{6}{8}\alpha_e F_e - \frac{1}{8}(1-\alpha_e)F_e - \frac{3}{8}(1-\alpha_w)F_w \\ a_{EE} = \frac{1}{8}(1-\alpha_e)F_e \end{cases} \tag{3.113}$$

and the direction of the flow is taken into account by the α coefficients:

$$\alpha_w = \alpha_e = 1 \quad \text{for positive flow}, \quad \alpha_w = \alpha_e = 0 \quad \text{for negative flow} \tag{3.114}$$

This scheme is conservative, respects transportiveness and the truncation error is of the third order, therefore it is the most accurate of those examined. However, under certain flow conditions, stability problems may arise; further developments of this scheme are reported in [14].

3.5.2 Solution algorithms for steady flows

In a generic flow it is usually necessary to compute not only the value of each scalar variable ϕ but the velocity field as well, which is a vector quantity. Assuming a steady flow in a two dimensional domain the governing equations (x, y momentum and continuity) are as follows:

$$\begin{cases} \frac{\partial}{\partial x}(\rho u u) + \frac{\partial}{\partial y}(\rho v u) = \frac{\partial}{\partial x}\left(\mu\frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial y}\left(\mu\frac{\partial u}{\partial y}\right) - \frac{\partial P}{\partial x} + S_u \\ \frac{\partial}{\partial x}(\rho u v) + \frac{\partial}{\partial y}(\rho v v) = \frac{\partial}{\partial x}\left(\mu\frac{\partial v}{\partial x}\right) + \frac{\partial}{\partial y}\left(\mu\frac{\partial v}{\partial y}\right) - \frac{\partial P}{\partial y} + S_v \\ \frac{\partial}{\partial x}(\rho u) + \frac{\partial}{\partial y}(\rho v) = 0 \end{cases} \tag{3.115}$$

where it is possible to discern velocity, stress, pressure and internal sources terms. This new equation system brings new problem concerning the solution of the entire flow, mainly the presence of non linear terms and the absence of a dedicated equation for the pressure, which nonetheless appears in both momentum equations. Usually the pressure gradient is not known from the start and, depending if the flow is compressible or not, a procedure must be followed to compute this variable: in the first case it is possible to leverage the energy conservation and the equation of state, in the latter there is no correlation between density and pressure and therefore the link with velocity becomes a new constraint in the flow. All this problems are solvable by adopting an iterative approach to the solution, which implies the start from a "guessed" value of the properties and the repetition of a series of steps until convergence is reached. In this section are

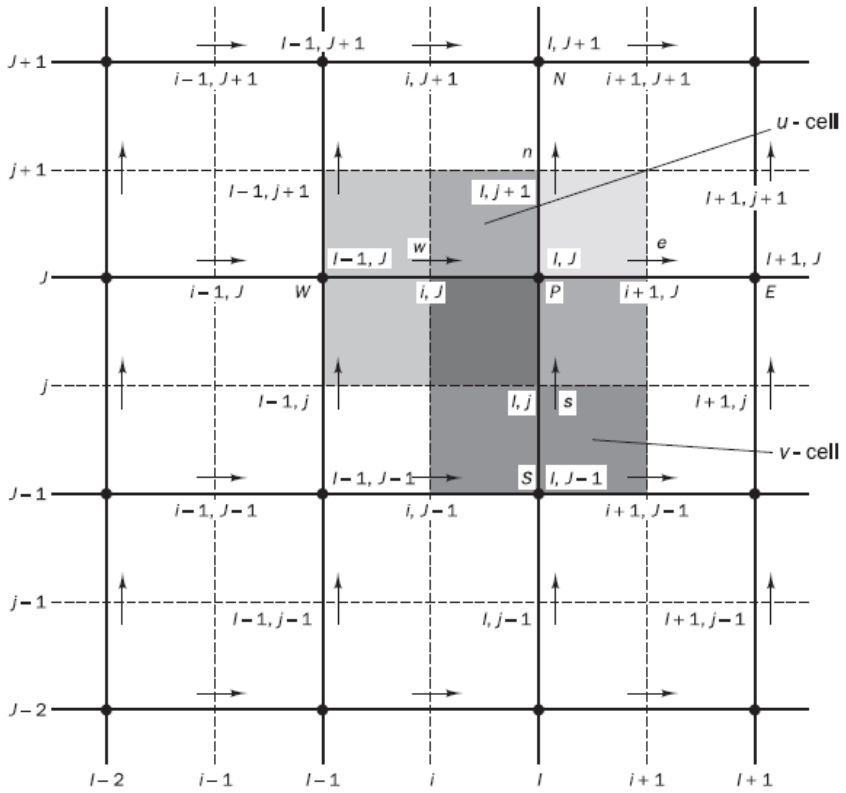


Figure 3.22: Staggered grid notation for scalar and velocity values. Continuous grid and capital letters for scalar values, segmented grid and lowercase letters for velocity [14]

presented the two main solution algorithms, SIMPLE and PISO, for a complete discussion of this aspects and the enhanced algorithms that have been developed chapter 6 of [14] is suggested.

It has been observed [14] that using the same grid for both scalar and velocity values results in a non physical representation of pressure in the flow, thus a *staggered grid* approach has been developed (Fig. 3.22): scalar quantities are computed at the nodes of the continuous grid whereas velocity values are defined at the in-between faces marked by the arrows, which identifies the dashed line grid. It is also possible to distinguish between two different control volumes for velocity, one for the u and the other for the v components. For this new grid it is possible, after some rearrangements, to express the discretised momentum equations:

$$\begin{cases} a_{i,J} u_{i,J} = \sum a_{nb} u_{nb} + (p_{I-1,J} - p_{I,J})\, A_{i,J} + s_{i,J} \\ a_{I,j} v_{I,j} = \sum a_{nb} v_{nb} + (p_{I,J-1} - p_{I,J})\, A_{I,j} + s_{I,j} \end{cases} \tag{3.116}$$

Applying the above formulas allows to calculate the u and v velocity fields for each cell of the staggered grid, the following algorithms are used for the pressure field.

- **SIMPLE algorithm**

  The Semi-Implicit Method for Pressure-Linked Equations was originally developed

by Patankar and Spalding [14]. The first step is the definition guessed fields for pressure and velocity: u^*, v^* and p^*; equations (3.116) are solved using them. Subsequently, corrections terms have to be defined:

$$\begin{cases} p = p^* + p' \\ u = u^* + u' \\ v = v^* + v' \end{cases} \tag{3.117}$$

subtracting the real and guessed momentum equations allows to write the corrected ones, where it is possible to consider null the effect of the sum of the neighbouring nodes:

$$\begin{cases} a_{i,J} u'_{i,J} = \sum a_{nb} u'_{nb} + \left(p'_{I-1,J} - p'_{I,J}\right) A_{i,J} \approx \left(p'_{I-1,J} - p'_{I,J}\right) A_{i,J} \\ a_{I,j} v'_{I,j} = \sum a_{nb} v'_{nb} + \left(p'_{I,J-1} - p'_{I,J}\right) A_{I,j} \approx \left(p'_{I,J-1} - p'_{I,J}\right) A_{i,J} \end{cases} \tag{3.118}$$

that results in:

$$\begin{cases} u_{i,J} = u^*_{i,J} + \frac{A_{i,J}}{a_{i,J}} \left(p'_{I-1,J} - p'_{I,J}\right) \\ v_{I,j} = v^*_{I,j} + \frac{A_{I,j}}{a_{I,j}} \left(p'_{I,J-1} - p'_{I,J}\right) \end{cases} \tag{3.119}$$

by following the same procedure for the values of $u_{i+1,J}$ and$v_{I,j+1}$ and considering the discretised form of the continuity equation:

$$\left[(\rho u A)_{i+1,J} - (\rho u A)_{i,J}\right] + \left[(\rho u A)_{I,j+1} - (\rho u A)_{I,j}\right] = 0 \tag{3.120}$$

it is possible to reach:

$$a_{I,J} p'_{I,J} = a_{I+1,J} p'_{I+1,J} + a_{I-1,J} p'_{I-1,J} + a_{I,J+1} p'_{I,J+1} + a_{I,J-1} p'_{I,J-1} + s'_{I,J} \tag{3.121}$$

where the values of the coefficients are obtained by previous rearranging, s' represent the imbalance due to the correction terms. Having computed p' it is possible to obtain the values of u, v and p from equations (3.119) and the first of (3.117).

- **PISO algorithm**

  The acronym stands for Pressure Implicit with Splitting of Operators and it has been originally proposed for unsteady flows by Issa [14], but it has been adapted for

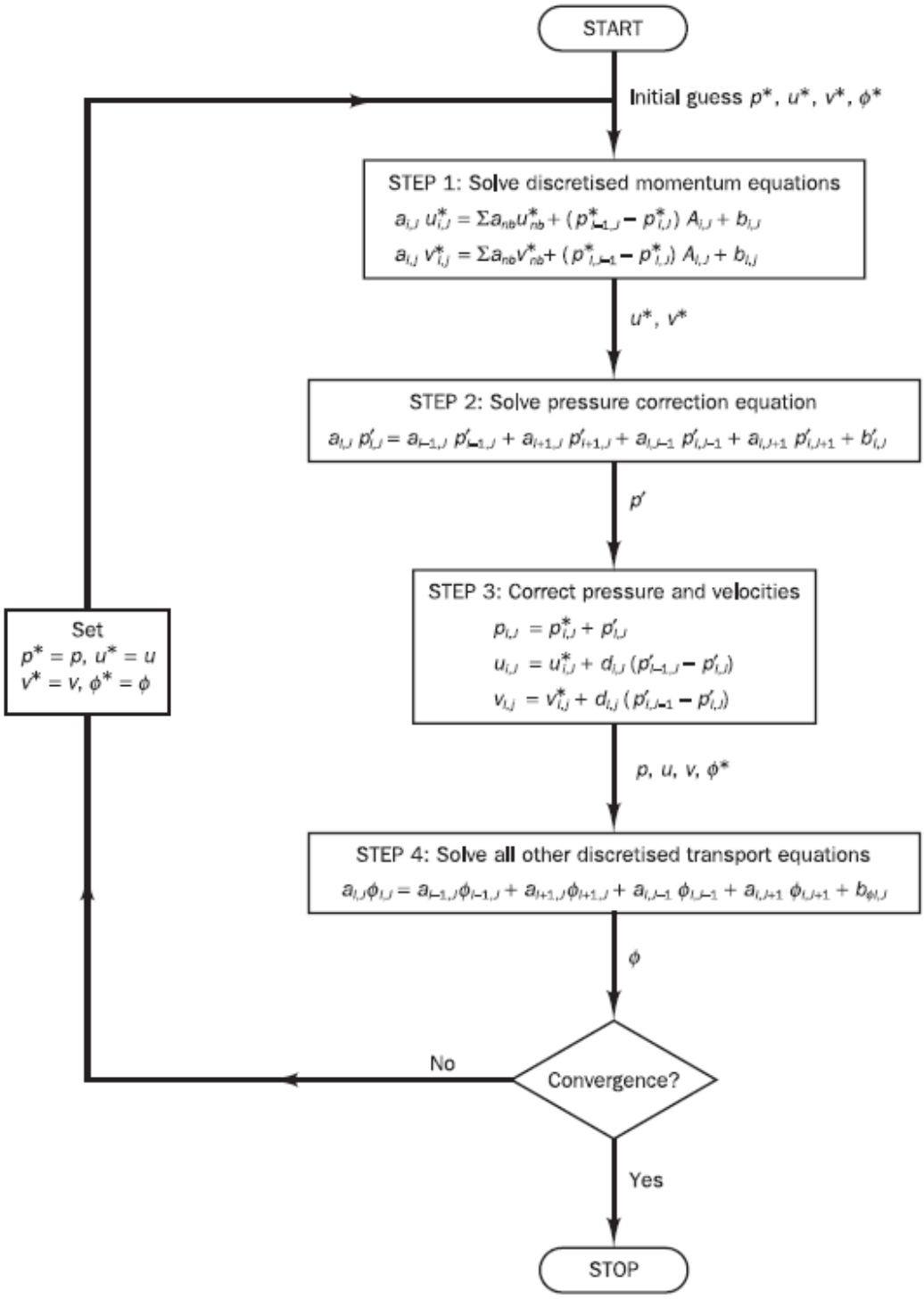


Figure 3.23: SIMPLE algorithm [14]

steady-state problems. It differentiates itself from SIMPLE with the inclusion of a corrector step to the algorithm, considering the already added steps of the enhanced versions of the latter. At the start the procedure involves a *corrector step*: the same workflow of the SIMPLE algorithm. The two successive sections are *corrector steps* to enhance twice the values of pressure and velocity, at first one employs the notation in (3.122) to obtain the corrected momentum equations:

$$\begin{cases} p^{**} = p^{*} + p' \\ u^{**} = u^{*} + u' \\ v^{**} = v^{*} + v' \end{cases} \tag{3.122}$$

$$\begin{cases} u^{**}_{i,J} = u^{*}_{i,J} + \frac{A_{i,J}}{a_{i,J}} \left(p'_{I-1,J} - p'_{I,J} \right) \\ v^{**}_{I,j} = v^{*}_{I,j} + \frac{A_{I,j}}{a_{I,j}} \left(p'_{I,J-1} - p'_{I,J} \right) \end{cases} \tag{3.123}$$

Then, by solving equation (3.121), it is possible to calculate u^{**} and v^{**} and the second correction can then be applied by considering both the discretised momentum equations for u^{**}, v^{**} and the further corrected velocities u^{***}, v^{***}, written in as similar manner to those in SIMPLE. After that it is possible to derive the second correction to the pressure and it's equation:

$$p^{***} = p^{**} + p'' = p^{*} + p' + p'' \tag{3.124}$$

$$a_{I,J} p''_{I,J} = a_{I+1,J} p''_{I+1,J} + a_{I-1,J} p''_{I-1,J} + a_{I,J+1} p''_{I,J+1} + a_{I,J-1} p''_{I,J-1} + s''_{I,J} \tag{3.125}$$

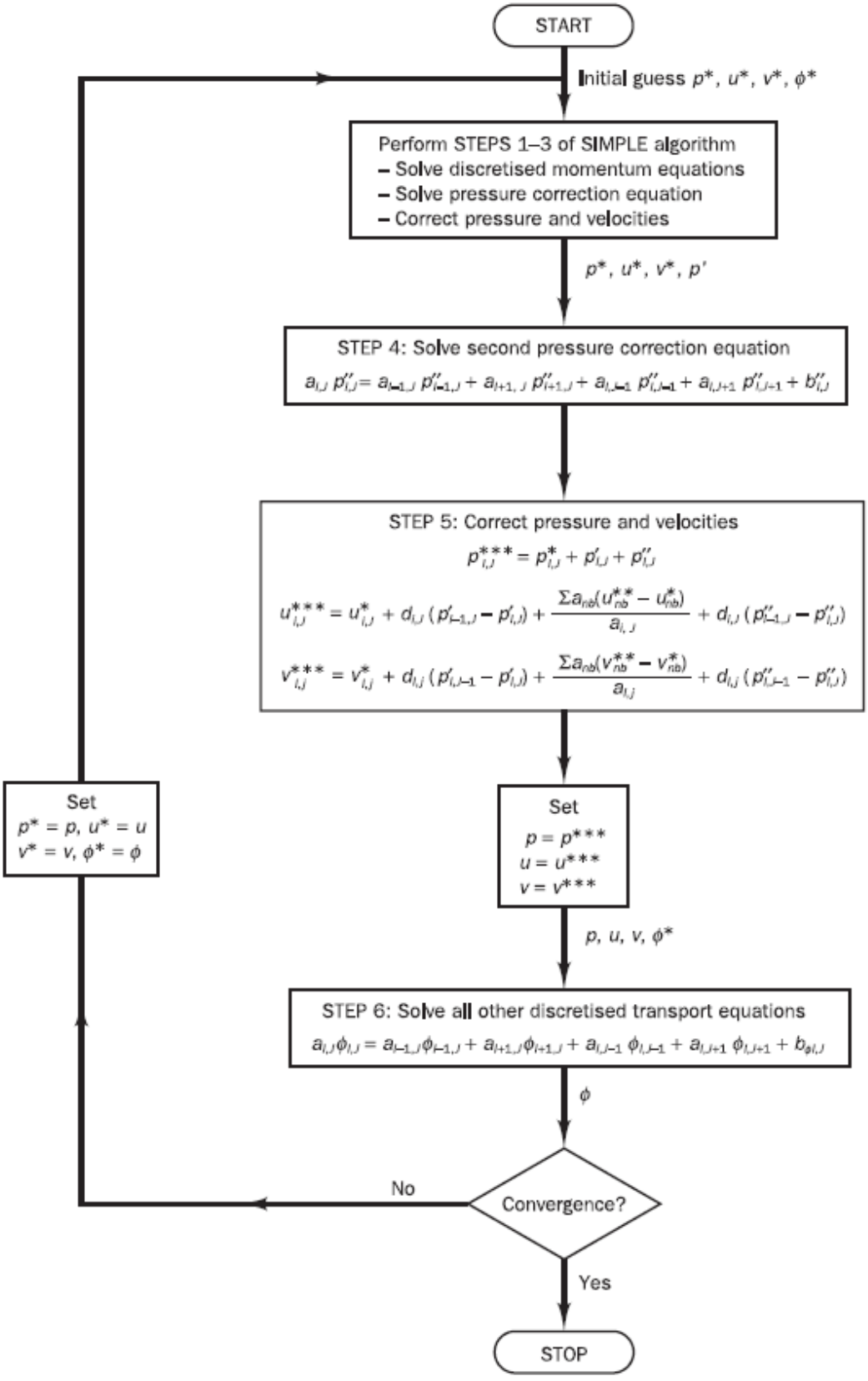


Figure 3.24: PISO algorithm [14]

Chapter 4

Automation workflow and models setup

The work presented in this thesis is consequential to the modern industry interest in optimization and automation, with particular attention to aspects of shape optimization. The need to obtain novel design methods for exhaust ports has sparked the necessity to develop an interface between morphing technologies and CFD software; a further FEM investigation of the resulting effects on an engine head has been added as well. It is important to consider that, even tough the examples presented in this work belong to the automotive field, the workflow can be easily adapted to other applications, provided that the corresponding software and models for fluid-structure interaction are prepared in a correct way.

In this chapter an explanation of the developed workflow and preparation of all the employed application software is given. A simple introduction to the mesh morphing of a elbow bend in a pipe is given in order to familiarize the reader with the key aspects of the work; two cases are then presented: one is the CFD analysis of a flow bench for an exhaust port used in motor sport; the other is a steady-state CFD analysis of an exhaust flow through the CAD model of an engine head (on which a preliminary study has been conducted by Matteo Marra in [23]), since a solid model is available a Conjugate Heat Transfer (CHT) study has been performed as well, the results of which have been used in a thermo-structural analysis.

4.1 Employed software

In order to perform the optimization several CAE applications have been employed, this section briefly describes them.

- **RBF-viewer**

  This tool allows to mesh morph complex geometries, applying the principles explored

in the first chapter. It is a proprietary software of *RBF morph* and represent an effort to expand the scope of possible applications of morphing, in fact its applicability is not restricted to a single type of file or program but it allows the use various input and output files, such as STL, STP and DAT, making it extremely flexible. Moreover, the software itself is a graphic user interface (GUI) to a JSON (Java Script Object Notation) file; so it becomes extremely easy to modify internal parameters or to use APIs (Application Program Interfaces) in an automated manner, a desirable characteristic for this type of work. As it will be discussed forward the employment of this program has regarded the use of DAT and STEP files to setup the morphing of the models and, subsequently, the automation of the shape optimization operation via a Python code, which acted directly upon the nodes of the JSON file. The RBF transformations currently available are: translation, scaling and rotation.

- **CONVERGE CFD** [20]

  Among the multiple software developed for the simulation of fluid dynamics phenomena CONVERGE CFD, from *Convergent Science*, is one the most recent and innovative. Its strength relies in the ability to simulate with extreme accuracy a wide range of applications related to internal flows, especially in the field of Internal Combustion Engines (ICEs) and fluid machinery. The graphic user interface of the software is called "Converge Studio". It posses multiple tools that allow for the modeling of turbulence, combustion, volume of fluid and many more physics dependent characteristics. Furthermore, the internal meshing tool permit easy grid scaling and adaptive mesh refinements of the stock cartesian grid, but inlaid and extruded meshes are available as well. Both steady-state and transient simulations may be performed, even though this work is concerned only with the former. To complete the picture, a robust optimization tool is present, together with an integrated copy of the post processing software Tecplot and a useful line-plotting interface. STL files of the CAD models geometries are converted into DAT files, which allows for mesh morphing in RBF-viewer without altering their boundary conditions, this allows for multiple variations of a geometry to be simulated without the need prepare new input files.

- **Ansys Mechanical** [24]

  One of the most widely used FEM software, Mechanical belongs to the Workbench ecosystem, which allows for easy integration between most applications under the *Ansys* family. With the Mechanical module it is possible to perform structural, thermal, modal and acoustics analysis for both linear and non linear problems, leveraging the extensive internal library of materials. Another powerful feature is the possibility to implement external plugin applications through the ACT (Application

Customization Toolkit) interface, so an internal version of RBF software is available to perform morphing on solid meshes. Map of temperature and pressure have been extracted from the Converge simulations to act as loads in structural analysis on an engine head and, through the use of the RBF add-on, it has been possible to investigate the behaviour of the different geometry variations.

- **Ansys SpaceClaim** [24]

  It is one of the CAD software available within the *Ansys* package. Its main purpose is the pre-processing for FEM analysis of already existing CAD models, but it is powerful enough to be used as proper design program, albeit for simpler geometries. The main use throughout this work has been the preparation of the auxiliary geometries for morphing, as well as the preparation of the engine head.

- **Python**

  One of the most widely used programming languages in the world, it has been extensively used in these thesis to take care of the automation and communication between different software. Libraries such as *pandas, json, shutil* and *psutil* have proved particularly useful.

4.2 Workflow

The work progression is rather complicated and needs careful examination in order to grasp all the important details, a general diagram is provided in Fig. 4.1. In this section the procedure to execute the entire work is explored, the preparation and details inside the single programs are defined in the following ones. The entire Python code is provided in Appendix A, whereas functions employed to perform specific tasks, such as creation of the data maps, are viewable in Appendix B. It is possible to divide the workflow in steps:

1. **Original geometry preparation**

   For a given geometry it is necessary to obtain a sufficiently accurate STL file to load in Converge, a STEP file is also necessary to define the morphing sources inside RBF-viewer. For the latter case it is possible to use the entire geometry or create a set of virtual entities limited to important zones, this last approach has proven to be more reliable during the work. Finally the surface DAT file must be prepared inside Converge, by assigning the triangles of the STL to different boundaries depending on the BC (Boundary Conditions) that have to be applied.

2. **RBF model setup**

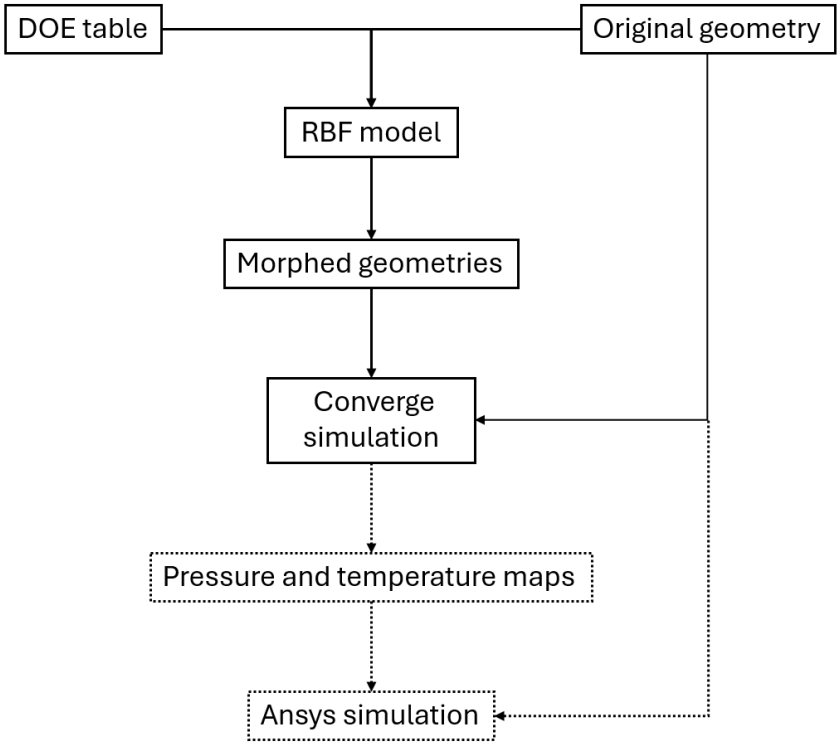


Figure 4.1: Workflow diagram for morphing and simulation. The solid lines refer to the CFD case, the dashed lines to the CSM one

The Converge surface DAT file and auxiliary STEP files have to be loaded inside RBF-viewer. The morphing is performed upon the surface file triangles by defining source points on the auxiliary geometries ($\mathbf{x_{si}}$ of equation (1.1)) through RBF Sources. By naming this later ones accordingly it is possible the access them via Python code in the JSON file and perform the transformation without interacting with the graphic user interface. For this later aspect one can refer to Fig. 4.2, where a generic RBF Source is defined. There is a series of *keys* on the left side of the colons and the corresponding *value* on the right; for this particular example it is possible to distinguish a translation transformation, with a value of -4 and 6 along the x and y axis respectively; the nodes upon which the transformation is prescribed are selected by their ID number (which is defined at the beginning of the JSON) and listed under the *"target nodes"*.

3. **Design Of Experiment table creation and morphing**

   In order to create a list of geometry variants, with different values for RBF Sources, it is necessary to use a format that allows for easy separation and access of numerical values. This thesis has employed Excel for its straightforwardness, but other software or formats may be used. After having created the table of variants a *for* cycle in the Python code goes through both the data frame and the JSON file, in order to create the desired number of morphed geometries in a DAT file format.

```
{
  "acting_on_deformed": false,
  "coord_filtering": false,
  "coordinate_system_serialized_id": 0,
  "function_degree": 3,
  "geometry_entities_ids": [
    556
  ],
  "id": 724,
  "is_region": false,
  "name": "RBF Source 4",
  "node_overlap_mode": 1,
  "scoping_method": 0,
  "serialized_id": 28,
  "target_nodes": [
    3698,
    3699,
    3749
  ],
  "transformation_definition_type": 0,
  "transformation_id": 0,
  "transformation_name": "Translation",
  "x": -4.0,
  "y": 6.0,
  "z": 0.0
}
```


Figure 4.2: Example of an RBF Source inside the JSON file

4. **Converge setup and simulations**

   The setup for the CFD simulations is the same for the original and morphed geometries, except for the DAT files. Thus, it is possible to create a single directory of input files that will be used for all the jobs, the Python code takes care of creating the correct directory, copying the input files into it, and changing the surface file with the one of corresponding morphed variant. In order to run a simulation the correct values for materials (whether gas, liquid or solid) and the corresponding species must be specified; subsequently the solver parameters, initial and boundary conditions and values for the computational grid have to be set, as well as turbulence modeling and/or other physics. The output files may be chosen as needed, for example, if there is the need to use the temperature of the solid body as a load in a further structural analysis, it is possible to create a map of coordinates of the central node of each cell with the corresponding temperature value, the Python code will then set it up in the correct format to be used inside Ansys Mechanical. After having completed the preparation of the simulations it is possible to export the settings into the aforementioned input files directory.

5. **Ansys mechanical setup, morphing and simulations**

   Having completed the CFD portion of the work, if there is the need to investigate the effects heat transferred to a body it is possible to continue with a structural

analysis. As a preliminary step it is necessary to create the solid temperature map, as well as one for the fluid pressure acting on the walls; moreover, since the morphing inside Ansys is performed through the internal extension, it is necessary to create a special file for each new geometry, where data about the original nodes coordinates and morphing displacements are stored. The Python code takes care of all the above. After that a Workbench project may be created containing a module to read external data (temperature and pressure maps) and one to perform a static structural simulation in Mechanical. Inside the latter it is necessary to define the geometry, create the mesh, prepare the morphing procedure for all the variants and apply the loads and constraints. After having performed the calculations it is possible to analyze contours and other diagrams in the post-processing results section.

