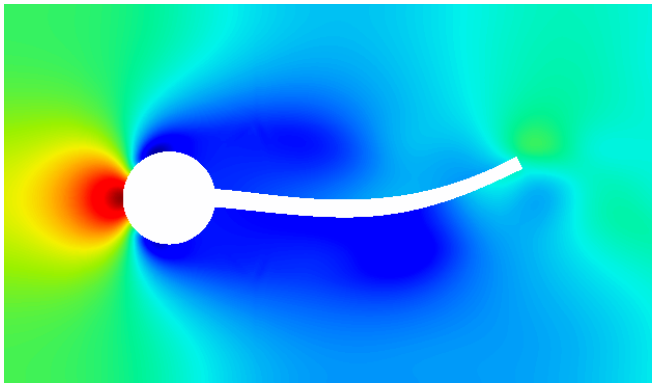
TÉCNICO LISBOA



Aeroelastic Effects of a Very Flexible Transonic Wing: Fluid-Structure Interaction Study

Dario Aresta

Thesis to obtain the Master of Science Degree in

Aerospace Engineering

Supervisor(s): Prof. Fernando José Parracho Lau
Dr. Frederico José Prata Rente Reis Afonso

Examination Committee

Chairperson: Prof. Filipe Szolnoky Ramos Pinto Cunha
Supervisor: Prof. Fernando José Parracho Lau
Member of the Committee: Prof. Afzal Suleman

November 2017

Dedicated to someone special...

Resumo

Os requisitos do sector de transporte exigem uma forma cada vez mais eficiente de consumo de combustíveis, dando origem a desenhos de asa esbeltas, com elevada alungamento. Devido à sua longa extensão, estas novas configurações exibem altas deformações. É então necessário prever as influências no desempenho de voo, especialmente a velocidades transónicas, onde os problemas de instabilidade são mais críticos. Nesta tese é efectuado um estudo computacional de aeroelasticidade num modelo 3D de uma asa transónica, já estudada no projecto FP7-NOVEMOR. Para alcançar este objectivo um novo solver de interacção fluído estrutura é compilado, usando open-source software como ponto de partida. Foram efectuados casos teste com escoamentos incompressiveis e também com escoamentos de alta velocidade. O resultado final demonstra grandes deslocamentos, levando a um claro movimento das ondas de choque ao longo da asa nos testes de desempenho de voo.

Palavras-chave: Fluid-Structure Interaction, Partitioned, Transonic, Aeroelasticity

Abstract

The challenging requirements of the transportation sector target a more efficient fuel consumption, leading to slender wing design. Because of the longer extension, the new configurations exhibit much higher deformations. Therefore, it is necessary to predict the influences in the flight performances, especially in transonic speeds, where instability problems such as flutter occur. This work performs a computational aeroelastic study of a three-dimensional transonic wing model, part of the FP7-NOVEMOR project. In order to reach this goal, a new partitioned Fluid-Structure Interaction (FSI) solver have been compiled, starting from Open-source software. It has been tested with an incompressible flow benchmark and then extended to high speed flows. The final results show large displacements, leading to a clear motion of the shocks waves along the wing chord and great variation in the flight performances.

Keywords: Fluid-Structure Interaction, Partitioned, Transonic, Aeroelasticity

Contents

List of Tables

List of Figures

Nomenclature

Greek symbols

α	Parameter of α method.
ϵ	Tolerance.
γ	Diffusivity field.
λ	Eigenvalue.
λ^s	First Lamé coefficient.
μ	Dynamic viscosity.
μ^s	Second Lamé coefficient.
ν	Kinematic viscosity.
ϕ	Volumetric flux.
ρ	Density.

Roman symbols

$[K]$	Stiffness matrix.
$[M]$	Mass matrix.
$\bar{U}$	Mean velocity.
a	Acceleration vector.
D	Deformation gradient tensor.
g	Gravity vector.
I	Identity matrix.
j	Diffusive heat flux.
S	Surface vector.
T	Viscous stress tensor.

C_D Coefficient of drag.

C_L Coefficient of lift.

F Gradient deformation tensor.

h Altitude.

J Jacobian.

k Thermal conductivity.

M Mach number.

Re Reynolds number.

S Surface.

T Temperature.

σ Cauchy stress tensor.

b Body forces vector.

E Energy.

p Pressure.

$\mathbf{u}$ Velocity vector.

$\hat{\mathbf{v}}$ Mesh velocity.

u, v Velocity Cartesian components.

c Convective velocity.

Subscripts

X Material coordinate system.

x Spatial coordinate system.

ζ Referential coordinate system.

Superscripts

ext External.

f Fluid.

s Solid.

T Transpose.

w Wing.

Glossary

CAD	Computer Aided Design.
CFD	Computational Fluid Dynamics.
CSS	Conventional Serial Staggered.
ENO	Essentially non-oscillatory scheme.
FEM	Finite Elements Method.
FSI	Fluid-Structure Interaction.
OpenFOAM	Open Field Operation and Manipulation.
PIMPLE	merged PISO-SIMPLE.
PISO	Pressure Implicit Split Operator.
PPM	Piecewise parabolic method.
RKDG	Runge–Kutta discontinuous Galerkin method.
SIMPLE	Semi Implicit Method for Pressure Linked Equations.
WENO	Weighted ENO scheme.

Chapter 1

Introduction

As the name could suggest, Fluid-Structure Interaction (FSI) refers to problems that concern the behavior of the structure in response to the surrounding fluid. In this broad category, it is possible to identify clinic applications, such as studies of the motion of the blood in the vascular wall, but also engineering problems that deal with very flexible structures. Clearly, this require a framework that considers the mutual dependence of the fluid and structure dynamics. In this sense it can be considered a two-way coupling problem.

The transport sector is nowadays challenging the new requests of the international community. The innovation, requested by the European Commission with the Horizon 2020 proposal, seek a more sustainable and resource efficient transport. In order to reduce the induced drag, hence to improve fuel consumption, a clear trend in aircraft design is the increased aspect-ratio of the wing.

These new configurations usually expose a slender and lightweight structure to airflow loads that induce an high deformation and fluctuations. Therefore, it is necessary to predict, in the early phase of the design process, the mutual dependence between the fluid and the structure to prevent undesired instabilities that could affect the flight performance. In the recent years these features are particularly important in unmanned aerial vehicles that have a long-time monitoring scope. In this case a flight performance analysis requires more sophisticated methods than a linear analysis adopted in small deformation case, yielding to a nonlinear aeroelastic study [1].

1.1 Motivations

A common way to perform aeroelastic studies is to couple a computational structural dynamics solver with an unsteady aerodynamic code. To make this method as simple as possible, the fluid equations are linearized and then discretized in a finite volume method. This technique is limited to subsonic and supersonic flows, where linearized unsteady flows may be considered. In the transonic flight regime, a nonlinear response approach should be preferred because of the motion of the shock waves and viscosity that play an important role in the stability prediction [2].

Different studies tried a different approach with the use of linear models through a correction in the

unsteady fluid solution to predict the flutter whereas the linear assumption is not valid [3]. Even if the semi-empirical correction is considered an adequate tool in the industrial sector, it has been discarded in this work. A more comprehensive method have been adopted, in order to develop a code that may be used for different flow Mach numbers.

When the flight conditions has a Mach number over the critical M_{cr}, but less than $M = 1$, local regions along the wing chord are dominated by a supersonic flow. They start with the acceleration right after the leading edge, and end with a nearly normal shock closer to the trailing edge. The shocks create discontinuities in both velocity and pressure fields, thus leading to a strong and rapid variation of the loads along the wing chord. As the wing deforms in bending and torsion, the supersonic region grows and shrinks. The resulting structure motion is suspected to be tightly related to nonlinear flutter behavior that the wing shows at Mach number near one [4].

Therefore this thesis focuses on the interesting and debated aeroelastic behavior of a commercial aviation wing in transonic flow. Nowadays the computational engineering tools are freely distributed to everyone, thanks the several open-source softwares and documentation that made this work possible. As explained later in subsection 2.1.1, even though similar solvers are already implemented, their restrictions make the utilization non compatible with more general cases. Thus the starting point is the coding of a new FSI solver that couple together a fluid and structural solver, and make it non-problem dependent in a way that it can be rearranged to different flow regimes, and a wide range of structure models, creating a general purpose FSI framework.

1.2 Previous and Related Work

FSI interactions have been studied since they gained the interest of the engineering community. The older aeroelastic studies and tools use simple aerodynamic models that do not capture complex flow phenomena, such as separation of the flow or transonic and sonic shocks [5]. By the way, the Computational Fluid Dynamics (CFD) softwares are gathering more and more interest, thanks to the improvement of their accuracy and the large availability of computational resources. The combination of a fluid solver with a structural solver remains a hard job, for several reasons, that will be discussed later.

The early stages of coupled systems analysis started with the firsts studies of deforming spatial domain, Tezduyar et al. [6] in 1992 calculated the flux over a cylinder with dynamic mesh motion of the boundaries. Later, researches regarding numerical methods applied to fluid-structure interaction [7, 8] investigated different approaches that range from segregated to monolithic schemes, starting the general framework that, even nowadays, is under study. In the recent years, classes of problems involving highly deformable structures and complex geometries, such as parachutes dynamics and arterial blood motion, have been studied in detail [9, 10].

Coming to the aerospace sector, the fluid-structure interaction of transonic flows gathered the attention of the researchers' community, to provide good a investigation method that address the aeroelastic phenomena, such as flutter and divergence [11]. In this sense, Stickan et al. [12] conducted a study of a three-dimensional wing model in a transonic wind-tunnel, from which they observed the limit cy-

cle oscillations in this regime and compared the experimental data with computational analysis. The results showed a strong relationship with the structural model employed. The conclusion is that it is required a detailed structural wing model to predict a correct motion. L. Campbell [13] investigated the flexible turbo-machinery response in time, validating the use of open-source softwares in a segregated approach to FSI. Even if classical flutter theory offers a reliable prediction for standard aircraft design, when structure and aerodynamic non-linearities are relevant, other approaches are preferred. Notable are the studies of large wind turbines, where the increasing need in generated power leads to very large rotor diameters, with composite flexible blades characterized by theirs high aspect-ratio and slenderness [14]. In this case, a good prediction of the fluid-structure interaction could allow an higher efficiency of the wind turbines, thus decreasing the drag.

1.3 Commercial FSI softwares

FSI could be considered as a subset of the broad category of multi-physical applications that the industry requires nowadays. Neglecting for the moment the implementation in open-source codes, the requirements of the industry technology introduced, in the last years, FSI support in the pre-existing software workbenches. The most relevant are Adina FSI and Ansys Multiphysics [15, 16].

The main advantage of commercial codes is that they rely on a largely developed fluid and structural software that can be coupled within the same software. Hence, they are able to simulate different fluids, ranging from the incompressible, compressible, transonic, and couple them with structure defined with a large variety of elements. On the other hand, if a customization is required to match the user needs, it is hard, if not impossible, to perform the necessary changes. For this reason, as will be explained in subsection 2.1.1, open-source softwares will be preferred. L. Campbell [13], after an investigation on the limited openness of these software to new modifications, concluded that the strongly coupled approach is not natively supported, thus the range of problems they can deal with is limited.

1.4 Thesis Outline

The thesis presentation follows the process that led the author to the goal discussed in section 1.1. chapter 2 starts with the selection of the software employed, that will represent the basis of the entire work. Fundamental in this case was the selection of appropriate criteria for the purpose of an FSI study. Then the dynamic mesh representation and the related conservation equations are presented along with the automatic mesh motion mechanism. So, the FSI algorithm is presented, highlighting the most notable features. At the end of the chapter the code produced is tested in a incompressible flow benchmark problem and the results are shown.

In chapter 3, the transonic wing model is introduced with a geometrical description and the flight condition is defined. In this case the flow is compressible, so the code is adapted to the new conservation laws and the new solution scheme is defined, followed by a mesh convergence study that aim at a good

trade-off between computational time and accuracy of the solution. The definition of the wing structural model closes the chapter.

In chapter 4 the results coming from the transonic aeroelastic study are presented and discussed. Finally, a conclusion ends the document.

Chapter 2

FSI Model

In this chapter the author presents the theoretical background of the solvers adopted in this study. The re-formulation of the standard equations for a dynamic mesh is introduced and how these are combined in the FSI algorithm is explained. At the end, a test problem is presented in order to compare the performance of the FSI solver with a literature benchmark.

2.1 Aerodynamic and structural solvers

As stated in the previous chapter, this thesis aims at the coding of a FSI solver, starting from pre-existent softwares. Commercial softwares have the advantage of a wide and verified stability, without a lot of interaction with the user, but they do not usually provide an easy developing environment for new applications. With that said, it was pretty straightforward to consider open-source projects that gathered the interest of the researchers' community in the last years.

Open-source softwares do not only guarantee the openness of their codes to new implementation, but also create an active community of researchers that share their knowledge and achievements for common purposes.

2.1.1 Aerodynamic Solver

The selection criteria of the fluid solver were connected to the essential specifications needed for this work. The main features considered were:

- A wide range of fluid solvers, namely steady/transient, compressible/incompressible with different turbulence properties;
- The implementation of the same codes for a dynamic mesh and automatic mesh motion solvers;
- Post-processing and visualization capabilities;
- An active community of developers.

OpenFOAM (for ”Open source Field Operation And Manipulation” [17]) respects all these criteria and thus was chosen. It has an extensive package that simulates anything from laminar to turbulent flows, chemical reactions and heat transfer. It also includes tools for meshing, data processing and visualization, among others. Moreover it has different mesh motion solvers for various applications, from the rigid body motion to more complicated motions.

Besides the official release, an extended version for developers is also available. It includes examples of FSI solvers that make use of an internal structural solver. By the way, the internal solver is only able to compute the dynamics of a structure discretized with volume elements, because *OpenFOAM* is designed for this mesh topology. Snippets of the code can be extracted and adapted for other purposes, such as the communication with a new structural solver that permits a wider range of problems to be solved. In conclusion, *OpenFOAM v4.1* has been adopted.

2.1.2 Structural solver

As for the fluid solver, a set of requirements were selected for the structural software. In particular it was sufficient to find a software able to run dynamic simulations with different material properties. These specifications were less restrictive than the previous since usually they figure as basic features of all finite elements method (FEM) applications.

For the simplicity of the input files, similar to the well known *ABAQUS*, *CalculiX v2.12* was used in this work. This package, written by Guido Dhondt [18], can execute static/dynamic simulations with the possibility of loads concentrated at some nodes of the mesh or distributed on faces or volumes. It comes with a pre- and post-processing application, *CalculiX GraphiX*, to visualize the mesh, apply constraints and loads and to analyze the results.

It is important to mention that if a different structural software is required, it will be easy to switch the applications, since the communication with *OpenFOAM* is just a matter of parsing in input and output data. In fact, all the FSI application have been compiled in the *OpenFOAM* environment, thus they do not interfere with the *CalculiX* package code.

2.2 Dynamic mesh description

2.2.1 Arbitrary Lagrangian-Eulerian (ALE) formulation

Dealing with problems that simulate a dynamic and strong deformation of the continuum requires a slightly different kinematic description of the phenomena; in order to take into account the relations between the moving mesh and the deformed domain [19]. The two classic continuum descriptions, *Lagrangian* and *Eulerian*, combined together into the so-called *Arbitrary Lagrangian-Eulerian* (ALE in short) description to take the advantages of the two methods and minimize their weakness.

In a Lagrangian description, each node of the mesh follows the material particle it belongs to, during time. This is the algorithm that is usually adopted in structural mechanics. On the contrary, an Eulerian description, popular in fluid dynamics, assumes that the nodes does not follow the particles of the flow,

and remains fixed in space, during time [19]. When it comes to FSI problems, the deformation of the structure implies some changes in the shape of the fluid domain. If the nodes of the fluid follow the deformation of the structure they will not anymore be fixed in space and will not track the displacement of the fluid particles as well. Thus, they will move in an arbitrary way, depending on the problem needs. This is why the new description of the continuum is needed for the purpose of this work.

The Eulerian and Lagrangian description refers respectively to the spatial domain $R_{\mathbf{x}}$ and the material domain $R_{\mathbf{X}}$, where the latter follow the material particles. Thus, two different reference frame, **x** and **X** are defined. The relation between them is the material velocity **v**:

$$\mathbf{v}\left(\mathbf{X}, t\right) = \frac{\partial \mathbf{x}}{\partial t}_{\mathbf{x}}, \tag{2.1}$$

that is the velocity of the particles in **X** holding the coordinate fixed. A great advantage of the Lagrangian description is that, at any instant, the mesh point coincide with the material particle. In this way the material derivative coincide with the time derivative: there are no convective effects. The drawback is that for problems that involve a large deformation of the continuum, the quality of the mesh decreases rapidly. For this reason, the fluid is usually described through an Eulerian point of view, where the flow deforms with respect to the mesh. With this approach, the material velocity corresponds to the instantaneous velocity of the material particle that in the considered time t belongs to a certain node, and it will be bounded to that node for the rest of the analysis. In this case the disadvantage is the presence of convective effects because of the relative motion between the material and the grid.

In the ALE description, a new reference system is required. It does not coincide either with $R_{\mathbf{x}}$, nor with $R_{\mathbf{X}}$. The new configuration is identified by R_{ζ} and the referential coordinate ζ.

Hence, the mesh velocity $\hat{v}$ is:

$$\hat{\mathbf{v}}\left(\zeta, t\right) = \frac{\partial \mathbf{x}}{\partial t}_{\zeta}. \tag{2.2}$$

Thus, the convective velocity c, that is the relative velocity between the material and the mesh, which can be rapidly expressed by the difference:

$$\mathbf{c} = \mathbf{v} - \hat{\mathbf{v}}. \tag{2.3}$$

It is also necessary to express the relations between the derivatives for a generic variable f:

$$\frac{\partial f}{\partial t}_{X} = \frac{\partial f}{\partial t}_{x} + \mathbf{v} \cdot \nabla f. \tag{2.4}$$

This equation states that the variation of the quantity f for a given particle is equal to the local variation plus a convective term. In order to have a lighter notation in the rest of the work, it will be

assumed that:

$$\frac{d\cdot}{dt} = \frac{\partial\cdot}{\partial t}_{\mathbf{x}}, \tag{2.5a}$$

$$\frac{\partial\cdot}{\partial t} = \frac{\partial\cdot}{\partial t}_{\mathbf{x}}, \tag{2.5b}$$

and the final relation between material and referential derivative is given by:

$$\frac{\partial f}{\partial t}_{X} = \frac{\partial f}{\partial t}_{\zeta} + \frac{\partial f}{\partial x}\cdot\mathbf{c} = \frac{\partial f}{\partial t}_{\zeta} + \mathbf{c}\cdot\nabla f. \tag{2.6}$$

In this case, the material derivative of the quantity f, for a certain particle, is equal to the local derivative in the reference system plus a convective term that consider the relative motion between the reference and material system.

