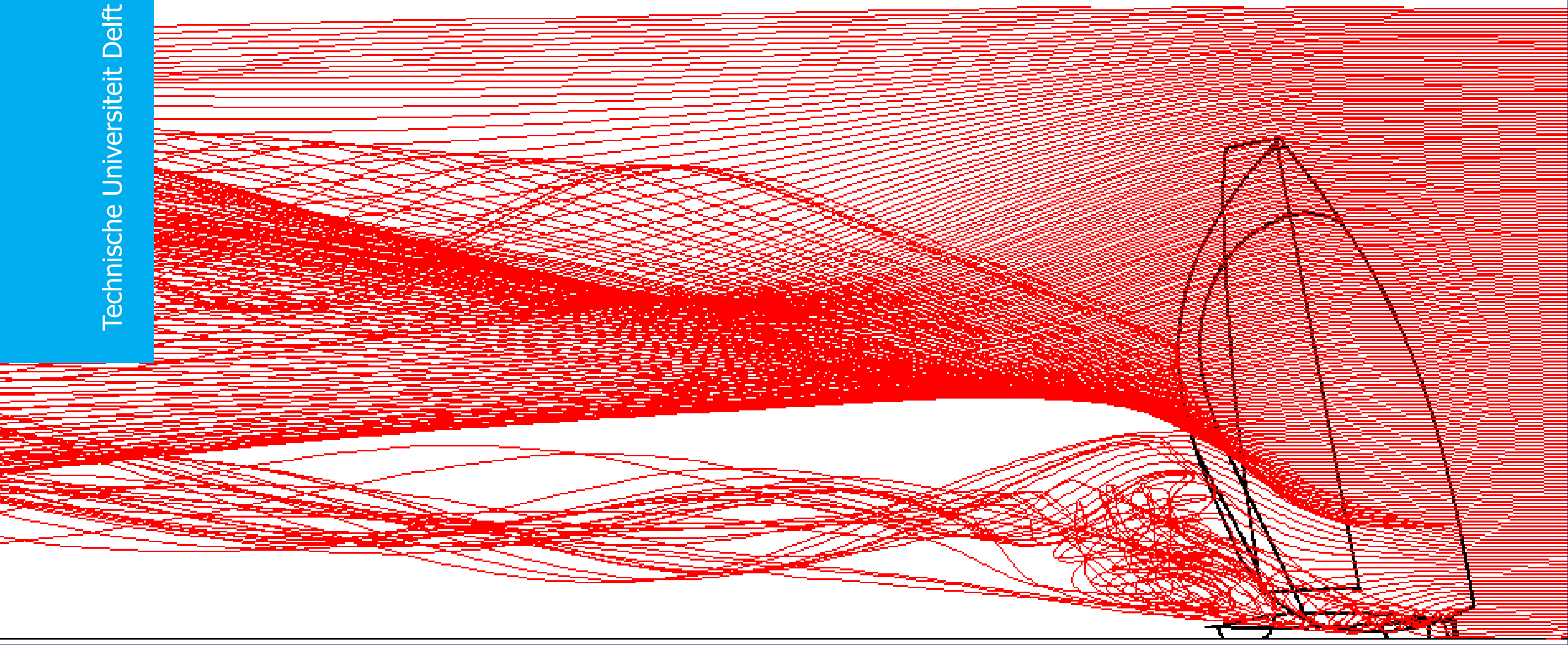
Implementation of a Fluid-Structure Interaction Solver for a Spinnaker Sail

Master Thesis

Anna Ramolini



Implementation of a Fluid-Structure Interaction Solver for a Spinnaker Sail

by

Anna Ramolini

in partial fulfillment of the requirements for the degree of

Master of Science
in Aerospace Engineering, Track Aerodynamics

at the Delft University of Technology,
to be defended publicly on Friday November 30, 2018 at 09:30 AM.