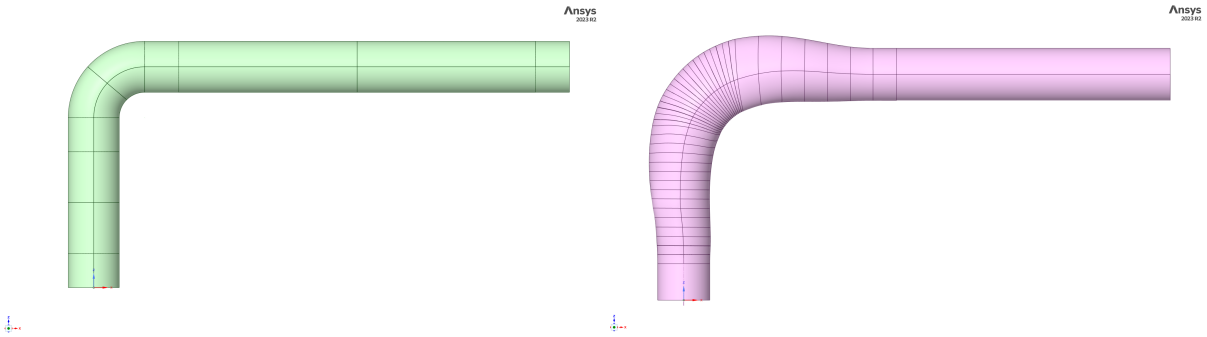
4.3 Introduction to mesh morphing: elbow bend in a pipe

A simple problem is hereby illustrated: the reduction of concentrated pressure loss from a 90 degree curve in a pipe; aspects concerning this application have been treated exhaustively by Andrea Lopez in [25], especially for what regards the implementation of CFD optimization through the adjoint method. The geometry is rather simple, a circular pipe with a diameter of 60 mm and an intrados curvature radius at the bend of 30 mm. An attempt was made to mimic the geometries obtained in the aforementioned study: this has particularly concerned the curve area in order to facilitate the change of direction of the flow. Fig. 4.3 shows the both the original geometry and the adjoint optimized one.

From an STL file of the original geometry it has been possible to obtain a surface DAT file where the triangles for the Inflow, Outflow and Wall are separated from one another using boundary fences. In order to perform the morphing inside RBF-viewer it has been necessary to create CAD representation of the geometry, which will then have to be meshed; it is possible to follow two approaches:

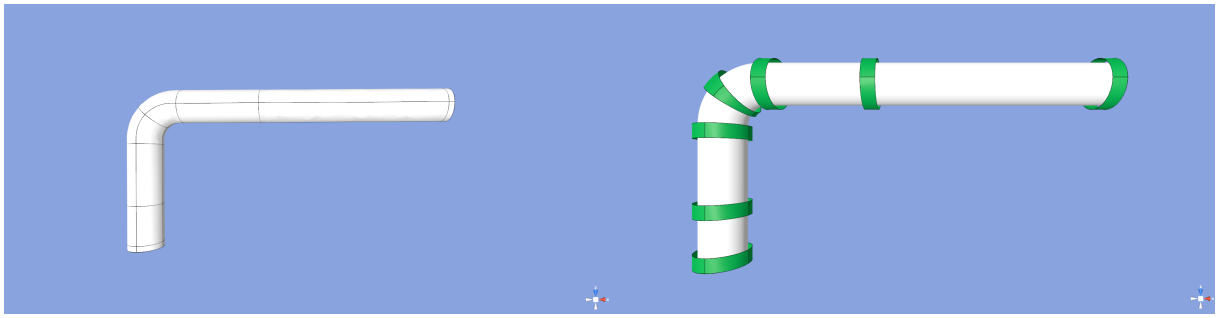
- A **CAD based approach** in which a twin CAD model of the entire geometry is created. It is important to utilize functions for surface separation in the proximity of the zones of interest, in order to permit the assignment of local RBF Sources. This strategy involves the creation of a complete STEP file of the original geometry.

- A **mesh based approach** that involves the creation of set of virtual geometries in the specific areas where the sources for the morphing are necessary, this includes both the zones that have to move and that must remain fixed. In this case the STEP file is comprised a series of smaller localized surfaces

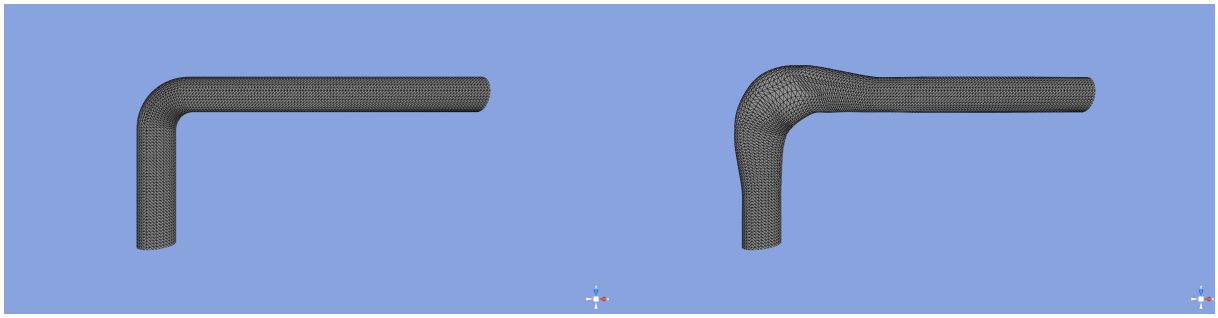


(a) Original geometry (b) Adjoint optimized geometry

Figure 4.3: Pipe geometries inside Ansys SpaceClaim.



(a) CAD based approach (b) Mesh based approach



(c) Original DAT file for the RBF Region (d) One of the morphed DAT files

Figure 4.4: Comparison of the two strategies inside RBF-viewer, original and morphed DAT files

Lift	10 mm
Valve diameter	28 mm
Curtain area	800 mm^2
Minimum flow area	484 mm^2
Cylinder bore	80 mm

Table 4.1: Exhaust port geometry for a single valve

Both of the two strategies are viable, but their efficacy may vary depending on the application. As an example, if the original geometry is rather complicated, the use of the entire CAD may prove more computationally costly compared to the creation of specific virtual geometries, whereas in some situation the former may give more precise results. This section is meant to give an overview on how to implement both the aforementioned strategies on a simple case, in the rest of this work the mesh based approach has been preferred.

The resulting RBF setup for the two methods is presented in Fig. 4.4: in the CAD based approach it is possible to see how the surface of the pipe has been divided with circular segments upon which the RBF Sources will be applied; the green smaller disks represent the virtual geometries of the mesh based strategy and are the sources for the morphing displacements. The DAT geometry upon which the RBF Region is applied and that will receive the effects of the sources is the same for both approaches.

4.4 Exhaust port setup description

The original geometries for the motor sport application consisted of an example DAT and a STEP files. The objective was the development of a connection between the RBF mesh morphing software and Converge and to attempt to increase the performance by simulating a flow bench with a pressure differential of 100 mbar and exhaust valves lift of 10 mm. Tab. 4.1 reports the main geometric parameters of the model. The CAD model of the geometry, a four valve cylinder, is shown in Fig. 4.5, to take advantage of the symmetry feature of CFD simulations only half of the volume has been created, the intake valves and ports are omitted and the large volume (plenum) after the exhaust duct absolves the function of limiting the effect of the outlet boundary conditions. The CAD model represent the "negative" of the real geometry: the computational domain includes the fluid regions, the external surfaces are the interfaces with the solid bodies.

In order to prepare the RBF-viewer setup a DAT surface file is needed, after importing the STL of the model (obtainable from most CAD programs) in Converge it is possible to perform a wide range of surface manipulation techniques, one of the most important is the separation of groups of triangles into distinct boundaries through the use of virtual fences; this allows to apply different BC. Fig. 4.6 shows the resulting boundaries applied

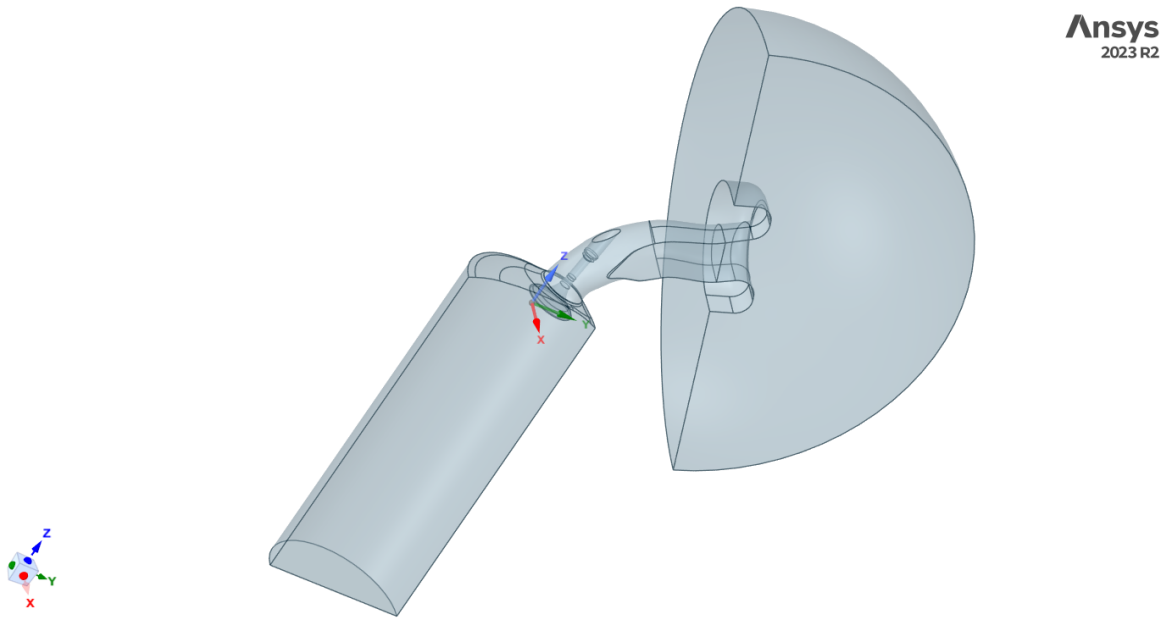


Figure 4.5: CAD model of the exhaust inside Ansys SpaceClaim

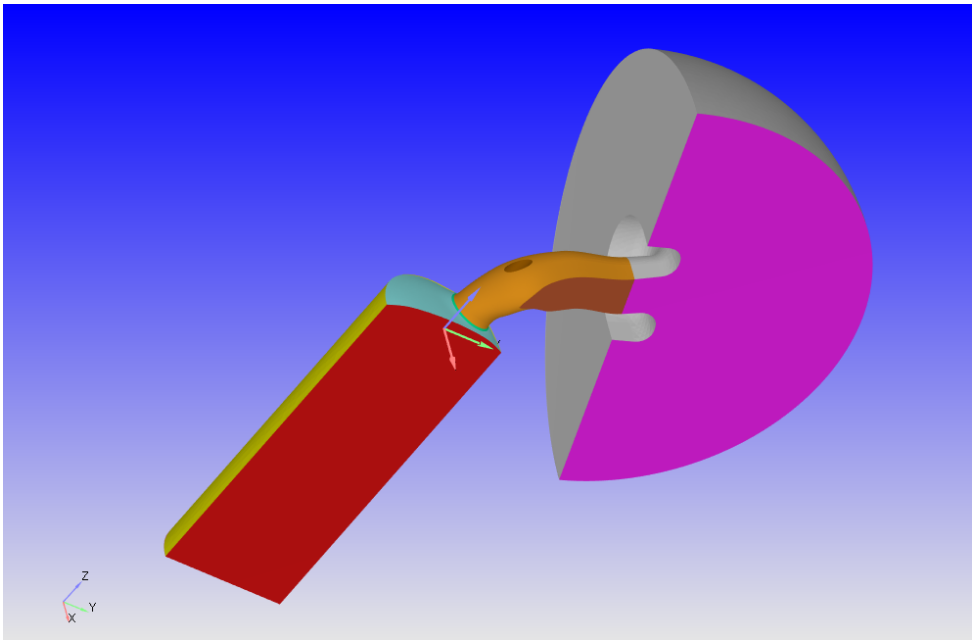


Figure 4.6: Boundaries subdivision of the exhaust model inside Converge Studio

on the STL file, they will be discussed further in the Converge subsection. Regarding STL files it is important to specify that the triangle arrangement has to be good enough for the representation of the fluid domain but also for maintaining a good mesh quality after the morphing has been performed. This aspect can be viewed in Fig. 4.7, which shows the difference between two type of surface meshes: the upper one is not suitable for morphing applications whereas the other is acceptable. For the rest of this work only those of the second kind have been taken in consideration.

4.4.1 RBF model setup

A set of virtual geometries had to be created inside SpaceClaim, shown in Fig. 4.8, to aid the definition of the RBF Sources, their location reflected the portions of the geometry upon which the morphing action was desired. They have mainly the shape of thin disks or somewhat circular laminae and are located around the exhaust duct, valve stem and seat. Their small form factor allows for an easier manipulation and lower computational cost, considering that once they are meshed they will posses a lower number of nodes, and subsequently a manageable number of RBF Source points. Not all have them have been used, but they have played different roles for the various DOEs that have been performed, as it will be explained in the next chapter. The RBF model for the exhaust is presented in Fig, 4.9.

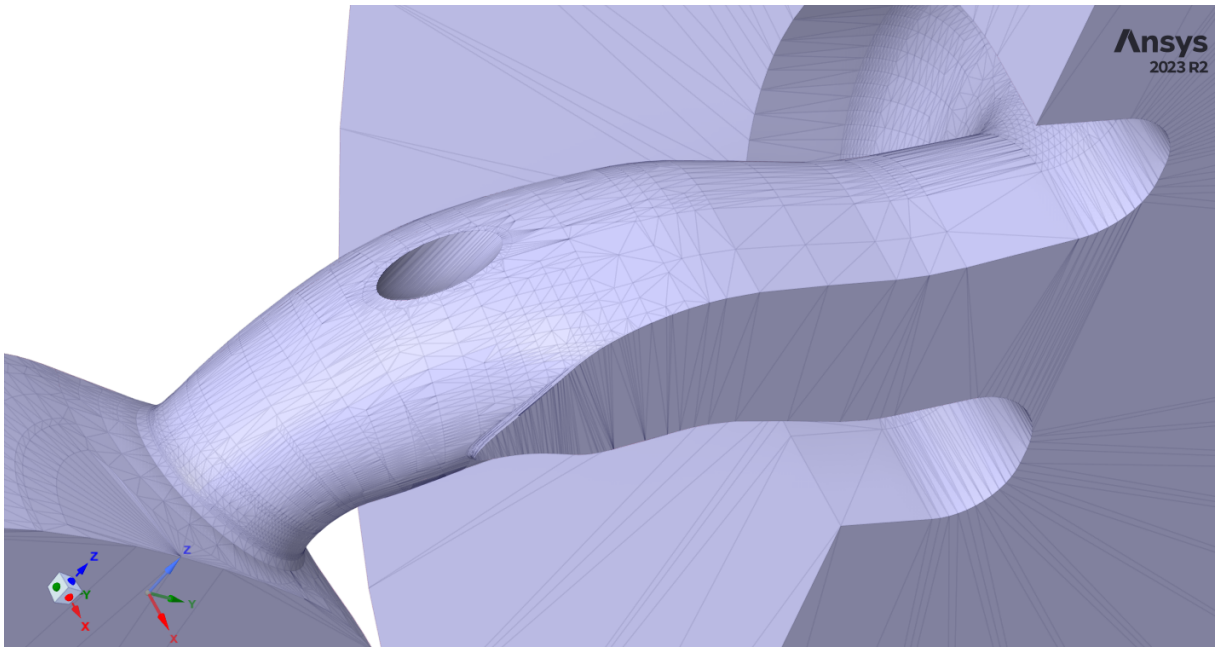
Two models have to be created inside RBF-viewer and their order is important to allow interaction with the Python code: the STEP files must be loaded in the first whereas the DAT files goes in the second, this permits the export of the correct file once the morphing is concluded. Attention must be paid to the measurements unit of different programs: since the imported step files are written in millimeters and Converge uses meters a scale factor of 1000 must be used for the DAT to maintain the correct proportions.

Furthermore a mesh must be created for the virtual geometries, it is possible to follow two approaches: setting a minimum and maximum value for the length of the STL triangles inside the program itself and let the internal mesher take care of the work or attaching an external mesh belonging to a twin STL model. In both cases the dimension of the elements is compromise between accuracy of representation and computational time of the RBF interpolation.

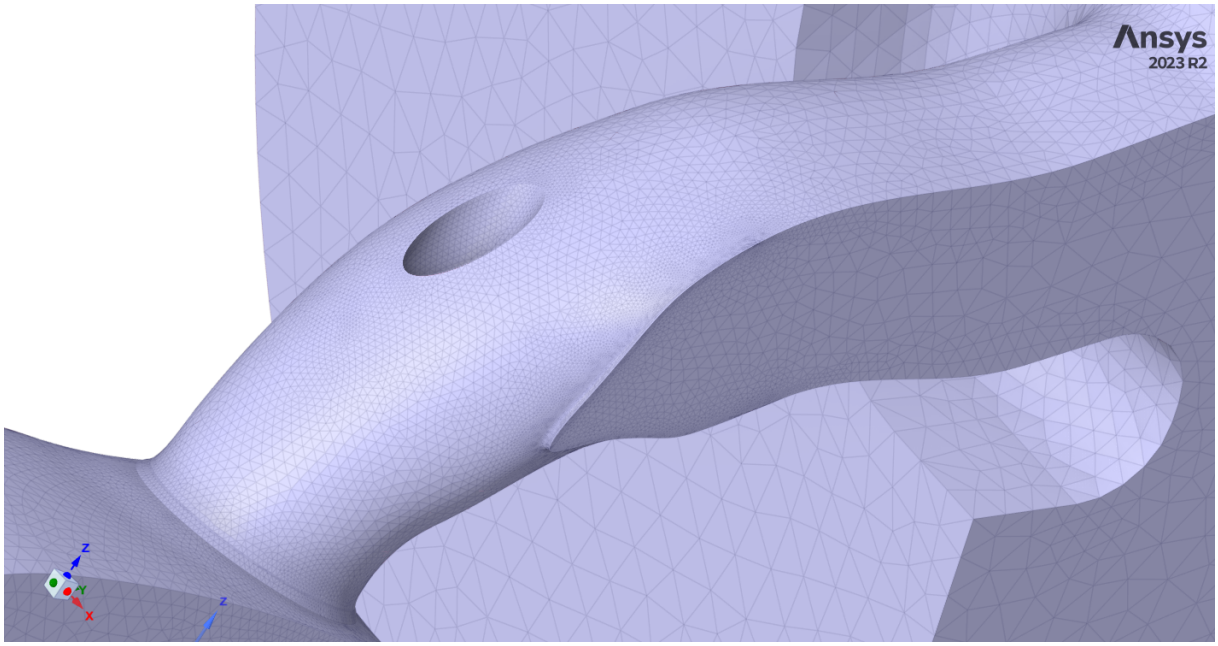
In order to represent the correct morphing transformations of the valve and the related geometries a local coordinate system with one of the axes parallel to that the valve had to created; this allows for an easier definition of the RBF sources acting on the component.

4.4.2 Converge model setup

A converge setup consists in a folder of inputs file in which a command to start a simulation is executed, all of which can be easily created from Converge Studio. After having



(a) Acceptable only for CFD



(b) Acceptable for CFD and mesh morphing

Figure 4.7: Comparison between two STL files

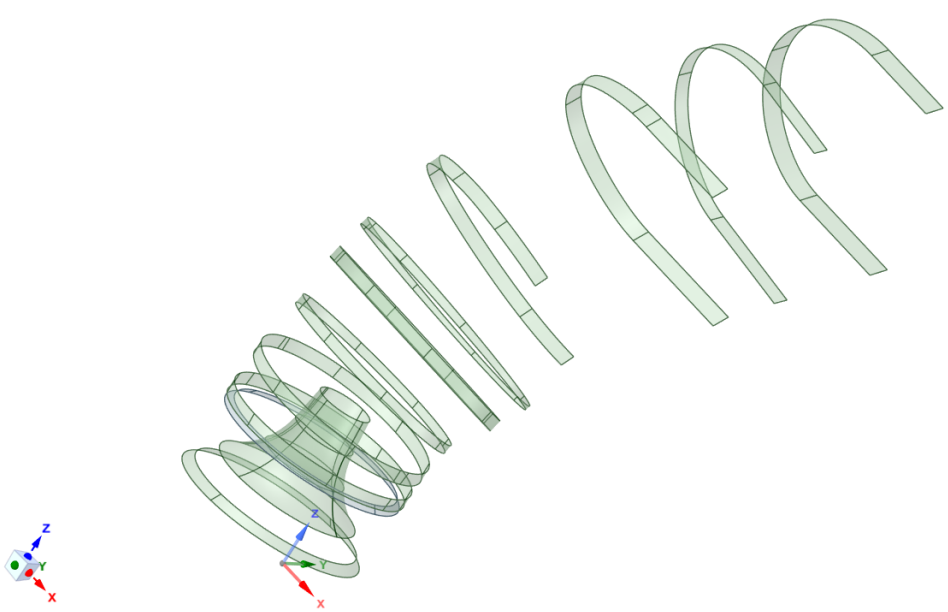
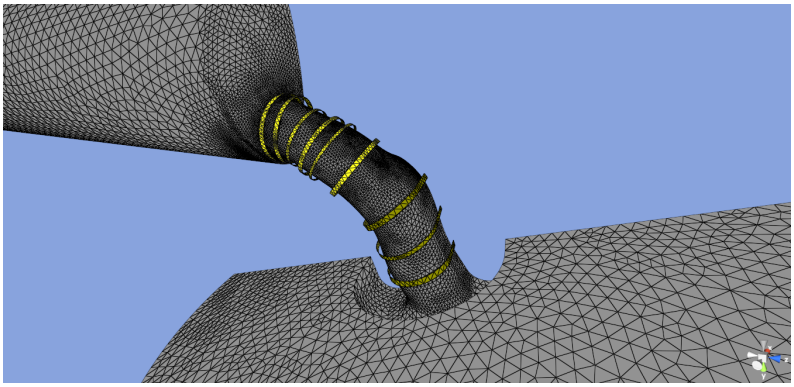
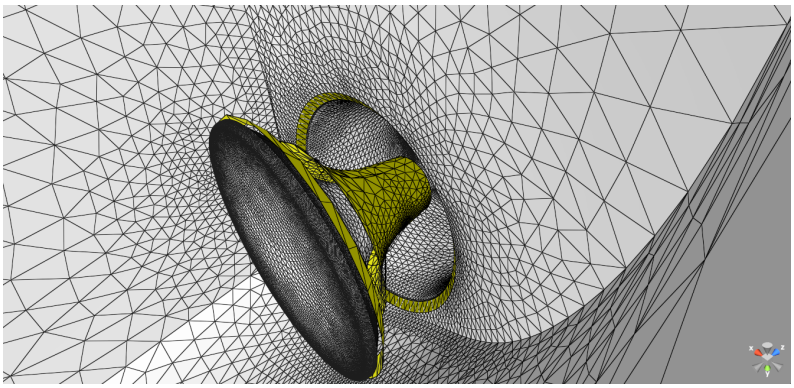


Figure 4.8: Exhaust auxiliary virtual entities inside Ansys SpaceClaim



(a) External view



(b) Valve internal view

Figure 4.9: RBF setup, in yellow the virtual auxiliary geometries

imported an STL or DAT file and prepared the surface it is possible to configure a multitude of parameters. The default option of only one stream was suitable for the problem, since it regards the flow of a single fluid.

Materials

In this section the species and materials employed are defined. The flow bench simulation of the exhaust is meant to work with air, which is modeled as a real gas composed of 77% N_2 and 23% O_2.

Simulation parameters

Here it is possible to select the type of simulation desired and the solver options. Since the current one is a steady-state analysis it is not necessary to set the parameters concerning simulation time, except for the maximum number of cycles, which is 2500. Since the velocities are so high that the Mach number exceeds 0.5, as it will be demonstrated further, it is necessary to consider a compressible flow. The algorithm chosen to solve the Navier-Stokes equations is a pressure based PISO, other than that the remaining parameters are taken from an existing model contained in the Converge library of example cases, named *"case setup SI8 intake flow bench 6mm"*. In order to allow the solver to decide if convergence has been reached for the entire flow a steady-state monitor must be set, this checks if the value of a specified property remains within a certain tolerance for a given number of iterations, if this is true then convergence is achieved. For this case the inlet and outlet mass flow rates have been selected as monitored variables.

Boundary conditions

It is now time to define all the boundaries within the problem, among these must be considered the inlet, outlet, walls and symmetry planes but other definitions are possible as well. Each boundary must be assigned to a region, which purpose will be explained later. Tab. 4.2 collects all the boundaries, their type and number.

In order to respect the pressure differential of 100 mbar at the inflow a total pressure of 111325 Pa has been considered, together with a static pressure of 101325 Pa at the outlet. Following the best practices outlined in the Converge training material turbulent kinetic energy (tke) and turbulent dissipation (eps) have been set at 0.01 and 0.008 m respectively, the latter is 10% of the hydraulic diameter. All the walls inside the setup utilize a Law of Wall condition, with the near wall treatment defined in the Turbulence modeling section. Since the heat exchange is irrelevant for the application a temperature of 300 K has been set at all boundaries. Finally, the symmetry condition has been applied to the corresponding surfaces, allowing to simulate only half of the domain for

Name	Definition	ID
Port symmetry	Symmetry	1
Valve stem	Wall	2
Exhaust port	Wall	3
Valve seat	Wall	4
Cylinder head	Wall	5
Plenum wall	Wall	6
Valve bottom	Wall	7
Liner symmetry	Symmetry	8
Exhaust seat	Wall	9
Valve guide	Wall	10
Plenum symmetry	Symmetry	11
Plenum (Outflow)	Outflow	12
Piston (Inflow)	Inflow	13
Liner	Wall	14

Table 4.2: Exhaust boundaries

lower computational cost, the solver treats the other half of the flow as mirrored to the simulated one.

Initial conditions and events

This branch of the setup tree allows the user to divide the domain in different regions, by assigning boundaries to them, and specify their initial conditions accordingly. Since the examined problem is rather simple only one region is needed, and the initial conditions have proven rather irrelevant in improving the computational time.

Physical models

Turbulence is the only physical property that needs modeling in this case. Among the various possibilities for RANS simulation the $RNG\ K-\epsilon$ model was chosen, following the examples present in the Converge library for similar application and its recommendation in the training material for ICE problems and indoor air simulations [26]. For near wall treatment the *standard wall function* was selected, which models the fluid in similar manner as the one described in chapter 3.

Grid control

Converge works primarily with a cartesian grid, meaning that the cells faces are always oriented as the global coordinate system axes, from the most recent version the possibility of employing an inlaid type of mesh is present as well, but it has not been utilized in this work. Aspects related to grid convergence will be discussed in the next chapter. The workflow for creating a grid is rather simple: one needs to specify the dimensions for the

Definition	Name	Scale	Layers
Cylinder	\\	1	\\
Boundary	Exhaust seat	4	2
Boundary	Valve stem	3	2
Boundary	Exhaust port	3	2
Boundary	Valve seat	3	2
Boundary	Cylinder head	3	2
Boundary	Liner	1	2
Boundary	Valve guide	3	2
Boundary	Valve bottom	3	2

Table 4.3: Exhaust grid embedding entities

Parameter	Value [m]
Lower face center coordinates	x = -0.02, y = 0.03, z= 0.05
Upper face center coordinates	x = -0.02, y = 0.01, z= -0.01
Lower face radius	0.025
Upper face radius	0.025

Table 4.4: Exhaust cylinder shape embedding geometry

x, y and z edges of the base grid, all other operations of refinement and embedding will follow the rule [26]:

$$dx_{scaled} = \frac{dx_{base}}{2^{scale}} \tag{4.1}$$

where dx_{scaled}, dx_{base} are the dimensions of the modified and base grid cells respectively and *scale* represent the desired value of the scale, which may be both for up or downsizing. There are multiple grid scaling control strategies that might be utilized:

- **Fixed embedding**

  The most simple type of grid scaling, allows for the variation of grid cell dimensions in specified regions, virtual geometries or near boundaries. In the option panel the entity of the scaling have to be specified, as well as the number of necessary layers and timing options, it is in fact possible to prescribe if the scaled portion of the grid has to be permanent or variable with time. For the present setup a series of wall embedding and a cylindrical region around the valve areas have been used, given the stationary nature of the flow they are all considered permanent; Tab. 4.3 resumes the employed entities and Tab. 4.4 specifies the cylinder geometry.

  The near wall embedding values have been selected to maintain a y+ value between 30 and 100, which is the optimal range if the near wall region has to be modeled using a wall function. The employment of the latter for this type of application is justified by the absence of fluid separation phenomena.

- **Grid scaling**

This tool is extremely useful to reach convergence faster, for transient simulations it also permits to coarsen or refine the grid at different values of the time variable. For this work it has been employed to coarsen the grid at the start of the simulations, in doing so the first cycles, for which the solution is extremely distant from the final values, are computed faster, allowing for less overall computational time. Convergence is evaluated through the variables in the steady-state monitor: once the convergence criterion is satisfied the entire grid is refined and the process is repeated. This case starts from a scale value of -2, then refines to -1 and finally reaches the effective base grid size until true convergence is reached.