The equations in the ALE differential form can be obtained starting from the Eulerian equations of mass, momentum and energy:

$$\frac{d\rho}{dt} = \frac{\partial\rho}{\partial t}_{\mathbf{x}} + \mathbf{v}\cdot\nabla\rho = -\rho\nabla\cdot\mathbf{v} \qquad mass, \tag{2.7a}$$

$$\frac{d\mathbf{v}}{dt} = \rho\left(\frac{\partial\mathbf{v}}{\partial t}_{\mathbf{x}} + (\mathbf{v}\cdot\nabla)\,\mathbf{v}\right) = \nabla\cdot\sigma + \rho\mathbf{b} \qquad momentum, \tag{2.7b}$$

$$\frac{dE}{dt} = \rho\left(\frac{\partial\mathbf{E}}{\partial t}_{\mathbf{x}} + \mathbf{v}\cdot\nabla E\right) = \nabla\cdot(\sigma\cdot\mathbf{v}) + \mathbf{v}\cdot\rho\mathbf{b} \qquad energy, \tag{2.7c}$$

where ρ is the mass density, **v** the material velocity, σ the Cauchy stress tensor, **b** the specific body force and E the specific total energy. Now substituting the relation in Equation 2.6 we have:

$$\frac{\partial\rho}{\partial t}_{\zeta} + \mathbf{c}\cdot\nabla\rho = -\rho\nabla\cdot\mathbf{v} \qquad mass, \tag{2.8a}$$

$$\rho\left(\frac{\partial\mathbf{v}}{\partial t}_{\zeta} + (\mathbf{c}\cdot\nabla)\,\mathbf{v}\right) = \nabla\cdot\sigma + \rho\mathbf{b} \qquad momentum, \tag{2.8b}$$

$$\rho\left(\frac{\partial\mathbf{E}}{\partial t}_{\zeta} + \mathbf{c}\cdot\nabla\mathbf{E}\right) = \nabla\cdot(\sigma\cdot\mathbf{v}) + \mathbf{v}\cdot\rho\mathbf{b} \qquad energy. \tag{2.8c}$$

The ALE description is widely used in the so-called *hydrocodes*, that are employed to analyze materials with large deformation, structure, fluids and FSI systems, thus when neither an Euler, nor an Lagrangian description can simulate the problem dynamics. On the other hand, the convective velocity **c** replaces the material velocity **v** in the convective terms of the Eulerian formulation, possibly leading to stability issues [19].

2.2.2 Mesh update algorithm

The purpose of a mesh update procedure is to keep the mesh as regular as possible in response to the deformation of the boundaries, to avoid mesh distortions that could prevent the correct execution of the

simulations [19]. In FSI problems, the motion of the boundary that represents the interface between the structure and the fluid is not known a priori, but is solution-dependent; thus the motion of the mesh must be recalculated for each time step.

The automatic update of the internal mesh nodes should be able to find the best disposition in order to match the movement of the boundaries, and, at the same time, produce a good quality mesh that fulfills the quality requirements. Different approaches, including the spring analogy, the elliptical smoothing and radial basis functions, have been studied in FSI applications [20].

Jasak and Tukovic [21] developed a vertex based finite volume method for automatic mesh update and implemented it in $OpenFOAM$. The motion is computed starting from a Laplace equation with a diffusion coefficient that can vary through the domain. The solution of the Laplace equation gives a regular mesh composed by lines of equal potential, which is a repositioning of the internal nodes in a way that are equidistant to the connected nodes. The drawback of this method is that in a non-convex domain the nodes could move outside the boundaries. Thus the equation

$$\nabla \cdot (\gamma \nabla \mathbf{u}) \tag{2.9}$$

is solved to find the new nodes positions. γ is the diffusion field, while **u** is the point velocity that is used to change the point position:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{u}\Delta t. \tag{2.10}$$

Alternatively, the spring analogy method relies on a physical-based approach. The edges of the meshes are discretized with fictitious linear springs. So, imposing a displacement on the boundary nodes, the internal nodes moves as a system of connected springs would do. The equilibrium state is described by the generalized Hooke law:

$$\{\mathbf{F}\} = [K]\{\mathbf{u}\}, \tag{2.11}$$

being $\{\mathbf{F}\}$ the force vector, $[K]$ the stiffness matrix and $\{\mathbf{u}\}$ the displacement vector.

Two different linear spring analogies can be developed depending on how the initial state is described. One option is to consider the springs at the start time to be in equilibrium, thus in an unstressed state. This is called the segment state. Otherwise the springs can be initialized in a pretensioned condition, leading to the vertex method. The latter considers the spring equilibrium position at zero length.

A comparative study conducted by Sudharsan et al. [22], found that even though the spring analogy, with some improvements, enhance the performance of the mesh motion schemes, it fails to maintain a good mesh quality, creating badly conditioned shapes and possibly causing a crash in code.

OpenFOAM includes an automatic mesh motion solver that is also able to deal with topological changes of the mesh. This method would require a re-meshing every time topological changes are needed, increasing considerably the computational time. For the scope of this work, the dynamic mesh algorithm will move the mesh points keeping the same connections between them, thus without any

topological change.

Discarding all the mesh motion that does not belong to the FSI analysis, like the rigid body motion, the only automatic mesh motion available in OpenFOAM is the $dynamicMotionSolverFvMesh$, where the motion of the boundary is set as boundary condition to determine the position of the internal points. In most of the cases, a Laplacian solver is employed to track the interface motion. For this solver, it is possible to specify the displacement of the boundaries or their velocity as boundary condition of the Laplace equation. Moreover, the diffusivity of the equation can vary from point to point. Particularly interesting is the $inverseDistance$ relation for the diffusivity. In this case the diffusion will be inversely proportional to a set of boundaries, resulting in a greater displacement of the nodes closer to the moving boundaries. Considering the geometry of the test case that will be discussed in section 2.4, the initial mesh in Figure 2.1 have been deformed through a translation of the cylinder and bar in the longitudinal direction. The results with a uniform diffusivity, in Figure 2.2, show worse quality of the mesh than the one obtained with an inverse relation, shown in Figure 2.3. It is clear that the former is worse in terms of skewness and orthogonality around the cylinder. Moreover, the cells closer to the tip of the bar are overlapping. Thus the $inverseDistance$ relation in this case is a very efficient method to avoid the motion of the mesh nodes outside the computational domain.

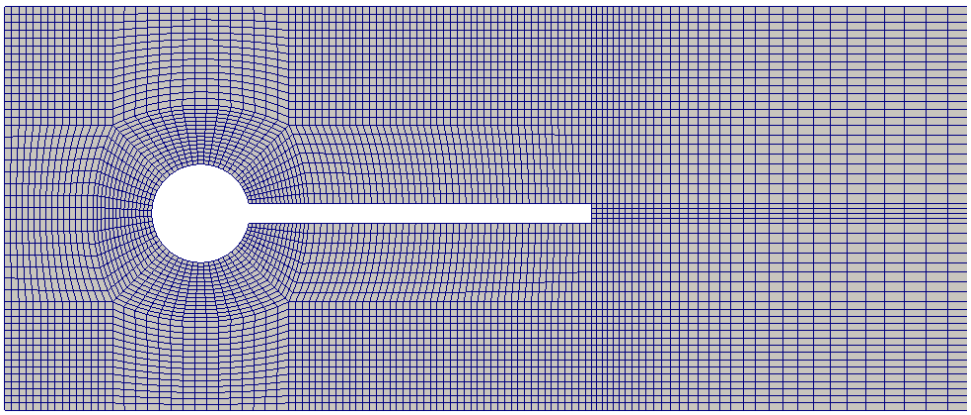


Figure 2.1: Initial mesh.

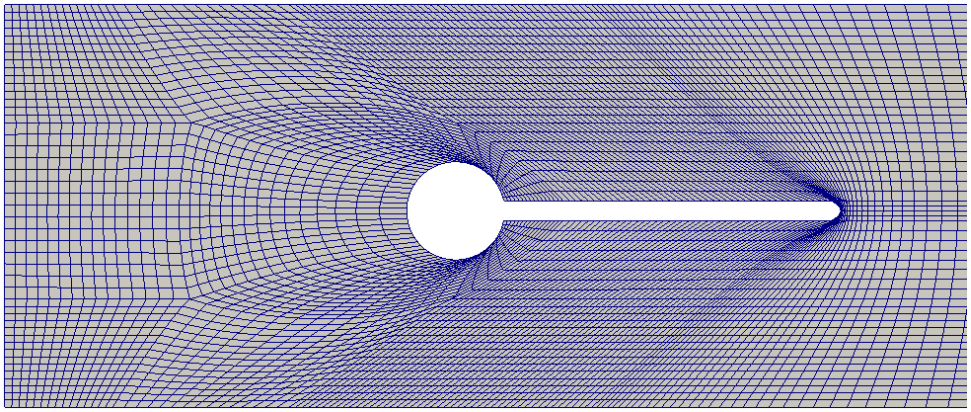


Figure 2.2: Mesh motion with uniform relation.

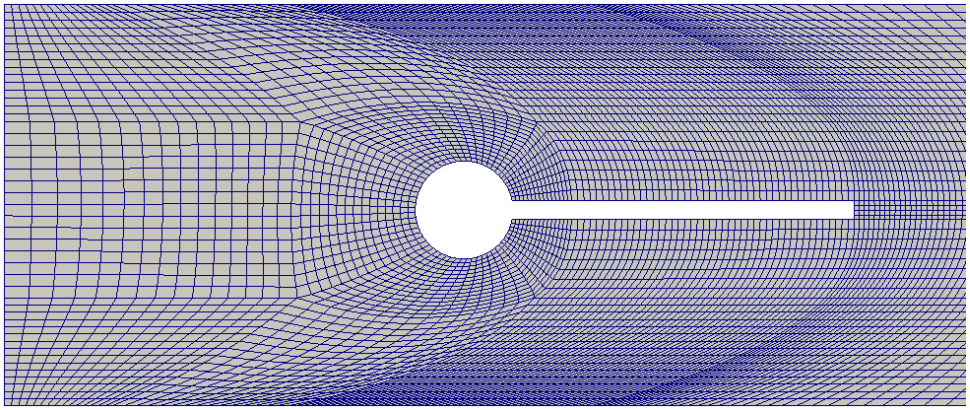


Figure 2.3: Mesh motion with inverse relation.

2.3 FSI algorithm

Fluid-structure interaction analysis is usually classified as a 3-field problem that involve the fluid, the structure and the motion of the mesh. The FSI algorithm aim at a correct communication between them and an overall convergence of the entire process.

As mentioned in chapter 1, a partitioned approach have been adopted for the FSI algorithm. This employs a segregated solution procedure where the fluid and structure are sequentially solved and some tools provide the communication between them. This allows to use largely developed software rather than cast the two domains in a single system, that would require a ground-up coding of the entire software.

The simplest example is referred to as Conventional Serial Staggered (CSS) algorithm [23]. In this scheme the fluid field is solved first, then the pressure and shear forces at the interface are transfered to the solid solver and a predicted motion of the solid is computed. Then the automatic mesh solver receive as input the displacements and velocities of the structure, and give as output the new mesh. Then the process is reiterated till the end of simulation is reached.

On the interface between fluid and structure, the following relations are set:

$$\sigma^f n = \sigma^s n \qquad on\ \Gamma; \tag{2.12a}$$

$$\mathbf{v}^f = \mathbf{v}^s \qquad on\ \Gamma; \tag{2.12b}$$

that state the correspondence, on the interface Γ, of stresses and velocities. In the following analysis, a conformed mesh will be used at the interface, so the two domains are discretized in a way that they have the same faces at the interface, in terms of nodes. In this way the transfer of the stresses and velocities is straightforward and there is no need to interpolate the values for non coincident nodes.

As described here, the CSS algorithm does not check that the predicted dynamics correspond to the actual motion of the structure. For these reason it is called loosely-coupled. Loosely coupled time

advancing methods does not have a correct energy balance at the interface, so they show the so-called *added-mass* effect [24]. The numerical difficulties related to this effect are relevant in elastic structures that interacts with an incompressible fluid in a partitioned approach, especially if a specific combination of fluid and solid properties is met [25], that is when the density of the fluid and structure are comparable.

When the high complexity and non-linearity of FSI problems are relevant, some mathematical explanations of this phenomenon have been proposed. The conclusion is that to ensure the correspondence between the predicted and actual motion it is necessary to apply a strongly-coupled algorithm. It consists in a corrector step through sub-iterative process that check residuals set by the user at each time step. The scheme of this procedure is illustrated in Figure 2.4.

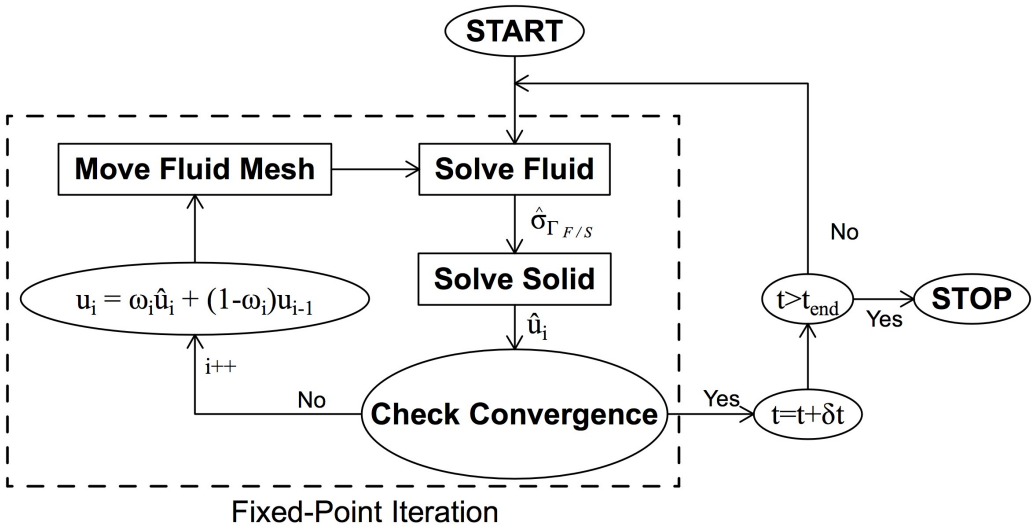


Figure 2.4: Sub-iterative solution scheme.

Some of the techniques that address this problem are known as block Newton methods and require the computation of cross derivatives between the fluid and structure fields. This is hardly feasible when dealing with black box solvers. Therefore *fixed-point* iteration are usually employed with much less effort and good results [26]. The relaxation step in a fixed-point iteration is given by:

$$u_i = u_{i-1} + \omega_i \left(\hat{u}_i - u_{i-1}\right) = \omega_i \hat{u}_i + \left(1 - \omega_i\right) u_{i-1}, \tag{2.13}$$

where i refers to the iteration number, u is the displacement applied, while $\hat{u}$ is the displacement computed by the structural solver. The relaxation factor ω_i can be constant during the sub-iterative process or can varies dynamically. Dynamic value allows for faster convergence. The Aitken's scheme [26] was adopted in this work, that gives a relaxation factor equal to:

$$\begin{aligned} \omega_{i+1} &= \frac{u_{i-2} - u_{i-1}}{u_{i-2} - \hat{u}_{i-1} - u_{i-1} + \hat{u}_i} = \omega_i \frac{u_{i-2} - \hat{u}_{i-1}}{u_{i-2} - \hat{u}_{i-1} - u_{i-1} + \hat{u}_i} = \\ &= -\omega_i \frac{r_{i-1}}{r_i - r_{i-1}} = -\omega_i \frac{(r_{i-1})^T (r_i - r_{i-1})}{|r_i - r_{i-1}|^2}, \end{aligned} \tag{2.14}$$

where $r_i = u_{i-1} - \hat{u}_i$. It demonstrated a good time convergence, moreover it is easy to implement and it is cheap to compute in the iterative process. It requires an initial relaxation factor for each time step that usually is set by the user. A value equal of 0.01 allows for a stable analysis for the problem presented in section 2.4. In Figure 2.5 the speed of the fixed-relaxation method is compared to the Aitken's relaxation method. The results refer to the test problem of section 2.4. The clear drop of the number of iterations leads to a great reduction of the CPU-time in the fixed-point relaxation process.

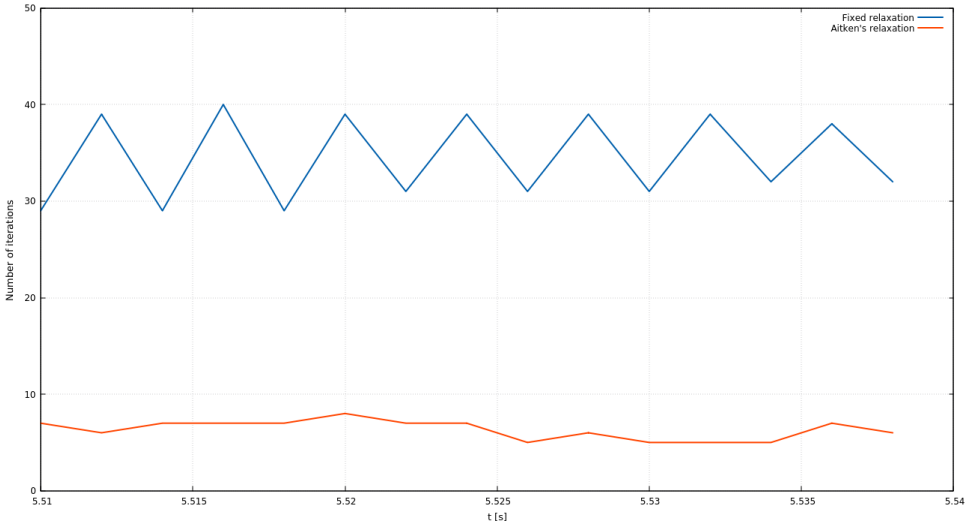


Figure 2.5: Comparison of the fixed-point iteration speed between fixed relaxation and Aitken's relaxation.

At the end end of each iteration, a check is executed to verify whether the inner-loop converged or not. In this work the residuals are evaluated on the displacements:

$$\frac{u_i - u_{i-1}}{u_0} < \epsilon, \tag{2.15}$$

where ϵ is a tolerance set by the user and u_0 is the displacements vector calculated in the first iteration.

2.3.1 Implementation

The algorithm described below has been coded in an *OpenFOAM*'s new application. The code can be summarized in the following snippet:

```
preProcessing.H
while (runTime.run())
{
  runTime++;
  solveSolid.H
  numIter=0;
  do
  {
```

```
    numIter++;
    setInterfaceDisplacement.H
    moveFluidMesh.H
    solveFluid.H
    solveSolid.H
    calcFsiResidual.H
  }
  while
  (fsiResidualNorm > outerCorrTolerance)
  runTime.write();
}
```

This C++ style list of commands can be described as:

1. The software starts with a pre-processing phase that finds the correspondence between the nodes and elements at the interface, included in $preProcessing.H$. That is required because, even if a conforming meshes is considered, when the elements are exported separately for fluid and structure mesh, the numbering of nodes and faces at the interface is not the same. At the beginning, the mesh corresponding to the fluid and the one corresponding to the structure are red, saving the nodes's coordinates and elements's matrix connection in different lists. Then the nodes' coordinates are compared to find a correspondence and the numbering is reordered in a way that it is the same for the different lists. Then the faces in the interface are compared to find the correspondence of the elements, to transfer the stresses. This process ensure the correctness of the mapping in Equation 2.12.

2. Afterwards the simulation starts with the external $while$ loop, that is active till the end of the runtime. At the beginning of each time step the structure is solved in $solveSolid.H$. In this header the interface loads are calculated as:

$$p_s = p_w, \tag{2.16a}$$

$$\tau_s = \rho\nu \, \nabla\mathbf{v}, \tag{2.16b}$$

being p_w the pressure on the generic interface face and p_s the pressure applied on the solid interface. The shear stress applied on the solid interface, τ_s, is calculated through the shear stress law for Newtonian fluids. $\nabla\mathbf{v}$ is the velocity gradient, ρ the fluid density, and ν the kinematic viscosity.

Then the external structural solver is called through a system call and the predicted displacements and velocities are stored into variables.

3. After that, the sub-iterative process starts with a relaxation factor $\omega_1 = 1$. Afterwards the velocities are transferred into $moveDynamicMesh.H$ to the mesh motion solver that computes the new

positions of the nodes and the convective velocity field.

4. Now the flow field is solved in $solveFluid.H$.

5. At the end of the iteration a new prediction of the structure motion is computed and compared with the previous through a check on the residuals. If the iteration did not converge, the new displacements and velocities are recalculated in $setInterfaceDisplacement.H$ using the Aitken's relaxation technique. The same scheme is repeated at each time step.