Student number:	4515536	
Project duration:	October 27, 2017 – November 30, 2018	
Supervisor:	Dr. ir. A. H. van Zuijlen	
Thesis committee:	Dr. ir. B. W. van Oudheusden,	TU Delft
	Dr. ing. R. Schmehl ,	TU Delft
	Ir. M. Folkersma,	TU Delft

List of Abbreviations

Abbreviation	**Meaning**
2D	two-dimensional
3D	three-dimensional
AWA	Apparent Wind Angle
AR	Aspect Ratio
BC	Boundary Conditions
BL	Boundary Layer
CFD	Computational Fluid Dynamics
CPU	Central Processing Unit
DOF	Degrees of Freedom
DES	Detached Eddy Simulation
FE	Finite Elements
FEA	Finite Element Analysis
FEM	Finite Element Method
FSI	Fluid-Structure Interaction
GUI	Graphical User Interface
LES	Large Eddy Simulation
MSc	Master of Science
NS	Navier-Stokes
PhD	Doctor of Philosophy
RANS	Reynolds Averaged Navier Stokes
RBF	Radial Basis Function
TU Delft	Technische Universiteit Delft
TWA	True Wind Angle
VPP	Velocity Prediction Program

List of Symbols

Symbol	**Meaning**	**Unit**
ε	Turbulent dissipation rate	$\frac{m^2}{s^3}$
μ	Dynamic viscosity	$\frac{Ns}{m}$
ν	Kinematic viscosity	$\frac{m^2}{s}$
$\tilde{\nu}$	Modified turbulent viscosity	$\frac{m^2}{s}$
ρ	Density	$\frac{kg}{m^3}$
ω	Specific dissipation rate	$\frac{1}{s}$
A	Area	m^2
C_μ	Turbulence model constant	[-]
C_D	Drag coefficient	[-]
C_L	Lift coefficient	[-]
C_p	Pressure coefficient	[-]
h	Model height from groud	m
k	Turbulent kinetic energy	$\frac{m^2}{s^3}$
I	Turbulence intensity	[-]
l	Turbulence length scale	m
L	Length	m
p	Pressure	Pa
Re	Reynolds number	[-]
St	Strouhal number	[-]
t	Time	s
U	Velocity Magnitude	$\frac{m}{s}$
$\mathbf{u}$	Velocity	$\frac{m}{s}$
u	x axis velocity	$\frac{m}{s}$
v	y axis velocity	$\frac{m}{s}$
w	z axis velocity	$\frac{m}{s}$
y^+	Non-dimensional wall distance	[-]

Preface

This project consists in developing a fluid-structure interaction solver to resolve the flow field and deformation of a spinnaker sail. The goal is to correctly capture the spinnaker's ability to deform and propel the sailboat. The need for this solver is motivated by the fact that estimating the sail thrust considering it rigid and therefore using only CFD can lead to wrong thrust estimations. Thus, if the CFD solver is coupled with a structural solver, that allows also the cloth deformation to be taken into account, the estimation becomes more reliable. This goal has only been partially achieved so far, because of the high computational costs of the operations and the complexity of the physics involved. For this reason this project, if completed correctly, can take the science behind sailing a step further. Given the complexity of the problem, three main areas need to be devloped: the fluid solver, the structure solver and the algorithm able to couple them. This document will illustrate all the steps taken in order to reach the final goal, presenting the used tools and the line of reasoning employed throughout the project.

Anna Ramolini
Delft, November 2018

Contents

List of Figures

List of Tables

1 Introduction

The physics underlying sailing is characterized by complex phenomena that are only lately beginning to be explored through numerical methods, since the computational power has only recently started to meet the high computational costs required by these types of analyses. The main difficulty lays in the fact that the flow regime is highly turbulent and, as it is well known, this type of flow is still not resolved fully in its whole complexity, but can only be approximated through turbulence models that require heavy approximations.