- **Adaptive mesh refinement (AMR)**

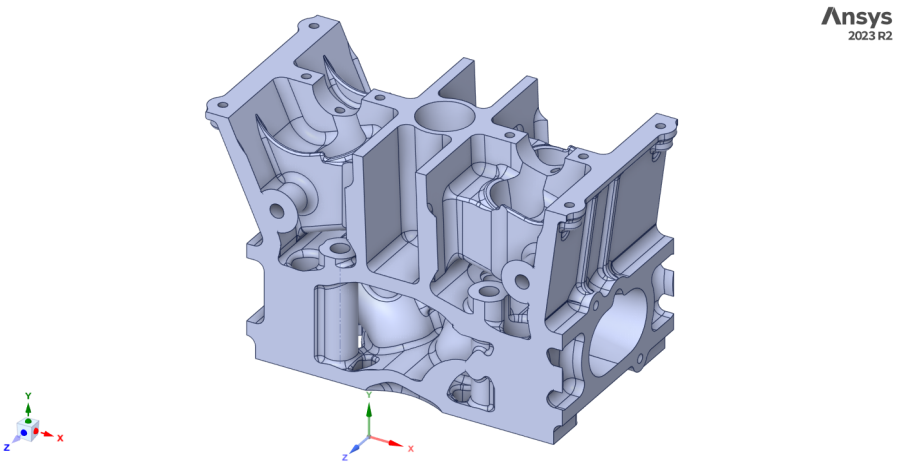
  AMR is another extremely useful tool that allows the solver to take care of the of grid manipulation. It works by defining a variable and a sub grid criterion, if its gradient within a cell exceeds the specified value than the cell will be split in smaller ones, allowing for a better resolution. A variety of other controls are available such as the maximum number of possible cells, the frequency at which the refinement (or release) takes place and timing controls. Despite its usefulness, in this work it has mainly been employed to monitor the y+ value at specified boundaries, ensuring that it remained under 100.

Output/Post-processing

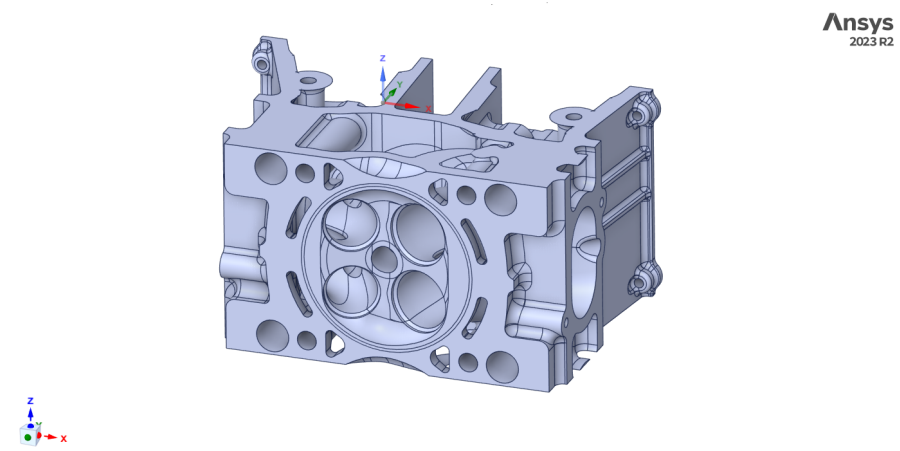
This last section controls which type of files have to be written for the post-processing, as well as the parameters contained within. For this setup only the value of pressure, velocity, y+ and other basic parameters of CFD were evaluated.

4.5 Engine head and conjugate heat transfer setup description

The previous example did not consider the effects that a fluid dynamic problem may have upon a structure. However, in the field of ICEs, temperature and thermal stresses are one of the main causes of failure; thus the ability to conduct a CFD and heat transfer analysis in parallel proves to be extremely powerful especially if the resulting values happen to be suitable to act as load in a structural simulation. The scope of this work did not limit itself to this investigation, but involved the creation of an added link to the previously explored two: mesh morphing was not only used to improve the fluid dynamics performance, but it was leveraged to modify the shape of stress concentration zones in order to diminish the effects of the acting thermal loads. The simulation itself still regards a stationary flow through an exhaust port, with the difference of considering hot air as a fluid. The case



(a) Upper view



(b) Bottom view

Figure 4.10: CAD model of the engine head

is not particularly realistic, but the focus of the thesis was to provide a functioning work frame for multiphysics analyses, future studies of more advanced models may leverage what has been done here on automation and optimization.

The used model derives from a previous thesis by Matteo Marra [23], where an optimization study was conducted through thermal and structural analysis by hypothesizing possible temperature loads in an engine head ducts. The present case aims to be an improvement of the previous study, by creating a way to automate the exporting process of the actual CFD loads and analyze their effect on the original and morphed geometries. Fig. 4.10 shows the CAD model used for the engine head, limited to one cylinder only. Since the model lacks the valve components, four in total, they had to be added in order to perform the flow bench simulation. They were created inside Ansys SpaceClaim by taking inspiration from the ones of the previous example, the intake side is closed and the exhaust ones have a lift of 10 mm. Their separate model is represented in Fig. 4.11 and the a detail of the resulting assembled geometry is reported in Fig. 4.12. It is important to note that, in order to simplify the overall analyses, the valve have been considered

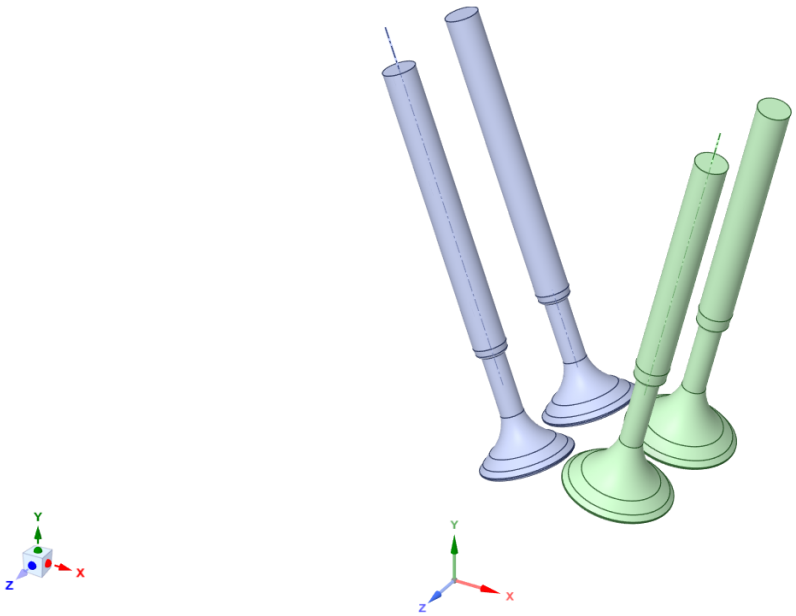


Figure 4.11: Valves CAD model. In gray and green the exhaust and intake ones respectively

Property	Value
Density	2700 $\frac{Kg}{m^3}$
Specific heat	900 $\frac{J}{Kg \cdot K}$
Thermal conductivity	140 $\frac{W}{m \cdot K}$
Young's Modulus	71 GPa
Poisson's ratio	0.33
Tensile yield strength	280 MPa
Tensile ultimate strength	310 MPa

Table 4.5: Aluminum alloy properties

of the same material of the head, whereas in reality they are usually made of stronger materials. Furthermore they are present inside the CFD simulations, since they have a crucial role in the flow, but are absent in the model used for the FEM analyses.

For the material of the engine head properties resembling the aluminum alloy *Silafont 30* (AlSi9Mg) have been selected, Tab. 4.5 reports the one used in both programs, to lower the computational cost they are of constant value, regardless of the temperature.

Since the discussed model refers to only the solid part of the engine head, it was necessary to create the missing boundaries for the CFD domain. Surfaces for the inflow, outflow, liner and covering of the injector seat where created inside Converge studio, making use of the surface preparation tools available; the initial DAT file and the final one are shown in Fig. 4.13.

Having a complete and functioning geometry it is possible to define Tab. 4.6 with the main parameters.

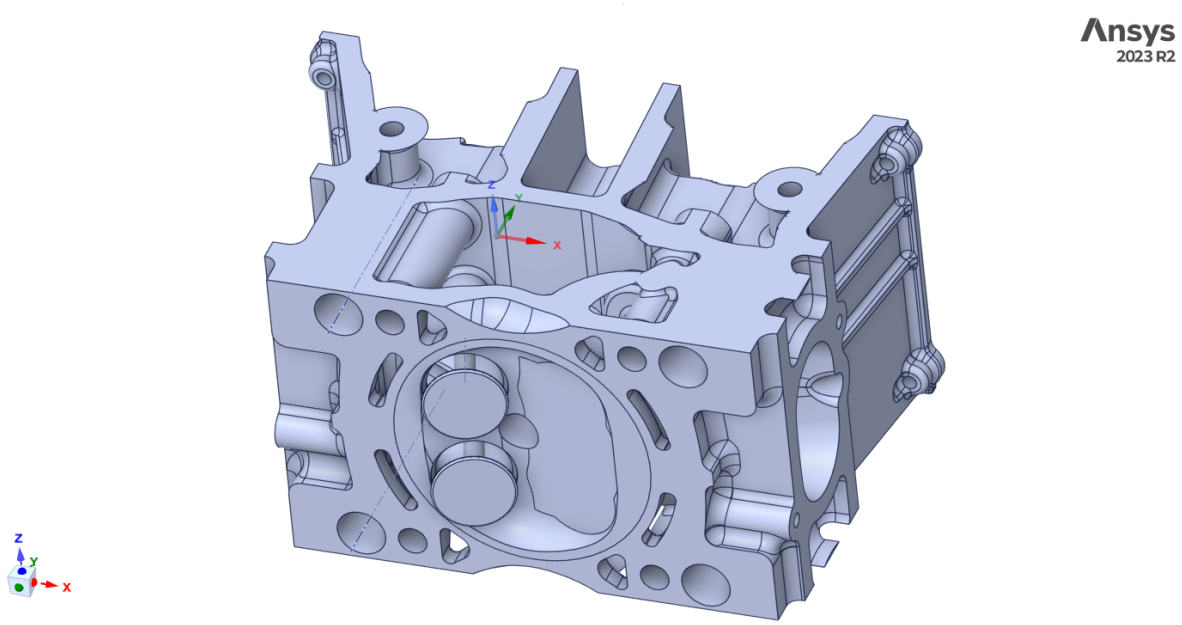
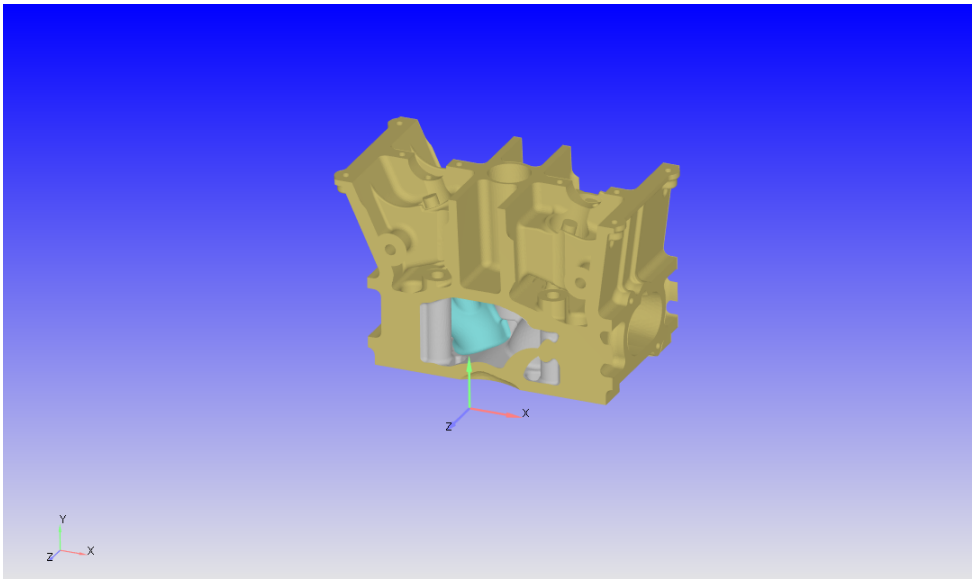


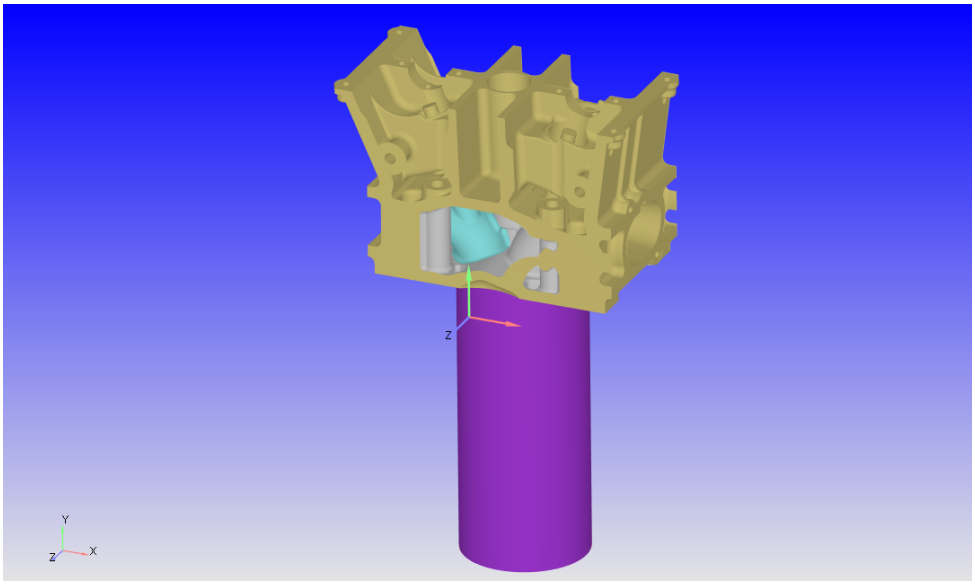
Figure 4.12: Valves and head CAD model, intake side is completely closed, the exhaust ones are lifted by 10 mm

Lift	10 mm
Valve diameter	$28,5$ mm
Curtain area	895 mm^2
Minimum flow area	457 mm^2
Cylinder bore	90 mm

Table 4.6: Engine head geometry for a single valve



(a) Initial DAT



(b) Complete DAT

Figure 4.13: DAT of the engine head

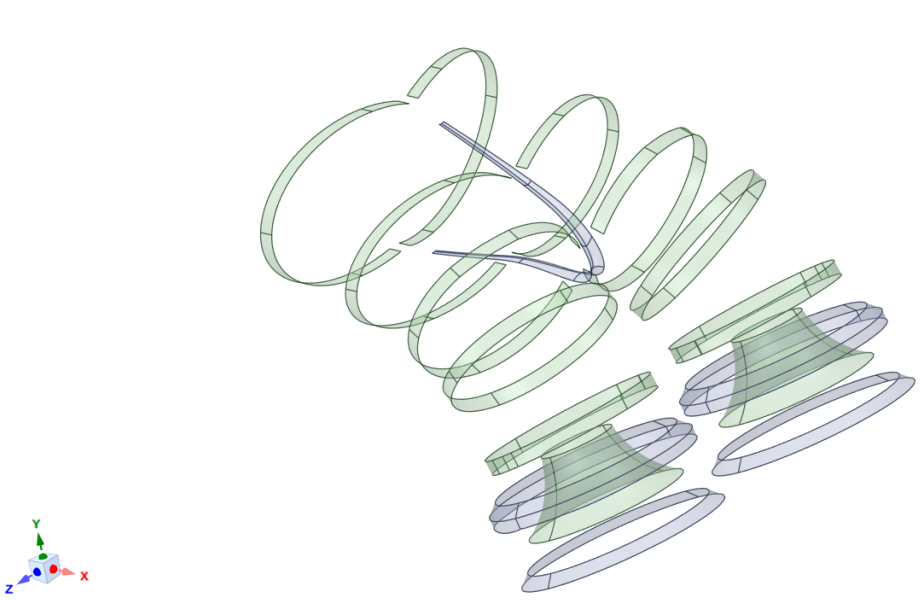


Figure 4.14: Engine head auxiliary virtual entities inside Ansys SpaceClaim

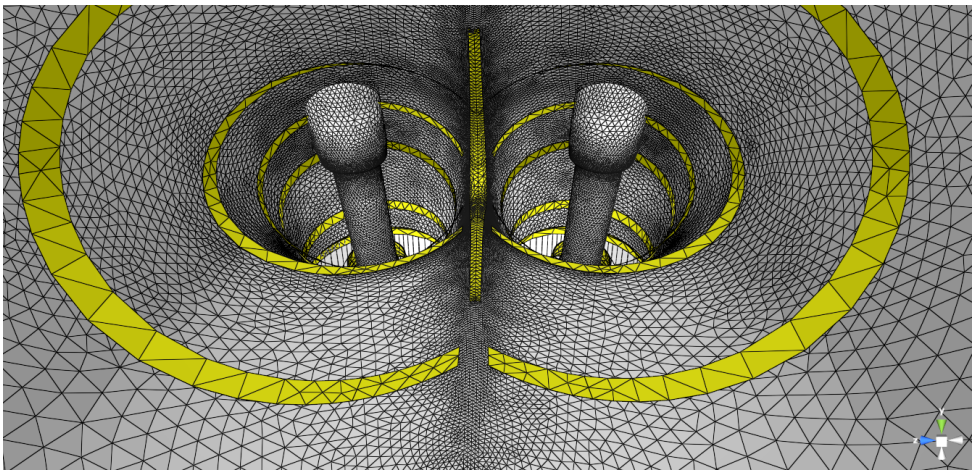


Figure 4.15: Inside view of the RBF model for the engine head, in yellow the auxiliary geometries

4.5.1 RBF model setup

In a similar manner as the previous case, a set of virtual geometries was created, visible in Fig. 4.14; the zones of interest where the inner side of the exhaust port, as well as the valve seats and stems. The model setup procedure is the same as the one described in the corresponding section of the exhaust case. Fig 4.15 shows the RBF model from the perspective of the outlet of the exhaust port. In this case two separate coordinate systems for the valves have been created; this is necessary due to the nature of the scaling transformation, which moves the nodes in a radial manner with the origin of the system as the center, with only one coordinate system the transformation would have not performed in a correct manner.

4.5.2 Converge model setup

This section follows the same path outlined in the exhaust case, additional aspects related to the heat transfer problem are also discussed. As a first stem two streams had to be defined in the defined: *stream 0* for the fluid and *stream 1* for the solid. This is necessary for a correct definition of the boundary conditions, regions and materials.

Material

In addition to the air, a material had to be defined for the solid portion of the domain. By considering the properties of density, thermal conductivity and specific heat defined in Tab. 4.5 it was possible to create a new custom material named *"Silafont30(AlSi9Mg)"*.

Simulation parameters

The simulation time and solver parameters are identical to the ones in the previous case. However, this time the mean temperature inside the solid has been added to the monitor for steady state convergence, thus accounting for both mass flow rate and temperature to be fixed to stop the simulation.

Boundary conditions

Particular attention must be paid to the definition of the boundary conditions, especially for the interface boundaries between the solid and flow domain. This special type of boundary is shared between two region and allows for the exchange of heat through itself. After having defined two regions in the following section, *Fluid* and *Solid*, it is possible to assign the related boundaries to them. For the Inflow and Outflow surfaces the pressure conditions are exactly the same of the previous case, but with a temperature of 500 K and a turbulent dissipation of 0.009 m for the inlet boundary. The wall surfaces have been defined in a similar manner as the exhaust setup, considering a Law of Wall condition for both velocity and temperature, the latter has a value of 293 K at the wall. The interfaces between the two regions are treated as wall at the fluid side and at the solid side their behaviour is coupled to the fluid one. Finally, for the solid walls a convection boundary condition was selected, with a convective heat transfer coefficient of 2 $\frac{W}{m \cdot K}$ and a far field temperature of 293 K (simulating the heat exchange with almost still air), since the application is a flow bench the same boundary condition has been applied to the water jacket surfaces. No symmetry condition was present in this setup. Tab. 4.7 resumes all the boundaries.

Name	Definition	Region	ID
Exhaust seat	Interface	Fluid-Solid	1
Engine head	Wall	Solid	2
Valve seat	Interface	Fluid-Solid	3
Valve stem	Interface	Fluid-Solid	4
Water jacket exhaust	Wall	Solid	5
Water jacket	Wall	Solid	6
Valve bottom	Interface	Fluid-Solid	7
Valve guide	Interface	Fluid-Solid	8
Injector seat	Wall	Fluid	9
Cylinder head	Interface	Fluid-Solid	10
Exhaust port	Interface	Fluid-Solid	11
Outflow	Outflow	Fluid	12
Inflow	Inflow	Fluid	13
Liner	Wall	Fluid	14

Table 4.7: Engine head boundaries and their corresponding region

Initial conditions and events

As it has been underlined in the previous setup, two regions for the different phases must be created in this section. As for the exhaust problem, the initial condition have proved rather ineffectual to accelerate the solution process, only the fluid temperature has been changed to 500 K, since is reasonable to assume that in flowing through the domain the single particles of fluid would not cool down much.

Physical models

This application requires no particular model other than the turbulence one. For the same reasons as before the $RNG\ K-\epsilon$ has been selected, together with the standard wall function for near wall treatment.

Grid control

The grid control strategies are exactly the same as the one previously described, the only change is in the fixed embedding definition and values, both for the fluid and solid portion; they are shown in Tab. 4.8.

Output/Post-processing

In order to transfer the distribution of temperature and pressure inside the solid and fluid domains the output files must present a section containing the coordinates of the center of the grid cells, as well as their value of temperature for the solid and pressure for the fluid. Thus, the temperature may be used as a thermal load inside Ansys Mechanical

Definition	Name	Scale - Fluid side	Scale - Solid side	Layers
Region	Solid	\\	1	\\
Boundary	Valve seat	3	2	2
Boundary	Cylinder head	3	2	2
Boundary	Valve guide	3	2	2
Boundary	Exhaust port	3	2	2
Boundary	Valve stem	3	2	2
Boundary	Valve bottom	3	2	2
Boundary	Injector seat	3	\\	2
Boundary	Exhaust seat	4	2	2
Boundary	Liner	1	\\	2

Table 4.8: Engine head embedding entities

and the fluid pressure in the near wall regions will be mapped upon the surfaces of the cylinder head and duct geometry of the FEM model.

4.5.3 Ansys model setup

The structural analyses have been performed inside the Mechanical module of Ansys Workbench, the layout of which is shown in Fig. 4.16. The project itself is rather simple and it is only comprised of a static structural module and an external data interface. Since the only entities that vary between each simulation are the maps of temperature and pressure (in the external data block) and the morphing parameters (inside the RBF extension in the static structural block), automation of the analyses is straightforward. The data regarding temperature, pressure and displacements of the mesh nodes are stored in specific folders that are accessed externally, after every simulation the Python code changes the files with the new ones, allowing for a different morphed geometry to be simulated with the correct CFD loads. For every variant a new Ansys Workbench process is launched and the Python code executes an internal script (Fig. 4.17) to refresh the settings, load the new data, launch the simulation, save and exit.

Considering the Mechanical static structural module, the original engine head geometry was imported from SpaceClaim and the "Aluminum alloy" under general material was utilized, the mechanical properties of which are listed in Tab. 4.5.

Virtual topology and structural constraints

The first step of the setup is the creation of a series of points on the model for the application of the structural constraints, this can be achieved inside Mechanical through the use of the virtual topology tool or by modifying the geometry with SpaceClaim. A series of four virtual points was created at the center of the lateral portions of the model, two of them are visible in Fig. 4.18 and the other are exactly opposite to them. Given the thermal nature of the problem, particular care must be paid to the constraint assignments:

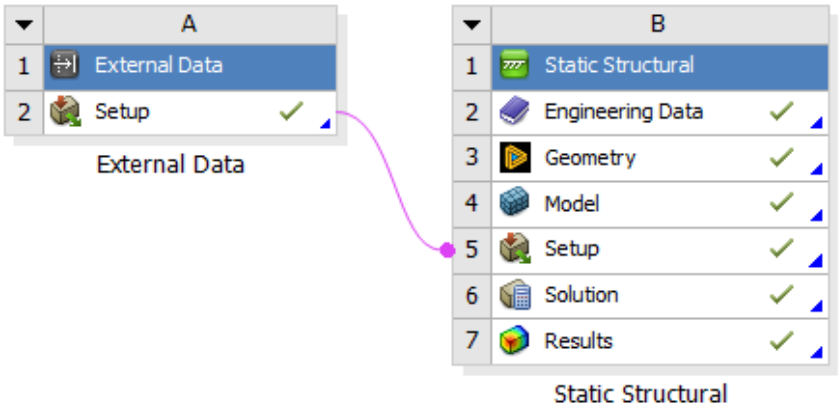


Figure 4.16: Workbench project layout for the structural analysis

```
# encoding: utf-8
# 2023 R2
SetScriptVersion(Version="23.2.142")
Refresh()
system1 = GetSystem(Name="SYS")
setup1 = system1.GetContainer(ComponentName="Setup")
setup1.RereadDataFiles()
Update()
Save(Overwrite=True)
```

Figure 4.17: Workbench internal script for the update command

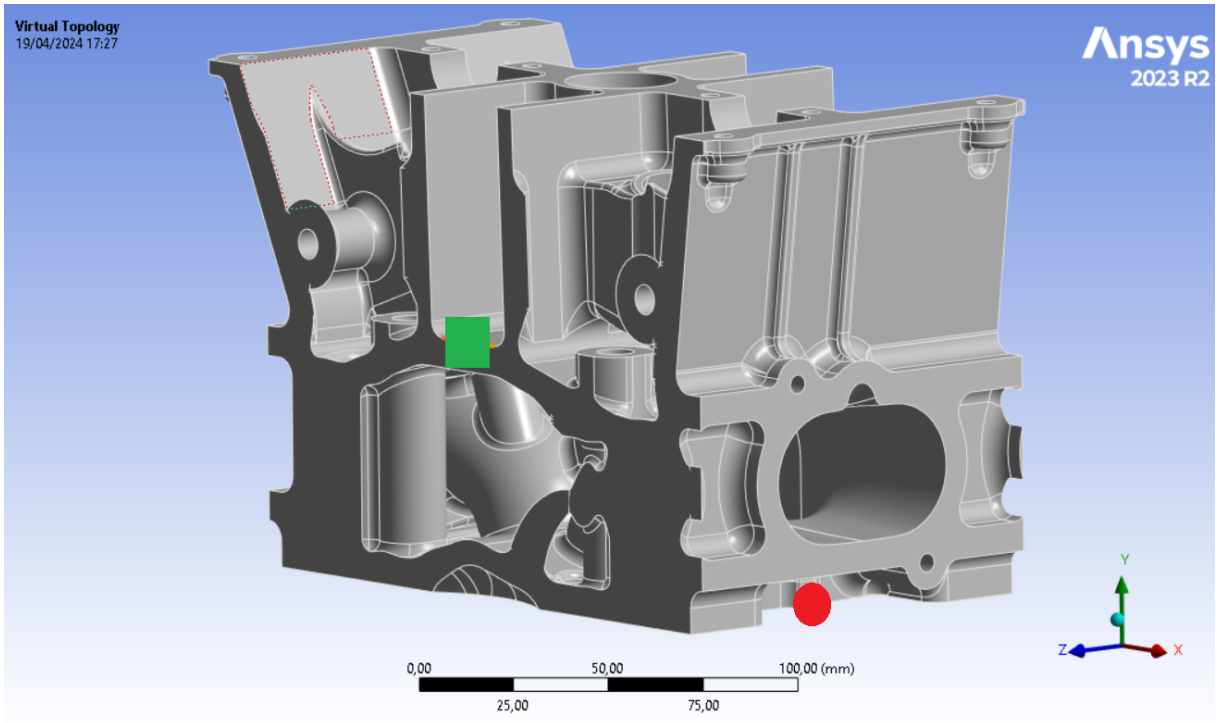


Figure 4.18: Virtual topology for structural constraints

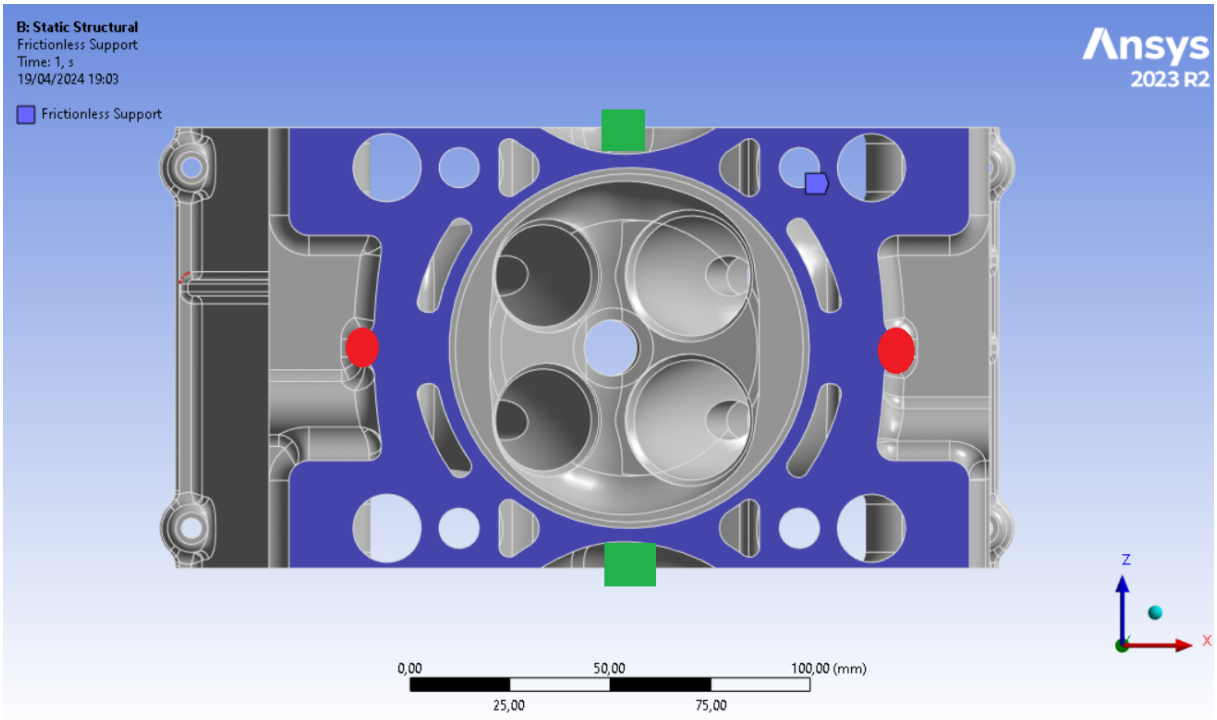


Figure 4.19: Structural analysis constraints: the blue surface represents the frictionless support along the y axis, the green square are prescribed zero displacements points along the x axis, the red circle are prescribed zero displacements points along the z axis

the translation and rotations along the three axes have to be blocked but the at the same time the structure must be left free to expand or contract, otherwise extremely high concentrations of non-realistic thermal stresses would give a false solution. The set of constraints utilized in this work is depicted in Fig. 4.19, it is comprised of:

- A frictionless support on the lower face of the model, which represents how the head is placed upon the flow bench and prescribes no translation along the y axis.
- The two virtual points belonging to the central z, y plane but placed at the extreme ends of the structure. They have a prescribed zero displacement along the x axis.
- The remaining two virtual points, at the extreme ends of the center plane of the lower face with a zero y displacement.