This is a general algorithm that can be adapted to different situations, in fact it does not depends on the fluid or solid model to solve, either they are compressible/incompressible etc. *OpenFOAM* natively include the possibility to create a new solver that, starting from the standard static mesh solver, supports dynamic mesh motion.

2.4 Benchmark

In order to verify the accuracy of the algorithm developed, a quantitative comparison with other solvers is required. Among all the literature's benchmarks, the Turek and Hron's proposal was preferred [27]. It describes a specific configuration, where a flow around an elastic body result in significant self-induced structural oscillations. Their study was conducted with a fully implicit monolithic approach.

2.4.1 Geometry

Coming to the domain definition, it is a simple 2D flow around a cylinder with an elastic bar attached. Referring to Figure 2.6 the geometry, at the start time, is fully described by the following quantities:

- The domain bounding box is $L = 2.5$ and $H = 0.41$;

- Considering the left bottom corner as the origin, the center of the cylinder is in $C = (0.2, 0.2)$ with radius $r = 0.05$. Thus the position of C gives to the geometry an asymmetrical configuration, since the channel on the top of the bar is larger than the other;

- The bar is defined by the length and height, respectively $l = 0.35$ and $h = 0.02$. The right bottom corner is at $(0.6, 0.19)$;

- A control point, called A, is located at $(0.6, 0.2)$.

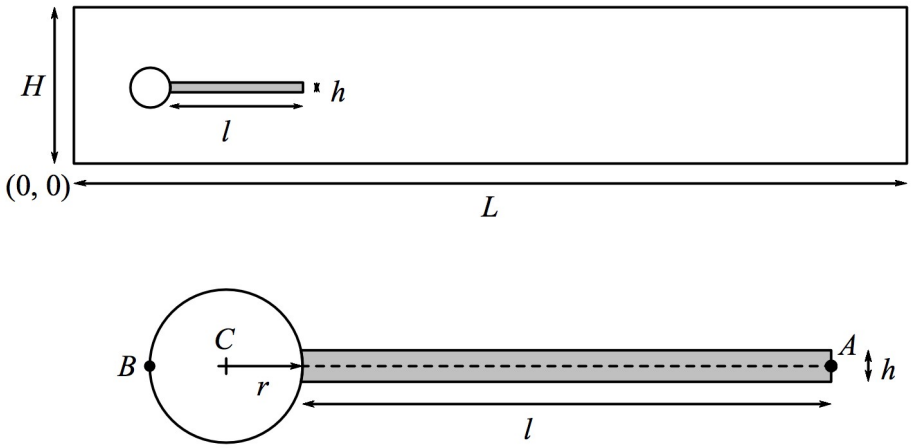


Figure 2.6: Geometry of the Turek's benchmark [27].

2.4.2 Incompressible fluid equations

The model of the fluid considered is Newtonian and incompressible. Fluid properties are defined through the density ρ^f and the viscosity ν^f. A starting point for the coding of the FSI solver was $pimpleDyMFoam$, distributed with the official *OpenFOAM*'s release. This is a version of $pimpleFoam$ that supports dynamic mesh. It is based on the incompressible governing equations [27]:

$$\nabla \cdot \mathbf{v}^f = 0 \qquad mass, \tag{2.17a}$$

$$\rho^f \frac{\partial \mathbf{v}^f}{\partial t} + \rho^f \left(\nabla \mathbf{v}^f\right) \mathbf{v}^f = \nabla \cdot \sigma^f \qquad momentum. \tag{2.17b}$$

and the constitutive equation:

$$\sigma^f = -p^f \mathbf{I} + \rho^f \mathbf{v}^f \left(\nabla \mathbf{v}^f + \nabla \mathbf{v}^{f\,T}\right). \tag{2.18}$$

The equations are solved through a PIMPLE (merged PISO-SIMPLE) algorighm [28]. It consists in a pressure-velocity coupling scheme and can be seen as a variation of the PISO (Pressure Implicit Split Operator). algorithm, with an external corrector that loops a fixed number of times. It does support the dynamic change of the time step in order to match the desired condition on the Courant number. In general, the SIMPLE (Semi Implicit Method for Pressure Linked Equations) is used for steady-state simulations, the PISO algorithm for small time-step calculations, and PIMPLE algorithm when large time-step is required, thus when the Courant number is greater than one [29].

Because steady state algorithm SIMPLE does not have a time derivation and it is not consistent, it is necessary to make use of under-relaxation factors between iterations to avoid floating point exception, that is the blow up of the solution. On the other hand low under-relaxation factors could lead to a slow convergence. The PISO algorithm includes the time derivation and the consistency of the pressure-

velocity coupling equation. This leads to better stability, thus the under-relaxation is not needed anymore. At the same time, a new limitation is introduced, it is necessary to have a Courant number lower than one; this number measures how fast the information travels, compared to the integration time.

Here the derivation of the PIMPLE algorithm will be presented following Versteeg and Malalasekera [30]. It is defined as a guess and correct procedure for the calculation of pressure in staggered grid. On a staggered grid the scalar pressure variable are stored in the cell centers of the control volumes, while the velocities are stored on the cell faces. Figure 2.7 shows the control volume of the pressure, whose center is P and the volumes correspondent to the two velocity component, u and v, in a two-dimensional case. The continuous lines refer to the position of the pressure cell centers and their numbering is given in capital letters, that is $I-1$, I, $I+1$ in one direction and $J-1$, J, $J+1$ in the other. With the same method, the broken lines correspond to the velocity control volumes, numbered as $i-1$, i, $i+1$ and $j-1$, j, $j+1$.

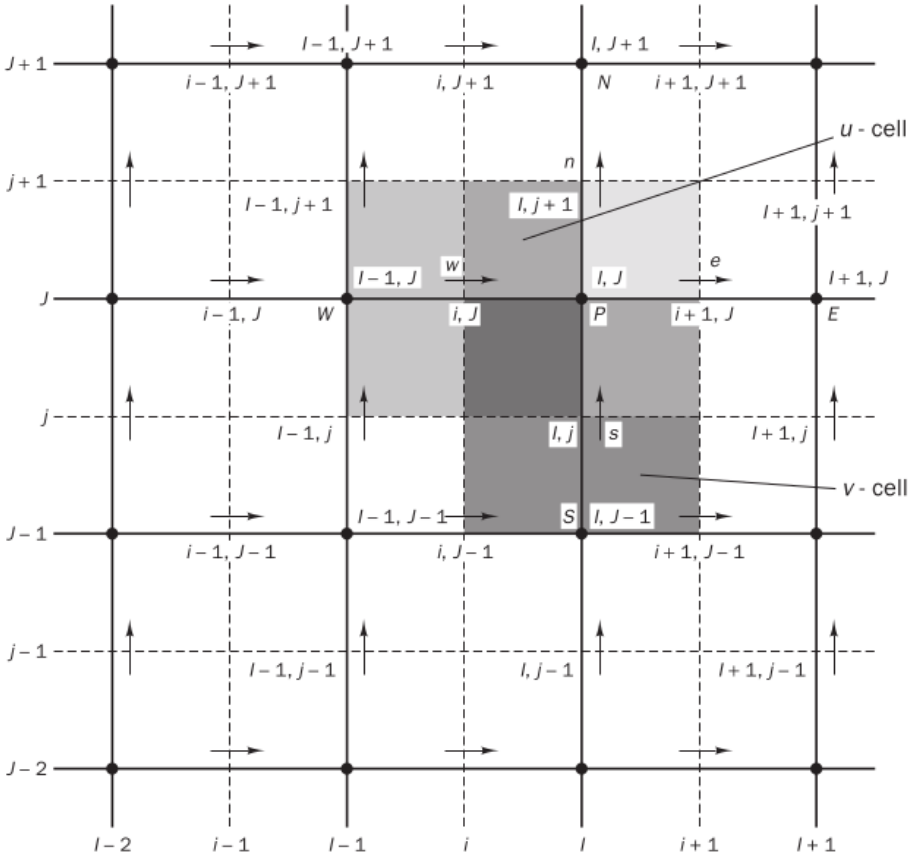


Figure 2.7: Representation of a staggered mesh.

The discretized momentum equation is given by:

$$a_{i,J}u_{i,J} = \sum a_{nb}u_{nb} + \left(p_{I-1,J} - p_{I,J}\right)A_{i,J} + b_{i,J}, \tag{2.19a}$$

$$a_{I,j}v_{I,j} = \sum a_{nb}v_{nb} + \left(p_{I,J-1} - p_{I,J}\right)A_{I,j} + b_{I,j}, \tag{2.19b}$$

where $a_{i,J}$ and a_{nb} can be calculated with any of the differencing methods. $\sum a_{nb}u^*_{nb}$ involve the summation at the four different faces that bound the pressure control volume: $(i-1,J)$, $(i+1,J)$, $(i,J-1)$ and $(i,J+1)$. $b_{I,j}$ is the source term of the momentum equation.

Considering a two-dimensional steady state flow, the algorithm starts with a pressure field guessed p^*. Using this guess, the discretized momentum equation is solved as:

$$a_{i,J}u^*_{i,J} = \sum a_{nb}u^*_{nb} + \left(p^*_{I-1,J} - p^*_{I,J}\right)A_{i,J} + b_{i,J}, \tag{2.20a}$$

$$a_{I,j}v^*_{I,j} = \sum a_{nb}v^*_{nb} + \left(p^*_{I,J-1} - p^*_{I,J}\right)A_{I,j} + b_{I,j}. \tag{2.20b}$$

Then the correction of the pressure equation, p', is defined as the difference between the guessed and correct field:

$$p = p' + p^*. \tag{2.21}$$

The same is valid for the velocity field:

$$u = u' + u^*, \tag{2.22a}$$

$$v = v' + v^*. \tag{2.22b}$$

Subtracting the Equation 2.19 to Equation 2.31 gives:

$$a_{i,J}\left(u_{i,J} - u^*_{i,J}\right) = \sum a_{nb}\left(u_{nb} - u^*_{nb}\right) + \left[\left(p_{I-1,J} - p^*_{I-1,J}\right) - \left(p_{I,J} - p^*_{I,J}\right)\right]A_{i,J}, \tag{2.23a}$$

$$a_{I,j}\left(v_{I,j} - v^*_{I,j}\right) = \sum a_{nb}\left(v_{nb} - v^*_{nb}\right) + \left[\left(p_{I,J-1} - p^*_{I,J-1}\right) - \left(p_{I,J} - p^*_{I,J}\right)\right]A_{I,j}. \tag{2.23b}$$

Using the corrections:

$$a_{i,J}u'_{i,J} = \sum a_{nb}u'_{nb} + \left(p'_{I-1,J} - p'_{I,J}\right)A_{i,J}, \tag{2.24a}$$

$$a_{I,j}v'_{I,j} = \sum a_{nb}v'_{nb} + \left(p'_{I,J-1} - p'_{I,J}\right)A_{I,j}. \tag{2.24b}$$

At this point, the SIMPLE algorithm introduces an approximation that discards the summations $\sum a_{nb}u'_{nb}$ and $\sum a_{nb}v'_{nb}$, leading to the simplified expressions:

$$u'_{i,J} = d_{i,J}\left(p'_{I-1,J} - p'_{I,J}\right), \tag{2.25a}$$

$$v'_{I,j} = d_{I,j}\left(p'_{I,J-1} - p'_{I,J}\right), \tag{2.25b}$$

where $d_{i,J} = \frac{A_{i,J}}{a_{i,J}}$ and $d_{I,j} = \frac{A_{J,i}}{a_{J,I}}$. Finally, the momentum equation is solved substituting the values into the correct velocity values:

$$u_{i,J} = u^*_{i,J} + u'_{i,J} = u^*_{i,J} + d_{i,J}\left(p'_{I-1,J} - p'_{I,J}\right), \tag{2.26a}$$

$$v_{I,j} = v^*_{I,j} + v'_{I,j} = v^*_{I,j} + d_{I,j}\left(p'_{I,J-1} - p'_{I,J}\right). \tag{2.26b}$$

So far only the momentum equation have been considered. In order to verify also the continuity equation, the following discretization should be satisfied:

$$[(\rho u A)_{i+1,J}) - (\rho u A)_{i,J})] + [(\rho u A)_{I,j+1}) - (\rho u A)_{I,j})] = 0. \tag{2.27}$$

Substituting the values of the correct velocity and rearranging the equation, a relation between the pressure corrections p' is obtained. Solving it, the pressure correction field is extracted, thus the correct pressure p is easily computed as $p = p' + p^*$. As said before, the SIMPLE algorithm uses under-relaxation factors to prevent divergence of the solution. Thus the new pressure is corrected as:

$$p^{new} = p^* + \alpha_p p', \tag{2.28}$$

where α_p is the relaxation factor. Also the velocity field is under-relaxed through the factors α_u and α_v.

In Figure 2.8 the whole algorithm is summarized. After the calculation of the corrected values, if the fields did not converge, a new loop starts with a guess of the variables equal to the output of the previous iteration.

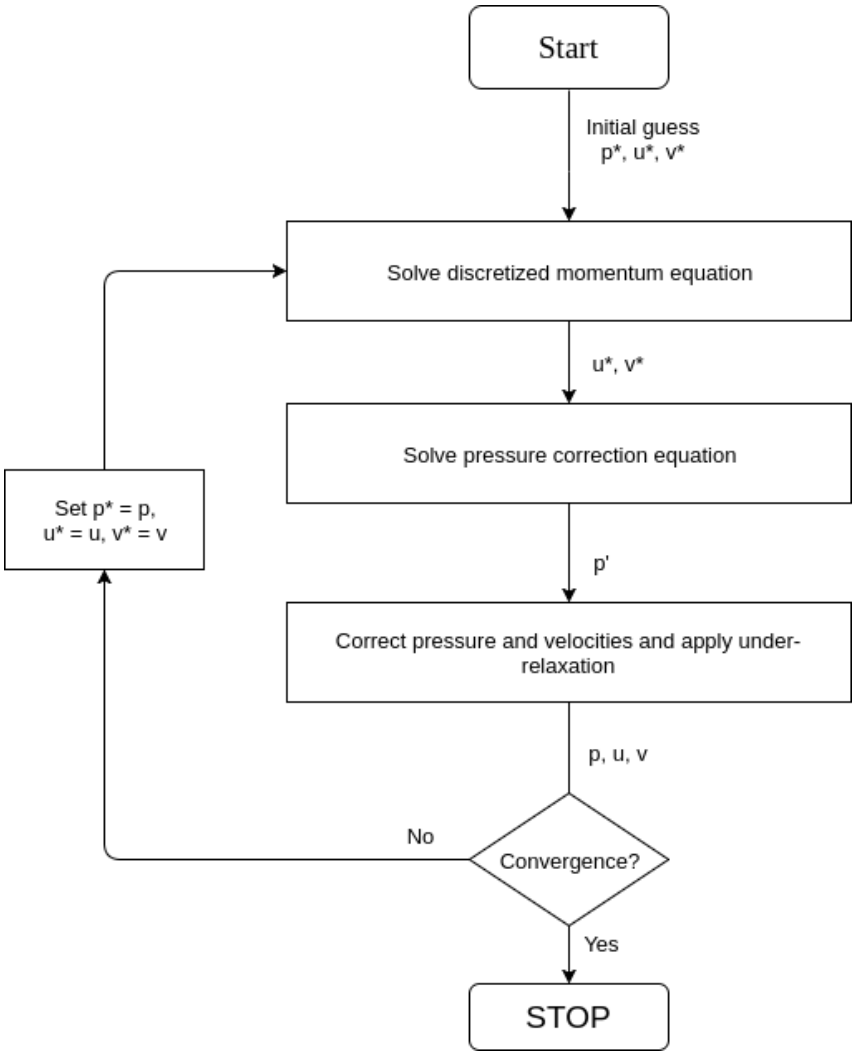


Figure 2.8: Simple algorithm flowchart.

The PISO algorithm can be seen as en extension of SIMPLE. It involves one predictor step and two corrector steps, one more than the steady-state SIMPLE. The corrector and first predictor steps are the same as the SIMPLE algorithm, thus the intermediate fields p^*, u^* and v^* are computed and the continuity equation is satisfied. Since there is a further correction step, the output fields of the first corrector are in a different notation:

$$p^{**} = p' + p^*, \tag{2.29a}$$

$$u^{**} = u' + u^*, \tag{2.29b}$$

$$v^{**} = v' + v^*. \tag{2.29c}$$

The corrected velocities are:

$$u^{**}_{i,J} = u^{*}_{i,J} + u'_{i,J} = u^{*}_{i,J} + d_{i,J}\left(p'_{I-1,J} - p'_{I,J}\right), \tag{2.30a}$$

$$v^{**}_{I,j} = v^{*}_{I,j} + v'_{I,j} = v^{*}_{I,j} + d_{I,j}\left(p'_{I,J-1} - p'_{I,J}\right). \tag{2.30b}$$

As before, substituting the values of the correct velocity in the continuity equation, p' field is obtained. In the case of PISO, this is defined as the first pressure correction equation. Once the correction is computed, u^{**} and v^{**} are obtained from Equation 2.30.

The second corrector step, introduces the twice corrected velocity field (u^{***} and v^{***}) that is calculated with the new momentum equation:

$$a_{i,J}u^{***}_{i,J} = \sum a_{nb}u^{**}_{nb} + \left(p^{**}_{I-1,J} - p^{**}_{I,J}\right)A_{i,J} + b_{i,J}, \tag{2.31a}$$

$$a_{I,j}v^{***}_{I,j} = \sum a_{nb}v^{**}_{nb} + \left(p^{**}_{I,J-1} - p^{**}_{I,J}\right)A_{I,j} + b_{I,j}. \tag{2.31b}$$

Then the new velocity field is computed and substituted, as previously, in the continuity equation that leads to the new pressure correction p''. Thus the corrected pressure is:

$$p^{***} = p^{**} + p'' = p^{*} + p' + p''. \tag{2.32}$$

Coming to the PIMPLE algorithm, it can be seen as a combination of the SIMPLE and PISO algorithm. The big advantage is the opportunity to use larger time-steps, due to the absence of limitations on the Courant number. This can considerably speed up the computational time. Holzmann [29] describes it as a solver that first finds a steady-state solution within the time-step, using under-relaxation factors, then employs outer correction loops to integrate in time and ensure the convergence.

It starts with a *Momentum predictor* that solves the velocity field, using the pressures coming from the previous iteration. Then a *Pressure corrector* step solve Poisson equation to correct the fluxes. Lastly, a new *Momentum corrector* phase update the velocity field, using the corrected pressure. These three steps are executed till the convergence is reached and the run time can go on. The flowchart of the algorithm is graphically recalled in Figure 2.9.