While for upwind sails like the jib or the mainsail some prediction works through CFD have already been performed (for example the work done by the Van Oossanen naval architects, [6], or Viola [21]), for downwind sails such as spinnaker and gennaker more sophisticated techniques are needed [31], [39], [41]. This is due to the fact that the flow in upwind sailing is mostly attached and the turbulence that can be present is limited to the small scales in the boundary layer; therefore the flow can be modelled with inviscid models such as the Vortex Lattice Method and potential flow codes, possibly with the help of a boundary layer correction for the small scale turbulence, or also with RANS.

On the other hand, when sailing downwind the functioning of the sail changes completely, with the flow detaching and largely separating, the presence of large scale turbulence and the propulsion being generated through drag and not lift. For this case, the inviscid methods are not suitable, as it is well known that they cannot resolve correctly large scale turbulence, detachment and separation. Some more elaborate techniques are then needed to unveil the physics, such as RANS, LES or DES

Moreover, the added complexity to the problem of downwind sails lays also in the fact that the cloth is made of a very light and deformable material, differently from the upwind sails. That results in big changes of shape of the sail when sailing, depending on the wind intensity and direction and the trim settings. This aspect has to be taken in account if a correct analysis is to be performed.

This project is aimed to resolve the flow around a spinnaker sailing downwind, and to find the techniques that allow the thrust prediction to be accurate enough to be used in the sail design analysis, instead of the costly experimental simulations. In order for this to happen, the first step is to capture correctly the nature of the turbulence around the sail: the main interest is to accurately predict the separation and the wake, so that the pressure distribution around the sail can be computed. In addition, some studies ([17],[15]) have shown the necessity to use a structure solver together with the fluid solver, making this a fluid-structure interaction problem, in order to have more realistic thrust predictions. In the next section this concept will be further examined.

1.1. Motivation

For many years, sailing has been done "on the field" and the sail constructors were designing sails based on the previous experiences on the water, without quantitative analyses. Now that the computational power allows it, more and more computer studies are being performed ([31], [39],[6], [21]) in order to avoid experimental testing with all the sail prototypes, using the preliminary testing to save materials and time. The objective of these simulations is to quantify the pressure field around the sails, which can be used to calculate the thrust the sail can yield. As it has been stated in the previous paragraph, some work on the upwind sails has been already done, while for the downwind sails there is still a lot of work to be carried out to have precise information on the sail performance.

1.1.1. Physics of spinnaker flow

The spinnaker, or spi, is a large, light triangular sail that is positioned in front of the bow of the boat. Being it the foremost sail, it is also called *headsail*. It can be used for *running*, which is merely going straight downwind, or *broad reaching*, that is going towards a point that has a wide angle with respect to the wind direction. Typical sailing regimes are shown in figure 1.1. The setting of the sail will change depending on the regime, being it more tightened to the side for the reaching regime or set free to fly in the front of the boat if running. This sail is generally used with apparent wind angles in the range 90°-180°, where between 90° and 120° the boat is reaching and between 120° and 180° the boat is running, indicatively. So while the other sails, mainsail and jib, are used for all points of sail, the spinnaker is only usable in the last two configurations of figure 1.1. This sail can be considered as elective, as it is not always used especially outside of competitions, but it can give major improvements to the boat speed if used, because of its huge dimensions and the consequent thrust it can provide.