This combination of constraints allows a free thermal deformation of the structure and limits all its possibles rigid movements.

Mesh and mesh morphing

A correct discretisation of the model in finite elements is extremely important to ensure reliable results. Since a suitable mesh for this geometry was already achieved in [23], the same settings have been utilized in this work (Tab. 4.9), with the addition of zone of refinement utilizing the "face sizing" tool. Preliminary simulations showed that a stress

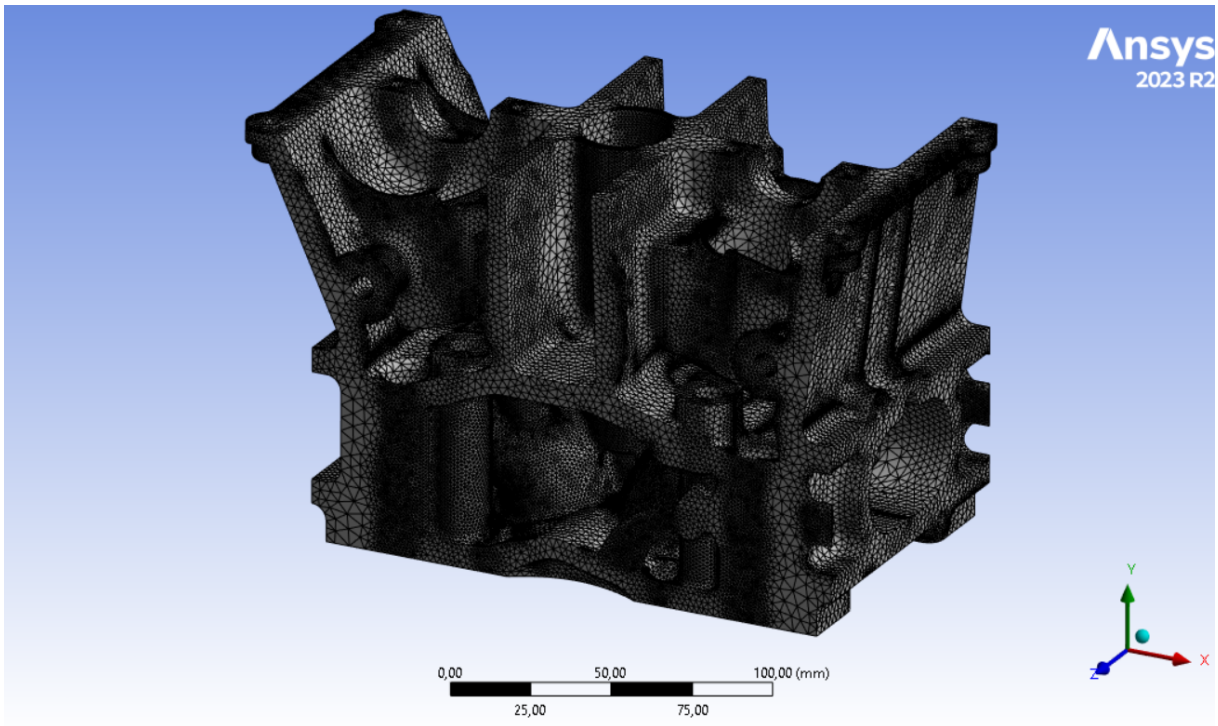
Setting	Value
Element size	90 mm
Adaptive sizing	No
Growth rate	1.85
Max size	180 mm
Mesh defeaturing size	0.45 mm
Capture curvature minimum size	0.9 mm
Curvature normal angle	70.395 mm
Capture proximity minimum size	0.9 mm
Proximity Gap Factor	3

Table 4.9: Mesh settings

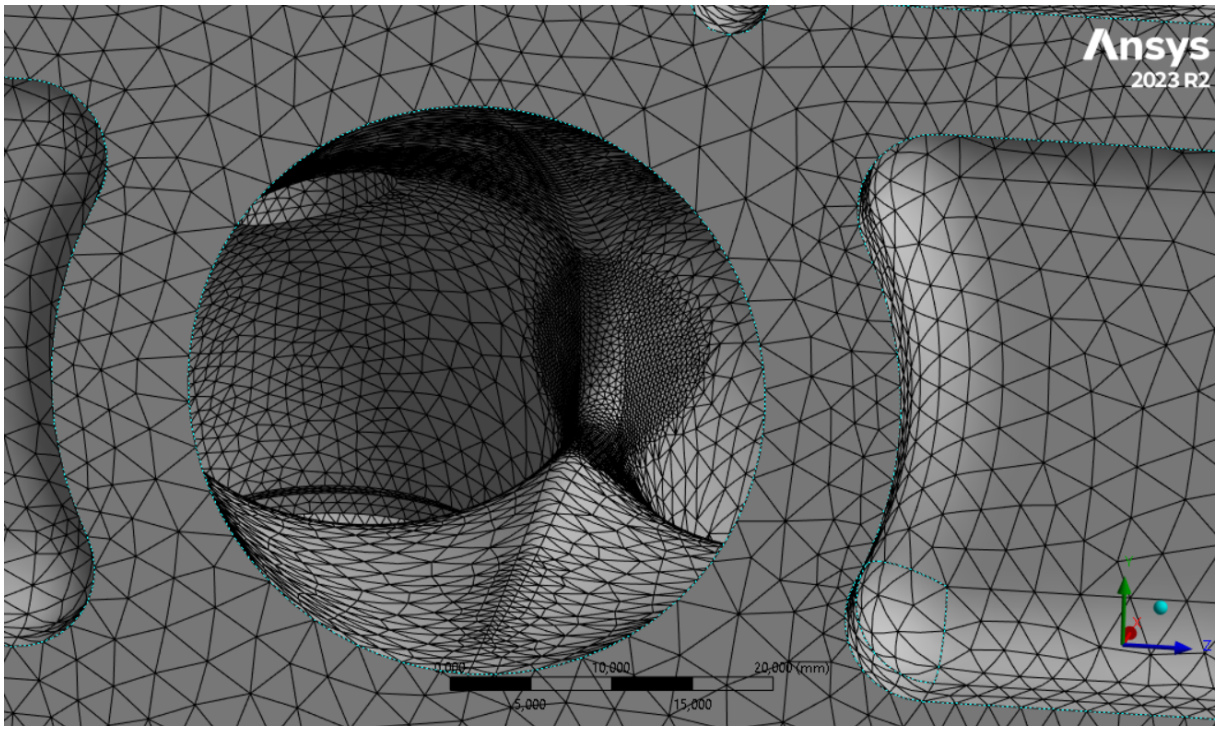
concentration spot was present in the exhaust duct portion where the two valve ports meet, thus virtual surfaces were created inside SpaceClaim to allow for the creation of a refinement zone; convergence tests have been conducted as well and will be presented in the following chapter. For the base grid size a total number of elements is 1621873, whereas the nodes are 2440244. The element order is left to the program to decide, but the majority of them is quadratic. Fig. 4.20 shows the overall model and the refined zone.

RBF morph extension

Mesh morphing inside Ansys Mechanical is possible through the use of the custom ACT extension. The way it functions is extremely similar to RBF-viewer but it possesses some additional features. One employed in this work is the ability to export the values for the morphing from an external PTS file, thus it is possible to achieve the same variations of geometry for both the Converge and Ansys model. The aforementioned PTS file is created by the Python code (Appendix B) by taking the original geometry surface DAT file and the corresponding one created after having performed the morphing inside RBF viewer, it contains the coordinates in space of the original points and their displacements. Since it is extremely probable that the surface mesh of the DAT files, which derives from an STL triangulation, and the one created inside Ansys Mechanical do not match an interpolation of the imported values is necessary. Given that the number of points is extremely high the computational cost is prohibitive but, since all the simulations performed showed that the stress concentration was located in the same area, it is possible to restrict the number of points that will take part in the morphing to only those that will be effected by an actual displacement (for all the variants the majority of nodes remains still, but the interpolation has to go through them nonetheless). Thus only the nodes surrounding the concentration of stress area are selected for the RBF Region, under this last one two RBF sources are created: one that takes the value from the external PTS file, the other that surrounds it and is fixed (a translation transformation of zero prescribed displacements). In order to select the correct nodes a set of named selections was created, which are detailed from

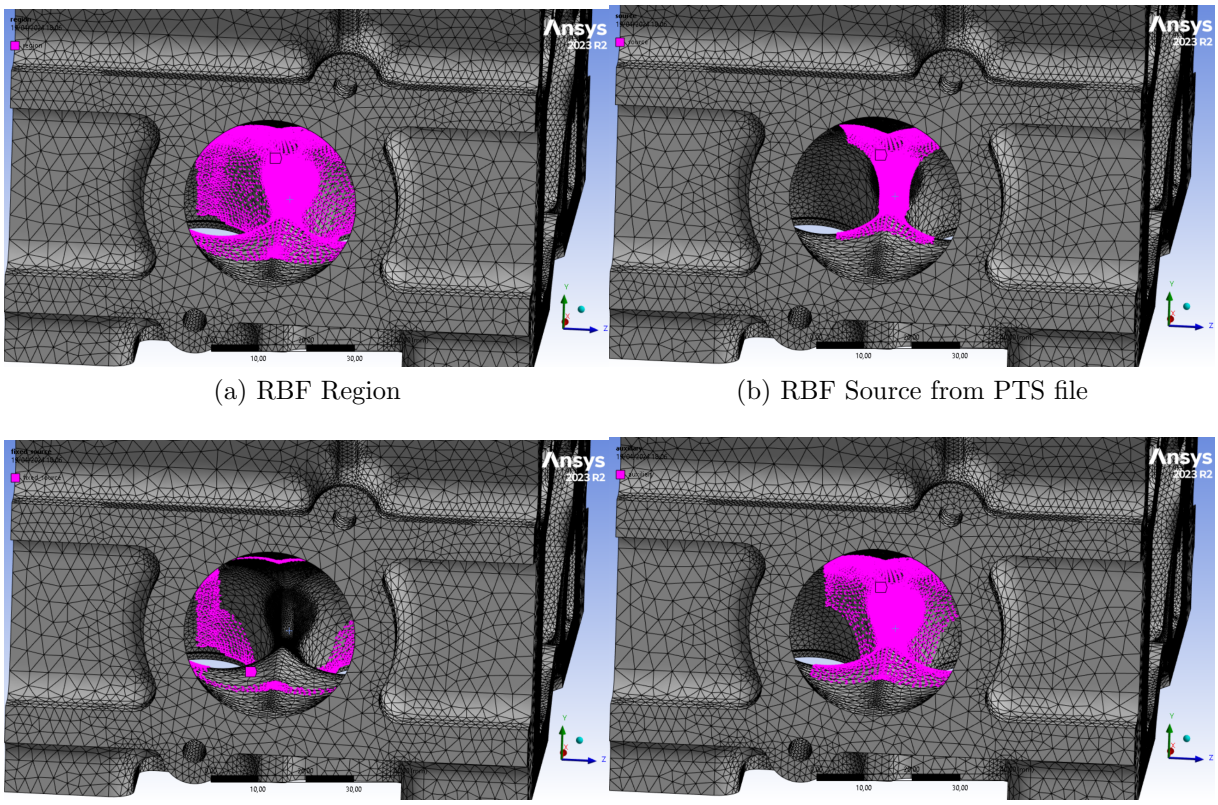


(a) Complete model



(b) Detail of the refinement zone

Figure 4.20: Mesh of the model in Ansys Mechanical



(a) RBF Region (b) RBF Source from PTS file

(c) RBF Source fixed (d) Auxiliary points

Figure 4.21: Named selections of mesh nodes for morphing

Action	Criterion	Operator	Lower bound [mm]	Upper bound [mm]
Add	Location X	Range	-70	-10
Filter	Location Y	Range	10	60
Filter	Location Z	Range	-110	-60

Table 4.10: RBF Region work sheet: contains the coordinates of the global coordinate system within which the mesh nodes are selected

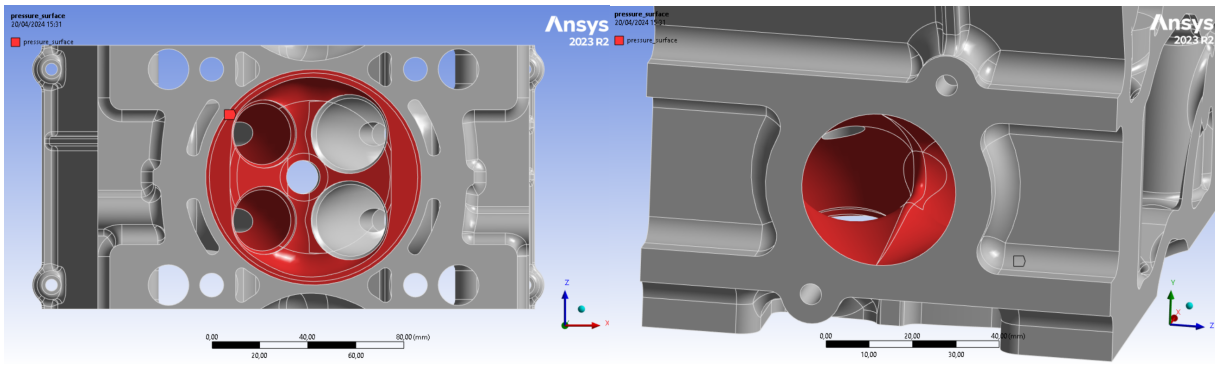
Tab. 4.10 to Tab. 4.12 and Fig. 4.21. Named selections for sorting nodes through a worksheet were created for the RBF Region and Sources, the fixed one is the result of the difference between the region and an auxiliary ones.

Action	Criterion	Operator	Lower bound [mm]	Upper bound [mm]
Add	Location X	Range	-65	-15
Filter	Location Y	Range	15	55
Filter	Location Z	Range	-105	-75

Table 4.12: Auxiliary work sheet: contains the coordinates of the global coordinate system within which the mesh nodes are selected

Action	Criterion	Operator	Lower bound [mm]	Upper bound [mm]
Add	Location X	Range	-56	-26
Filter	Location Y	Range	22	48
Filter	Location Z	Range	-100	-80

Table 4.11: RBF Source from PTS file work sheet: contains the coordinates of the global coordinate system within which the mesh nodes are selected



(a) Cylinder surfaces (b) Port surfaces

Figure 4.22: Surfaces upon which the external pressure loads acts

Temperature and pressure loads

The loads from the CFD simulations are imported in the External Data module, then they are transferred in the Setup section of the Structural Analysis block. The temperature is interpolated on the entire volume of the geometry, whereas the pressure acts only on the cylinder head and exhaust duct surfaces, as shown in Fig. 4.22.

Chapter 5

Results

In this chapter the results of the optimization workflow are explored for both the proposed models. A series of DOE tables have been created to investigate how the effects of morphing specific areas influences the overall performances, a single all-encompassing table was not possible due to the high complexity of the RBF-viewer setup. Since both cases are steady-state flows with a prescribed pressure differential the performance parameter is the mass flow rate.

5.1 Exhaust port

The results regarding first exhaust port will be discussed in this section: at first the analyses for grid independence are presented, followed by a discussion of the most promising DOE studies.

5.1.1 Grid convergence

In order to trust the results obtained from a simulation, whatever the field of application, a grid convergence test is needed. This process involves the monitoring of a specific variable in a series of simulation where the only varying parameter are the mesh settings. With coarser grids the simulation results are not reliable at all and they tend to converge towards a specific value as the mesh is refined, or in other words when the cells dimension diminish. The value at which it is reasonable to stop is when the percentage difference of the monitored variable between two successive refinements becomes less than 5 %, however this is strongly dependent on the application filed and available computational power.

For this case three base grid values, from coarser to finer, have been explored: 4 mm, 2 mm and 1 mm. As it has been explained in the previous chapter, Converge computes all grid scaling parameters based on equation (4.1), and so, by changing only the size of the base grid, it is possible to maintain all local modification and AMR settings in

Base grid $[mm]$	Mass flow rate $\left[\frac{Kg}{s}\right]$	Number of cells	Solution time [s]	Percentage difference
4	0.14	215000	1726	\\
2	0.147	738000	11307	5 %
1	0.152	4500000	$>$ 24 h	3.4 %

Table 5.1: Exhaust grid convergence, the number of cells refers to the displayed value once convergence is reached

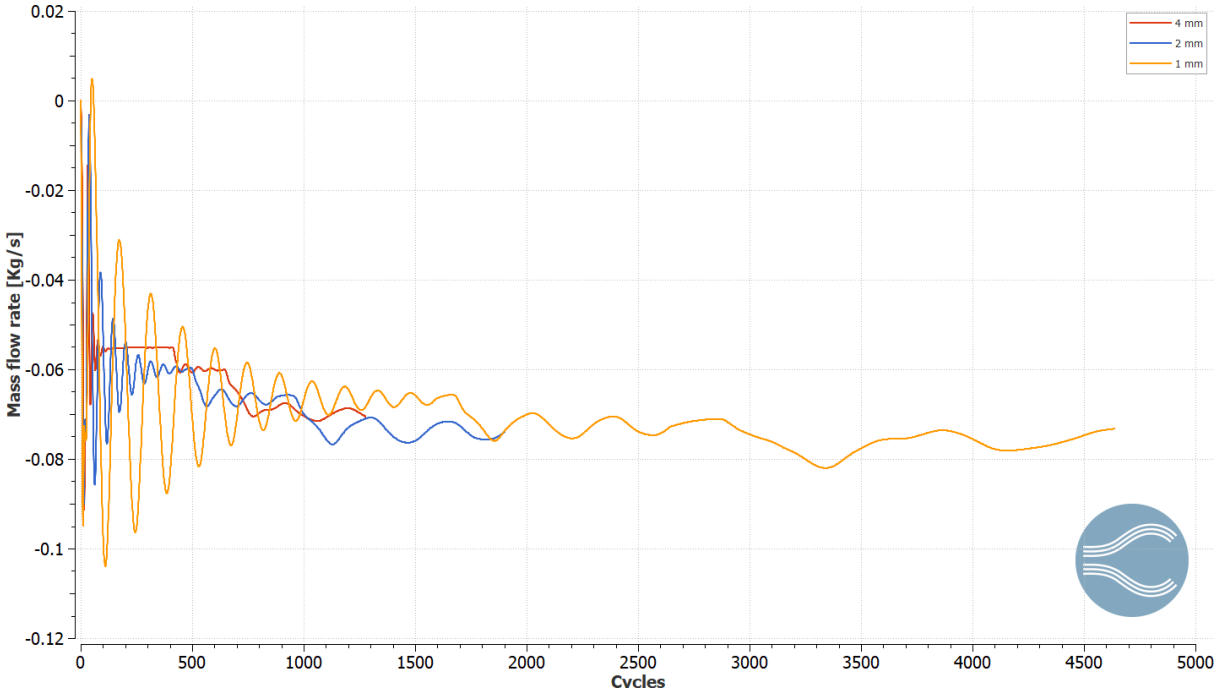


Figure 5.1: Exhaust grid convergence line plot: on the x axis the number of cycles performed, in the y axis the mass flow rate. The values refer to a single valve flow, since the model is only half the geometry

the successive refinements of the grid independence study. Results of the test are shown in Tab. 5.1 and Fig. 5.1, it is possible to note that total grid convergence is almost completely achieved in the 2 mm case and the 4 mm one is extremely close as well.

However, given the completely different solution times, the 4 mm setup has been chosen as the default value for the DOE study, given the high number of simulations that needed to be performed. It would have been also more correct to analyze one case with $y+ \approx 1$ and the wall function option disabled, in order to simulate the near wall treatment without a model, but the computational cost have proven out of reach for the hardware employed. The values of Fig. 5.1 are negative because they are relative to the inlet mass flow rate and Converge treats entering properties as negative, Fig. 5.2 shows the effects of the grid scaling option: the mesh is at first extremely coarse and it is refined two times, the fluctuating values of the last iterations are the cells being activated (or deactivated) through AMR. Finally it is worth noting that since the simulations involve a compressible gas and RANS turbulence modeling the properties posses a fluctuating behaviour, thus it is necessary to compute a mean value; since the grid scaling tool has been used only the

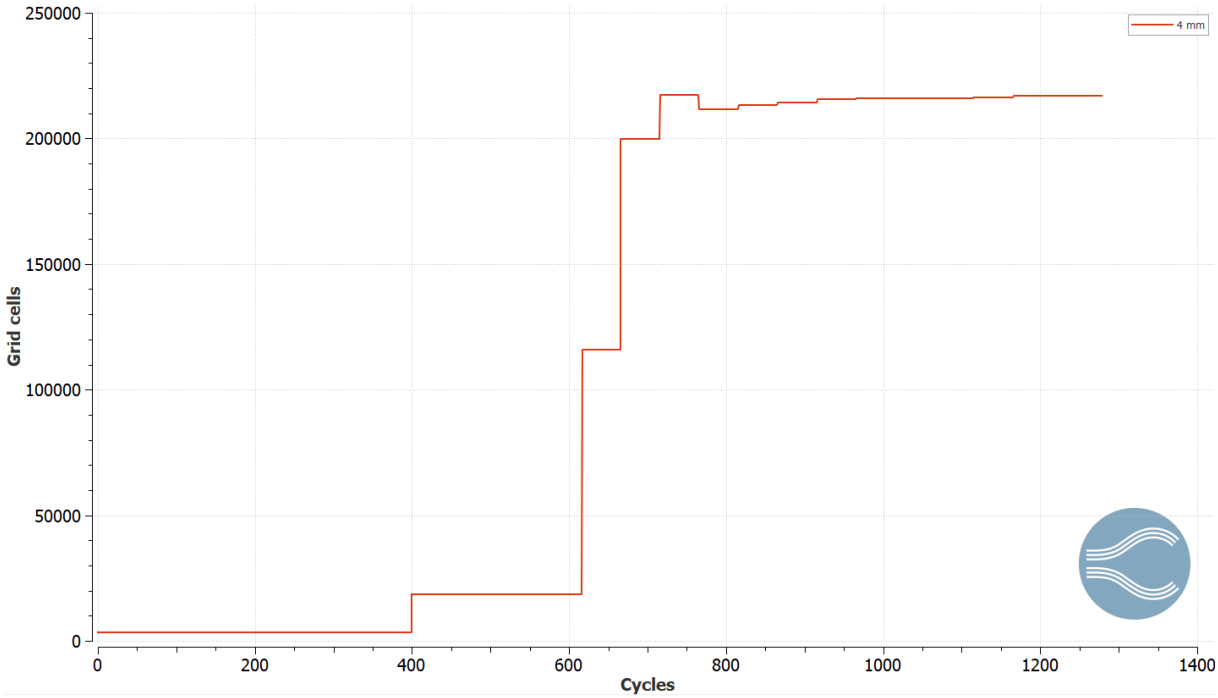


Figure 5.2: Grid scaling effects on the number of cells in the simulation

last iterations have to be considered.

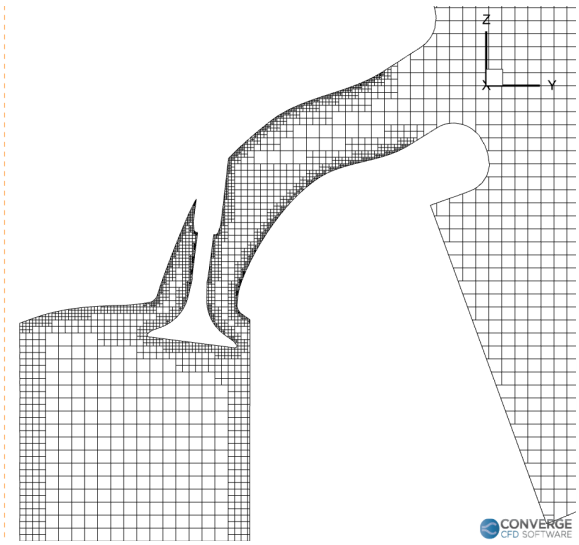
Fig. 5.3 shows the final mesh employed in the different cases, the represented plane is obtained from a slice cutting along in the vicinity of the valve, in order to capture the most critical zone discretisation.

5.1.2 Results discussion

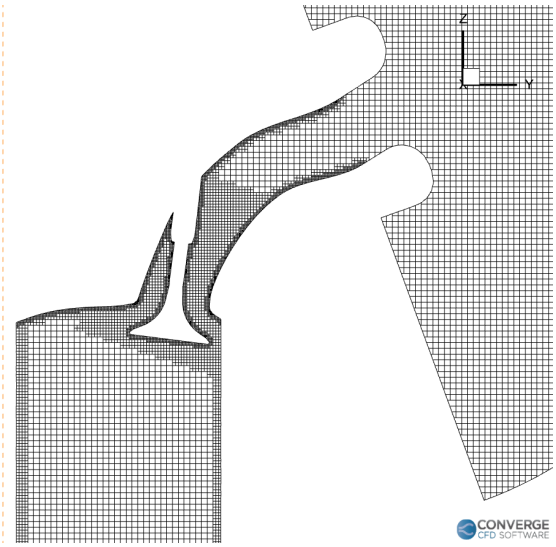
Various design of experiment involving different morphing parameters have been explored. Multiple attempts to vary the geometry of the duct between the valve and the plenum have been performed, but their difference in performance has been negligible or, in some cases, worse. Of the many morphing setups the two that brought the best results are presented in section: one regarded the duct while the other effected the valve seat. The improvements are reported together with the relative DOE table and RBF-viewer setup.