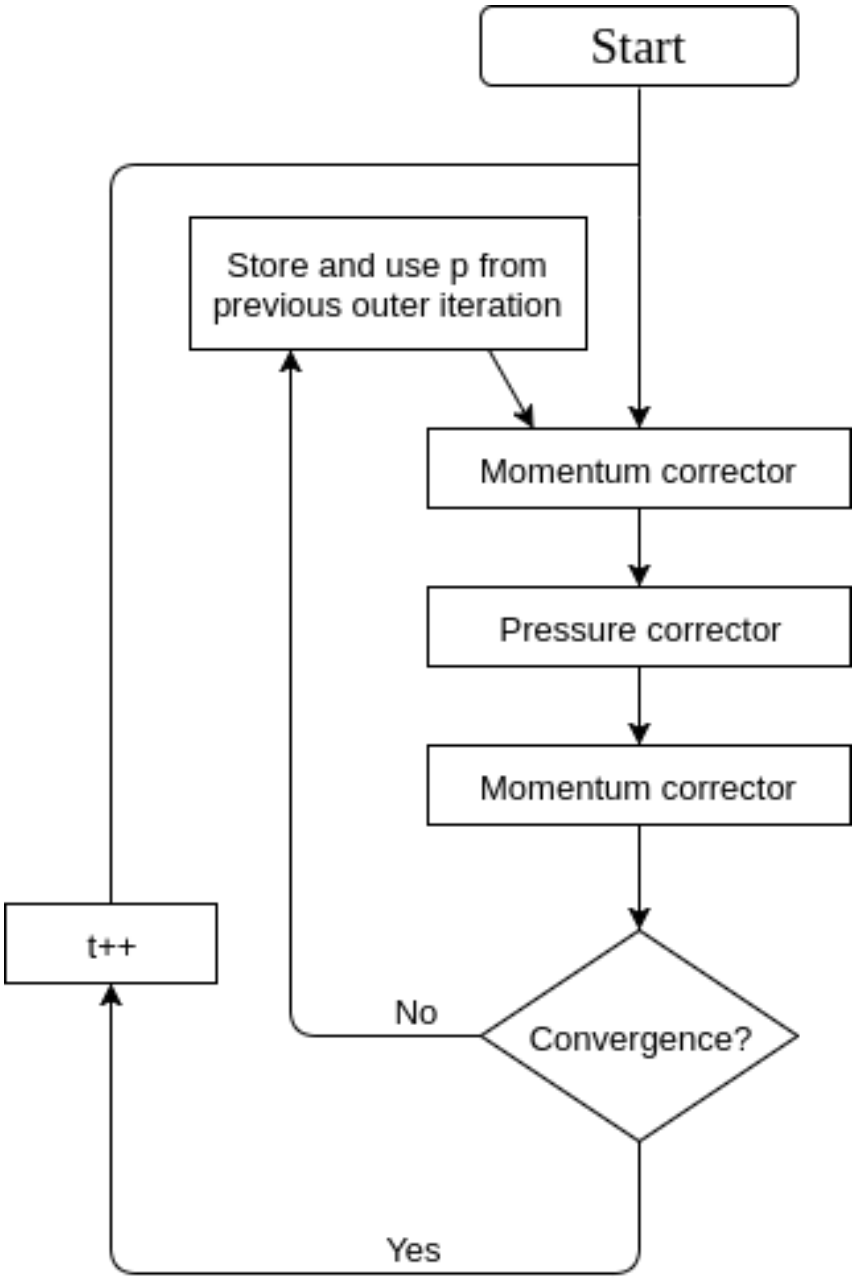


Figure 2.9: Pimple algorithm flowchart.

2.4.3 Structure equations

The structure is elastic and compressible. The corresponding properties are declared as Young modulus E, Poisson ratio ν^s and the shear modulus μ^s. The corresponding equation is [27]:

$$\rho^s \frac{\partial \mathbf{v}^s}{\partial t} + \rho^s \left(\nabla \mathbf{v}^s\right) \mathbf{v}^s = \nabla \cdot \sigma^s + \rho^s \mathbf{g}, \tag{2.33}$$

where **g** is the gravity vector. Traditionally a finite element approach is employed in structural modeling, while a finite volume method is used in fluid dynamics, because of the complicated constitutive equations and the high deformations that the fluid continuum encounters in time. The open-source *Calculix* is a

finite element solver, based on the Lagrangian formulation of the conservation law, that is:

$$\rho^s \frac{\partial^2 \mathbf{v}^s}{\partial t^2} = \nabla \cdot \left(J\sigma^s \mathbf{F}^{-T} \right) + \rho^s \mathbf{g}, \tag{2.34}$$

where J is the jacobian, $F = I + \nabla u^s$ the gradient deformation tensor and I the identity matrix. The corresponding constitutive equation is:

$$\sigma^s = \frac{1}{J} \mathbf{F} \left(\lambda^s \left(tr\mathbf{E} \right) \mathbf{I} + 2\mu^s \mathbf{E} \right) \mathbf{F}^T, \tag{2.35}$$

where λ^s adn μ^s are the two Lamé coefficient. For sake of clarity, the equation will be simplified into the following:

$$[M]\{\ddot{\mathbf{u}}\}\} + [K]\{\mathbf{u}\} = \{\mathbf{F}\}^{ext}, \tag{2.36}$$

where $[M]$ is the mass matrix, $[K]$ the stiffness matrix, $\{\ddot{u}\}$ the nodal acceleration vector, $\{u\}$ the nodal displacements vector and $\{F\}^{ext}$ the external forces vector. Given the boundary conditions:

$$\{\mathbf{u}\}_{t=0} = \{\mathbf{u}\}_0 \qquad initial\ displacements, \tag{2.37a}$$

$$\{\dot{\mathbf{u}}\}_{t=0} = \{\dot{\mathbf{u}}\}_0 \qquad initial\ velocities, \tag{2.37b}$$

the integration method should find the displacements that satisfy Equation 2.36. The solution method implemented in $CalculiX$ is defined as $\alpha - method$ [31]. The space domain is discretized with finite elements method, while the integration in time relies on finite differences, thus the fields are calculated in discrete time, starting from a simple time integration of the acceleration:

$$\mathbf{v}_{n+1} = \mathbf{v}_n + \int_{t_n}^{t_{n+1}} \mathbf{a}(\xi) d\xi, \tag{2.38}$$

where $\mathbf{a} = \ddot{u}$ the acceleration and $\mathbf{v} = \dot{u}$ the velocity. Being the acceleration function $\mathbf{a}(\xi)$ unknown, a solution can be reached by a linear combination of the known values in the discrete finite difference scheme, thus at $t = t_n$ and $t = t_{n+1}$:

$$\mathbf{a}(t) \approx (1 - \gamma)\mathbf{a}_n + \gamma \mathbf{a}_{n+1}. \tag{2.39}$$

The integration leads to:

$$\mathbf{v}_{n+1} = \mathbf{v}_n + \Delta t[(1 - \gamma)\mathbf{a}_n + \gamma \mathbf{a}_{n+1}]. \tag{2.40}$$

Applying the same procedure for the displacement:

$$\mathbf{u}_{n+1} = \mathbf{u}_n + \int_{t_n}^{t_{n+1}} \mathbf{v}(\eta) d\eta = \mathbf{u}_n + \mathbf{v}_n(t_{n+1} - t_n) + \int_{t_n}^{t_{n+1}} \int_{t_n}^{t_{n+1}} \mathbf{a}(\xi) d\xi d\eta. \tag{2.41}$$

Again, acceleration is interpolated using the known values (in general $2\beta \neq \gamma$):

$$\mathbf{a}(t) = (1-2\beta)\mathbf{a}_n + 2\beta\mathbf{a}_{n+1}, \tag{2.42}$$

thus one obtain:

$$\mathbf{u}_{n+1} = \mathbf{u}_n + \mathbf{v}_n(t_{n+1} - t_n) + 1/2(t_{n+1} - t_n)^2[(1-2\beta)\mathbf{a}_n + 2\beta\mathbf{a}_{n+1}]. \tag{2.43}$$

Indicating the time interval as $\Delta t = t_{n+1} - t_n$, the velocity vector as $\{\mathbf{V}\} = \{\dot{\mathbf{u}}\}$ and the acceleration vector $\{\mathbf{A}\} = \{\ddot{\mathbf{u}}\}$, with $\{\mathbf{A}\}_{n+1} = \{\mathbf{A}\}_n + \{\Delta\mathbf{A}\}$, Equation 2.40 and Equation 2.43 can be rewritten as:

$$\{\mathbf{V}\}_{n+1} = \{\mathbf{V}\}_n + \Delta t[(1-\gamma)\{A\}_n + \gamma\{A\}_{n+1}], \tag{2.44a}$$

$$\{\mathbf{U}\}_{n+1} = \{\mathbf{U}\}_n + \Delta t\{\mathbf{V}\}_n + 1/2\Delta t^2[(1-2\beta)\{A\}_n + 2\beta\{A\}_{n+1}]. \tag{2.44b}$$

Equation 2.36 is satisfied at any instant, thus considering a friction term, the new relation is:

$$[M]\{\mathbf{A}\}_{n+1} + [K]\{\mathbf{V}\}_{n+1} + [C]\{\mathbf{V}\}_{n+1} = \{\mathbf{F}\}^{ext}_{n+1}. \tag{2.45}$$

An option could be the substitution of the velocity and displacement vector in this expression, however the experience showed that it is much better, in terms of high frequency dissipation, to evaluate all the terms, except the acceleration, in an intermediate time between t_n and t_{n+1}:

$$[M]\{\mathbf{A}\}_{n+1} + (1+\alpha)[C]\{\mathbf{V}\}_{n+1} - \alpha[C]\{\mathbf{V}\}_n + (1+\alpha)[K]\{\mathbf{U}\}_{n+1} - \alpha\{\mathbf{U}\}_n = (1+\alpha)\{\mathbf{F}\}^{ext}_{n+1} - \alpha\{\mathbf{F}\}^{ext}_n, \tag{2.46}$$

with $-1 \leq \alpha \leq 0$. Considering the stiffness and friction term as internal forces:

$$[M]\{\mathbf{A}\}_{n+1} + (1+\alpha)[\mathbf{F}]^{int}_{n+1} - \alpha[\mathbf{F}]^{int}_{n+1} = (1+\alpha)\{\mathbf{F}\}^{ext}_{n+1} - \alpha\{\mathbf{F}\}^{ext}_n. \tag{2.47}$$

Now, substituting Equation 2.44 into Equation 2.47, the $\alpha - method$ is obtained. It can be proven that if $\beta = 1/4(1-\alpha)^2$ and $\gamma = 1/2 - \alpha$, the method is second-order accurate, consistent and unconditionally stable for $\alpha \in [-1/3, 0]$ [31]. The method can be extended to nonlinear problems, where $[C]$ and $[K]$ are nonlinear functions of $\{\mathbf{U}\}$ and $\{\mathbf{V}\}$.

2.4.4 Test conditions

The initial condition is a flow with velocity equal to zero. The boundary condition on the inlet is a parabolic velocity profile with a smooth increase in time, starting from a null velocity. The parabolic profile has a mean velocity $\bar{U}$. The upper and lower boundaries, as well as the cylinder and bar patches, have a no-slip condition.

Three different benchmarks have been proposed by Turek and Hron [27]. The first results in a steady state solution, while the others show a periodic behavior in time. For the purpose of this comparison,

the *FSI3* case was reproduced. The values for the fluid and structure properties and boundary condition are recalled in Table 2.1 and Table 2.2. For this test, the Reynolds number is:

$$Re = \frac{2r\bar{U}}{\nu^f} = 200.$$

Table 2.1: Structure constants.

Structure	
$\rho^s[10^3 \frac{kg}{m^3}]$	1
$E[10^6 \frac{N}{m^2}]$	5.6
ν^s	0.4

Table 2.2: Fluid constants.

Fluid	
$\rho^f[10^3 \frac{kg}{m^3}]$	1
$\nu^f[10^{-3} \frac{m^2}{s}]$	1
$\bar{U}[\frac{m}{s}]$	2

Many computational studies investigated the transition of the wake of the flow passing a circular cylinder [32]. It is well known that, even if the boundary conditions are steady, the flow is not and shows periodic oscillations. To capture this flow motion, shown in Figure 2.11, it was necessary to increase the mesh resolution. The domain has been discretized with a fully structured mesh for both structure and fluid, respectively they have 35600 and 400 hexahedrons. Because OpenFOAM works only with 3D meshes, the 2D mesh shown in Figure 2.10 was extruded in the third direction using a single element. Because of the larger displacements of the structure, which require a finer mesh, the dimension of the elements closer to the structure is higher than the ones closer to the outlet section, through a mesh size scaling in the flow direction.

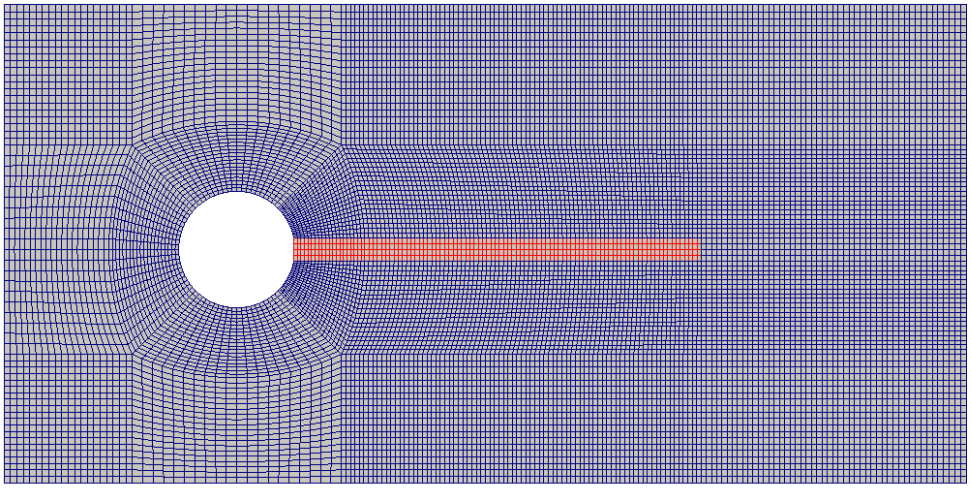


Figure 2.10: Fluid (in blue) and structural (in red) mesh.

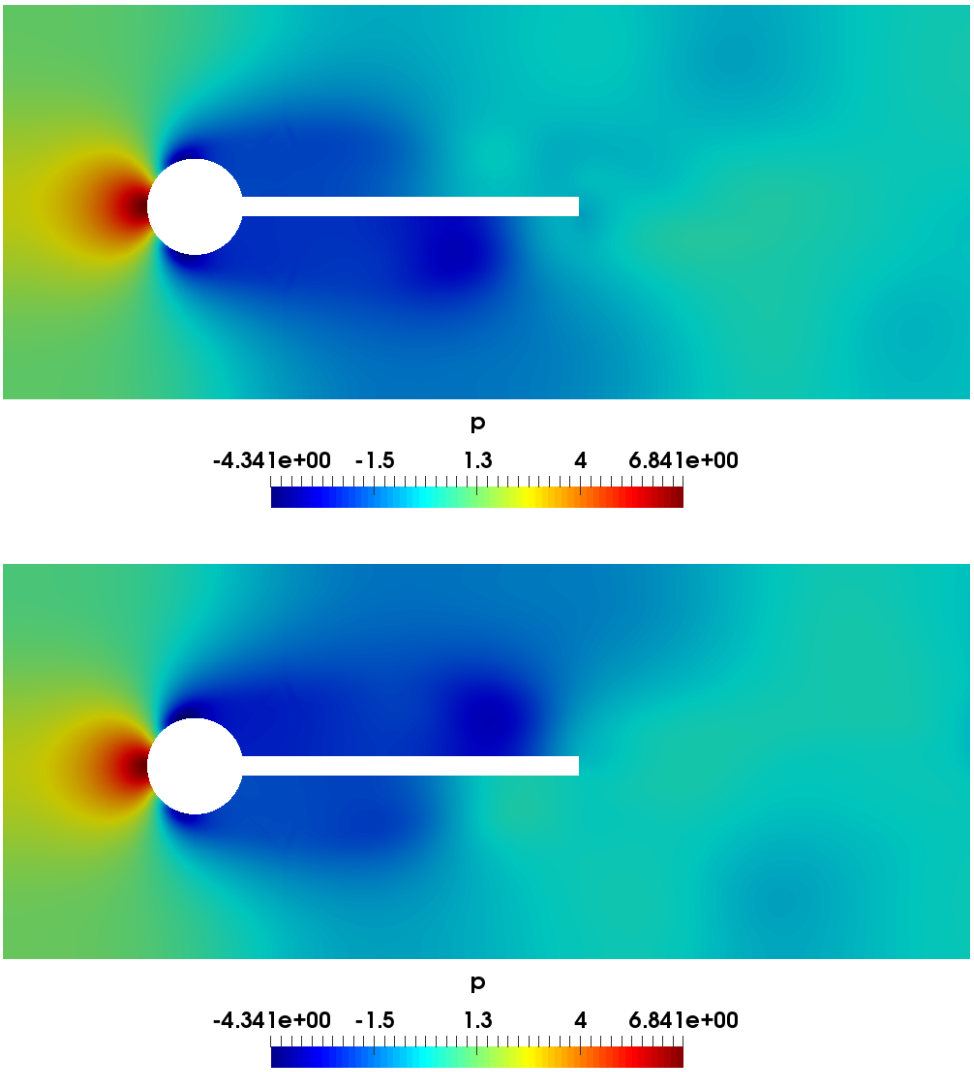


Figure 2.11: Wave transient pressure field in static mesh test, captured in two different moments (in SI base unit). The pressure is referred as the incompressible value used in *OpenFOAM*, p/ρ.

To ensure the simulation to be 2D, in OpenFOAM a special a boundary condition, called $empty$, was used. It is used for reduced dimensions' cases, in patches whose normal is aligned with a geometrical direction that does not represent solution direction [33]. Regarding the structural solver, it was sufficient to impose a null displacement on all points in the extrusion direction and a constraint on the displacements in all directions for the points attached to the cylinder. To mesh the structure, 8-point linear brick elements have been employed. The linear brick elements is an isotropic three-dimensional element characterized by linear shape functions in all directions [34], where each node has three degree of freedom. The mapping of the element into a general hexahedron through the natural coordinates ξ, η and μ, as in Figure 2.12.

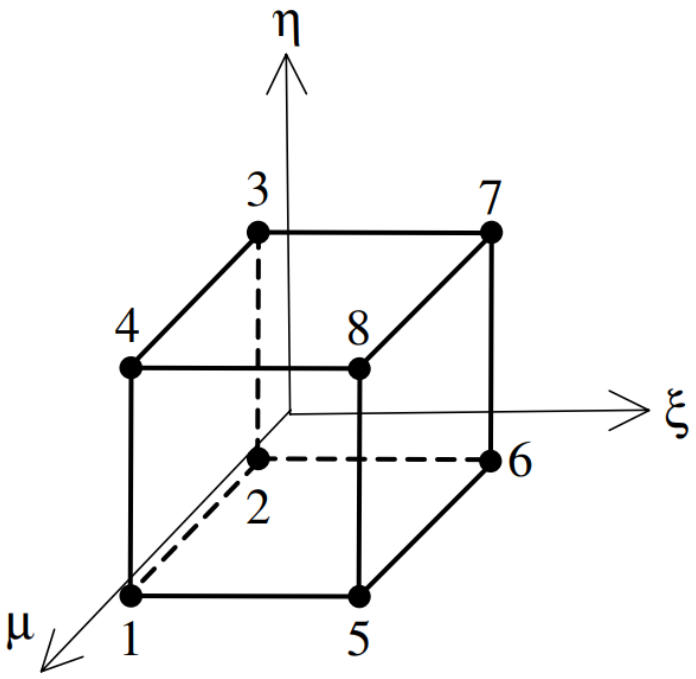


Figure 2.12: Linear brick element in natural coordinates.

The corresponding eight shape functions are:

- $N_1 = \frac{1}{8}(1-\xi)(1-\eta)(1+\mu)$
- $N_2 = \frac{1}{8}(1-\xi)(1-\eta)(1-\mu)$
- $N_3 = \frac{1}{8}(1-\xi)(1+\eta)(1-\mu)$
- $N_4 = \frac{1}{8}(1-\xi)(1+\eta)(1+\mu)$
- $N_5 = \frac{1}{8}(1+\xi)(1-\eta)(1+\mu)$
- $N_6 = \frac{1}{8}(1+\xi)(1-\eta)(1-\mu)$
- $N_7 = \frac{1}{8}(1+\xi)(1+\eta)(1-\mu)$
- $N_8 = \frac{1}{8}(1+\xi)(1+\eta)(1+\mu)$

First a structural test is executed in order to validate the mesh and scheme adopted in the structural solver. In this case the load is given by gravitational force only, specifying the gravity vector **g**. The values for this test are reported in Table 2.3.