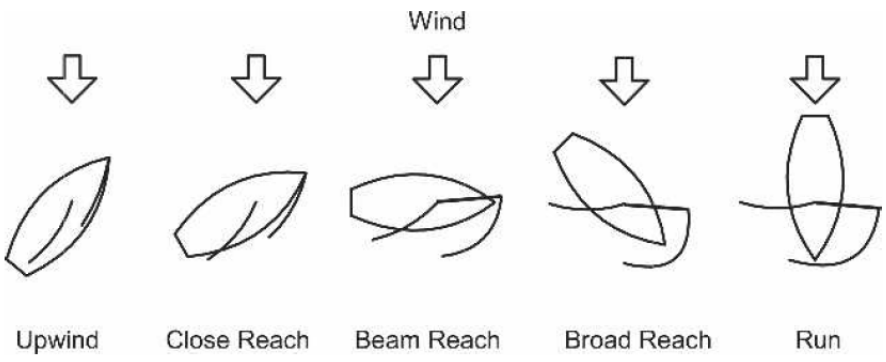


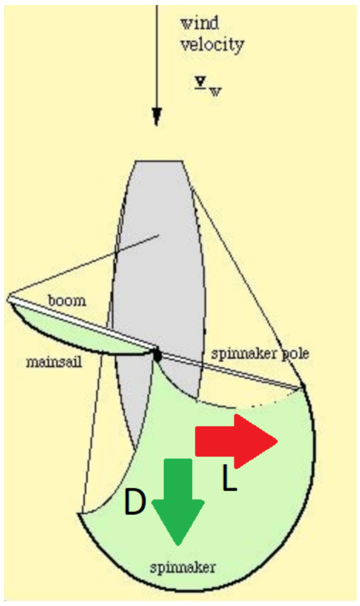
Figure 1.1: Points of sail, from [41]

The reason why this sail presents such different features from the other sails is that its functioning is completely unlike. In particular when sailing downwind its propulsion relies almost completely on the drag produced by the sail, and not the lift. For the spinnaker this is the only way to generate driving force, while for example the mainsail can be used as a lift or drag device depending on the wind angle. Consider figure 1.1: the first three images depict a situation in which the driving force is the lift generated by the mainsail and jib, while in the last two the mainsail is using the drag that it generates to propel the boat.

As it has been said before, the spinnaker can only be used in the last two (occasionally three) points of sailing of figure 1.1: this means that the air will arrive to the spi from the back of the boat, and the aerodynamic forces generated will be the lift, directed perpendicularly to the flow and therefore to the side of the boat, and the drag, parallel to the flow and directed in front of the boat. In this case, the drag will be the driving force. The lateral force, namely the lift, will constitute a heeling force that will tend to make the boat roll. On the other hand, in a different situation without the spinnaker the lift generated by the mainsail and the resistance of the keel will play a fundamental role in the propulsion of the boat. A schematic representation of this process is shown in figure 1.2.

Being the spinnaker a drag device, it goes without saying that the estimation of the drag coefficient must be as accurate as possible in order to have a correct estimate of the propulsion the sail can provide. Some online tools for a fast estimation of the sail performance coefficients are available, for example the tool SailPowerCalc[1], which is a fast Velocity Prediction Program that, given some sail data, can give an estimate of the driving and heeling coefficients of the boat. In order to have an idea of the performance of this fast estimator, some data has been taken from Viola ([22]) and filled in to test the online tool. In the paper, some data about a model used in the wind tunnel for testing is given. Table 1.1 summarizes the input data and the resulting values from the two cases.

[1] https://cdn2.hubspot.net/hub/209338/news/SailPowerCalc/SailPowerCalc.htm#SailPowerCalc



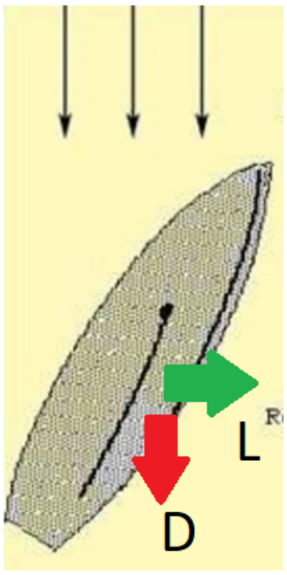


Figure 1.2: Aerodynamic forces generated by a running spinnaker (left) and by mainsail in close reach (right) [2]

	Viola [22]	SailPowerCalc
Mainsail Luff [m]	0.21	0.21
Mainsail Foot [m]	0.62	0.62
Mainsail Area [m^2]	**0.09**	**0.1**
Spinnaker Area [m^2]	2.4	2.4
Total Sail Area [m^2]	**2.49**	**2.5**
Apparent Wind Speed [m/s]	3.5	3.5
Heel Angle [deg]	10	10
Driving force coeff. [-]	**0.9**	**1.04**