Duct morphing

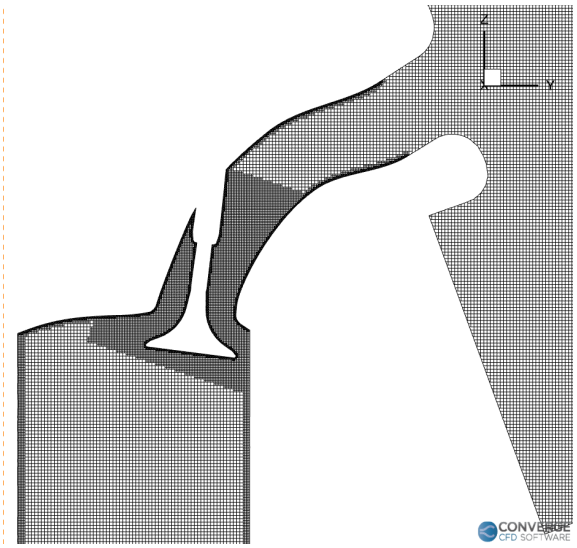
Following the procedure described in chapter four a RBF-viewer project containing a surface file and some auxiliary geometries was created, of the STEP files presented in Fig. 4.8 only some were necessary for this DOE and are shown in Fig. 5.4 as the colored disks: the gray ones are fixed whereas the other two are movable RBF sources. The morphing zone has been restricted to the duct portion by only assigning the corresponding nodes to the RBF Region (blue dots of Fig. 5.5 (a)), then four RBF sources have been created: the first one is fixed and contains the two gray auxiliary disks previously discussed, the second one in shown in Fig. 5.5 (b) and is fixed as well, the remaining two contain some



(a) 4 mm base grid



(b) 2 mm base grid



(c) 1 mm base grid

Figure 5.3: Grid representations from the mesh convergence test for the exhaust port

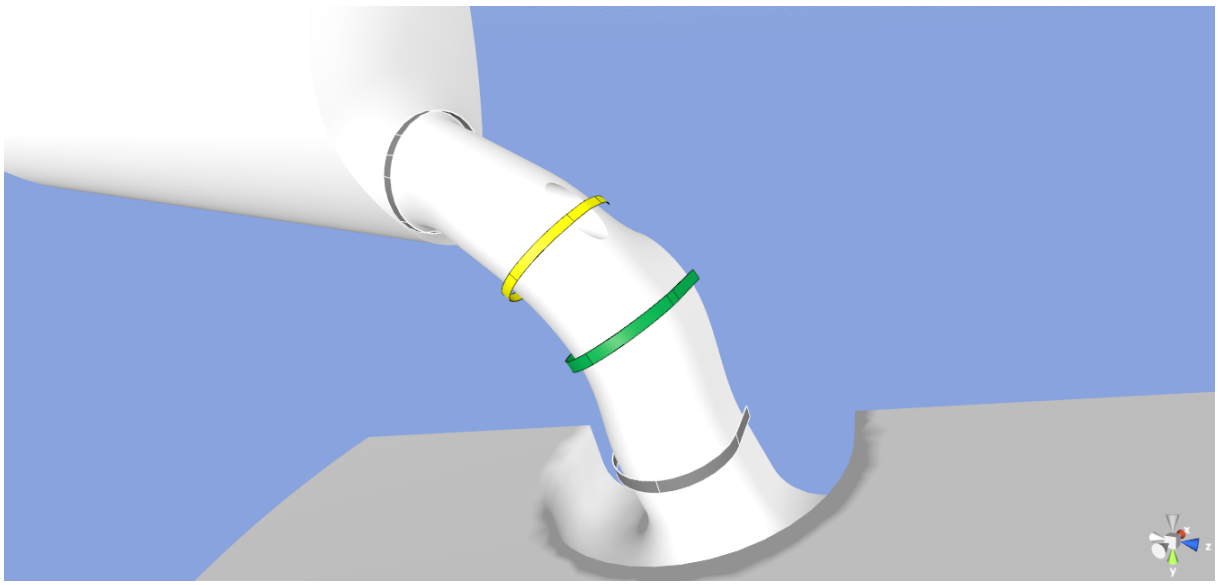


Figure 5.4: Detail of the duct morphing setup for the exhaust: the colored disks are the virtual geometries. Gray: fixed, green and yellow: movable

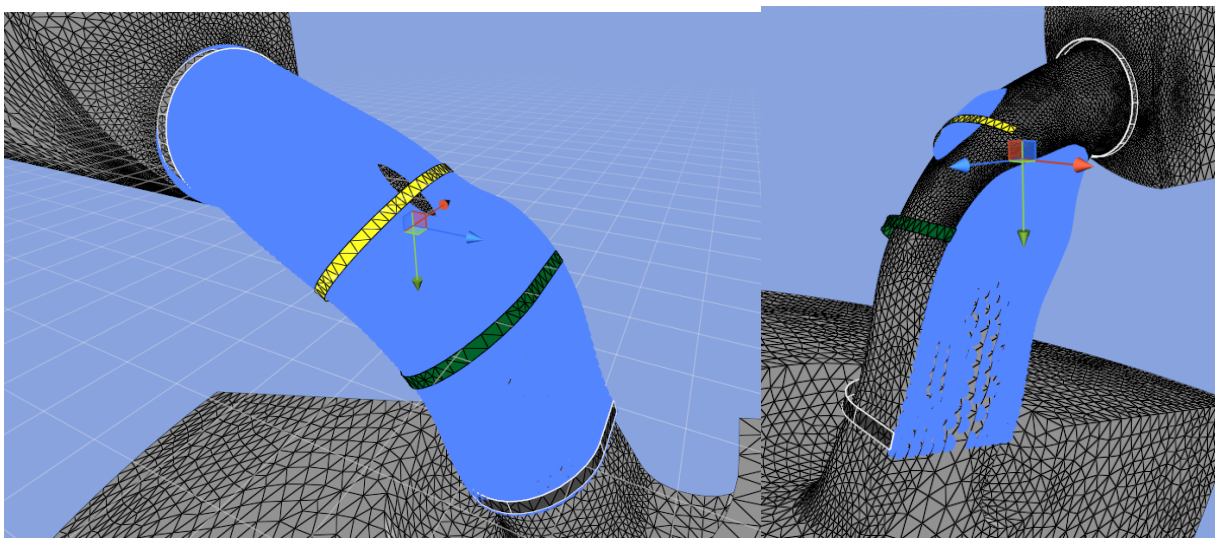
ID	RBF Source 3.T.X	RBF Source 3.T.Z	RBF Source 4.T.X	RBF Source 3.T.Z
1	-2	-4	-1	2
2	-4	-6	-2	4
3	-6	-8	-4	6

Table 5.2: DOE table for the exhaust duct morphing. Each morphed geometry is marked by the ID, the corresponding transformation values are inserted in the JSON file

nodes of the two other virtual entities and are used to impress a translation on the region nodes, they are reported in Fig. 5.6.

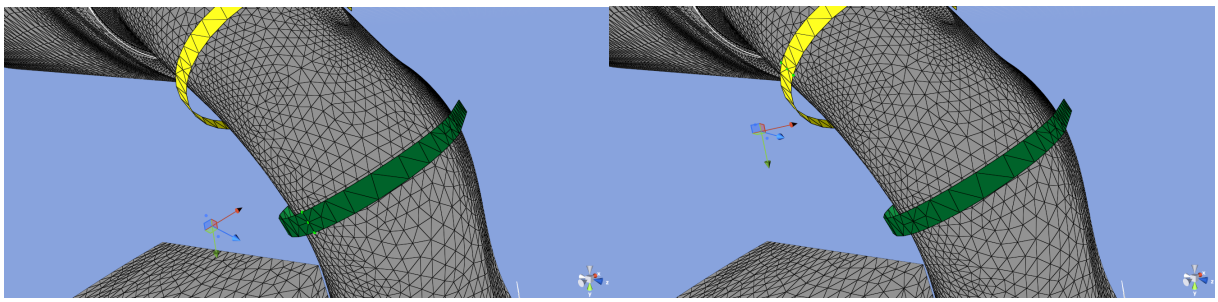
The excel table for the corresponding DOE is reported in Tab. 5.2, for every modification of the source values a new variant is created, corresponding to a new row, the fixed sources are omitted since they posses a constant value of zero. All transformations are performed with respect to the global coordinates system and the degree of the radial function is linear. The transformations impressed by the sources are reported in the header of Tab. 5.2 and follow a similar definition throughout this work; the content of the cell is divided by two points: before the first one is placed the name of the RBF Source, in between is the type of transformation that can be expressed by either using the full name ("Translation","Rotation" or "Scaling") or initial letter ("T","R" or "S"), the last item is the direction of the transformation ("X","Y" or "Z") with respect to the coordinate system of the source. This denomination scheme allows for easy interaction with the Python script, allowing the automation of the morphing procedure.

A comparison between the original duct geometry and the third variant, the one which nodes move the most, is reported in Fig. 5.7. The line plot shown in Fig. 5.8 reports the mass flow rate of the original geometry and the variant which performed the best (the first one), the results are extremely underwhelming and the they were the best one among all



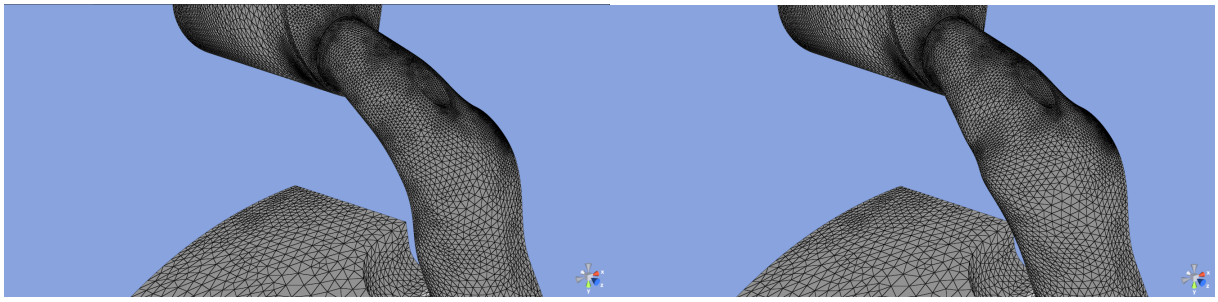
(a) RBF Region (b) RBF Source 2

Figure 5.5: Nodes corresponding to the RBF Region and the second fixed RBF source



(a) RBF Source 3 (b) RBF Source 4

Figure 5.6: Nodes corresponding to the RBF sources: the green and blue dots are the initial and final positions respectively



(a) Original geometry (b) Morphed geometry. ID = 3

Figure 5.7: Comparison between the original and morphed geometry for the exhaust duct DOE

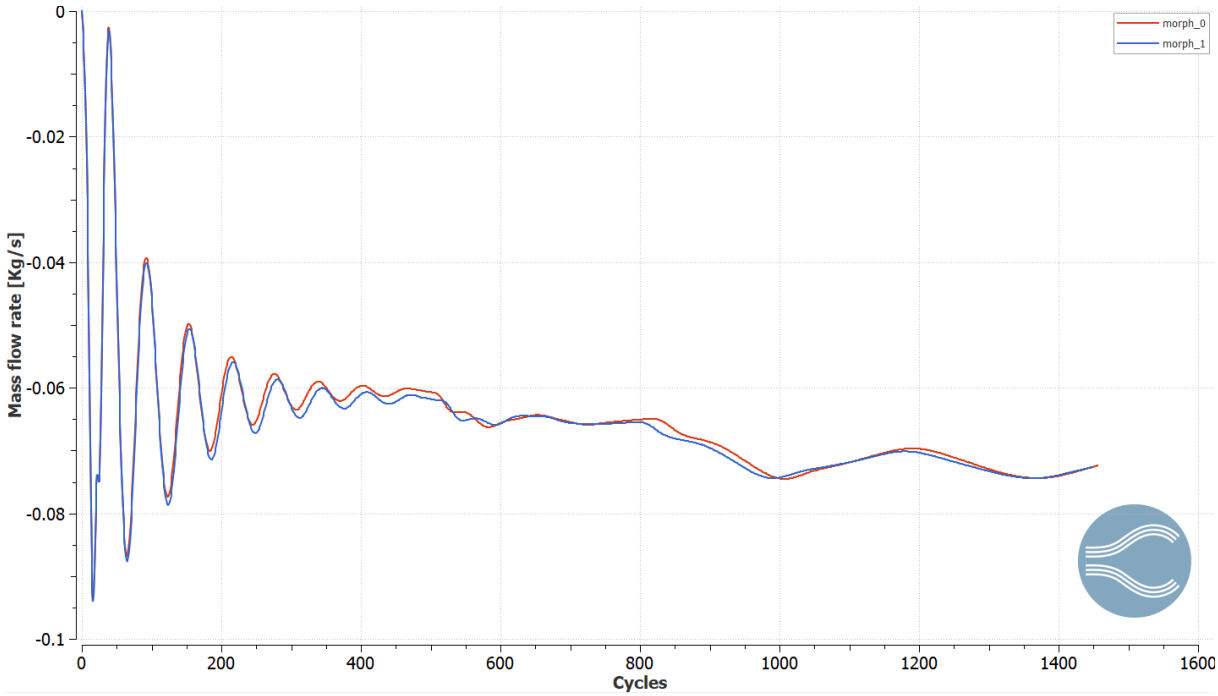
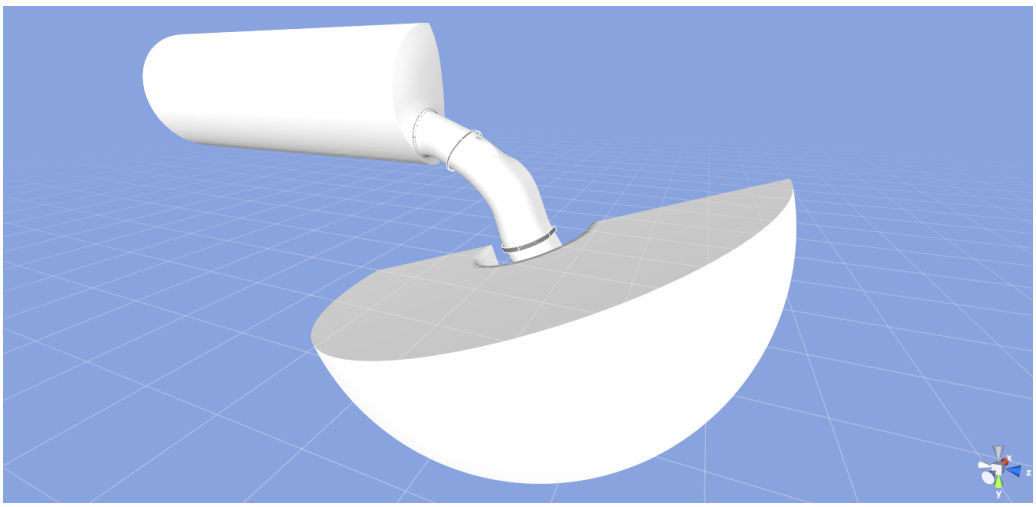


Figure 5.8: Converge line plot for the duct exhaust morphing: *morph 0* corresponds to the original geometry, *morph 1* to the first morphed variant

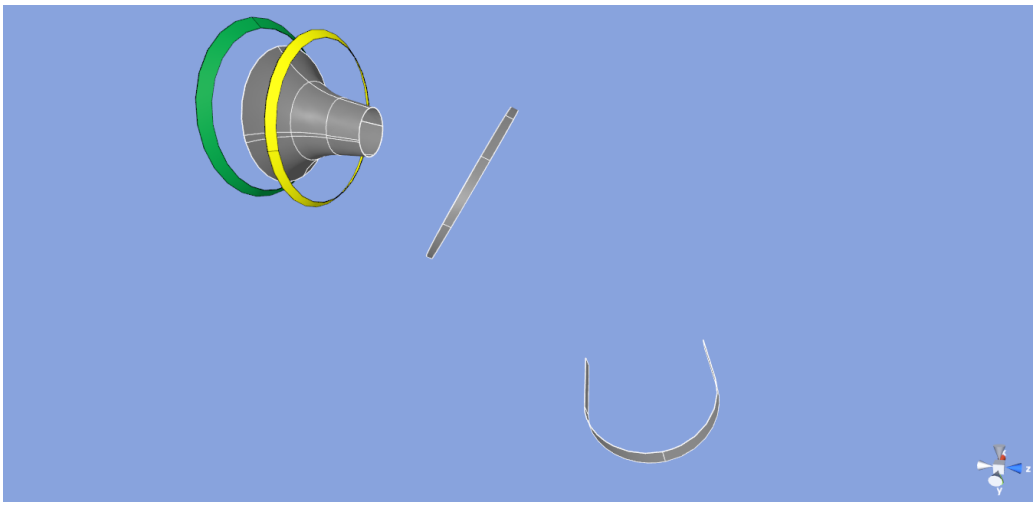
the studies that focused on morphing the duct region. Better performances where obtained by modifying the seat angles of the valve, and are described in the following paragraphs.

Valve seat morphing

Considering that the most critical condition for a fluid flowing in an exhaust port is the moment at which it goes through the minimum cross section area a preliminary study to investigate the effects of mesh morphing in this region has been conducted. Particular attention has been payed to the portion of duct where the valve resides once it is closed, called "valve seat". Translation and scaling transformation have been prescribed to the nodes on the valve and those on the corresponding portion of the seat, in order to obtain a still functioning geometry; in other words the valve would still be able to be closed even in the morphed configurations. Following the same procedure of the previous design of experiment, Fig. 5.9 shows the RBF-viewer setup and the auxiliary entities, Tab.



(a) Setup for the valve seat morphing DOE



(b) Auxiliary entities, the gray ones are fixed

Figure 5.9: RBF model and auxiliary geometries for the exhaust.

5.3 and Tab. 5.4 are the corresponding DOE tables. The first considers only scaling transformations, the second consider also a translation along the direction of the valve axis. As already underlined in the previous chapter, in order to simplify the morphing procedures, a new coordinate system with the z axis coincident with the valve one was created, to which the RBF Sources of this DOEs refer.

Fig. 5.10 shows the two zones belonging to the same RBF region, since the morphing has a coupled effect on both the valve and duct the corresponding nodes possess the same displacements. The RBF Sources are depicted in Fig. 5.11, the translation and scaling transformations act on the same geometries which are the smaller circular curves of the green an yellow auxiliary entities. A comparison between the original and a morphed variant is reported in Fig. 5.12, Tab. 5.5 and Tab. 5.6 show the results in terms of mass flow rate, discharge coefficient and percentage difference with respect to the original geometry. The discharge coefficient for a single valve has been calculated from equation (2.15), assuming the curtain area in Tab. 4.1 and an inlet static pressure of 111000 Pa

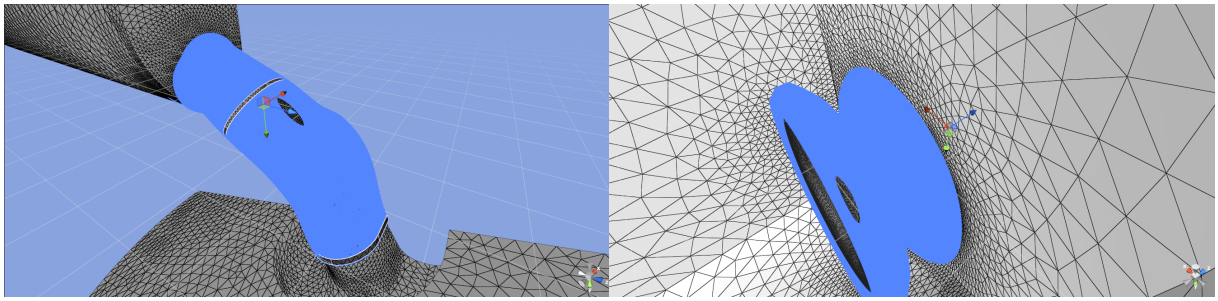
ID	RBF Source 3.S.X	RBF Source 3.S.Y
1	1.025	1.025
2	1.05	1.05
3	1.06	1.06
4	1.02	1.02
5	1.01	1.01

Table 5.3: First DOE table for the exhaust valve seat morphing, containing only scaling transformations.

ID	RBF Source 3.S.X	RBF Source 3.S.Y	RBF Source 5.T.Z
1	1	1	0.25
2	1	1	0.5
3	1	1	0.75
4	1	1	1
5	1.025	1.025	0.25
6	1.025	1.025	0.5
7	1.025	1.025	0.75
8	1.025	1.025	1

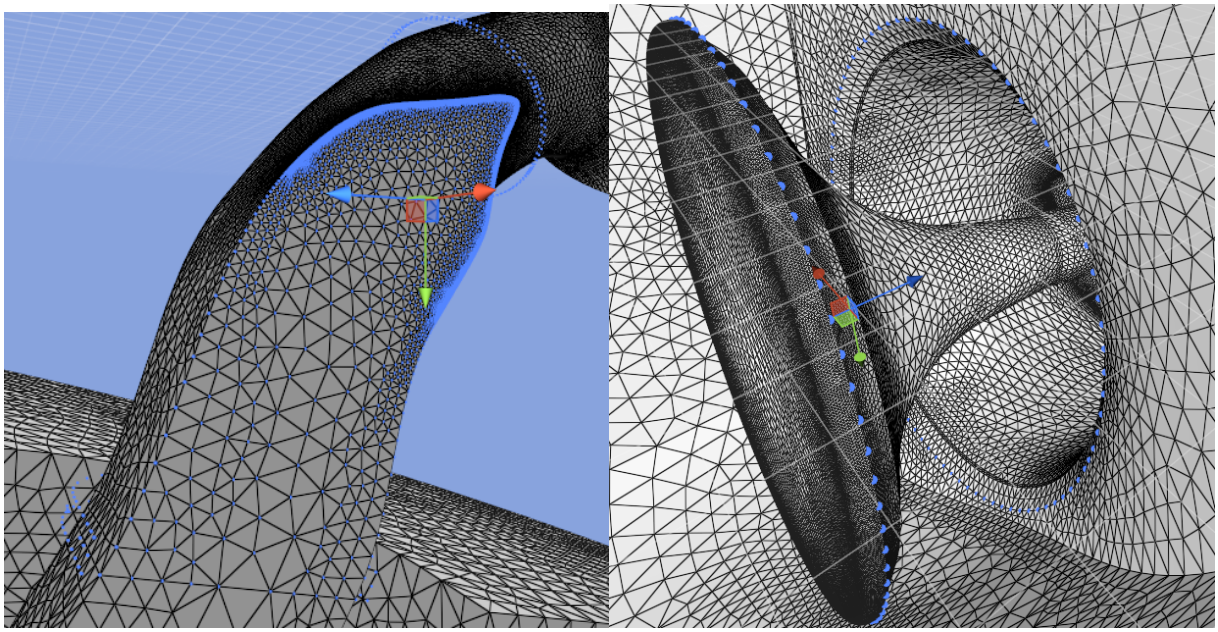
Table 5.4: Second DOE table for the exhaust valve seat morphing, containing both translation and scaling transformations. RBF Source 3 and 5 are applied on the same nodes.

(value obtained from the simulations). The line plot in Fig. 5.13 compares the former shape with the best variant obtained, whereas the contour depicted in Fig. 5.14 shows the difference in velocity and pressure fields. Considering this last ones, especially those for the velocity, it is possible to se how the morphing of the valve seat helped to smooth a velocity hot spot, simplifying the path for the flow.

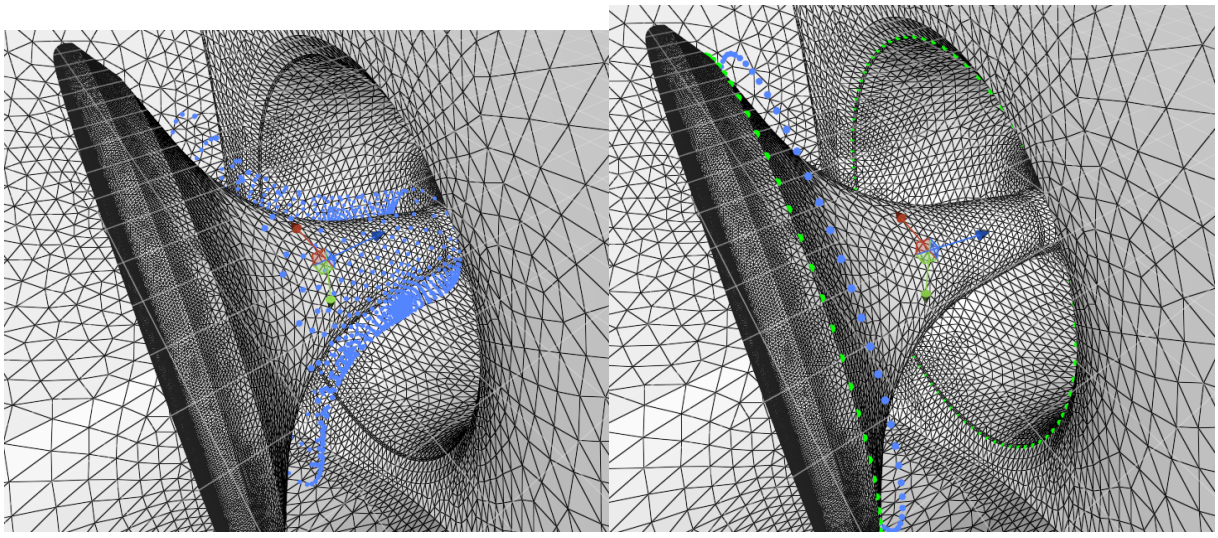


(a) External RBF Region (b) Internal RBF Region

Figure 5.10: Nodes belonging to the RBF Region

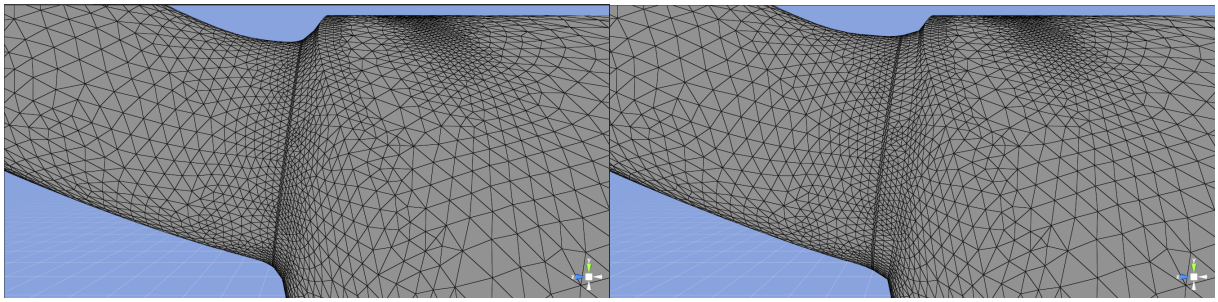


(a) External Constraints (b) Valve stem constraints



(c) Valve and duct seat constraints (d) RBF source 3 and 5

Figure 5.11: Nodes belonging to the RBF Sources



(a) Original geometry (b) Best morphed geometry, ID = 2 from Tab. 5.4

Figure 5.12: Comparison between the original and morphed geometries.

ID	Mass flow rate $\left[\frac{Kg}{s}\right]$	Discharge coefficient (single valve)	Percentage difference
0	0.144	0.598	\\
1	0.146	0.607	1.39 %
2	0.140	0.582	-2.07 %
3	0.136	0.565	-5.6 %
4	0.145	0.603	0.7 %
5	0.145	0.603	0.7 %

Table 5.5: Results for the first DOE table (Tab. 5.3) for the exhaust valve seat morphing, ID 0 refers to the original geometry.