Table 2.3: Structural test parameter.

$\rho^s[10^3 \frac{kg}{m^3}]$	1
$E^s[10^6 \frac{kg}{ms^2}]$	1.4
ν^s	0.4

The results shows an amplitude of $0.13m$, equal to the one of Turek and Hron [27], while the frequency of $1.07Hz$ is slightly lower than the benchmark, that is $1.09Hz$. The results are shown in Figure 2.13.

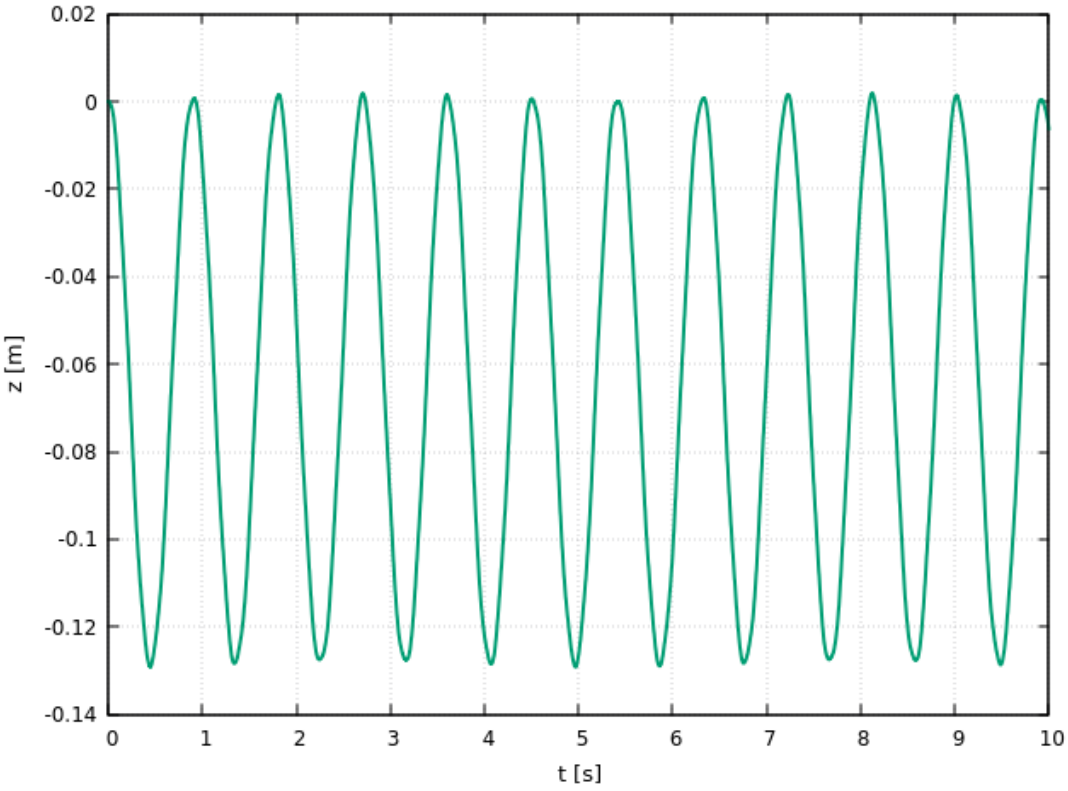


Figure 2.13: Vertical displacement of A in structural test.

2.4.5 Results

The results obtained with the FSI algorithm test are shown in Figure 2.14. The amplitude registered is $0.0681m$ and the frequency is $5.55Hz$. A comparison with the Turek's value are in Table 2.4.

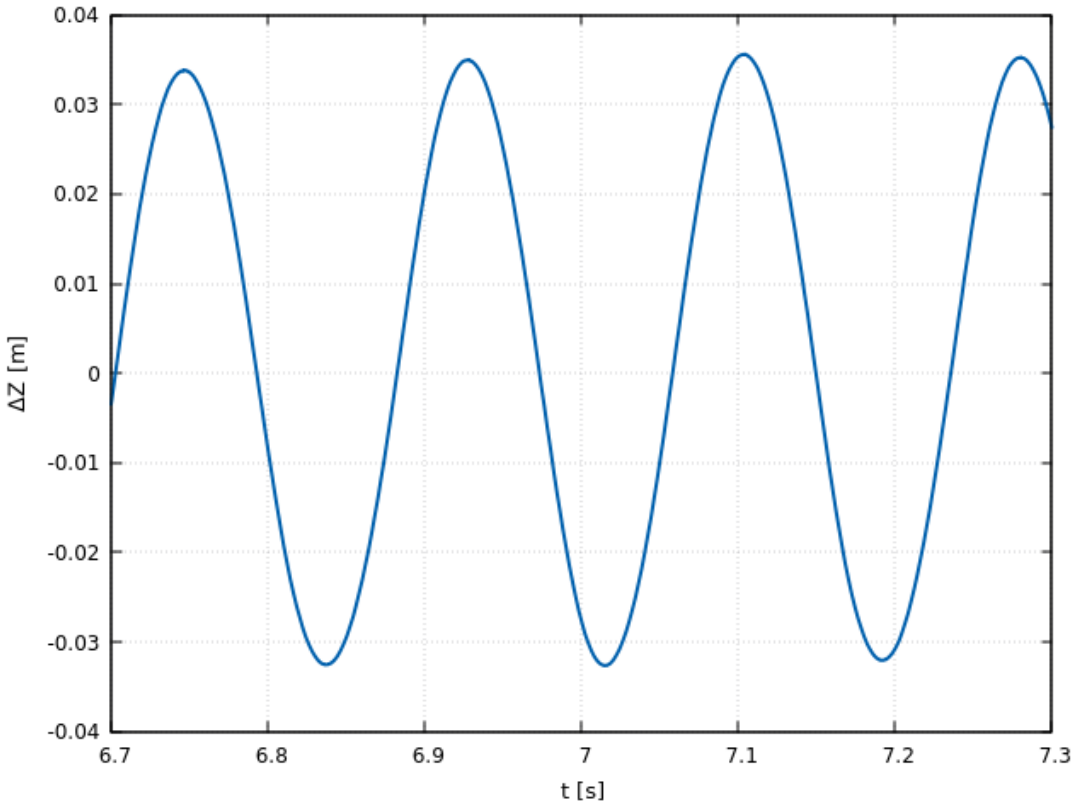
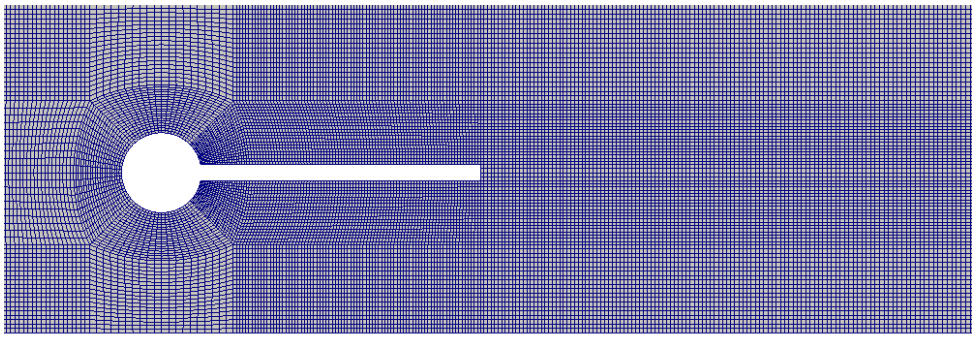


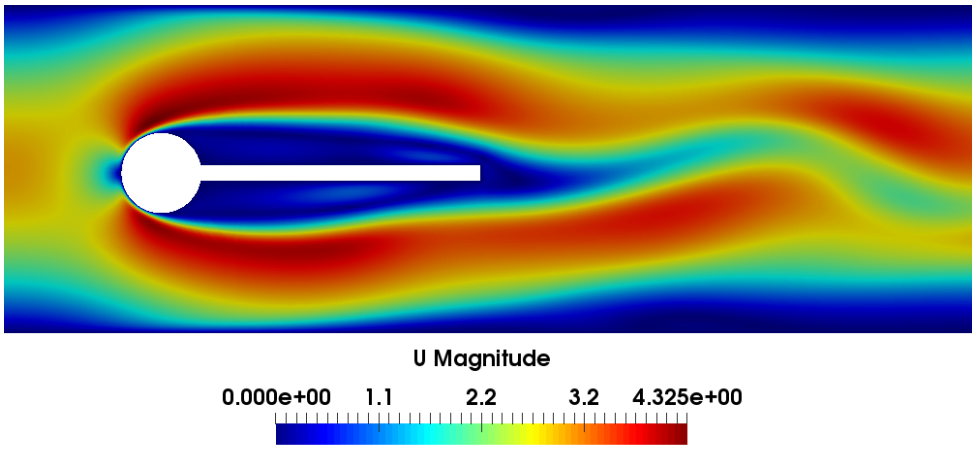
Figure 2.14: Vertical displacement of A in FSI test.

Table 2.4: Comparison of benchmark's results [27].

	Calculated	**Turek**	**Error**
$\Delta z[m]$	0.0681	0.0684	0.44%
$Frequency[Hz]$	5.55	5.47	1.4%

This chapter ends with the graphical representation of the pressure, velocity and mesh fields, at the beginning of the simulation in Figure 2.15, when the tip has maximum vertical negative displacement in Figure 2.16 and when it has the maximum vertical positive displacement in Figure 2.17.





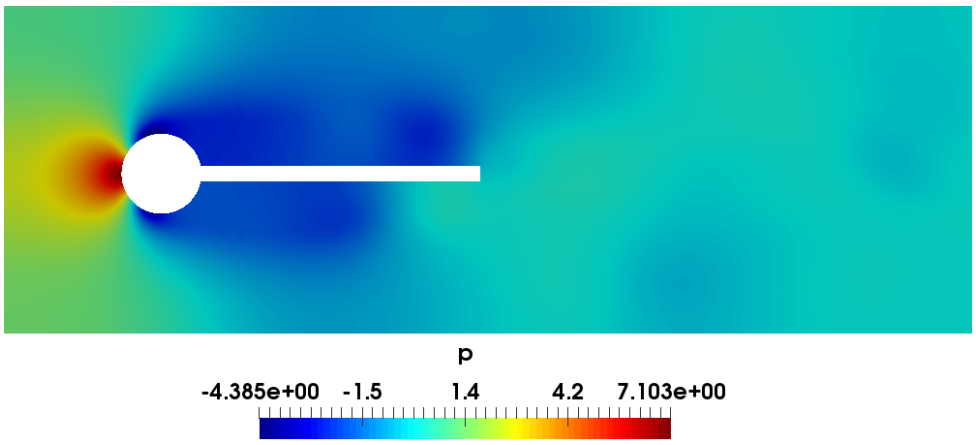
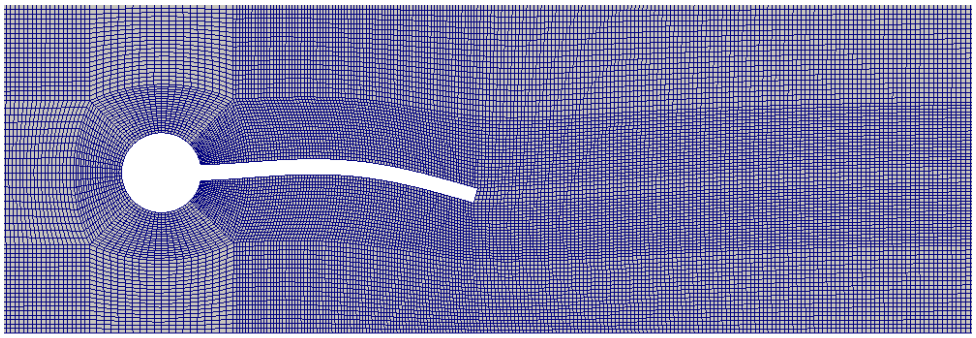
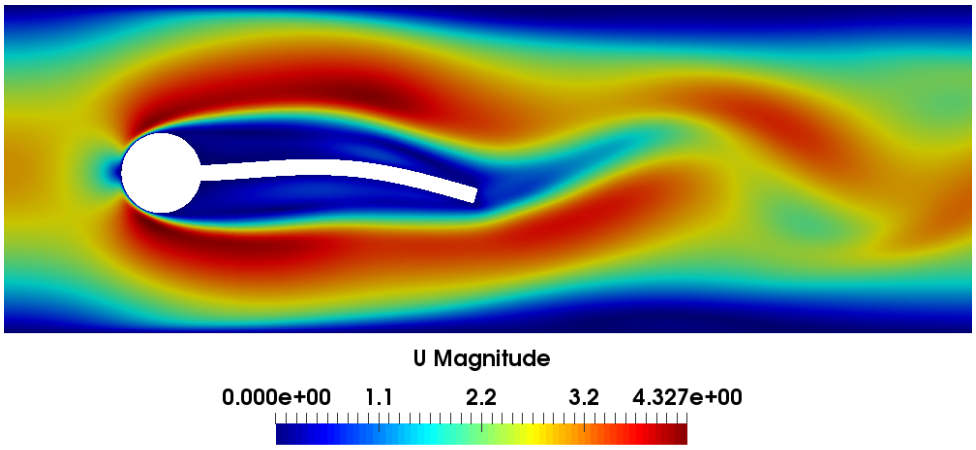


Figure 2.15: Mesh, velocity and pressure fields at the beginning of the simulation. Velocity and pressure in SI base unit. The pressure is referred as the incompressible value used in *OpenFOAM*, p/ρ.





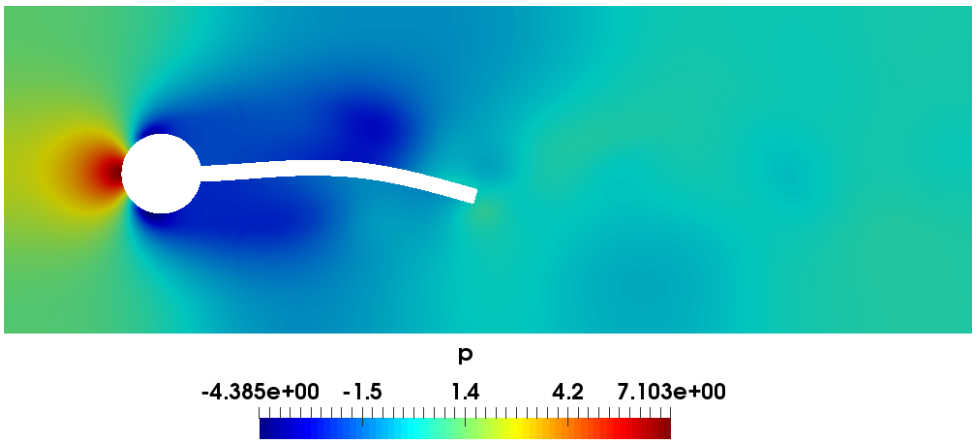
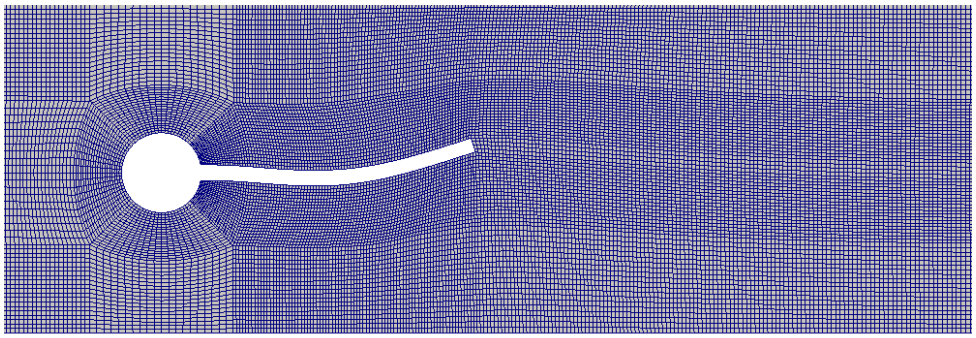
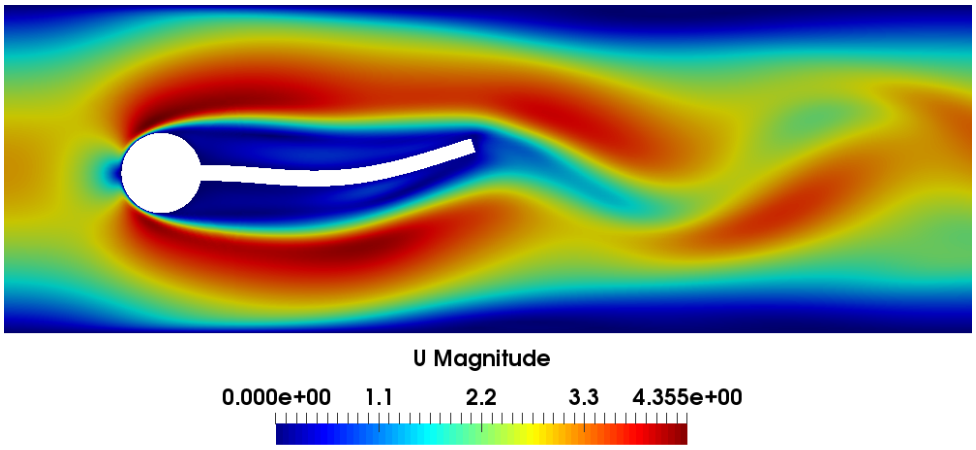


Figure 2.16: Mesh, velocity and pressure fields when the tip has maximum vertical negative displacement. Velocity and pressure in SI base unit. The pressure is referred as the incompressible value used in *OpenFOAM*, p/ρ.





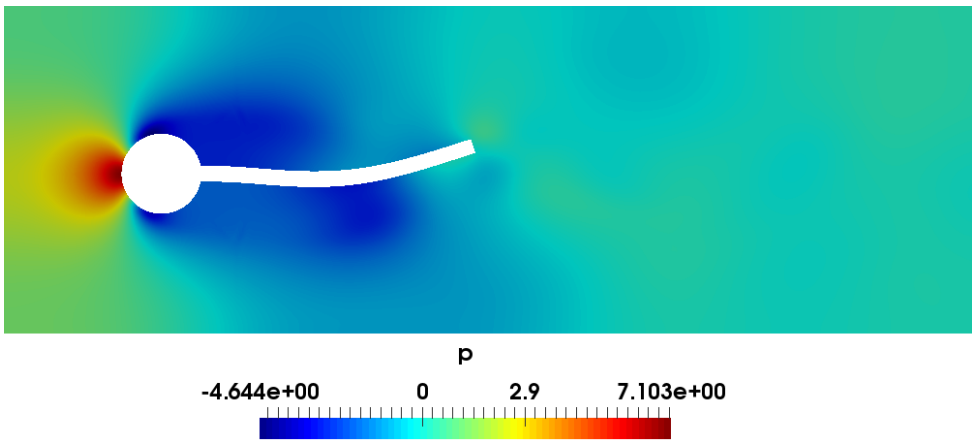


Figure 2.17: Mesh, velocity and pressure fields when the tip has maximum vertical positive displacement. Velocity and pressure in SI base unit. The pressure is referred as the incompressible value used in *OpenFOAM*, p/ρ.

Chapter 3

Transonic Wing Model

The purpose of this chapter is to explain the configuration of the wing studied and show some insight on how the results have been obtained. It starts with a brief description of the geometry, then the second section highlights the differences between this case and the incompressible flow of section 2.4. In the end, a structural model of the wing is presented.

3.1 Wing geometry

The wing considered in this work is part of the FP7-NOVEMOR project (Novel Air Vehicle Configurations: From Fluttering Wings to Morphing Flight) [35]. The Reference Aircraft is a regional jet with 113 PAX in a single economic class and provides operational flexibility to fly different missions at the transonic regime.