Table 1.1: Comparison of experimental data from [22] with estimated data from online tool SailPowerCalc

In table 1.1 the main differences are highlighted in bold: it can be observed that the final value of the driving force coefficient estimated with the online tool has an error of approximately 15% with respect to the experimental data. What can be observed is that the estimation of the total sail area is performed with an error by the online tool: it computes a mainsail area of 0.1 m^2, while for Viola [22] the mainsail area is given to be 0.09 m^2. This error can be influencing the final results. More inaccuracies can be attributed to the fact that the online tool is based on a Velocity Prediction Program. That is an estimation tool based on empirical observations only on particular sets of yachts and sails, so it can be inaccurate when considering a type of sail that is not present in the VPP database. The goal of this analysis is to show that CFD is necessary to have more accurate predictions of the sail performance data, and fast estimators like the one presented above are not sufficient to have an accurate analysis.

Another important difference between the spinnaker and the other sails is the presence of transient phenomena such as large separation and transition to turbulence in the flow, due to the position of the sail with respect to the wind. This does not happen with the mainsail in beam reach regime for example, where the

[2]from https://sites.google.com/site/crisflopt/navegacao/sail-trim

flow stays attached for most of the chord length (since the sail is aligned with the flow), the transition to turbulence is limited to the boundary layer and the turbulent eddies remain confined to the small scale range. Whereas the spinnaker in running regime is perpendicular to the flow, and therefore the turbulent eddies are large and the separation is massive. The two different flow situations are schematically shown in figure 1.3.

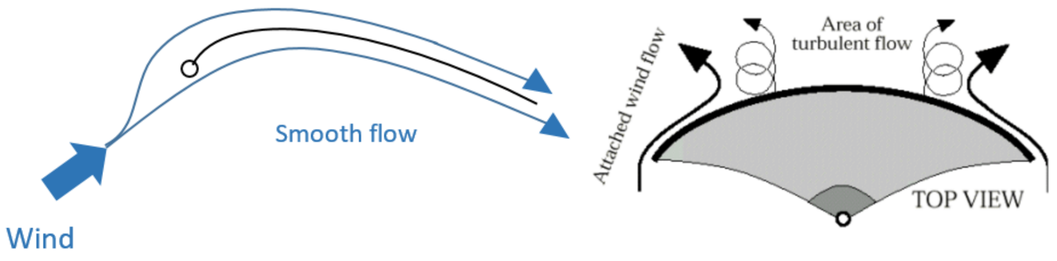


Figure 1.3: Attached (left) and detached (right, from [2]) flows around mainsail and spinnaker, respectively

Therefore, while for the mainsail and jib in reaching regimes a potential flow code coupled with a boundary layer correction can be suitable and yield relevant results, for the spi requires a more elaborate code. The upwind and downwind turbulence has to be modelled correctly in order to have significant results. The perfect CFD tool for this application would be able to capture the flow around the mainsail with its wake and separation point, and then would use this flow condition as a starting point to compute the flow around the spinnaker, which will also have its wake and separation point. Figure 1.2 (left) shows the typical configuration of mainsail and spinnaker: it can be noted that the wake of the mainsail only affects a small portion of the spinnaker, so the air that will inflate the spinnaker would be mostly the free stream one. Anyhow, taking into account the wake of the mainsail surely makes the simulation more significant and representative of reality. The challenge is then to find a tool able to describe all these complex phenomena.

Once the functioning of this sail is explained, an important observation has to be made: as mentioned before, this sail has a high ability to deform under the pressure loads of the flow due to its shape, material and setting - it is free to move from two of the three points where it is attached to the boat, and therefore has much more freedom than a mainsail or jib, which are fixed in two points. This means that it will assume various shapes during its usage, depending on the wind speed and the sailing direction. and this is a crucial point for CFD simulations - a body that can deform is harder to simulate than a rigid one. Therefore, maintaining a constant geometry during the whole simulation, by assuming a rigid body, will not yield an accurate representation of the reality.