ID	Mass flow rate $\left[\frac{Kg}{s}\right]$	Discharge coefficient (single valve)	Percentage difference
0	0.144	0.598	\\
1	0.1466	0.609	1.81 %
2	0.1470	0.611	2.08 %
3	0.1468	0.61	1.94 %
4	0.1458	0.606	1.25 %
5	0.1462	0.608	1.53 %
6	0.1452	0.603	0.83 %
7	0.1442	0.599	0.14 %
8	0.143	0.594	-0.69 %

Table 5.6: Results for the second DOE table (Tab. 5.4) for the exhaust valve seat morphing, ID 0 refers to the original geometry

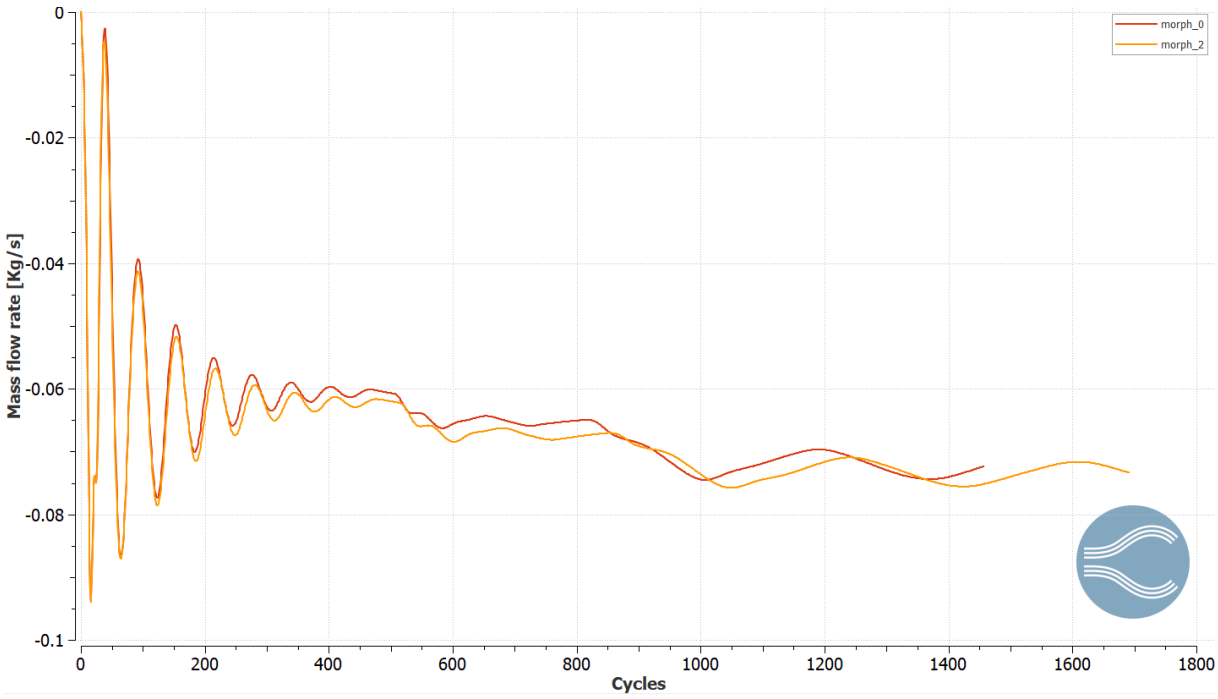
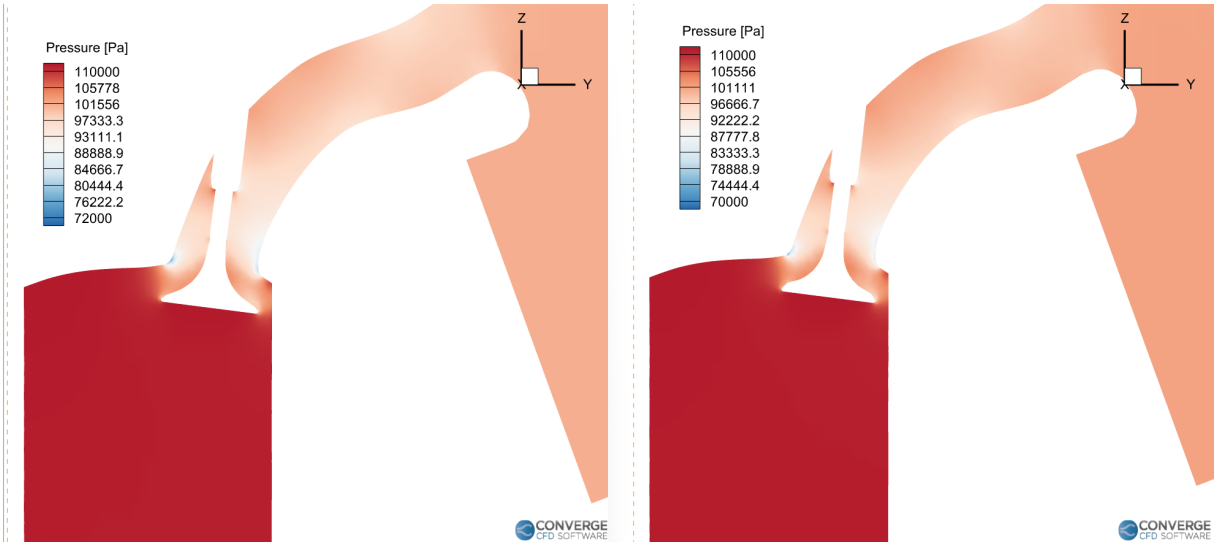
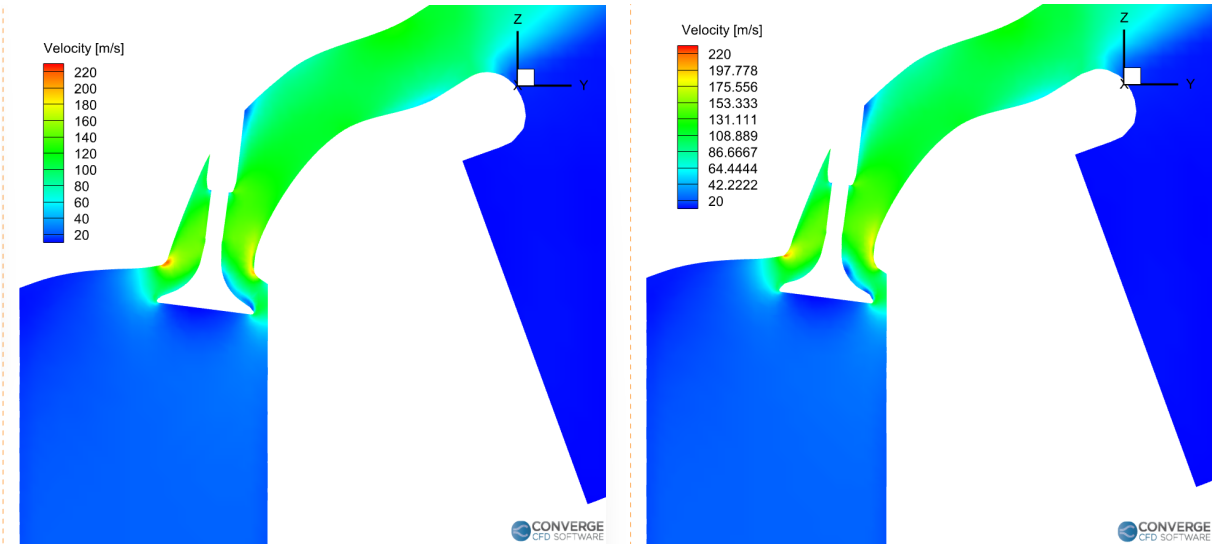


Figure 5.13: Converge line plot for the valve seat exhaust morphing: *morph 0* corresponds to the original geometry, *morph 2* to the second variant of Tab. 5.4



(a) *Morph 0* pressure

(b) *Morph 2* pressure



(c) *Morph 0* velocity

(d) *Morph 2* velocity

Figure 5.14: Contours of pressure and velocity fields for the valve seat morphing of the exhaust port original geometry and second variant of Tab. 5.4

Base grid $[mm]$	Mass flow rate $\left[\frac{Kg}{s}\right]$	Number of cells	Solution time [s]	Percentage difference
4	0.088	645000	1950	\\
4 (y+=30)	0.089	1383000	5444	1.14 %
2	0.091	3185000	11300	2.25%

Table 5.7: Engine head grid convergence, the number of cells refers to the displayed value once convergence is reached

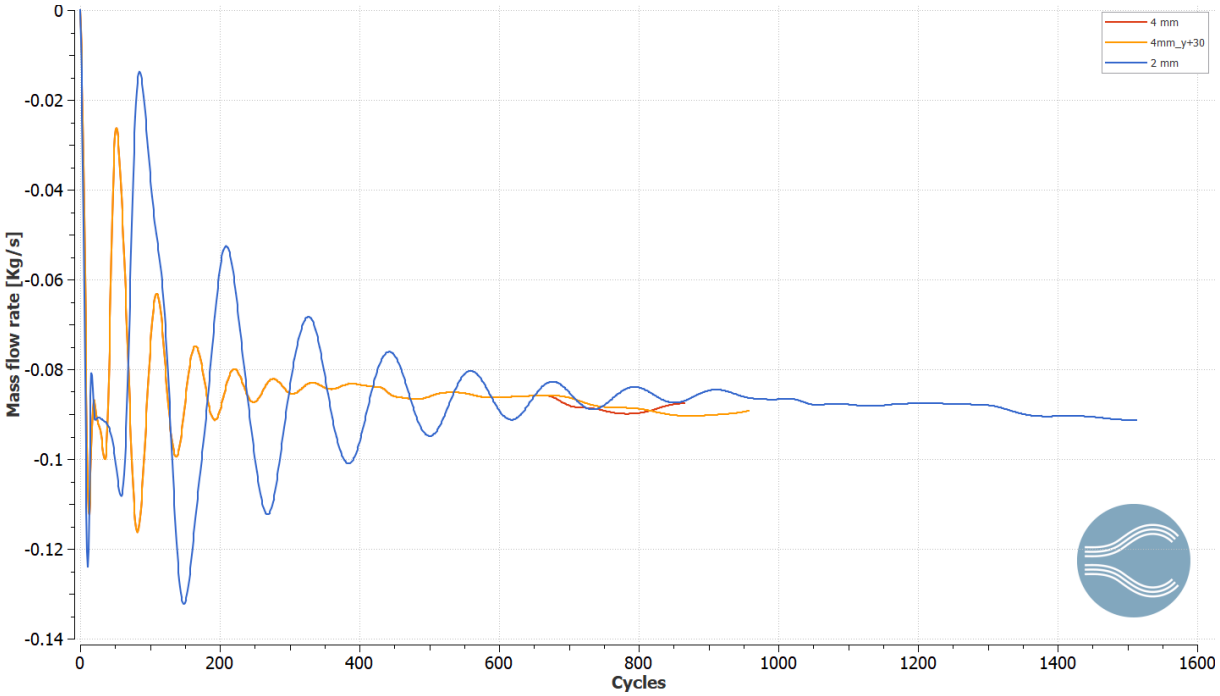


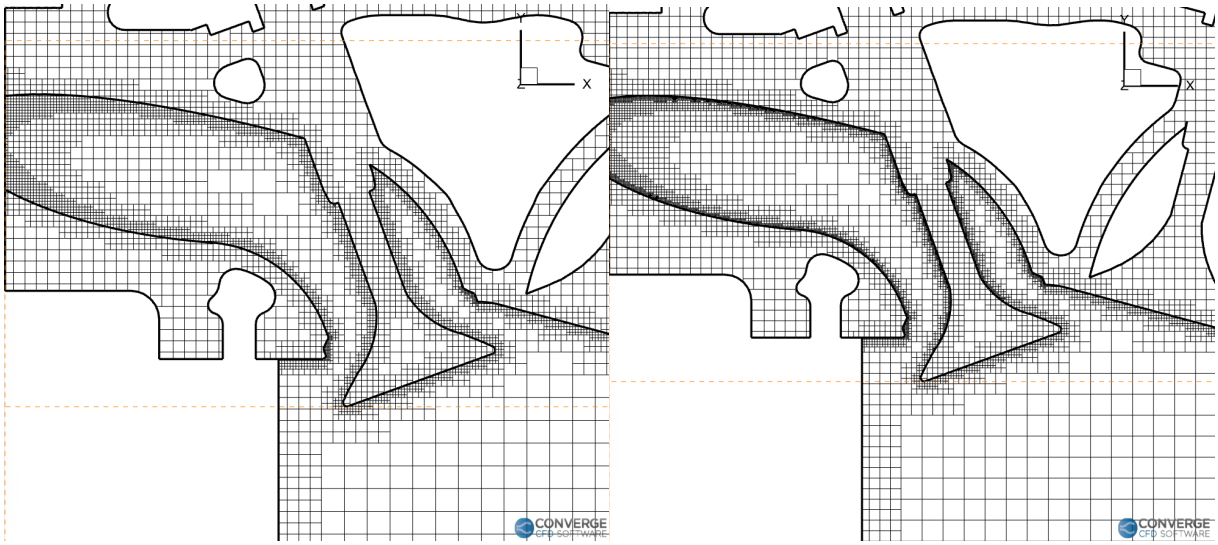
Figure 5.15: Exhaust grid convergence line plot

5.2 Engine head and CHT

More interesting results have been obtained for this case, probably due to the less optimized geometry of the original model, which left more room for performance improvements. This section follows a similar structure as the last one, separating the considerations between CFD and structural studies.

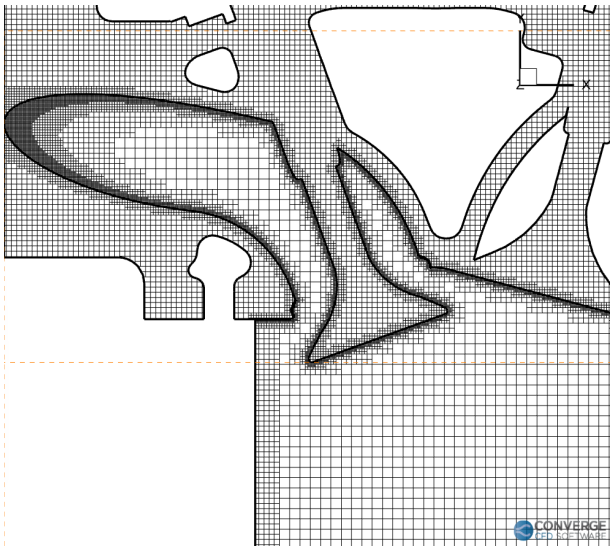
5.2.1 Converge grid convergence

Similarly to the last case, for the grid convergence test two base grid sizes of 4 mm and 2 mm have been evaluated, the local scaling results are taken from Tab. 4.8; for the 4 mm setup the effect of enforcing a y+ of 30 through the use of AMR has been explored as well; as it can be seen from Tab. 5.7 and Fig. 5.15 the coarser grid is already at convergence, Fig. 5.16 shows the three proposed grids for a slice plane which contains the axis of one of the valves.



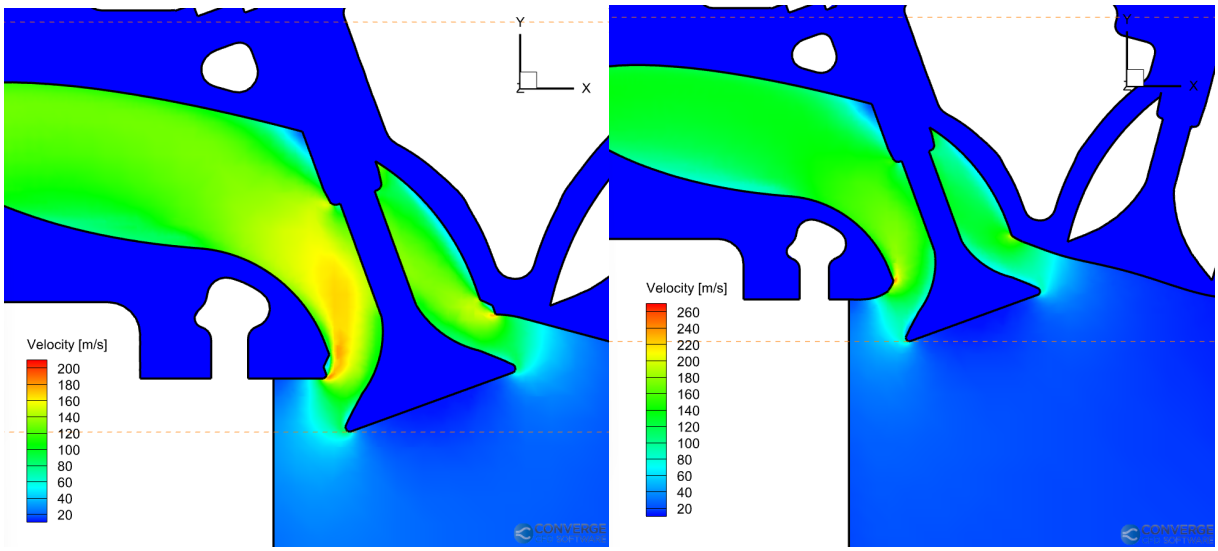
(a) 4 mm base grid

(b) 4 mm base grid and y+=30



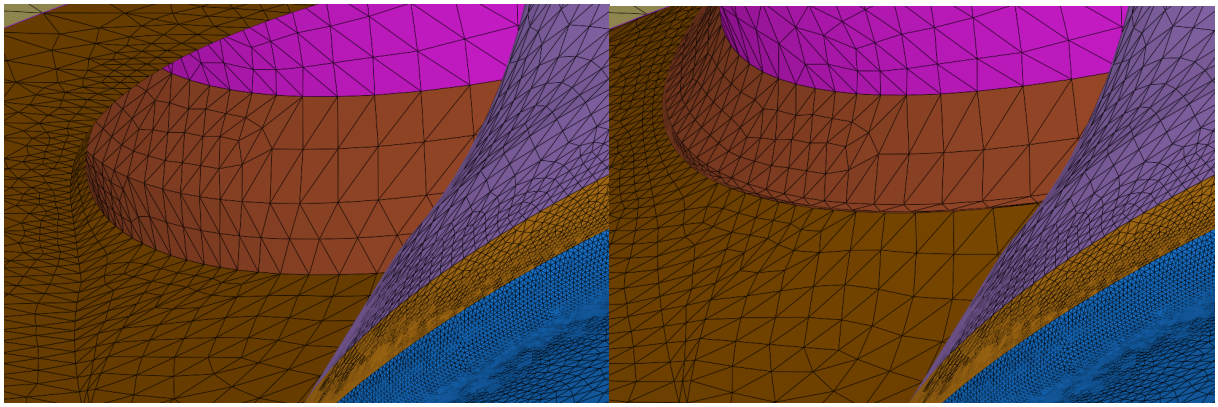
(c) 2 mm base grid

Figure 5.16: Grid representations from the mesh convergence test for the engine head



(a) Original geometry (b) No socket optimization

Figure 5.17: Contour velocities for socket removal comparison



(a) Original geometry (b) No socket optimization

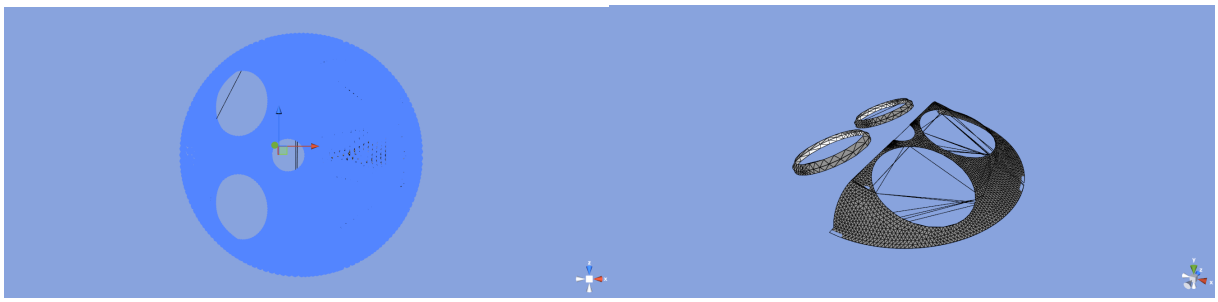
Figure 5.18: Geometry comparison for socket removal inside Converge Studio

5.2.2 Converge results discussion

Valve "socket" removal

Having already seen, from the previous examples, that the morphing of the duct brings lackluster results compared to a variation of the valve seats a similar strategy has been followed for this model. However as it can be seen in Fig. 5.17 the "socket" within which the valve enters once it is closed produces a recirculation bubble that heavily impacts the performances, in order to investigate the effects of its absence a specific morphing setup has been created. The geometry differences between this optimized case and the original one are shown in Fig. 5.18; the RBF Region, which in this setup involves only the cylinder head, and the CAD auxiliary entities for both fixed and moving RBF sources are reported in Fig. 5.19.

This shape variation alone brings a considerable jump in performance: the mean mass



(a) RBF Region (b) CAD auxiliary entities

Figure 5.19: RBF setup for socket removal. The cylinder head is the RBF Source, the big CAD geometry has been used to keep the intake side of the fixed while the exhaust side nodes are free to move

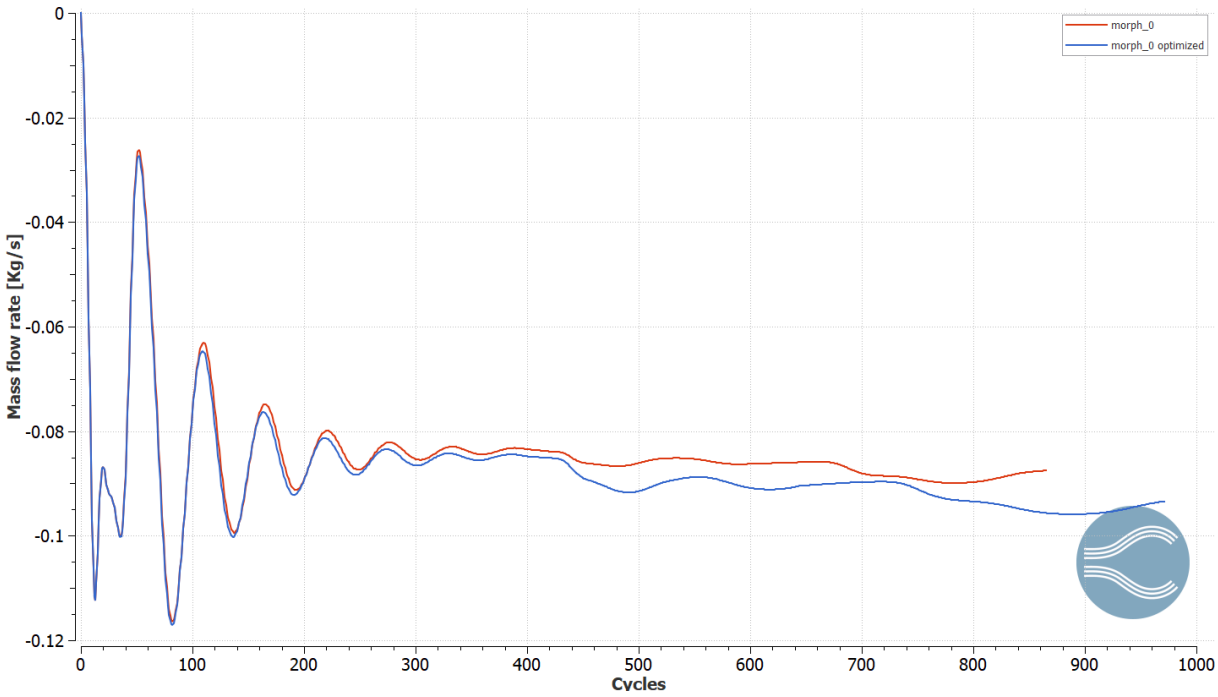


Figure 5.20: Converge line plot comparison for the socket removal morphing. *Morph 0 optimized* refers to the no socket geometry

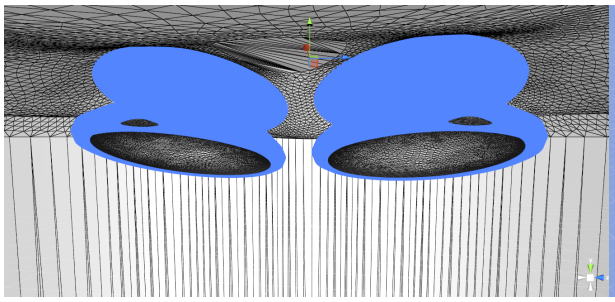
flow rate becomes $0.0948\frac{Kg}{s}$ with a percentage difference increase of about 7%; a line plot comparing the two cases is shown in Fig. 5.20.

Valve seat morphing

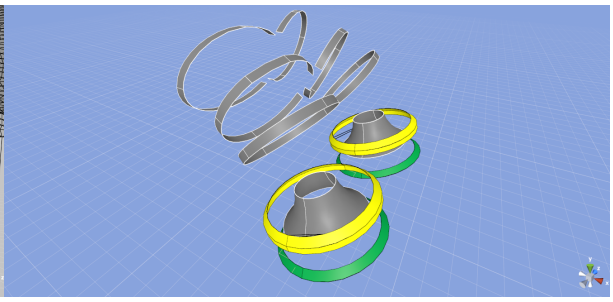
Using the previously obtained geometry as a new baseline, a DOE to explore the effects of the morphing on the valve seat has been performed. The setup is rather similar to the exhaust port one, the main difference is the need to use two different sets of transformations (possessing the same values) for the two valves, given that the local coordinates systems employed are not the same. In this setup the RBF region contains the valve stems and the exhaust duct itself, it is depicted in Fig. 5.21 together with the virtual geometries employed and an example of the displacement for the nodes. The transformations are,

ID	1	2	3	4	5	6	7
translation seat 2.T.Y	0.25	0.5	0.75	0	0	0	0
translation seat 1.T.Y	0.25	0.5	0.75	0	0	0	0
scaling seat 2.S.X	1	1	1	1.01	1.03	1.05	1.06
scaling seat 2.S.Z	1	1	1	1.01	1.03	1.05	1.06
scaling seat 1.S.X	1	1	1	1.01	1.03	1.05	1.06
scaling seat 1.S.Z	1	1	1	1.01	1.03	1.05	1.06

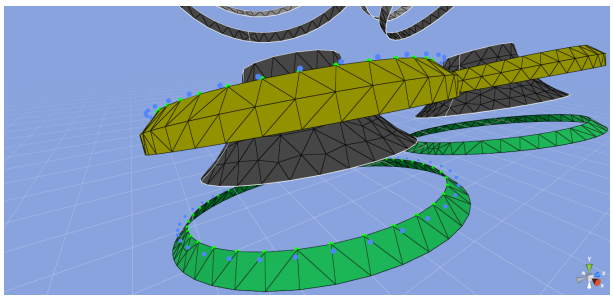
Table 5.8: DOE table for the valve seat morphing of the engine head



(a) Detail of the RBF Region



(b) CAD auxiliary entities



(c) Example of nodes displacements

Figure 5.21: RBF setup for the valve seat morphing of the engine head. The gray entities are the fixed ones

similarly as before, a combination of a translation and a scaling of the CAD models that trace the seats, accompanied by the necessary constraints. The values of the sources, respect their local coordinate system, are reported in Tab. 5.8.

The results for this DOE are visible in Tab. 5.9, they increase of performance after morphing is rather significant, in Fig. 5.22 a line plot depicts the mass flow rate for the original geometry, for the one obtained after the socket removal and for the best morphed one. The discharge coefficient from equation (2.15) was calculated with the curtain area from Tab. 4.6 and an upstream stagnation pressure of about 111175 Pa (obtained from the simulations). Fig. 5.23 show the pressure and velocity distribution maps for the original and best variation.

Duct morphing for stress concentrations

All the geometries explored until now where modified in regions far from the stress concentrations derived from morphing, so their structural impact is rather limited. A new

ID	Mass flow rate $\left[\frac{Kg}{s}\right]$	Discharge coefficient (single valve)	Percentage difference
0	0.0885	0.421	\\
0'	0.0945	0.449	6.78 %
1	0.095	0.452	7.34 %
2	0.0965	0.459	9.04 %
3	0.097	0.461	9.6 %
4	0.0945	0.449	6.78 %
5	0.096	0.456	8.47 %
6	0.097	0.461	9.6 %
7	0.0975	0.463	10.17 %

Table 5.9: Results for the valve seat morphing of the engine head. ID = 0 refers to the original geometry, ID = 0' to the the one obtained after the socket removal

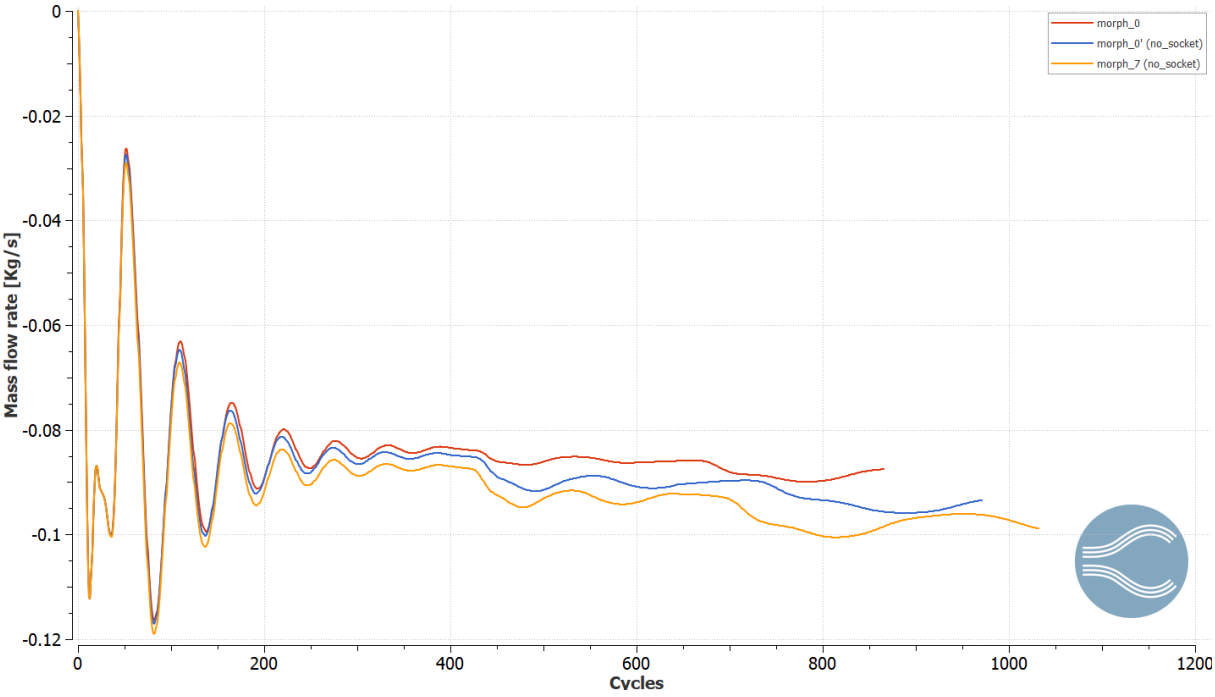
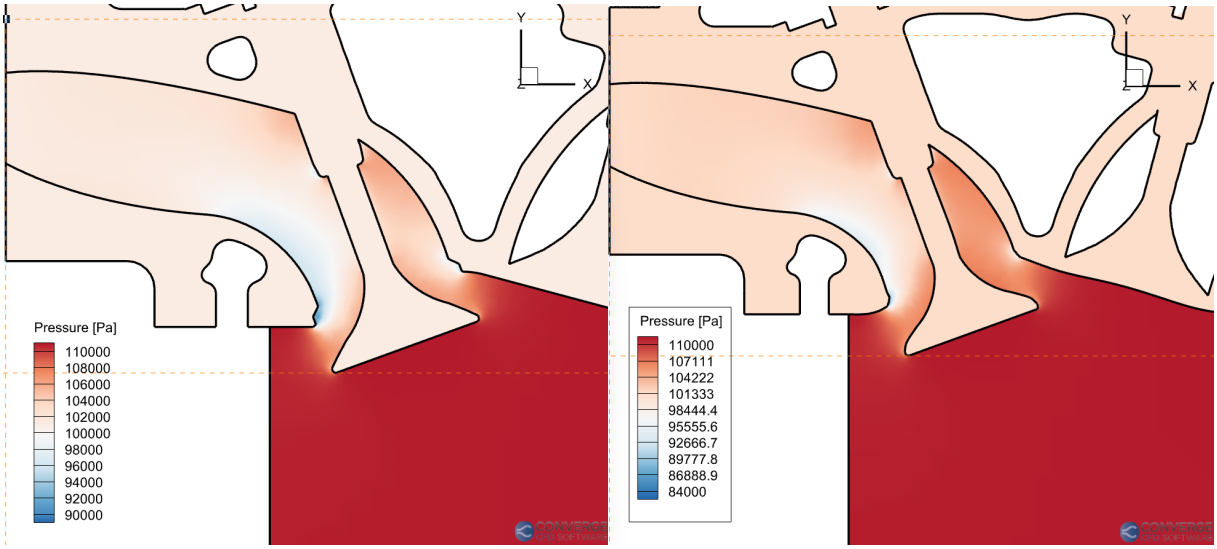
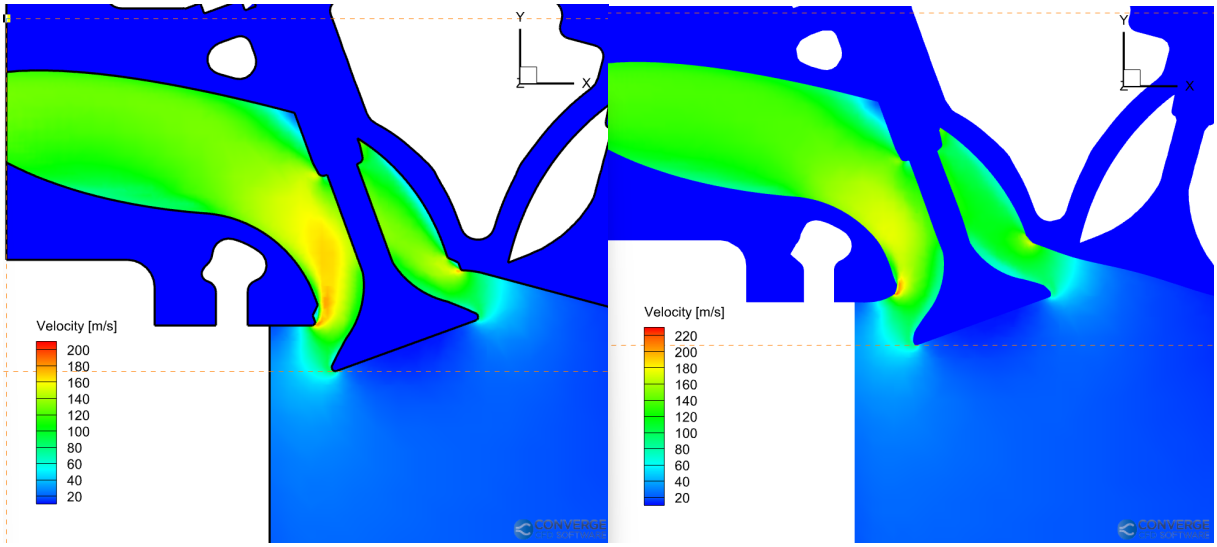