The wing geometry was defined using three airfoils, distributed span-wisely. The first is at the root, then one is located at the trailing-edge discontinuity, also known as wing break, and finally one at the tip. The points that describe the airfoils have been interpolated using splines. The positions of the sections, as well as the respective chords, are presented in Table 3.1,where Y is the span direction. The wing was designed with a loft that joins the curves defined previously. A graphical representation is illustrated in Figure 3.1 (top view) and Figure 3.2 (front view).

Table 3.1: Geometrical data of the wing.

Station	Chord $[m]$	$Y[m]$
Root	7.57	0
Break	3.99	6.18
Tip	1.73	15.73

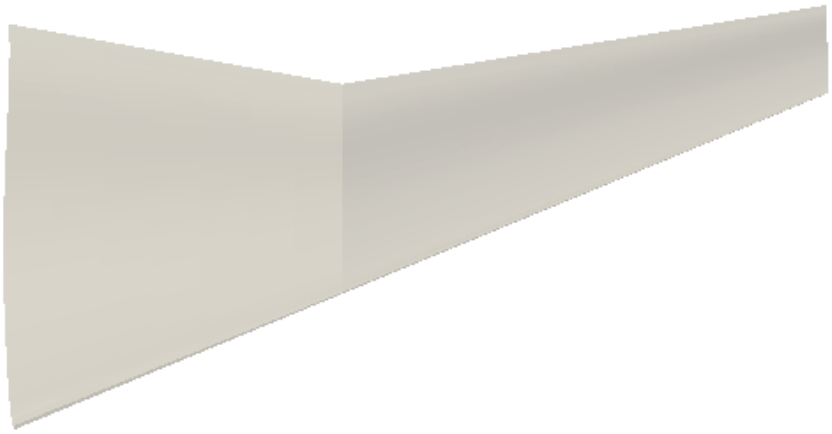


Figure 3.1: Top view of the transonic wing.

Figure 3.2: Front view of the transonic wing.

The geometrical model, drawn in a CAD software, has a reference area $S_w = 63.8m^2$. This will be later used to calculate the force coefficients.

3.2 Aerodynamic Model

3.2.1 Fluid solver

The flight condition is transonic, the Mach number is $M = 0.78$ and the altitude $h = 38000$ft. The standard atmosphere has been considered to evaluate the density, temperature and static pressure at this altitude [36]; the values are recalled in Table 3.2.

Table 3.2: Flight conditions.

Altitude $[ft]$	Mach	Pressure $[Pa]$	Temperature $[K]$	Speed of sound $[m/s]$	Density $[kg/m^3]$
38000	0.78	20646	217	295	0.332

The development of new fluid solvers is mainly driven by the need of simulating large geometries

and model complex physical behavior. For these reason, differently to the SIMPLE scheme explained in subsection 2.4.2, a "colocated" mesh is preferred. With this solution, all the scalar and vector fields are stored in the same points, giving to user much more freedom in the mesh generation phase.

To model the flow, a new solver is required since the compressibility can not be neglected in this case, and discontinuities, such as shocks, must be captured. $OpenFOAM$'s suite includes different applications designed for high speed flows that support mesh motion. Gutierrez et al. [37] compared the pressure-based $sonicFoam$ with the density-based $rhoCentralFoam$, in a set of different cases that includes a flow over a wedge, a diamond airfoil and a two dimensional blunt body. The main difference is that $sonicFoam$ employs a PISO method, using pressure and velocity as dependent variables, while $rhoCentralFoam$ introduces an alternative approach to Riemann solvers based on the central upwind scheme. In the tests, $rhoCentralFoam$ demonstrated better performance. In fact, to reproduce results of the same quality, $sonicFoam$ requires three times more cells than its competitor. Moreover comparing them with the same amount of cells, the computational time is greater with the pressure-based solver. For these reasons, $rhoCentralFoam$ have been preferred for the transonic wing study.

In order to model the presence of strong discontinuities in high speed flows, avoiding spurious oscillations, different methods have been proposed. Most of them, including the monotone upstream-centred schemes for conservation laws, Piecewise Parabolic Method (PPM), Essentially Non-Oscillatory (ENO) schemes, Weighted ENO (WENO) schemes and the Runge–Kutta Discontinuous Galerkin (RKDG) method, make use of Riemann solvers, characteristic decomposition and Jacobian evaluation, making them complex and difficult to implement in a colocated polyhedral framework [38]. A new alternative, the so-called central scheme, has been introduced by Greenshields et al. [38] and implemented in the $OpenFOAM$ solver $rhoCentralFoam$.

The governing equations, in Eulerian frame are recalled in Equation 3.1.

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\mathbf{v}\rho) = 0 \qquad mass, \tag{3.1a}$$

$$\frac{\partial \rho \mathbf{v}}{\partial t} + \nabla \cdot [\mathbf{v}\,(\rho \mathbf{v})] + \nabla p + \nabla \cdot \mathbf{T} = 0 \qquad momentum, \tag{3.1b}$$

$$\frac{\partial \rho E}{\partial t} + \nabla \cdot [\mathbf{v}\,(\rho E)] + \nabla \cdot (\mathbf{v}p) + \nabla \cdot (\mathbf{T} \cdot \mathbf{u}) + \nabla \cdot \mathbf{j} = 0 \qquad totalenergy, \tag{3.1c}$$

where E is the total energy, **j** is the diffusive heat flux, **T** is the viscous stress tensor [38] defined as,

$$\mathbf{T} = -2\mu\, dev(\mathbf{D}), \tag{3.2}$$

being μ the dynamic viscosity, **D** the deformation gradient tensor

$$\mathbf{D} = \frac{1}{2}\left[\nabla \mathbf{u} + (\nabla \mathbf{u})^T\right], \tag{3.3}$$

and $dev(\mathbf{D})$ is the deviatoric component. The heat flux can be obtained through the Fourier's law:

$$\mathbf{j} = -k\nabla T, \tag{3.4}$$

where k is the conductivity and T the temperature. Substituting Equation 3.3 and Equation 3.4 in Equation 3.1, the Navier-Stokes equations are obtained.

In compressible flows the properties are transported not only by the flow, but also by the propagation of waves. Thus the computation of the fluxes needs to consider the transport in each direction. In $rhoCentralFoam$ the Kurganov and Tadmor [39] method is applied.

The convective terms of the conservation equations, $\nabla \cdot (\mathbf{v}\rho)$, $\nabla \cdot [\mathbf{v}\,(\rho\mathbf{v})]$, $\nabla \cdot [\mathbf{v}\,(\rho E)]$, and $\nabla \cdot (\mathbf{v}p)$ are integrated in each cell as:

$$\int_V \nabla \cdot (\mathbf{v}\Psi)\, dV = \sum_f (\mathbf{S}_f \cdot \mathbf{v}_f)\, \Psi_f \approx \sum_f \phi_f \Psi_f, \tag{3.5}$$

where the summation is executed on all the faces of the cell, $\mathbf{S}_f$ is the surface vector and ϕ_f is the volumetric flux that is the volume of fluid that goes through the face per second. Ψ_f is linearized splitting the flux in the direction of the incoming and outgoing cell as:

$$\phi_f \Psi_f = [\alpha \phi_{f+} \Psi_{f+} + (1-\alpha)\phi_{f-}\Psi_{f-} + \omega_f(\Psi_{f-} - \Psi_{f+})]\,, \tag{3.6}$$

where $f+$ and $f-$ refer to the incoming and outgoing direction, respectively. The first terms are flux evaluations at the interface, while the third is required in case the convective term is part of a total derivative. It consists of an additional diffusion term that uses the volumetric flux ω_f, related to the maximum speed of discontinuity propagation that could exist on the face between $f+$ and $f-$. The volumetric fluxes are evaluated as:

$$\varphi_{f+} = max\,(a_{f+} \mid \mathbf{S}_f \mid\, + \phi_{f+}\,,\; a_{f-} \mid \mathbf{S}_f \mid\, + \phi_{f-}\,,\; 0)\,, \tag{3.7a}$$

$$\varphi_{f-} = max\,(a_{f+} \mid \mathbf{S}_f \mid\, - \phi_{f+}\,,\; a_{f-} \mid \mathbf{S}_f \mid\, - \phi_{f-}\,,\; 0)\,, \tag{3.7b}$$

where a_f is the speed of sound on the face. In a central scheme, the weighting coefficient $\alpha = 0.5$, while in a central upwing scheme $\alpha = \frac{\varphi_{f+}}{(\varphi_{f+} - \varphi_{f+})}$. According to Greenshields et al. [38], the additional volumetric flux is calculated as:

$$\omega_f = \begin{cases} \alpha\; max\,(\varphi_{f+}\,,\; \varphi_{f-}) & \text{Kurganov and Tadomr scheme,} \\ \alpha\,(1-\alpha)\,(\varphi_{f+} + \varphi_{f-}) & \text{Kurganov, Noelle and Petrova scheme.} \end{cases}$$

Finally, a flux limiter function, $\beta(r)$, is employed to switch between low and high order schemes, ensuring the boundedness. It could be a van Leer, Minmod or van Albada limiter [39]. Thus the interpolation of the generic variable Ψ on the face is:

$$\Psi_{f+} = [1 - \beta(1-w_f)]\Psi_P + \beta(1-w_f)\Psi_N, \tag{3.8}$$

where P and N refer to the two cells that share the same face. The new weighting coefficient is given by:

$$w_f = \frac{\mid \mathbf{S}_f \cdot \mathbf{d}_{fN} \mid}{\mid \mathbf{S}_f \cdot \mathbf{d}_{PN} \mid}, \tag{3.9}$$

being $\mathbf{d}_{fN}$ the vector that connects the centroid of the cell N to the center of the face and $\mathbf{d}_{PN}$ the one that connects the two centroids.

The gradient terms of the conservation laws in Equation 3.1 are discretized as:

$$\int_V \nabla \Psi dV = \int_S \Psi d\mathbf{S} \approx \sum_f S_f \Psi_f, \tag{3.10}$$

and the interpolation of the variable, similarly to the convective fluxes, as:

$$\mathbf{S}_f \Psi_f = \left[\alpha \, \mathbf{S}_f \Psi_{f+} + (1-\alpha) \, \mathbf{S}_f \Psi_{f-}\right], \tag{3.11}$$

with the limiters used, as before, to bound the interpolated values, Ψ_{f+} and Ψ_{f-}.

Finally, the Laplacian terms are discretized as:

$$\int_V \nabla \cdot (\Gamma \nabla \Psi) dV = \int_S (\Gamma \nabla \Psi) d\mathbf{S} \approx \sum_f \Gamma_f \mathbf{S}_f \cdot (\nabla \Psi)_f, \tag{3.12}$$

where Γ is the diffusion coefficient. In case of non-orthogonality, that is when $\mathbf{S}_f$ is not parallel to the vector that joins the cell centers, the inner product is split into two terms:

$$\mathbf{S}_f \cdot (\nabla \Psi)_f = A(\Psi_N - \Psi_p) + \mathbf{a} \cdot (\nabla \Psi)_f, \tag{3.13}$$

with $A = \frac{|\mathbf{S}_f|}{\mathbf{S}_f \cdot \mathbf{d}_{PN}}$ and $\mathbf{a} = \mathbf{S}_f - A\mathbf{d}_{PN}$.

3.2.2 Aerodynamic Mesh

When dealing with transonic flows, the viscous effects strongly affects the performance of the airfoil. In order to capture them, it is important to select a solver, schemes and a mesh that fulfill the requirements of the simulation [40]. In this section the mesh adopted will be discussed in detail.

Speaking in general, two chances are offered to the user: unstructured or structured meshes. In one hand, the first can be defined also as a ”free” mesh, because it consists of an irregular pattern that covers the whole domain, through elements of different topology. On the other hand, a structured mesh is characterized by a ”mapping” of the domain with a repeatable pattern. Even if the unstructured method allows an easy meshing of complex geometries with less interaction of the user, it usually leads to inaccuracy and non-convergence of the solution [41]. The author noticed that, in *OpenFOAM*, the higher skewness of the unstructured meshes can cause severe oscillations and crashes of the code, especially when a grading of the volumes size is required to capture wall interaction. Thus, a structured mesh was preferred in this study.

The first step to compute a structured mesh around the wing is to divide the geometrical domain in blocks. This is a simple job in a two-dimensional case. In the example of Figure 3.3, a block has been

defined around the leading edge, on the top and lower curves; finally another two start at the trailing edge.

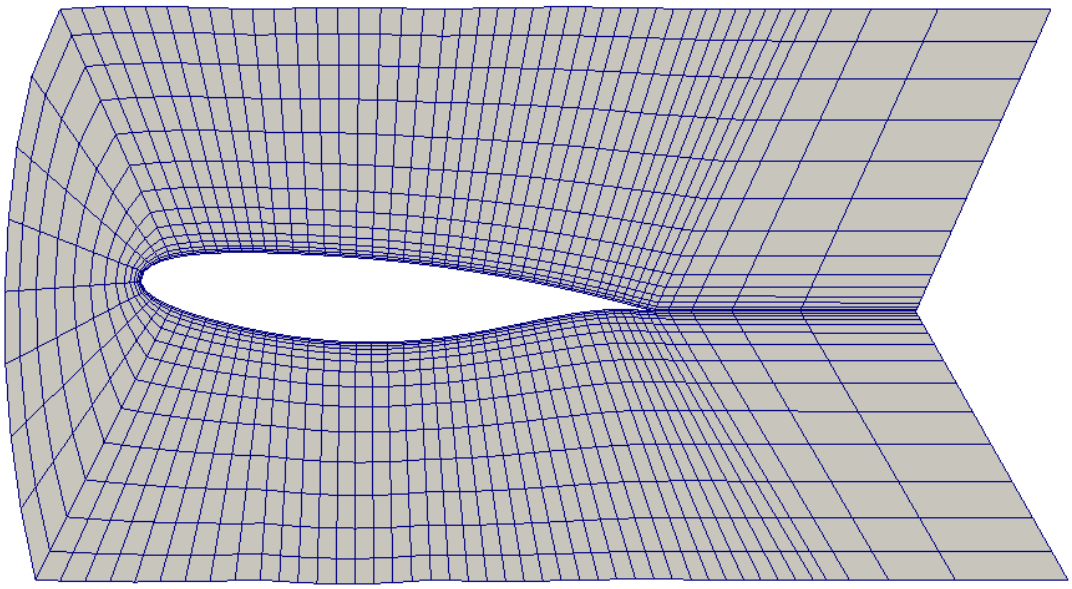


Figure 3.3: Blocking example in 2D case.

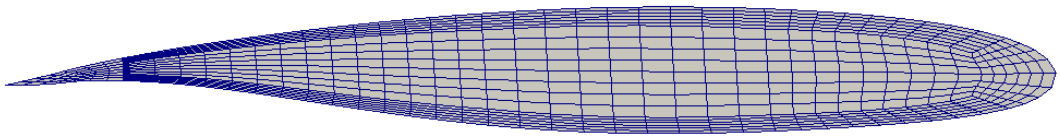


Figure 3.4: Y-Block at the trailing edge.

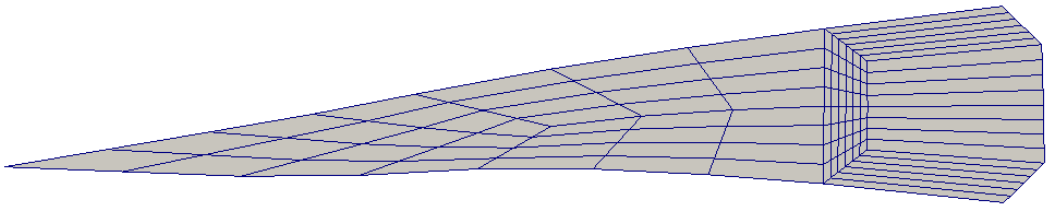


Figure 3.5: Zoom of the Y-Block at the trailing edge.

This method can be repeated for the three sections defined in section 3.1, obtaining the required blocks around the airfoil. A different approach have been employed to mesh the wing tip. The trailing edge can be discretized through a Y-block, that allows the meshing of sharp edges. The method is illustrated, in Figure 3.4, and a zoom on the trailing edge is shown in Figure 3.5.

For an external flow analysis is required to specify the geometrical boundaries that limit the domain extension. In this case a bounding box have been chosen. Supposing that the root of the air-

foil is attached to one face of the box, and its center is at the origin, the bounding box extends from $(-40,\ 0,\ -40)$ m to $(100,\ 30,\ 40)$ m, where the first direction is the flow direction and the second is the span-wise direction. These lengths have been designed in a way that the boundaries' gradient is so small that there is no interaction with the flow around the wing.

In order to reduce the resources and computational time of the simulations, different meshes have been compared to find a good trade-off between accuracy and computational cost. The number of cells and the CPU-time is presented in Table 3.3. The computational time is referred to a simulation with the same boundary conditions and integration time, with an angle of attack equal to zero.

Table 3.3: Mesh comparison.

Mesh	Cells	CPU Time $[h]$
Fine	260694	115
Coarse	108894	27

The lift and drag coefficients, coming from the different simulations have been compared to reference values, obtained experimentally by Tuling and Morelli [42]. These are defined as:

$$C_L = \frac{2L}{\rho v^2 S_w}, \tag{3.14a}$$

$$C_D = \frac{2D}{\rho v^2 S_w}, \tag{3.14b}$$

where L is the lift, D the drag, ρ and v the density and velocity of the air at the inlet boundary, respectively. The values of the force coefficients are in Tab.3.4. During the simulations a Shear Stress Transport (SST in short) $k-\omega$ turbulence model have been used.

It is clear that the run time does not exactly scale with the number of cells but it also depends on the overall quality of the mesh. A comparison of the mesh quality can be done by evaluating the skewness and non orthogonality of the cells. The skewness measures how much the line connecting two cell centers across the merged patches is far from the center of the face, while the other refers to the non-orthogonality between this line and the face.

Even though the finer mesh, as expected, leads to better results, the CPU-time required have been considered not compatible with a FSI analysis. For this reason the coarser mesh have been adopted for the rest of the work.

Table 3.4: Force coefficient comparison, experimental data from [42].

Data	C_L	C_D
Exp.	1.30	0.0123
Fine	1.32 (1.5%)	0.0095 (22.8%)
Coarse	1.34 (3.1%)	0.0141 (14.6%)

Table 3.5: Mesh quality comparison.

Mesh	Max Skewness	Non orthogonal cells
Fine	1.21	1232
Coarse	1.35	227

A plot of the Mach number in the plane of the wing root is shown in Figure 3.6, from which one can clearly observe a first shock wave, right after the leading edge, that represents the start of the supersonic region. This ends with the second shock, allowing the flow to slow down to subsonic speeds at the trailing edge. The small number of cells adopted in the coarser mesh does not allow to visualize the strong discontinuity across the shock wave, that appears to be more like a soft variation.