Renzsch et al. [17] show that the flying shape computed with their Fluid-Structure Interaction software and the moulded one given by the sail constructor not only are widely different, but also yield significantly different values for the driving and heeling forces, given the same wind speed and angle. In their work, Renzsch et al. developed a fluid-structure interaction solver able to capture the turbulence around the sail and its deformed shape, using the RANS solver CFX for CFD and an in-house structural solver. The CFD solver uses the $k-\omega$ model for turbulence and it is able to compute the separation point and wake behind the spinnaker. This solver is then coupled with the structural solver, and the final flying shape is computed. This is shown in figures 1.4 and 1.5, where the difference both in shapes and forces can be appreciated: in figure 1.5 the light blue line is the result of a mere CFD analysis, while the dark blue comes from a Fluid-Structure interaction analysis. It can be noted that the trend of the forces is well described, but the values are still off. The difference in the results between the rigid and flexible body simulation increases as the wind angle increases, so it can be argued that the deformation of the sail has a higher influence on the results the more the yacht is sailing away from the wind. For example, from figure 1.5 the highest error in the heeling force is for the highest measured angle of attack, and the error with respect to the deformed shape is in the range of 60%. For the driving force, the higher measured error is for the lowest angle and is in the range of 5%.

From this observation one can deduce the need for a more elaborate software that can not only compute the pressure field around the sail, but also the deformation of the sail due to this pressure field, seeing the big difference in results in the two cases. The need for a more accurate estimation is motivated by the fact

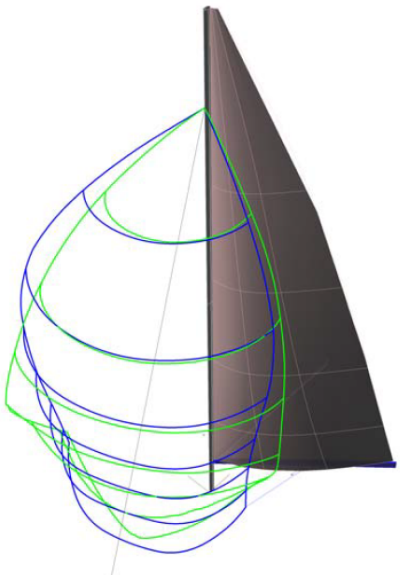


Figure 1.4: Design shape (blue) and computed flying shape (green) from [17]

that in the design analysis process of the whole yacht, for example a wrong estimation of the heeling force of one of the sails can lead to structural problems due to the under- or over-dimensioning of the structure. Similarly, an error in the driving force estimation will lead to an over- or under-estimation of the propulsion given by the sail, with the consequent over- or under-estimation of the dimensions of the sheets, wires and load-bearing beams. This project will then address this topic and try to create a reliable tool able to solve this problem.

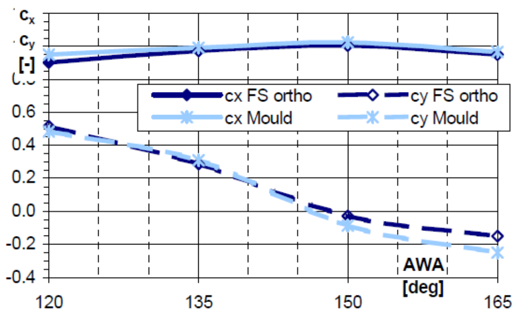


Figure 1.5: Normalised driving and side forces using the deforming shape (dark blue) and the rigid design shape (light blue), from [17]

The accuracy of the thrust estimation is necessary to the design analysis process, both for the constructor, who can give more detailed information about its product to the customer, and most importantly for the customer, who will use the sails and might have to dimension and rig the rest of the boat based on the provided data. These types of coupled solvers already exist (for example the North Sails tool Flow/MemBrain and the company K-Epsilon's FSI solver[3]) but are not open source and therefore directly available. The aim of this project is, therefore, to create a tool that can be used by everyone, even small sail constructors that might not be able to afford the other available options.