Figure 5.22: Converge line plot comparison for the valve seat morphing of the engine head. *Morph 0* refers to the original geometry

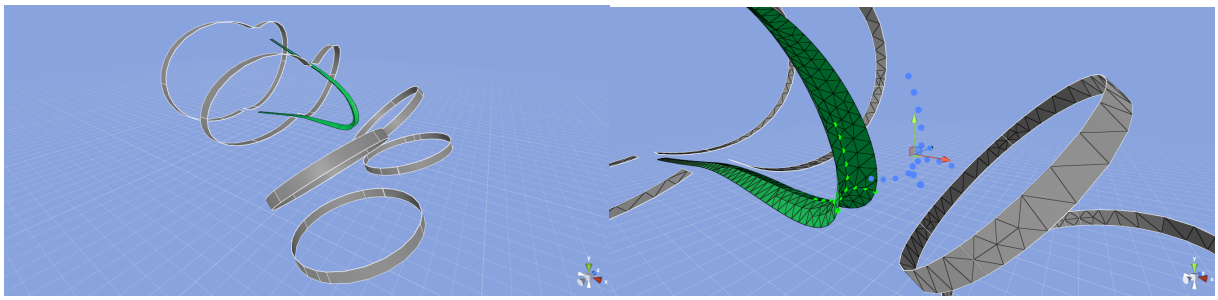


(a) *Morph 0* pressure (b) *Morph 7 (no socket)* pressure



(c) *Morph 0* velocity (d) *Morph 7 (no socket)* velocity

Figure 5.23: Contours of pressure and velocity fields for the valve seat morphing of the engine head



(a) RBF Region (b) CAD auxiliary entities

Figure 5.24: RBF setup for the duct morphing of the engine head. The gray entities are the fixed ones

ID	RBF Source 2.T.X	RBF Source 2.T.Y
1	-2	0
2	-4	0
3	-2	2
4	-4	2
5	2	0
6	4	0
7	2	2
8	4	2

Table 5.10: DOE table for the duct morphing of the engine head

DOE, for the duct intersection of the valve ports, has been created to see the effects on the fluid dynamics performances of the morphing destined to structural reinforcement and subsequently export the pressure and temperature maps. The RBF region contains only the duct surface nodes and the CAD entities for the sources are shown in Fig. 5.24 (a). A simple sequence of two translations is considered and they are imposed on two curves of the green auxiliary geometry in Fig. 5.24 (b), the DOE table is reported in Tab. 5.10. The morphing involves the translation of the curve of the duct along the x axis, to investigate how the concentrated stress may vary with the amount of material present in the region. As it can be seen in Fig. 5.25, the shape variation is practically irrelevant to the mass flow rate, thus allowing to improve the structural aspects without sacrificing fluid dynamics performance. A temperature contour is given in Fig. 5.26 for the original geometry, and is representative of the entire set of morphed variants as well. It is important to note that the most effected zone is the valve itself, however this component is not present inside the structural simulation.

5.2.3 Ansys grid convergence

Following the setup process described in chapter four a mesh convergence test has been performed for the Ansys Mechanical model as well, since the maximum stress are localized in a specific region only the local element size was modified, while the rest of the mesh

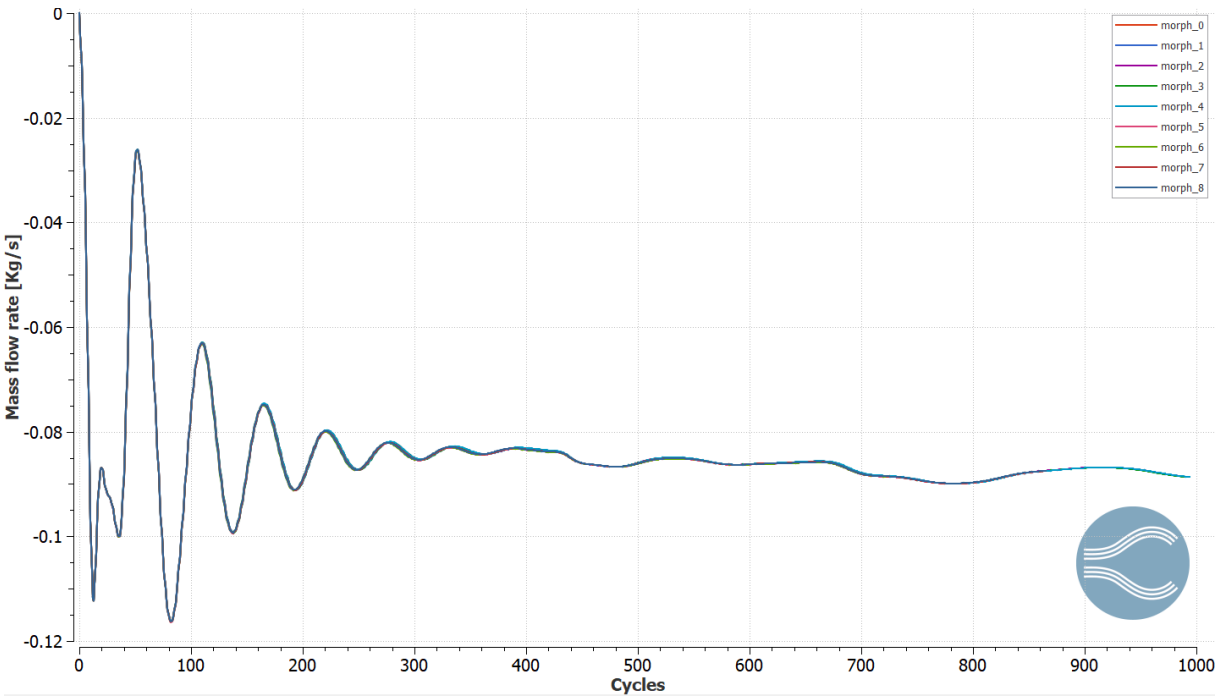


Figure 5.25: Converge line plot comparison for the duct morphing of the engine head. The plots for the different geometries are substantially identical

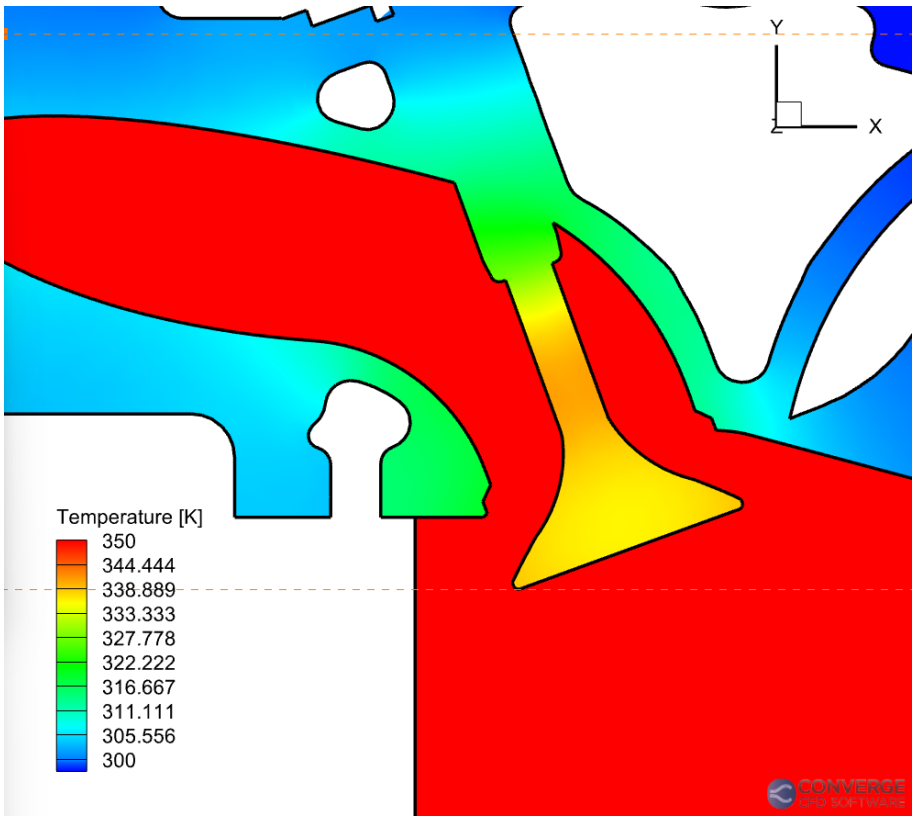


Figure 5.26: Temperature contour for the engine head. The maximum represented value is set to 350 K to capture the gradient inside the solid, in reality the fluid is at 500 K

Element size .[mm]	Mesh elements	Mesh nodes	Solution time [s]	Maximum stress [MPa]	Percentage difference
1	1613200	2426900	1379	131.12	\\
0.5	1624300	2443400	1444	143.46	9.41 %
0.25	1663800	2504100	1492	148.32	3.39 %
0.1	1971400	2971000	1384	148.26	-0.04 %
0.05	3024400	4569100	2534	148.28	0.01 %

Table 5.11: Ansys mesh convergence results.

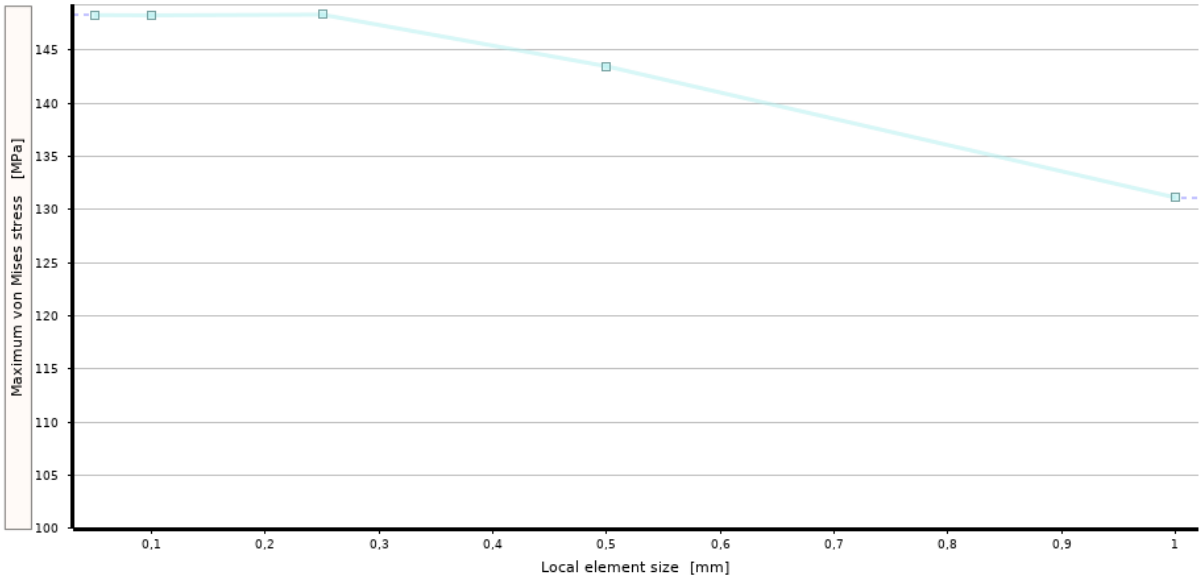
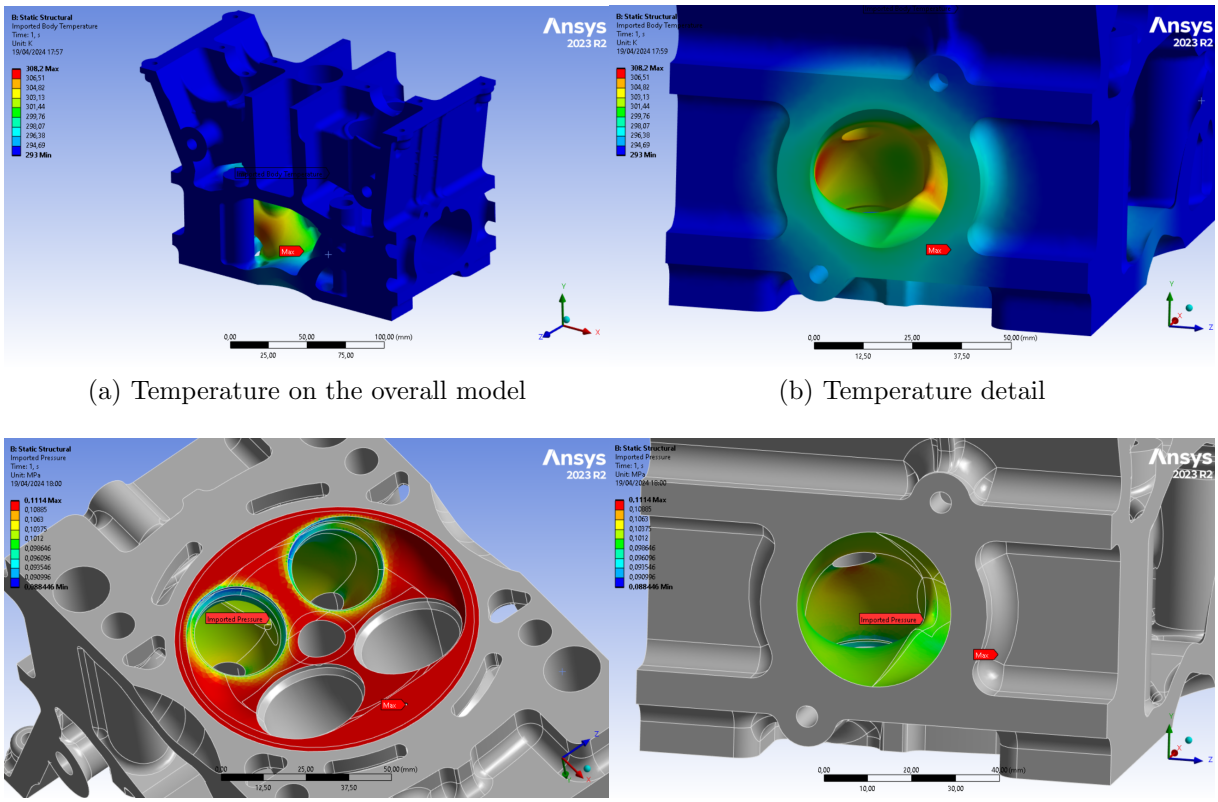


Figure 5.27: Stress convergence plot for element size of the structural mesh

remained untouched. In order to accomplish this a new Workbench project was created, employing the original geometry, the same loads and constraints as seen before but with the addition of the "Parameter Set" block. This component is employed to perform parametric studies inside the same Workbench project and, in this specific case, took the local mesh element size as the input parameters, the output is comprised of the maximum equivalent von Mises stress, the number of nodes and elements and the total elapsed time. The results are reported in Tab. 5.11 and the von Mises stress with respect to the local element size is plotted in Fig. 5.27. Since the refined zone is rather limited the solution time never becomes to excessive and the finer meshes could be utilized, however since the nodes of the region must undergo an interpolation operation during morphing, the overall elapsed time rapidly increases. For this study a local refinement of 0.5 mm has been selected, the stress values have practically converged and the overall simulation time remains manageable; the chosen mesh is shown in Fig. 4.20.

5.2.4 Ansys results discussion

The contours of the imported CFD loads described in the previous chapter are depicted in Fig. 5.28, they are computed and interpolated for each variant and are substantially



(a) Temperature on the overall model

(b) Temperature detail

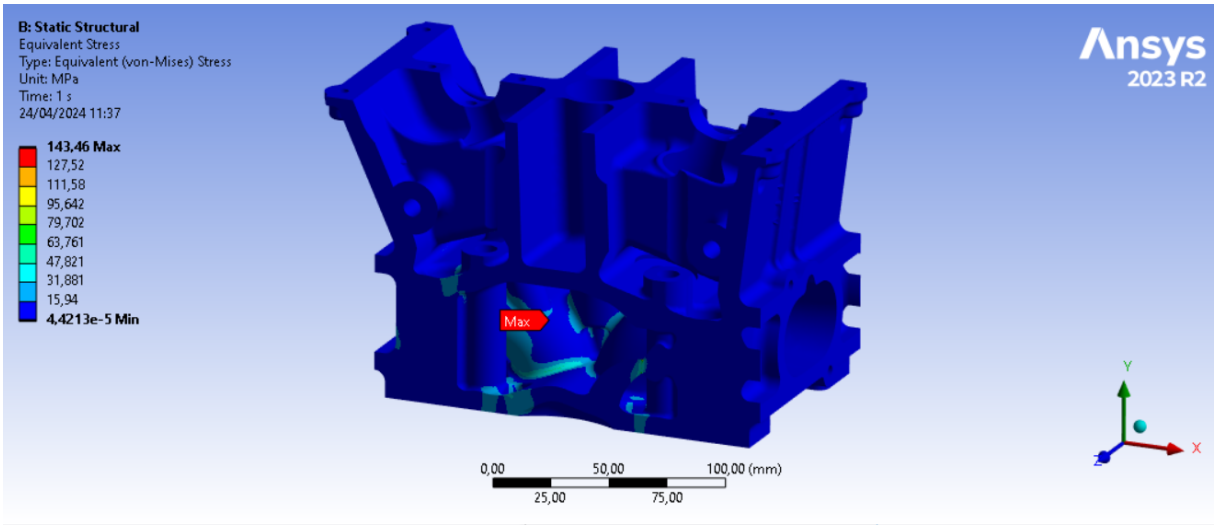
(c) Pressure on the cylinder head

(d) Pressure detail

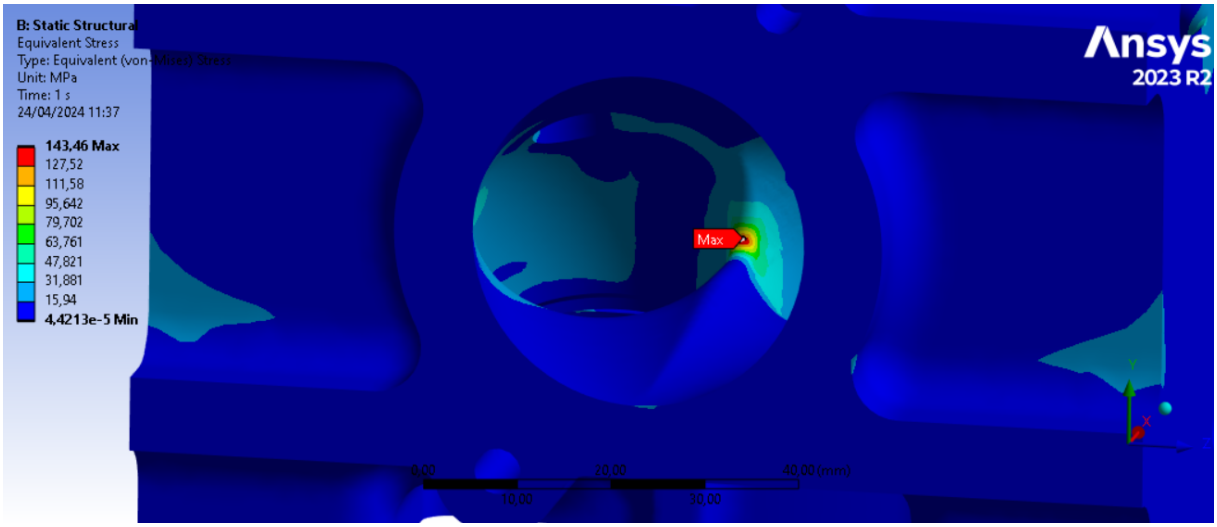
Figure 5.28: Contours of the pressure and temperature loads imported in Mechanical: The temperature acts on the entire body, the pressure only on the internal surfaces

similar for all the geometries. The resulting stresses acting on the original geometry are shown in Fig. 5.29, they are concentrated in the duct portion and the rest of the structure is not heavily loaded.

The corresponding values of stress obtained for the DOE table reported in Tab. 5.10 are shown in Tab. 5.12; a great improvement in some of the morphed variations is achieved, particularly for those that involve pushing inward (along a positive x direction) the curve of the duct. A comparison between the morphed region of the original mesh and best cases is provided in Fig. 5.30, the distribution of stress for the best variant is shown in Fig. 5.31. For this last geometry the maximum stress is located elsewhere than the considered zone, this is due to how the model was prepared for the analysis in [23]: the engine head initially involved four cylinders, three of which were then cut; the new concentration derives from a geometry feature (Fig. 5.32) resulting from the cuts and, since it does not represent a realistic situation, it can be ignored.



(a) Stress on the overall model

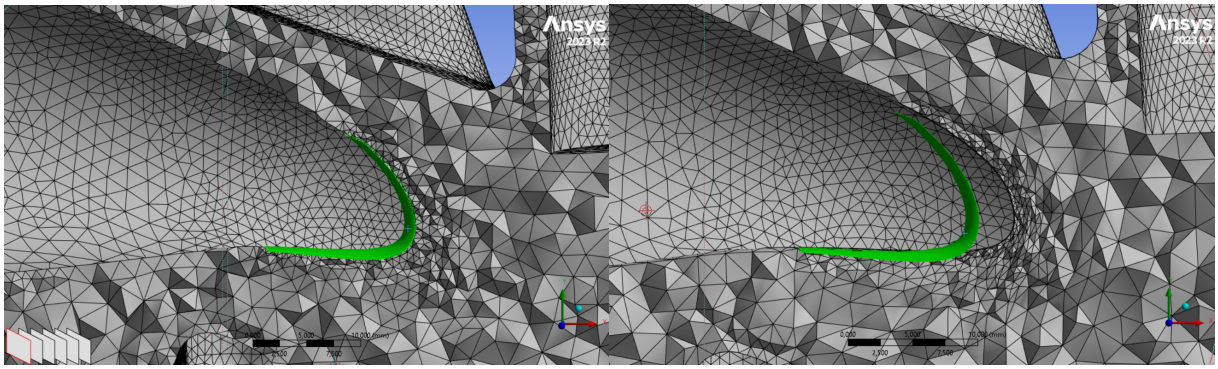


(b) Concentration of stress

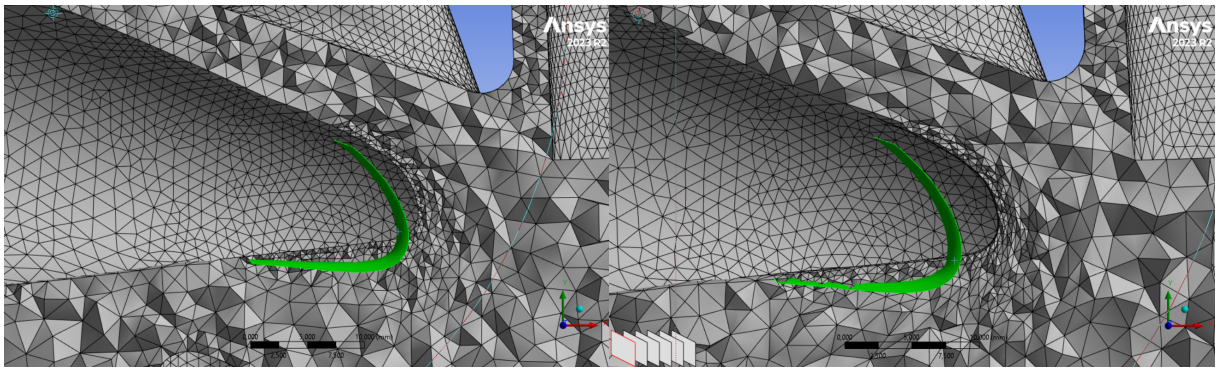
Figure 5.29: Stress distribution on the original geometry

ID	Maximum stress [MPa]	Percentage difference
0	140.40	\\
1	152.49	+8.61 %
2	153.10	+9.05 %
3	136.99	-2.43 %
4	139.48	-0.01 %
5	132.44	-5.67 %
6	121.20	-13.68 %
7	125.76	-10.43 %
8	118.11	-15.88 %

Table 5.12: Stress concentration results for the DOE of Tab. 5.10. ID = 0, refers to the original geometry



(a) Original geometry (b) Morph 6



(c) Morph 7 (d) Morph 8

Figure 5.30: Comparison of the localized morphed meshes along the mid plane section, the green surface shows the initial position of the duct curve