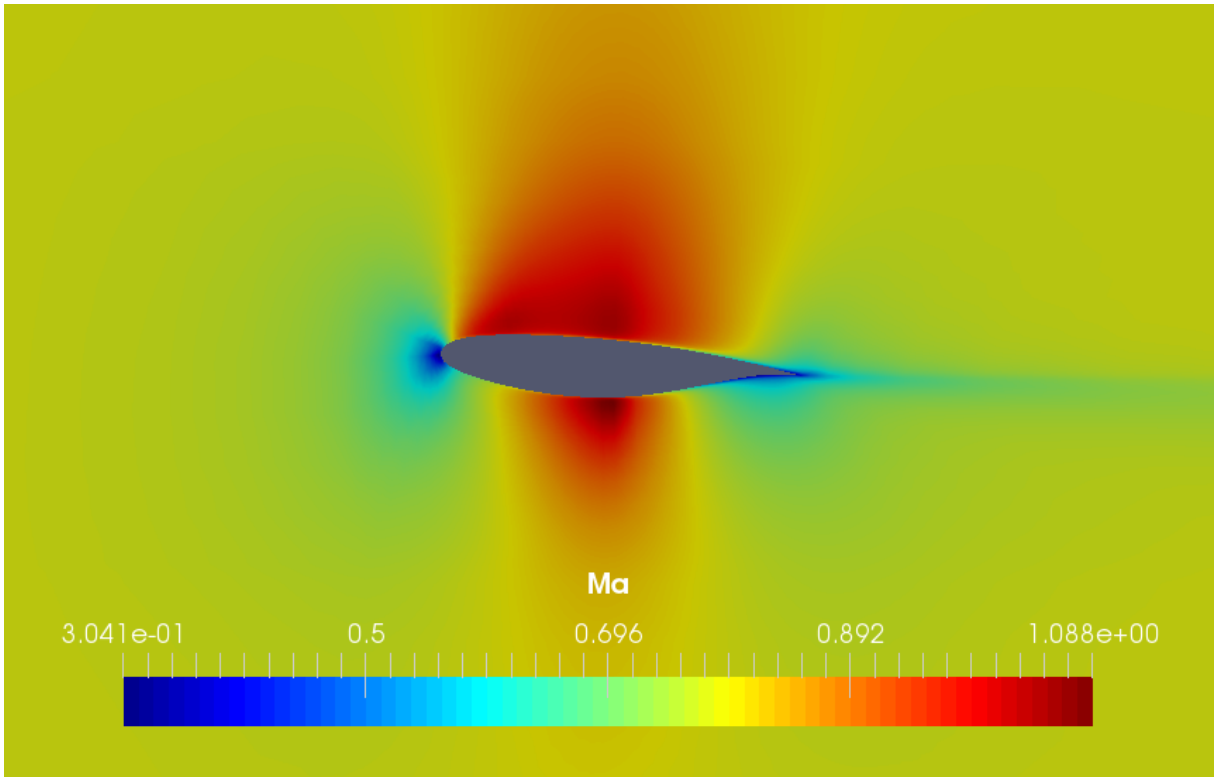


Figure 3.6: Mach number plot in the wing root plane.

3.3 Structural Model

3.3.1 Structural Solver

In FSI analysis, in addition to the fluid model, it is required to specify a structural model that responds to the aerodynamics loads and others that depend on the structure itself. Note that the structural method described in subsection 2.4.3 is still valid, because it does not depends on the fluid model.

3.3.2 Structural Mesh

Modern wing structures generally include spars that extend span-wisely, ribs in the chord direction and stiffeners connected to the skin, plus a set of others equipments like fuel maneuvering systems, fuel tanks and engines, among the others. This thesis does not aim at a comprehensive description of this complex configuration, thus a simplified discretization was preferred.

A reduced structural representation of the wing through a beam model have been investigated in other works, leading to unsatisfactory results [12]. Even though the wing geometry presented in section 3.1 fulfill the slenderness hypothesis, the preservation of the cross-section is not. For this reason, a more complex structural model have been used.

Starting from the 2D mesh, that represents the wing skin in $OpenFOAM$, 2D elements were used in span-wise direction and along the chord, such they prevent an excessive deformation in bending and torsion. The number of elements used in the skin surface is 2582, while the internal structure required 5584 for the internal structure. All of them have been modeled with three or four nodes linear shell elements, so they are characterized by linear shape functions. These are automatically expanded by $CalculiX$ in three-dimensional wedge and brick elements, respectively [18].

The internal structure is in Figure 3.7 and the correspondent displacements in Figure 3.8. The two plots show contours of the total displacements. The results of this dynamic simulation have been obtained using pressure loads on the skin, calculated in the fluid solver. Figure 3.8 demonstrate that this structural model displays the expected motion driven by torsion and bending.

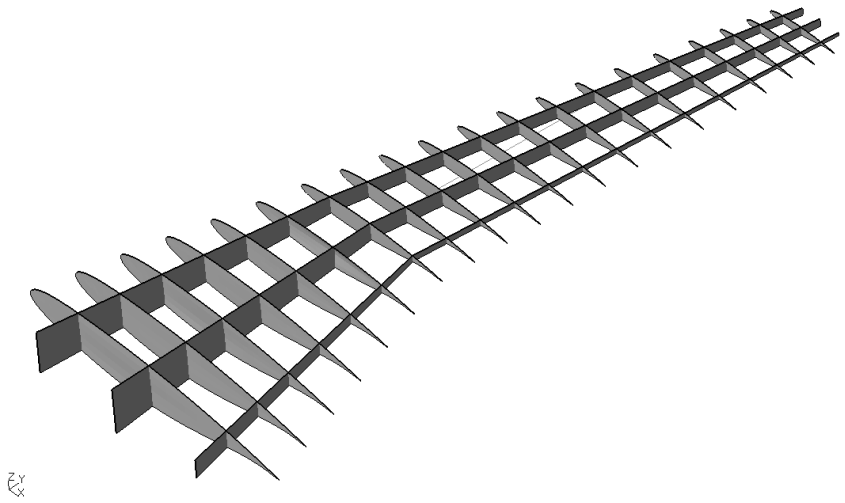


Figure 3.7: Wing structure model with 2D elements (without skin elements).

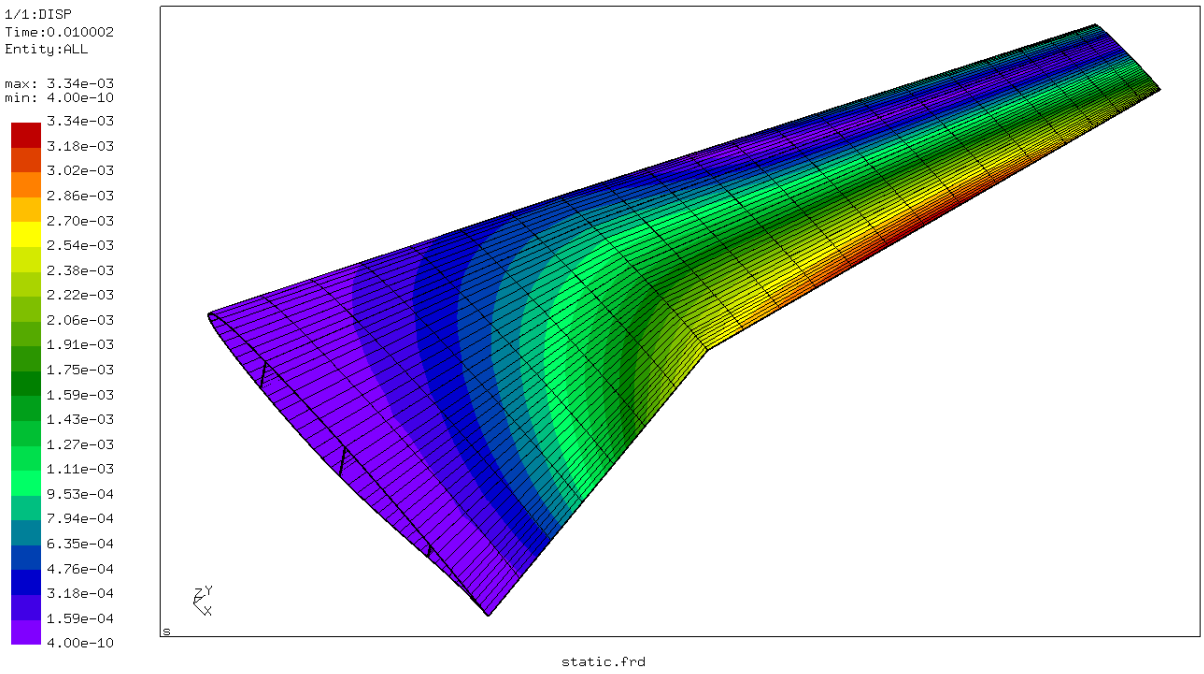


Figure 3.8: Wing total displacements with 2D elements, in [m].

In order to predict the preferred shapes that the wing will assume during the loading, a modal analysis is performed. It finds the eigenmodes, oscillating homogeneous solutions of the linearized governing equation. Each eigenmode has an related frequency of oscillation, the eigenfrequency [31]. The modal study starts from the homogeneous solution of the structural governing equation, already seen in subsection 2.4.3:

$$[M]\{\ddot{\mathbf{u}}\} + [K]\{\mathbf{u}\} = \{0\}^{ext}, \tag{3.15}$$

with boundary conditions:

$$\{\mathbf{u}\}_{t=0} = \{\mathbf{u}\}_0, \qquad initial\ displacements, \tag{3.16a}$$

$$\{\dot{\mathbf{u}}\}_{t=0} = \{\dot{\mathbf{u}}\}_0, \qquad initial\ velocities. \tag{3.16b}$$

A solution in the form $\{\mathbf{u}\} = \{\mathbf{u}_j\}e^{i\omega_j t}$ is proposed. Substituting in the previous equation, the classical eigenvalue problem is obtained:

$$[K]\{\mathbf{u}_j\} = \omega_j^2[M]\{\mathbf{u}_j\}. \tag{3.17}$$

It can be demonstrated that, since $[K]$ is symmetric and $[M]$ is symmetric and positive-definite, the eigenvalues $\lambda_j = \omega_j^2$ are real, and so are the corresponding eigenmodes [31]. Considering two different

solutions: $\{\mathbf{u}_j\}$ and $\{\mathbf{u}_i\}$, the following equations are verified:

$$[K]\{\mathbf{u}_j\} = \lambda_j[M]\{\mathbf{u}_j\}, \tag{3.18a}$$

$$[K]\{\mathbf{u}_i\} = \lambda_i[M]\{\mathbf{u}_i\}. \tag{3.18b}$$

Multiplying respectively by $\{\mathbf{u}_j\}^T$ and $\{\mathbf{u}_i\}^T$ and subtracting them:

$$(\lambda_i - \lambda_j)\{\mathbf{u}_j\}^T[M]\{\mathbf{u}_i\} = 0, \tag{3.19}$$

that, for $\lambda_i \neq \lambda_j$, gives $\{\mathbf{u}_j\}^T[M]\{\mathbf{u}_i\} = 0$. This is the prove that the eigenmodes, hence the induced oscillations, are orthogonal. Even though a continuous system has infinite eigenmodes, only the lowest (5-10 modes) are practically relevant [31]. Figure 3.9 exhibits the first bending mode, and Figure 3.10 the first torsional mode. The second shows a strong deformation of the cross-section, that obviously does not respect the hypothesis of a beam model, where the preservation of the cross-section is guaranteed.

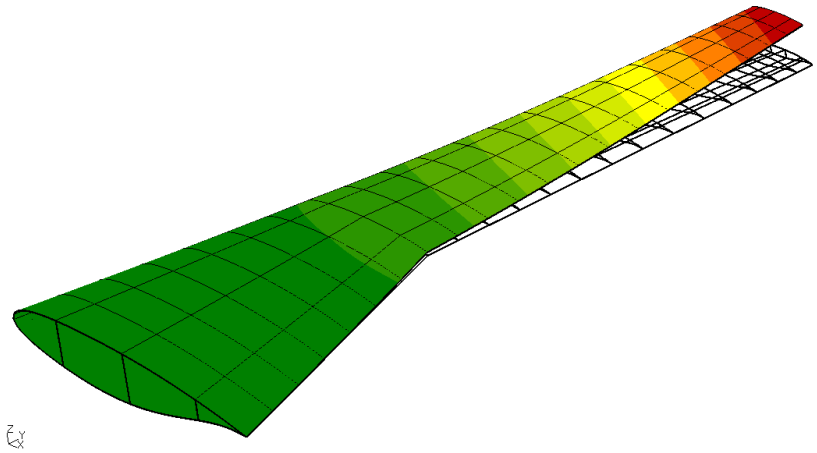


Figure 3.9: First bending mode, total displacements.

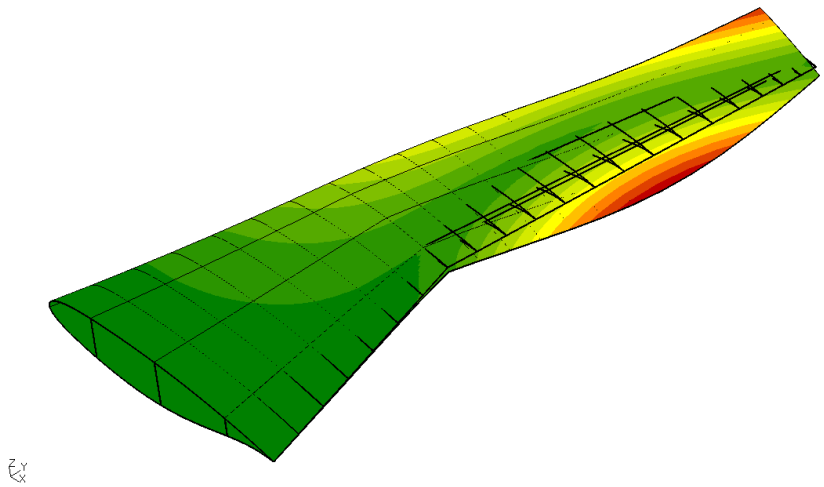


Figure 3.10: First torsional mode, total displacements.

Chapter 4

Transonic Wing FSI Study

As explained in section 2.3, even though loosely coupled time advancing methods introduce the *added-mass* effect [24], the inaccuracy of this method arise in the case of internal fluid analysis of incompressible flow with comparable density value for structure and fluid [25]. Obviously, none of these hypotheses is met in the FSI transonic wing study. Xiang et al. [43] developed a loosely-coupled method, concluding that the accuracy is good enough to capture transonic aeroelastic instability problems, including the transonic dip phenomenon prediction. Thus this approach was preferred for the transonic wing FSI study.

Using $rhoCentralDyMFoam$, that is the dynamic mesh version of $rhoCentralFoam$, the algorithm described in the previous chapter have been implemented in the new solver, without the fixed-point iterative process of the strongly coupled method. Since also in this case conformed mesh have been considered at the interface, the mapping procedure remained untouched. In fact, in a conformed mesh the faces of the solid and fluid elements at the interface are the same, thus there is no need to interpolate the loads and displacements.

Regarding the initial conditions of this analysis, the fields coming from the CFD simulations of the mesh convergence study have been transferred to this simulation. The structure was supposed to be unstressed at initial time, so the displacements are null.

4.1 Structure parameter definition

In order to simulate different cases, two configurations have been considered. In particular, the thickness of the ribs and spars elements have been changed to analyze the response of wing structures with different stiffnesses. Both of them are entirely made of a material with properties similar to an aluminum alloy. The relevant mechanical properties for this study are recalled in Table 4.1.

Table 4.1: Wing material mechanical constants.

$\rho^s[10^3 \frac{kg}{m^3}]$	2.7
$E[10^9 \frac{N}{m^2}]$	73
ν^s	0.33

The two configurations are presented in Table 4.2. Clearly, it is expected to have larger deformation in the second configuration, since the decreased thickness of the spars leads to a drop in the structural stiffness in both bending and torsion.

Table 4.2: Thicknesses of the 2D elements in the structural model.

Data	**Configuration 1**	**Configuration 2**
Skin [mm]	2	2
Rib [mm]	40	20
Spar [mm]	40	20

4.2 Results

The displacements recorded in four points distributed along the wing surface have been stored into results to compare motion of points in the two structural configurations. Two of them are on leading edge, the other on the trailing edge. For each group one is located at the tip of the edge, the other in the middle between the tip and the wing break. For the sake of clarity, the positions are highlighted in Figure 4.1.

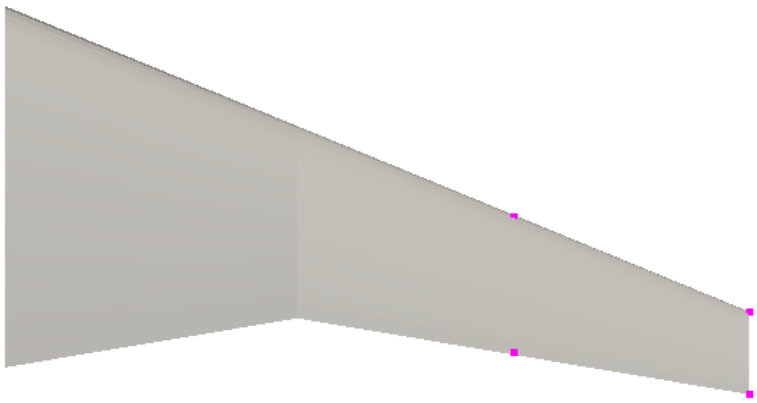


Figure 4.1: Positions of the four control points distributed along the wing surface.

Figure 4.2, Figure 4.3, Figure 4.4 and Figure 4.5 compare the displacements of the first configuration (in blue) with the second (in red). As expected, the second configuration has larger displacements. In this short simulation time, a torsion deformation drives the wing motion. This is clear observing that the points on the trailing edge have positive displacements, while the ones on the leading edge have negative displacements.

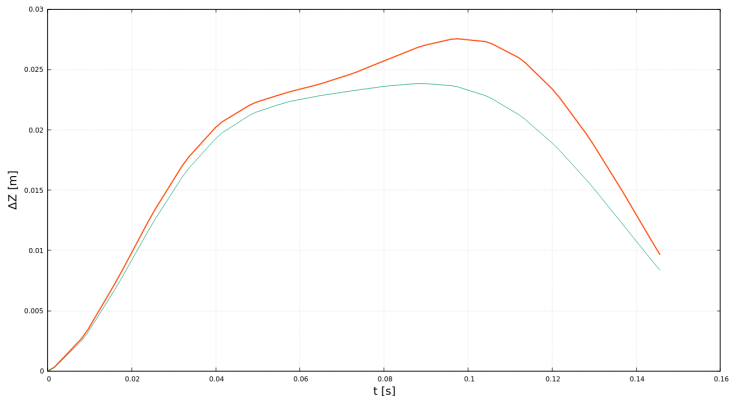


Figure 4.2: Displacements in the z direction for the control point on the trailing edge, at tip. Configuration 2 in red, Configuration 1 in blue.

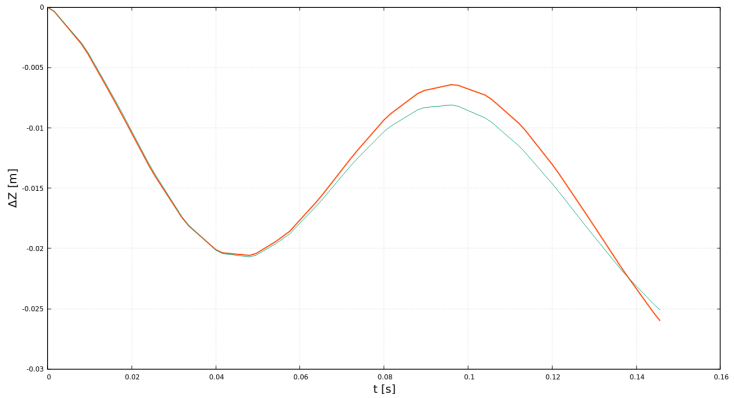


Figure 4.3: Displacements in the z direction for the control point on the leading edge, at tip. Configuration 2 in red, Configuration 1 in blue.

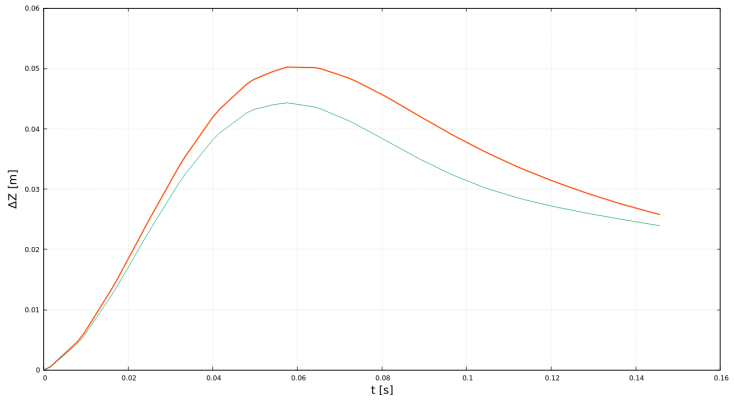


Figure 4.4: Displacements in the z direction for the control point on the trailing edge, between wing break and tip. Configuration 2 in red, Configuration 1 in blue.