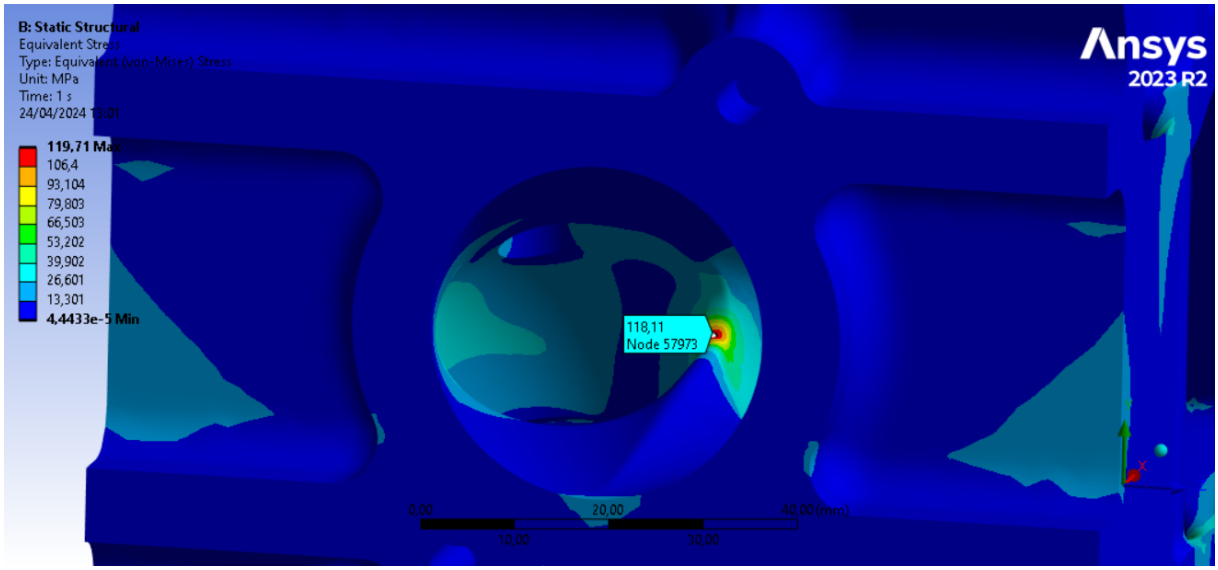
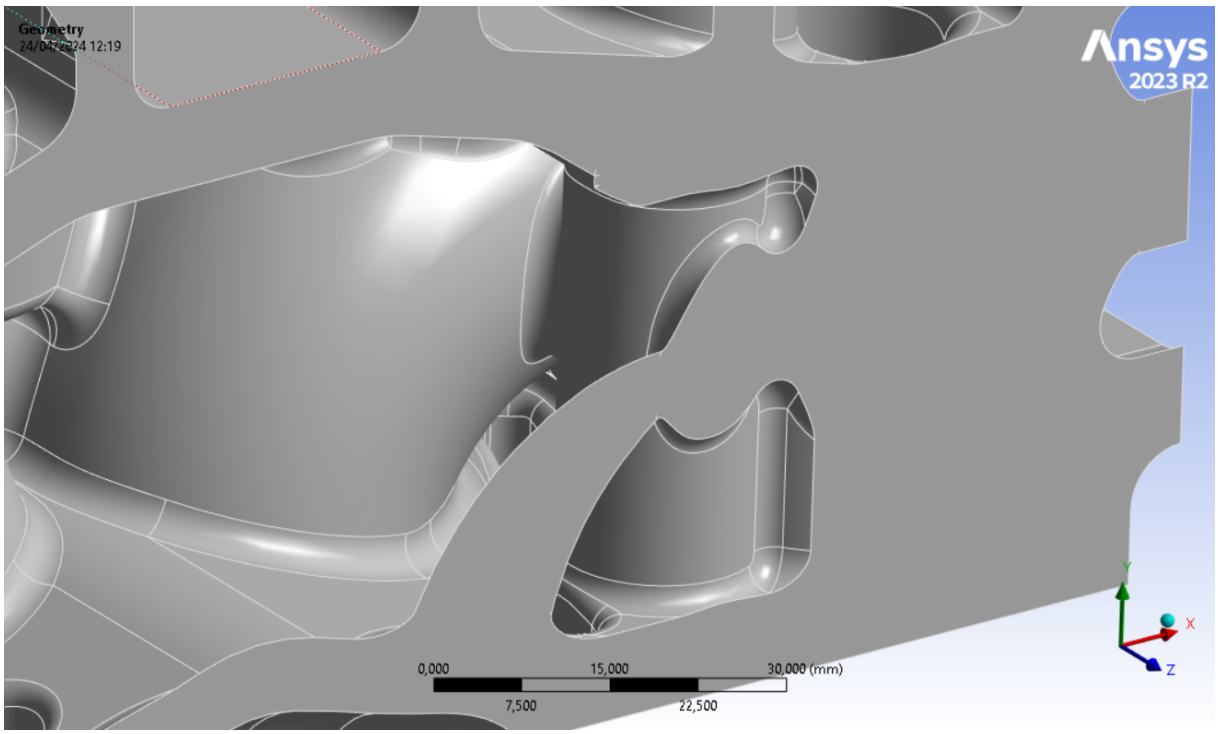
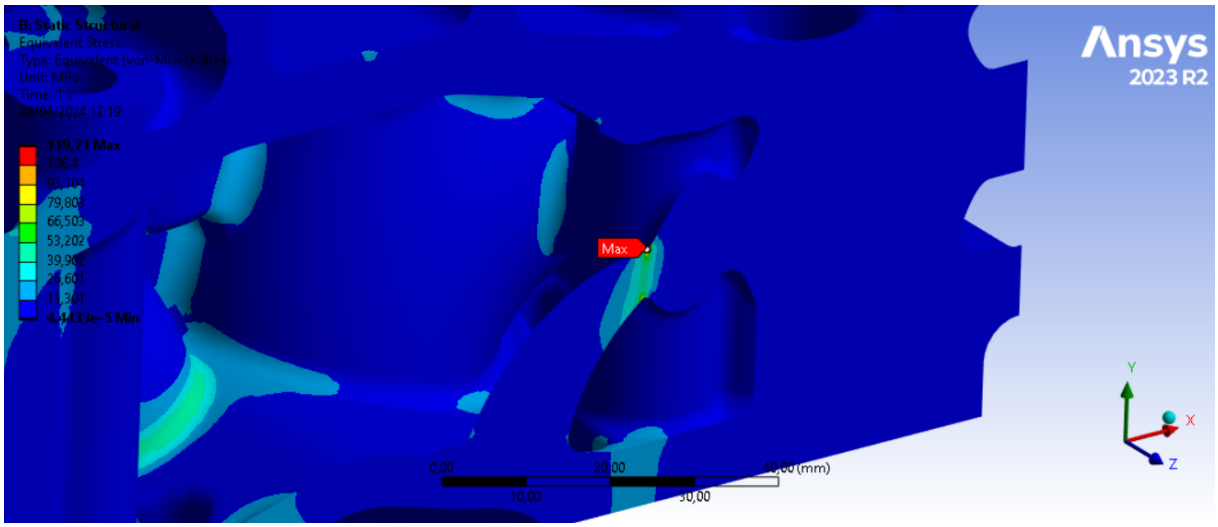


Figure 5.31: Concentration of stress for the best morphed variant



(a) Geometry



(b) Stress distribution

Figure 5.32: Negligible concentration of stress of *morph 7*, due to geometry preparation in [23]

Conclusions and future developments

The objective of this master's degree thesis has been the study and development of a multiphysics optimization workflow, with particular attention to automation and mesh morphing aspects. The scope of the work is well integrated with the modern industry reliance on numerical methods for validation and design, the optimization problem in particular is present in the waste and emissions reduction effort and in the improvement of already existing designs. The automotive and motor sport industries in particular are one of the cradle of studies in efficiency and performance increase; the spark that ignited core idea of this work was indeed born from the challenges of modern motor sport.

Thus this thesis main purpose has been the creation of a link between three software belonging to different cornerstones of engineering: fluid dynamics, structural mechanics and optimization; this was achieved through the implementation of a Python code, which also allowed for automation of the problem. The resulting workflow has been tested via CFD simulations on two exhaust port geometries, one of which also concerned the conjugate heat transfer problem and consequent effects on structural mechanics; the mesh morphing has been achieved leveraging the radial basis function theory.

A guide for the preparation of the models in the different programs has been provided, together with possibility of automating all the necessary simulations for an infinite number of variants. In fact, after having properly prepared the setup for the analyses of interest and having manually specified the mesh morphing parameters for each geometry variant, the proposed Python code allows for a one-click optimization.

The first exhaust port optimization has given tepid results; the main factors behind this are probably the already rather optimized shape and high number of functional constraints, which limits the scope of the morphing actions. Nonetheless an increase in performance of a couple percent point has been achieved, which for the field of application is not a negligible result. Additional studies leveraging the adjoint method have been performed to validate what have been achieved and their results have been comparable to those obtained here.

The engine head model case has presented more favorable results: the fluid dynamics performances have been greatly improved over the base geometry and the structural resistance has increased as well. Furthermore, the shape variations that reduce the stress concentrations have no influence over the flow rate performance allowing for a joint opti-

mization without having to resort to a compromise.

As it has already been said, the main objective of the thesis was to provide an automation workflow to multiphysics optimization, this is reflected as well in the somewhat simplistic nature of the presented analyses: the internal combustion engine field is heavily concerned with time dependent problems whereas the simulation hereby performed are all stationary. Future studies built upon this foundation could concern themselves with combustion analysis and its effect on engine structures, employing the automation developed here. Regarding the DOE aspects of the work, which have been manually prepared, a further integration with specialized software for automated management of the optimization would greatly enhance the workflow both in terms of accessibility for the user and performance gain, since a program would be more precise than a human in identifying promising designs.

Adjoint studies inside Converge may also prove extremely valuable for optimization purposes: they could provide a guiding line for designs of experiment such as those executed in this work, but the adjoint method may also provide extremely competitive results on its own especially considering its intrinsic affinity with mesh morphing for shape optimization, as it has been already demonstrated in previous studies.

In the foreseeable future of ICEs the employment of hydrogen as a combustible is gaining popularity, this new energy vector presents challenges diametrically opposed to those encountered in other fuels, especially in the combustion phase. Mesh morphing might prove a valuable tool to investigate such new problems, in particular for what concerns the intake port, cylinder head and valve shape optimization.

Finally it is important to note that despite the applications explored in this work belong to the automotive industry, this workflow is easily adapted to any field concerned with the coupling of computational fluid dynamics and thermo-structural analyses.

Appendix A:
Python code for automation

This appendix contains the Python code developed to morph and simulate using CONVERGE CFD and Ansys Mechanical. The presented code performs the complete work: morphing, CFD and then FEM simulations; it is possible to extract fractions of the code in order to enact single tasks. After having created the models for RBF-viewer, CONVERGE, Ansys Mechanical and the DOE table as described in chapter 4 one needs to modify only the indicated directories and the code will perform the entire work.

The RBF functions contained in the "rbfmorph" package used in this work have been originally developed by Marco Camponeschi.

```
import os
import json
import shutil
import time
import glob
import rbfmorph as rbf
import pandas as pd
import numpy as np
import subprocess
import functions as fun

# Modify the strings in line 20 and 22 for the Converge and Ansys
                                work directories
# If necessary modify the directories on line 27,31 and the command
                                prompt on line 29 (following the
                                Converge getting started guide)

# The RBF file must contain two models:
# In the first import the STEP files
# In the second import only the DAT mesh

# Enter the work directory for Converge
converge_work_dir = r"..."
# Enter the work directory fon Ansys Workbench
wb_work_dir = r"..."
```

```
# Enter the drive in which Converge installed
drive = "C:"

# Enter the directory to the executable file for Converge
converge_exe_path = r"...\Convergent_Science\CONVERGE\3.1.9\bin\
                                intelmpi\converge.exe"
# Enter the valid command prompt to run the simulation
run_command = "mpiexec -n 4 converge.exe -l super"
# Enter the directory to the executable file for Ansys Workbench
wb_exe_path = r"...\ANSYS 2023 R2\v232\Framework\bin\Win64\RunWb2.
                                exe"

setup_dir = rf"{converge_work_dir}\setup" # Defines the path of the
                                setup folder directory
rbf_filepath = glob.glob(os.path.join(converge_work_dir, "*.rbf"))
                                # Creates a list of the .rbf files
rbf_filepath = rbf_filepath[0]  # Selects the first (and only) one
doe_table_path = glob.glob(os.path.join(converge_work_dir, "*.xlsx")
                                ) + glob.glob(os.path.join(
                                converge_work_dir, "*.xls"))  #
                                Creates a list of the Excel files
doe_table_path = doe_table_path[0] # Selects the first (and only)
                                one
data_frame = pd.read_excel(doe_table_path)
print(data_frame)
ID_list = data_frame["ID"].tolist()  # Creates a list with the IDs
                                of the morphed geometry
header = data_frame.columns.values.tolist()  # Creates a list with
                                the elements of the header
header = header[1:]  # Subtracts the first label of the header (ID)
                                (can be used to skip the "fixed" RBF
                                 Sources)

for ID in ID_list:  # Goes through the rows of the Excel file
  exported_filepath = rf"{converge_work_dir}\morph_{ID}.dat" #
                                Identifies the current .dat file of
                                the morphed geometry

  for source in header:  # Goes through the columns of the Excel
                                file
    source_list = source.split(".")  # Creates a list with the
                                elements in a cell
    rbf_source_name = source_list[0]
    rbf_transformation_type = str.lower(source_list[1])
    rbf_transformation_direction = str.lower(source_list[2])
    cell_value = data_frame[source]  # Select the cells under the
                                current   element of the header
```

```
print(f"Working on: ID = {ID}, Source:  {rbf_source_name}.{
                            rbf_transformation_type}.{
                            rbf_transformation_direction}")

with open(rbf_filepath, 'r') as file:
data = json.load(file)

# Checks if the current RBF Source exists in the .rbf file
if 'scene' in data and 'rbf_tree' in data['scene'] and 'rbf' in
                              data['scene']['rbf_tree']:
rbf_nodes = data['scene']['rbf_tree']['rbf']

if not any(node['name'] == rbf_source_name for node in rbf_nodes
                            ):
raise ValueError(f"ERROR: Source: '{rbf_source_name}' not found
                            in the  .rbf file.")

for node in data['scene']['rbf_tree']['rbf']:  # Goes through
                            the nodes   in the .json file

  # Checks if the source in the database is present, if it is
                            the   correct one and then
                            overwrites the values
  if node['name'] == rbf_source_name:

  if rbf_transformation_type in ["translation", "t", "scaling",
                            "s"]:
  node[rbf_transformation_direction] = float(cell_value[ID - 1])

    elif rbf_transformation_type in ["rotation", "r"]:
    node["angle"] = np.pi * float(cell_value[ID - 1]) / 180

      if rbf_transformation_direction == "x":
      node["axis"] = 0

      elif rbf_transformation_direction == "y":
      node["axis"] = 1

      elif rbf_transformation_direction == "z":
      node["axis"] = 2

  else:
  raise KeyError(f"ERROR: The name for the type of
                            transformation is not correct -> {
                            rbf_transformation_type}")

with open(rbf_filepath, 'w') as file:
```

```
        json.dump(data, file, indent=2)


        # 1. Initialization.
        # 2. Load JSON file modified.
        # 3. Perform morphing.
        # 4. Export morphed mesh in DAT format.

        geometries = rbf.POINTER(rbf.INT64)

        rbf.init()
        base_path, extension = os.path.splitext(bytes(rbf_filepath, 'utf
                                    -8'))
        rbf.load(base_path)
        rbf.morph()
        models = rbf.get_models()
        meshes = rbf.get_model_mesh_nodes(models[1])
        rbf.morph()
        rbf.export(bytes(exported_filepath, 'utf-8'), models[1], meshes,
                                     1, 0)
        rbf.clear()

    # Creates a folder for the morphed geometry and copies the input
                                    files from   the setup folder
    morph_dir = rf"{converge_work_dir}\morph_{ID}"
    shutil.copytree(setup_dir, morph_dir)

    # Overwrites the original geometry .dat file with the current
                                    morphed   geometry .dat file with
                                    the name "surface.dat"
    dat_to_overwrite_path = rf"{morph_dir}\surface.dat"
    shutil.copy(exported_filepath, dat_to_overwrite_path)

    # Copies the executable file in the simulation folder
    shutil.copy(converge_exe_path,morph_dir)

print("*** Finished morphing geometries ***")

for ID in range(1, len(ID_list) + 1): # Goes through the folders of
                                    the morphed geometries
    start_time = time.time()
    # Selects the directory to the current simulation
    sim_dir = rf"{converge_work_dir}\morph_{ID}"

    # Opens the command prompt, changes the directory and runs
    subprocess.Popen(f"cmd.exe /c {drive} && cd /d {sim_dir} && {
                                    run_command}",
```

```
shell=True, stdin=subprocess.PIPE, stdout=subprocess.PIPE, stderr=
                                subprocess.PIPE)
time.sleep(20)

# Checks if the current simulation has started or not
if fun.is_converge_running():
print(f"Converge simulation for morph_{ID}")

else:
print(f"*** WARNING: Unable to simulate morph_{ID} ***")

# Checks if the simulation is still running
while fun.is_converge_running():
time.sleep(10)

time.sleep(10)
# Creates a .pts file with the original coordinates and the
                                displacements after morphing
original_points_path = rf"{converge_work_dir}\morph_0\surface.dat"
morph_points_path = rf"{sim_dir}\surface.dat"
difference_path = rf"{sim_dir}\difference.pts"
fun.difference_points(original_points_path, morph_points_path,
                                difference_path)

# Creates a filtered .pts file for a faster interpolation iside
                                Ansys
interval_x = (-56, -26)  # Interval for X coordinate [mm]
interval_y = (22, 48)    # Interval for Y coordinate [mm]
interval_z = (-100, -80)  # Interval for Z coordinate [mm]
filtered_filepath = rf"{sim_dir}\filtered_difference.pts"
fun.filter_difference_points(difference_path, filtered_filepath,
                                interval_x, interval_y, interval_z)

# Creates a map of the temperature (for the solid region) and the
                                pressure (for the fluid region)
h5_output_path = rf"{sim_dir}\outputs_original\output"
temperature_mapping_path = rf"{sim_dir}\temp_map.csv"
pressure_mapping_path = rf"{sim_dir}\pres_map.csv"
files = os.listdir(h5_output_path)  # Gets a lis of the the files
                                in the output folder
files.sort(key=lambda x: os.path.getmtime(os.path.join(
                                h5_output_path, x)), reverse=True)
                                # Rarranges them so that the first
                                is the last modified
most_recent = files[0] if files else None  # Selects the first one
h5_path = os.path.join(h5_output_path, most_recent) # Creates a
                                path fot the last created output
```

```
                                    files
    fun.temp_mapping(h5_path, temperature_mapping_path)
    fun.pres_mapping(h5_path, pressure_mapping_path)

    end_time = time.time()
    elapsed_time = end_time - start_time
    print(f"Finished simulation for morph_{ID}, elapsed time: {int(
                                    elapsed_time)} s")
    print("-----------------------------------------------------")
print("*** Converge simulations completed ***")

# Defines the path of the Workbench setup
wb_setup_path = rf"{wb_work_dir}\setup"
# Defines the path for the Workbench script journal
update_command_path =rf"{wb_work_dir}\update_command.wbjn"
update_command = f"-x -r {update_command_path}"
# Defines the path for the mapped data that will be used in
                                    Workbench
current_temp_map_path = rf"{wb_work_dir}\current_map\temp_map.csv"
current_pres_map_path = rf"{wb_work_dir}\current_map\pres_map.csv"
current_displacements_path = rf"{wb_work_dir}\current_displacements\
                                    filtered_difference.pts"

for ID in range(1, len(ID_list) +1 ):  # Goes through the folder for
                                     the morphed geometries for the
                                    Ansys simulations
    # Creates a folder for the morphed geometry and copies the
                                    original files
    morph_dir = rf"{wb_work_dir}\morph_{ID}"
    shutil.copytree(wb_setup_path, morph_dir)
    # Selects the current workbench file for the structural simulation
                                     of the morphed geometry
    simulation_path = f"-f {morph_dir}\setup.wbpj"

    # Overwrites the current mapped data with those of the
                                    corresponding morph
    temp_map_path = rf"{converge_work_dir}\morph_{ID}\temp_map.csv"
    shutil.copyfile(temp_map_path, current_temp_map_path)
    pres_map_path = rf"{converge_work_dir}\morph_{ID}\pres_map.csv"
    shutil.copyfile(pres_map_path, current_pres_map_path)

    # Overwrites the current displacements .pts file (only for the
                                    morphed geometries)
    displacements_path = rf"{converge_work_dir}\morph_{ID}\
                                    filtered_difference.pts"
    shutil.copyfile(displacements_path, current_displacements_path)
```

```
    print(f"Starting Ansys Mechanical simulation for morph_{ID}")
    start_time = time.time()
    ansys_process = subprocess.Popen("%s %s %s" % (wb_exe_path,
                                    simulation_path, update_command))
    ansys_process.wait()

    end_time = time.time()
    elapsed_time = end_time - start_time
    print(f"Finished simulation for morph_{ID}, elapsed time = {int(
                                    elapsed_time)} s")
    print("------------------------------------------------------------")
print("*** Ansys simulations completed ***")
```

Appendix B:
List of utility Python functions

This appendix contains the functions used inside the main code, written separately to improve readability.

```
import psutil
import h5py
import pandas as pd
pd.set_option("mode.copy_on_write", True)

# This file contains all the functions created for the "
                                morph_and_run.py" script


def is_converge_running():
  """Checks if converge.exe is running
  Returns: True if it is running, False otherwise

  Args: Process (str): Name of the process
  """
  for process in psutil.process_iter(['name']):
    if process.info['name'] == "converge.exe":
      return True
  return False


def difference_points(original_points_path, morph_points_path,
                                difference_path):
  """Creates a .pts file with the coordinates of the nodes of the
                                original points
  and the displacements after morphing

  Args:
    original_points_path (str): Path of the original points file
    morph_points_path (str): Path of the morphed points file
    difference_path (str): Path of the output file
  """
```

```
    # Opens the .dat file and reads the number of nodes and triangles
    with open(original_points_path, "r") as file:
      first_line = file.readline().split(" ")
      triangle_number = int(first_line[2])

    # Creates a dataframe for the original .dat file
    original_df = pd.read_csv(original_points_path, delim_whitespace=
                                True, header=None, index_col=None,
                                skiprows=1)
    # Creates a dataframe only for the node coordinates and adds an
                                header
    original_df_node = original_df.iloc[:-triangle_number]
    original_df_node.columns = ["Node", "X_o", "Y_o", "Z_o"]

    # Creates a dataframe for the morphed .dat file
    morph_df = pd.read_csv(morph_points_path, delim_whitespace=True,
                                header=None, index_col=None,
                                skiprows=1)
    # Creates a dataframe only for the node coordinates and adds an
                                header
    morph_df_node = morph_df.iloc[:-triangle_number]
    morph_df_node.columns = ["Node", "X_m", "Y_m", "Z_m"]


    # Creates a new dataframe and a file for coordinates difference
    difference_df_node = pd.DataFrame({
      "Node" : original_df_node["Node"],
      "delta_X" : morph_df_node["X_m"] - original_df_node["X_o"],
      "delta_Y" : morph_df_node["Y_m"] - original_df_node["Y_o"],
      "delta_Z" : morph_df_node["Z_m"] - original_df_node["Z_o"]})

    # Drops the first columns from the two dataframes
    original_df_node.drop(original_df_node.columns[0], axis=1, inplace
                                =True)
    difference_df_node.drop(difference_df_node.columns[0], axis=1,
                                inplace=True)

    # Creates a new datframe merging the other two columnwise
    result_df = pd.concat([original_df_node, difference_df_node], axis
                                =1)

    result_df.to_csv(difference_path, sep=" ", index=False, header=
                                False, mode="w")


def filter_difference_points(difference_filepath, filtered_filepath,
                                 interval_x, interval_y, interval_z)
```

```
                                    :
    """Takes a .pts file and filters it according to the intervals
                                    given

    Args:
      difference_path (str): Path to the complete .pts file
      filtered_filepath (str): Path to the new filtered.pts file
      interval_x (tuple): X coordinates range [mm]
      interval_y (tuple): Y coordinates range [mm]
      interval_z (tuple): Z coordinates range [mm]
    """
    df = pd.read_csv(difference_filepath, header=None) # Reads the
                                    original .pts file
    df = df.map(lambda x: [float(val) for val in x.split()]) #
                                    Transforms the dataframe in rows of
                                    lists, leaving the structure the
                                    same
    filtered_rows = []

    # Iterate through each row of the dataframe and filters the rows
                                    so that only those that are in the X
                                    , Y and Z range are retained
    for index,row in df.iterrows():
      if interval_x[0] <= df[0][index][0]*1000 <= interval_x[1] and \
      interval_y[0] <= df[0][index][1]*1000 <= interval_y[1] and \
      interval_z[0] <= df[0][index][2]*1000 <= interval_z[1]:
        filtered_rows.append(df[0][index][:])

    # Creates a new dataframe with the retained rows and create a .pts
                                     file
    filtered_difference_points = pd.DataFrame(filtered_rows, columns=
                                    None)
    filtered_difference_points.to_csv(filtered_filepath, sep = " ",
                                    header = False, index=False)


def temp_mapping(h5_file, csv_file):
    """Allows to export in a .pts file the data
    for the temperature and the x,y,z coordinates of the
    center of every grid cell contained in a .h5 file

    Args:
      h5_file (str): Path of the .h5 file
      pts_file (str): Path of the .pts file
    """
    # Open the .h5 file
    with h5py.File(h5_file, 'r') as h5:
```

```
    # Extracts the data

    temp_data = h5["STREAM_01"]["CELL_CENTER_DATA"]["TEMPERATURE"][:
                                ]  #  Temperature [K] dataset
    xcen_x_data = h5["STREAM_01"]["CELL_CENTER_DATA"]["XCEN_X"][:]
                                # X   coordinate [m] dataset
    xcen_y_data = h5["STREAM_01"]["CELL_CENTER_DATA"]["XCEN_Y"][:]
                                # Y   coordinate [m] dataset
    xcen_z_data = h5["STREAM_01"]["CELL_CENTER_DATA"]["XCEN_Z"][:]
                                # Z   coordinate [m] dataset

  num_cells = temp_data.shape[0]

  with open(csv_file, 'w') as csv:
    for i in range(num_cells):  # Writes: "temperature" "X
                                coordinate" "Y   coordinate" "Z
                                coordinate" on each line
      csv.write(f"{temp_data[i]} {xcen_x_data[i]} {xcen_y_data[i]}
                                {xcen_z_data[i]}\n")


def pres_mapping(h5_file, csv_file):
  """Allows to export in a .pts file the data
  for the pressure and the x,y,z coordinates of the
  center of every grid cell contained in a .h5 file

  Args:
    h5_file (str): Path of the .h5 file
    pts_file (str): Path of the .pts file
  """
  # Open the .h5 file
  with h5py.File(h5_file, 'r') as h5:
    # Extracts the data
    pres_data = h5["STREAM_00"]["CELL_CENTER_DATA"]["PRESSURE"][:]
                                # Pressure [Pa] dataset
    xcen_x_data = h5["STREAM_00"]["CELL_CENTER_DATA"]["XCEN_X"][:]
                                # X coordinate [m] dataset
    xcen_y_data = h5["STREAM_00"]["CELL_CENTER_DATA"]["XCEN_Y"][:]
                                # Y coordinate [m] dataset
    xcen_z_data = h5["STREAM_00"]["CELL_CENTER_DATA"]["XCEN_Z"][:]
                                # Z coordinate [m] dataset

  num_cells = pres_data.shape[0]

  with open(csv_file, 'w') as csv:
    for i in range(num_cells):  # Writes: "temperature" "X
                                coordinate" "Y coordinate" "Z
```

```
                                    coordinate" on each line
            csv.write(f"{pres_data[i]} {xcen_x_data[i]} {xcen_y_data[i]} {
                                    xcen_z_data[i]}\n")


def h5_structure(h5_path):
    """Use this to explore the structure of the h5 output file from
                                    converge

    Args:
      h5_path (str): Path of the h5 output file
    """
    with h5py.File(h5_path, 'r') as h5:
      # List all groups and datasets in the file
      print("Groups and Datasets:")
      print(list(h5.keys()))
      print("\nDetailed Structure:")


      def print_hdf5_item(name, obj):
        # Iterate over all groups and datasets
        print(name, obj)


      h5.visititems(print_hdf5_item)
```

Bibliography

[1] Marco Evangelos Biancolini et al. *Fast radial basis functions for engineering applications*. Springer, 2017.

[2] Rolland L Hardy. Multiquadric equations of topography and other irregular surfaces. *Journal of geophysical research*, 76(8):1905–1915, 1971.

[3] Rolland L Hardy. Theory and applications of the multiquadric-biharmonic method 20 years of discovery 1968–1988. *Computers & Mathematics with Applications*, 19(8-9):163–208, 1990.

[4] Guang-Bin Huang, Paramasivan Saratchandran, and Narasimhan Sundararajan. A generalized growing and pruning rbf (ggap-rbf) neural network for function approximation. *IEEE transactions on neural networks*, 16(1):57–67, 2005.

[5] Juntao Fei and Hongfei Ding. Adaptive sliding mode control of dynamic system using rbf neural network. *Nonlinear Dynamics*, 70:1563–1573, 2012.

[6] John P Boyd and Kenneth W Gildersleeve. Numerical experiments on the condition number of the interpolation matrices for radial basis functions. *Applied Numerical Mathematics*, 61(4):443–459, 2011.

[7] Holger Wendland. Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Advances in computational Mathematics*, 4:389–396, 1995.

[8] Ubaldo Cella, Corrado Groth, and Marco Evangelos Biancolini. Geometric parameterization strategies for shape optimization using rbf mesh morphing. In *Advances on Mechanics, Design Engineering and Manufacturing: Proceedings of the International Joint Conference on Mechanics, Design Engineering & Advanced Manufacturing (JCM 2016), 14-16 September, 2016, Catania, Italy*, pages 537–545. Springer, 2017.

[9] Marco Evangelos Biancolini. Mesh morphing and smoothing by means of radial basis functions (rbf): a practical example using fluent and rbf morph. In *Handbook of*

research on computational science and engineering: theory and practice, pages 347–380. IGI Global, 2012.

[10] ME Biancolini, U Cella, G Travostino, and M Mancini. Shaping up–mesh morphing reduces the time required to optimize an aircraft wing. *ANSYS Advantage Magazine*, 7(1):32–34, 2013.

[11] Emiliano Costa, Marco E Biancolini, Corrado Groth, Ubaldo Cella, Gregor Veble, Matej Andrejasic, et al. Rbf-based aerodynamic optimization of an industrial glider. In *Proceedings of International CAE Conference*, 2014.

[12] John B Heywood. Internal combustion engine fundamentals. *(No Title)*, 1988.

[13] Giancarlo Ferrari. *Motori a combustione interna*. Società Editrice Esculapio, 2019.

[14] HK Versteeg and W Malalasekera. Computational fluid dynamics. *The finite volume method*, 1995.

[15] John Anderson. *Fundamentals of Aerodynamics*. McGraw hill, 2011.

[16] Roberto Verzicco. Appunti di turbolenza. *Universita degli studi di Roma Tor Vergata Dipartimento di Ingegneria Meccanica*, 2006.

[17] David C Wilcox et al. *Turbulence modeling for CFD*. DCW industries La Canada, CA, 1998.

[18] Hendrik Tennekes and John Leask Lumley. *A first course in turbulence*. MIT press, 1972.

[19] Nicoud Franck. Unsteady flows modeling and computation. 12 2007.

[20] Senecal P.K. Richards, K.J. and E. Pomraning. *CONVERGE 3.1*. Convergent Science, Madison, WI (2024).

[21] Frank Kreith. *Principles of heat transfer*. 1962.

[22] Prof. Paolo Coppa. *Dispense per il corso di Termotecnica 2*. A.A. 2022/2023.

[23] Matteo Marra. *Ottimizzazione della testata di un motore mediante mesh morphing*. A.A. 2021/2022.

[24] *Ansys Academic Research Workbench, Mechanical and SpaceClaim Release 23.2*.

[25] Andrea Lopez. *Ottimizzazione di flussi esterni ed interni mediante metodi CFD adjoint e mesh morphing*. A.A. 2019/2020.

[26] *CONVERGE 3.1 user manual*. Convergent Science, Madison, WI (2023).