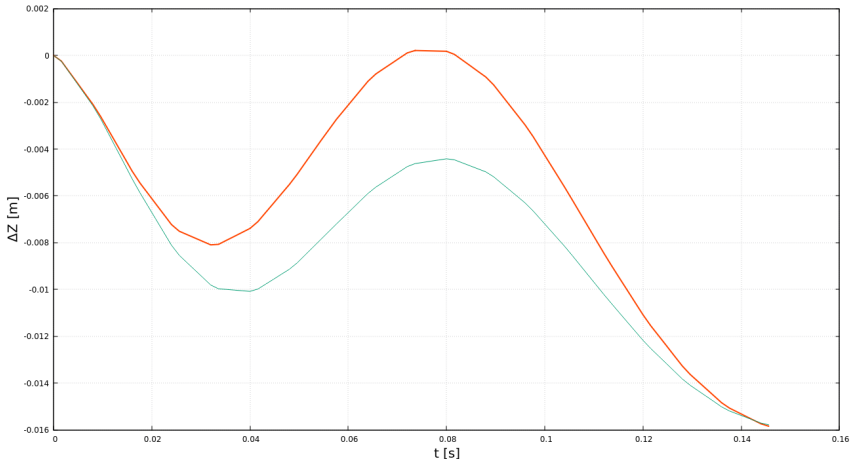


Figure 4.5: Displacements in the z direction for the control point on the leading edge, between wing break and tip. Configuration 2 in red, Configuration 1 in blue.

The lift and drag values during time are plotted in Figure 4.6 and Figure 4.7, respectively. The red refers to the first configuration, the ticker, while the blue lines refer to the second configuration. As expected, the larger displacements of the second configuraton lead to greater changes in both drag and lift. The torsion motion of the wing leads to a drop of 15% in the lift force during the simulation. Even though the drag force varies during time, the percentual change is much less than the one registered in the lift force. The change of the drag force, in percentage, is much lower, around 1%.

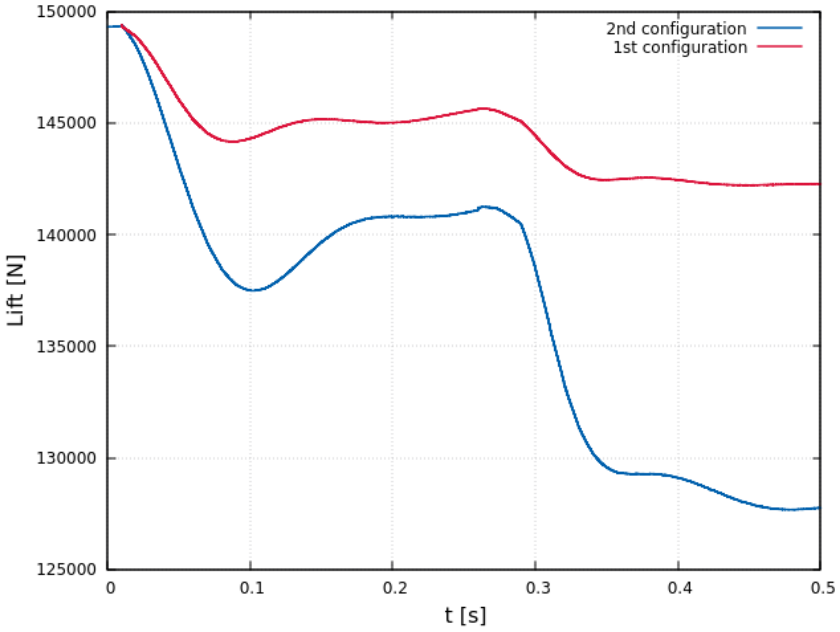


Figure 4.6: Plot of the lift force during the simulation.

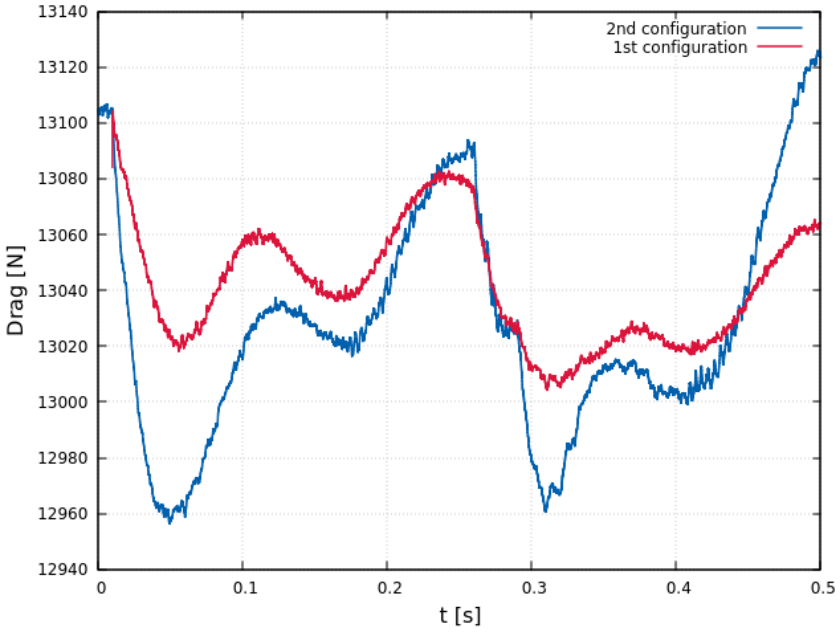


Figure 4.7: Plot of the drag force during the simulation.

In order to describe the effects of elasticity in bending and torsion in the aerodynamics, the dynamics of the second configuration are here presented. A section at 12m from the wing root along the span direction have been selected because it is characterized by greater displacements, thus greater changes in the flow fields. The pressure field at the beginning of the simulation is presented in Figure 4.8 and in Figure 4.9 at the end of the simulation. It can be noticed that the torsional motion of this section alignes the airfoil with the upstream flow velocity, leading to a lower acceleration of the fluid along the wing surface. For this reason the shock that is in Figure 4.8 vanishes and become a more gentle variation of the pressure field in Figure 4.9.

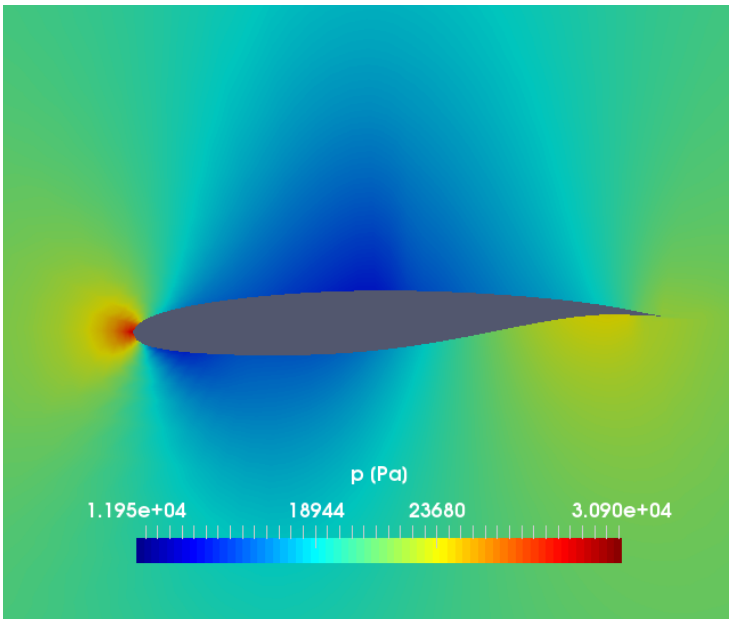


Figure 4.8: Plot of the pressure field at the beginning of the simulation in the selected section.

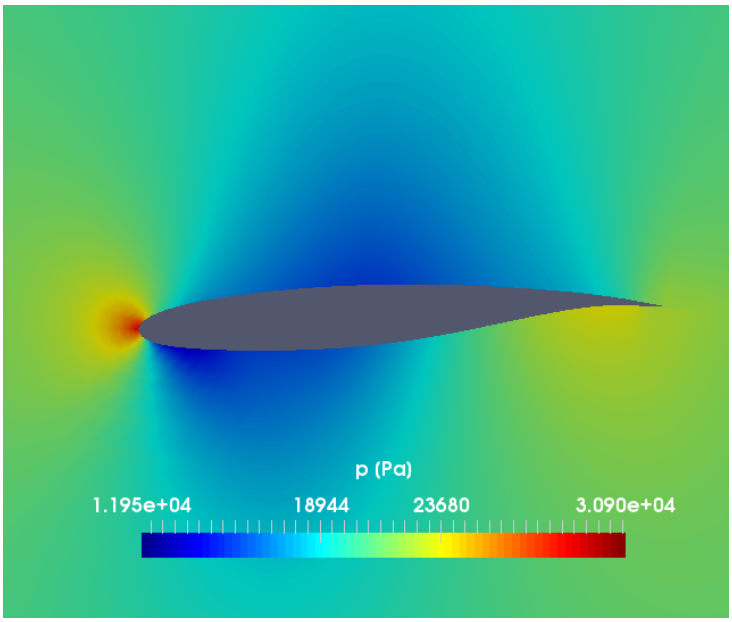


Figure 4.9: Plot of the pressure field at the end of the simulation in the selected section.

Chapter 5

Conclusions

In this thesis different problems have been addressed, starting from the implementation of a new FSI algorithm into an open-source software environment. Even though the algorithm can be easily described through a simple flowchart, the implementation was, for some aspects, cumbersome. In this case a big help was provided by all the active developers of the open-source application employed.

Even if the analysis performed on the incompressible test has no relations with the main goal of this thesis, it was necessary to test if the algorithm has been implemented correctly. The good performance demonstrated on the benchmark confirmed the correctness of the whole system built.

Then, the swept wing analyzed exhibited some instabilities in the calculations because of its complex geometry. Trying different solutions, it was possible to find a meshing method, in particular the Y-block at the tip, that allowed to have stable solutions and run the transonic simulations with a relatively small number of cells.

The work is concluded with the presentation of the FSI results of the transonic wing. Even though the final results presented could be considered not satisfactory because of the short run time, the expected motion of a very flexible wing is captured. The torsional motions, especially for the second configuration leading to a drop in the lift force that is very relevant. Regarding these simulations, the very long computational time was a bottleneck for the fast advancing of the thesis.

5.1 Future Work

The author considers that this work can be enhanced, spending more time on the following aspects:

- The support to non conformed meshes should be taken into account. In fact, the transient simulations of complex structures, like the transonic wing here discretized, can be quicken with the use of a coarser mesh at the interface;
- Thanks to the use of an external and extended structural solver, like *CalculiX*, different material models can be tested in FSI analysis, composite materials for example;
- More simulations can be executed, changing the geometry, turbulence model and flight conditions;

- The coding of an internal structural solver should be considered, calling the structural solver with a system call, as the author did in this work, does not allow to run it in parallel, thus leading to a longer computational time.
- The simulations related to the results presented in chapter 4 should be continued in order to conclude this work with more details.

Bibliography

[1] F. Afonso, J. Vale, E. Oliveira, F. Lau, and A. Suleman. Non-linear aeroelastic response of high aspect-ratio wings in the frequency domain. *The Aeronautical Journal*, 121:1–19, 05 2017.

[2] R. G. A. Silva, J. L. F. Azevedo, and O. A. F. Mello. An investigation on viscous effects in downwash weighting methods for transonic aeroelastic stability analysis. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, 35(2):153–162, 06 2013.

[3] R. Palacios, H. Climent, A. Karlsson, and B. Winzell. Assessment of strategies for correcting linear unsteady aerodynamics using cfd or experimental results. *International Forum of Aeroelasticity and Structural Dynamics*, 06 2001.

[4] O. Bendiksen. Influence of shocks on transonic flutter of flexible wings, 50th aiaa/asme/asce/ahs/asc structures, structural dynamics, and materials conference. *American Institute of Aeronautics and Astronautics*, 05 2009.

[5] M.-A. Mihai, R. Constantinu, and G. M. Pericle. Staggered approach for fluid-structure interaction phenomena of an agard 445.6 wing using commercial cfd/csm software. *Journal of Aerospace Engineering*, 28(4):401–409, 07 2015.

[6] T. E. Tezduyar, M. Behr, S. Mittal, and J. Liou. A new strategy for finite element computations involving moving boundaries and interfaces; the deforming-spatial-domain/space-time procedure. ii: Computation of free-surface flows, two-liquid flows, and flows with drifting cylinders. *Comput. Methods Appl. Mech. Eng.*, 94(3):353–371, 02 1992.

[7] H. J.-P. and R. O. Morand. *Fluid-Structure Interaction: Applied Numerical Methods*. 1995.

[8] C. Michler, E. van Brummelen, S. Hulshoff, and R. de Borst. The relevance of conservation for stability and accuracy of numerical methods for fluid–structure interaction. *Computer Methods in Applied Mechanics and Engineering*, 192(37):4195–4215, 09 2003.

[9] K. Takizawa and T. E. Tezduyar. Computational methods for parachute fluid-structure interactions. *Archives of Computational Methods in Engineering*, 19(1):125–169, 03 2012.

[10] K. Takizawa, J. Christopher, C. Moorman, J. Martin, J. Purdue, T. McPhail, P. Chen, J. Warren, and T. Tezduyar. Space–time finite element computation of arterial fsi with patient-specific data. *International Journal for Numerical Methods in Biomedical Engineering*, 26:101–116, 06 2009.

[11] J. T. Batina, D. A. Bennett, Robert M.and Seidel, H. J. Cunningham, and S. R. Bland. Recent advances in transonic computational aeroelasticity. 30(1):29–37, 09 1998.

[12] B. Stickan, J. Dillinger, and G. Schewe. Computational aeroelastic investigation of a transonic limit-cycle-oscillation experiment at a transport aircraft wing model. *Journal of Fluids and Structures*, 49 (Supplement C):223–241, 08 2014.

[13] R. L. Campbell. Fluid-structure interaction and inverse design simulations for highly flexible turbomachinery. *Journal of The Acoustical Society of America - J ACOUST SOC AMER*, 129, 04 2011.

[14] M. S. Maza, S. Preidikman, and F. G. Flores. Unsteady and non-linear aeroelastic analysis of large horizontal-axis wind turbines. *International Journal of Hydrogen Energy*, 39(16):8813–8820, 05 2014.

[15] A. R. . D. Adina fsi (accessed on 10 2017). URL `http://www.adina.com/fluid-structure-interaction.shtm`.

[16] ANSYS. Ansys multiphisics (accessed on 10 2017). URL `http://www.ansys.com/products/platform/multiphysics-simulation/fluid-structure-interaction`.

[17] H. G. Weller, G. Tabor, H. Jasak, and C. Fureby. A tensorial approach to computational continuum mechanics using object-oriented techniques. *Computers in Physics*, 12:620–631, 11 1998.

[18] G. Dhondt. Calculix crunchix user's manual version 2.12. 04 2017.

[19] J. Donea, A. Huerta, J.-P. Ponthot, and A. Rodríguez-Ferran. Arbitrary lagrangian-eulerian methods. *Encyclopedia of Computational Mechanics*, 1:413–437, 11 2004.

[20] D. Wei, S. M. Spence, A. Kareem, and A. Jemcov. A structured mesh boundary motion approach for simulating wind effects on bluff bodies with changing boundaries. *Journal of Wind Engineering and Industrial Aerodynamics*, 126(Supplement C):118–131, 01 2014.

[21] H. Jasak and Z. Tukovic. Automatic mesh motion for the unstructured finite volume method. *Transactions of FAMENA*, 30:1–20, 11 2006.

[22] N. M. Sudharsan, R. Ajaykumar, K. Murali, and K. Kumar. A comparative study of dynamic mesh updating methods used in the simulation of fluid-structure interaction problems with a non-linear free surface. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 218(3):283–300, 03 2004.

[23] G. Fourey, C. Hermange, D. Le Touzé, and G. Oger. An efficient fsi coupling strategy between smoothed particle hydrodynamics and finite element methods. *Computer Physics Communications*, 217(Supplement C):66–81, 08 2017.

[24] P. Causin, J.-F. Gerbeau, and F. Nobile. Added-mass effect in the design of partitioned algorithms for fluid-structure problems. *Computer Methods in Applied Mechanics and Engineering*, 194:4506–4527, 10 2005.

[25] W. G. Dettmer and D. Perić. On the coupling between fluid flow and mesh motion in the modelling of fluid-structure interaction. *Computational Mechanics*, 43:81–90, 12 2008.

[26] U. Küttler and W. Wall. Fixed-point fluid-structure interaction solvers with dynamic relaxation. *Computational Mechanics*, 43:61–72, 01 2008.

[27] S. Turek and J. Hron. Proposal for numerical benchmarking of fluid–structure interaction between an elastic object and laminar incompressible flow. *Fluid-Structure Interaction: Modelling, Simulation, Optimisation*, 53:371–385, 06 2007.

[28] E. Robertson, V. Choudhury, S. Bhushan, and D. Walters. Validation of openfoam numerical methods and turbulence models for incompressible bluff body flows. *Computers and Fluids*, 123:122–145, 12 2015.

[29] T. Holzmann. *Mathematics, Numerics, Derivations and OpenFOAM®*. 12 2016.

[30] H. Versteeg and W. Malalasekera. *An Introduction to Computational Fluid Dynamics, Second edition*. Pearson, 2007.

[31] G. Dhondt. *The Finite Element Method for Three-Dimensional Thermomechanical Applications*. Wiley, 12 2004.

[32] S. Behara and S. Mittal. Wake transition in flow past a circular cylinder. *Physics of Fluids*, 22, 11 2010.

[33] C. J. Greenshields. Openfoam user guide. 07 2017.

[34] P. I. Kattan. Matlab guide to finite elements: An interactive approach. *Springer Berlin Heidelberg*, 2008.

[35] S. Ricci, A. De Gaspari, A. Antunes, F. Odaguil, and G. Rodrigues de Lima. Application of active camber morphing concept to a regional aircraft, 01 2014.

[36] International standard iso 2533. 05 1975.

[37] F. Gutierrez, J. Tamagno, and S. Elaskar. High speed flow simulation using openfoam. 11 2012.

[38] C. J. Greenshields, H. G. Weller, L. Gasparini, and J. M. Reese. Implementation of semi-discrete, non-staggered central schemes in a colocated, polyhedral, finite volume framework, for high-speed viscous flows. *International Journal for Numerical Methods in Fluids*, 63(1):1–21, 5 2010.

[39] A. Kurganov and E. Tadmor. New high-resolution central schemes for nonlinear conservation laws and convection–diffusion equations. *Journal of Computational Physics*, 160(1):241–282, 2000.

[40] F. Frunzulică, H. Dumitrescu, A. Dumitrache, and V. Cardos. Transonic airfoil flow simulation. part i: Mesh generation and inviscid method. *INCAS BULLETIN*, 2, 06 2010.

[41] A. Javidinejad. Fea practical illustration of mesh-quality-results differences between structured mesh and unstructured mesh. *ISRN Mechanical Engineering*, 2012, 07 2012.

[42] S. Tuling and M. Morelli. Novemor d5.2: Test report for the transonic wind-tunnel tests, 2015.

[43] Z. Xiang, Z. Yongfeng, and Z. Sijun. Transonic wing flutter predictions by a loosely-coupled method. *Computers and Fluids*, 58(Supplement C):45–62, 2012.