1.2. Research Questions

The research questions that will be addressed in the thesis are:

[3]http://www.k-epsilon.com/fsi_products.html

1. What are the appropriate techniques to compute the correct pressure distribution around the spinnaker sail through a CFD solver?
2. What techniques are able to capture the spinnaker's flying shape by coupling the CFD solver with a structural solver in an efficient manner?
3. With the results from the two previous points, can the driving and side forces generated by the spinnaker be computed correctly and, possibly, optimized using various trim settings?

This project will then work towards the resolution of these problems, with the final goal of having a reliable prediction of the spinnaker's performance before testing it in reality. Ideally, in the future, this coupling tool could be embedded in an automated design environment where changes to the shape of the sail will be predicted to maximize its performance. It is worth mentioning that the research questions need to be answered in the presented order, as the answer of the first question is the starting point for the second and so on.

1.3. Structure of the Report

In the *Theoretical Basis* chapter the necessary information regarding the CFD, FEM and FSI techniques used in the project will be reported, after being carefully selected through the literature.
Afterwards, the *CFD Simulations* chapter will explain all the preliminary work that has been done before the simulations, together with the motivation for all the choices made for the setup, run and post processing. Within this chapter the sections regarding the presentation of the solver, the 2D simulations and 3D simulations can be found.
The chapter *FEM Simulations* will explain how the used FEM solver works, what equations it is based on and what types of inputs and outputs it deals with. The solver will also be validated with some references cases of which the analytic solution is known. Eventually, the obtained results for the test case at hand will be qualitatively validated with some real life data, due to the lack of reference data.
Following, the *FSI Simulations* chapter will present the core of this thesis project: how the coupling between the fluid and structural solver has been done and what tools have been used to obtain a successful coupling.
In the *Results* chapter, the obtained results will be presented, commented and validated with data obtained from the literature.
Lastly, in the *Conclusions* and *Recommendations* chapter the performed work will be analyzed and its validity asserted, adding some comments that can be useful for future projects in this field.

2
Theoretical Basis

In this chapter, all the preliminary knowledge necessary for the completion of this project is presented. Differently from the literature study, where all the relevant available techniques were presented, here only the ones actually used in the project will be reported in detail. In the literature study, the suitability of the available techniques for this specific case was evaluated and the more likely to be successful choices were made.

2.1. Computational Fluid Dynamics

In this section, the most common numerical method for capturing the physics of sailing with a spinnaker are explained. As it has been mentioned in the conclusions of chapter 1, the numerical method needed for this application should be able to correctly predict unsteady phenomena such as the separation point on the sail and the wake, and therefore it needs to have an accurate representation of turbulence. The goal is to obtain the pressure distribution on the sail and consequently the driving and side forces generated by the sail. Particular attention will be given to the RANS equations, underlining their the advantages and disadvantages for this particular application.

2.1.1. RANS

First of all, it is worth mentioning the incompressible Navier Stokes Equations, which describe the fluid motion:

$$\frac{\partial u_i}{\partial x_i} = 0, \tag{2.1}$$

$$\rho\frac{\partial u_i}{\partial t} + \rho\frac{\partial}{\partial x_j}(u_i u_j) = -\frac{\partial p}{\partial x_i} + \frac{\partial \tau_{ij}}{\partial x_j}, \tag{2.2}$$

where u is the velocity, p is pressure, τ the shear stress and ρ the density. Equation 2.1 expresses the mass continuity, while equation 2.2 is the momentum conservation equation. The Reynolds averaging procedure, which results in the RANS equations, consists in decomposing the unknown velocity u in a mean component $\bar{u}$ and a fluctuating component u':

$$u = \bar{u} + u'. \tag{2.3}$$

The mean component is obtained by averaging in time:

$$\bar{u}(x) = \lim_{N\to\infty}\frac{1}{N}\sum_{i=0}^{N} u(x,t_i). \tag{2.4}$$

The Reynolds Averaged Navier Stokes equations are obtained by substituting 2.3 into the incompressible Navier Stokes equations and then averaging the result, which gives:

$$\frac{\partial \bar{u}_i}{\partial x_i} = 0, \tag{2.5}$$

$$\rho\frac{\partial \bar{u}_i}{\partial t} + \rho\frac{\partial}{\partial x_j}(\bar{u}_i\bar{u}_j) = -\frac{\partial \bar{p}}{\partial x_i} + \frac{\partial}{\partial x_j}(\bar{\tau}_{ij} - \rho\overline{u_i' u_j'}), \tag{2.6}$$

where the term $\rho \overline{u'_i u'_j}$ is known as the Reynolds Stress. Equation 2.6 is derived from the momentum equation, where the fluctuating velocities u'_i caused by the turbulence have been averaged over time, so the equation represents a time-averaged balance of momentum.

Equations 2.5 and 2.6 do not form a closed set, as there are more unknowns than equations. Additional models have to be introduced to describe the Reynolds stress. Various expressions for this term have been proposed, and some equations have been added in order to close the system. The expression of this term can vary based on the choice of the turbulence model. In the literature, many turbulence models have been applied to this specific case and their performance evaluated. Table 2.1 shows the turbulence model that yielded the best results according to the authors.

Author	**Choice**
Lasher and Sonnemeier [40]	Realizable k-ε
Renzsch and Graf [17]	Baseline k-ω
Renzsch and Graf [15]	SST k-ω
Viola [39]	No turbulence model
Lombardi et al. [25]	SST k-ω
Wright et al. [43]	SST k-ω

Table 2.1: Chosen turbulence models in literature

It can be argued that unanimity on the choice of the model has not been reached and that there is no perfect turbulence model for this application. That is because different authors chose different models and some authors even tried with various models ([39, 40, 17], only to conclude that all considered none is better than the other. This makes the choice of the turbulence model complicated, however it seems like the most preferred model is the SST k-ω, probably because of its accurate prediction of the separation point in adverse pressure gradient, which is crucial in this application. For these reasons, this model was used in the current project and its details are presented below, first giving an overview of the standard k-ε and k-ω model and then of the SST k-ω model.

2.1.1.1. The standard k-ε model

This model, developed by Launder and Spalding in 1972 [4], is based on the Bousinnesq eddy viscosity hypotesis, for which the Reynolds stress is related to the turbulent kinetic energy k, the mean flow strain rate tensor s_{ij} and the turbulent viscosity ν_t by the relation:

$$-\overline{u'_i u'_j} = -\frac{2}{3}k\delta_{ij} + 2\nu_t s_{ij}, \tag{2.7}$$

where δ_{ij} is the Kronecker delta and s_{ij} is defined as

$$s_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right), \tag{2.8}$$

and the turbulent kinetic energy is a function of the fluctuating velocity in all 3 directions:

$$k = \frac{1}{2}\left(\overline{(u')^2} + \overline{(v')^2} + \overline{(w')^2}\right). \tag{2.9}$$

In order to close the system, the quantities k and ν_t have to be defined as functions of the mean flow variables $\bar{u}_i$ and $\bar{p}$. There are different definitions for these quantities found over the years, and one of the most common ones is the standard $k-\varepsilon$. This ε represents the dissipation of turbulent kinetic energy per unit mass and can be used to model ν_t through the following relation:

$$\nu_t = c_\mu \frac{k^2}{\varepsilon}. \tag{2.10}$$

Two more equations are required to calculate k and ε, and they are defined as follows:

$$\frac{Dk}{Dt} = \frac{\partial}{\partial x_j}\left[\left(\nu + \frac{\nu_t}{\sigma_k}\right)\frac{\partial k}{\partial x_j}\right] + G - \varepsilon, \tag{2.11}$$

$$\frac{D\varepsilon}{Dt} = \frac{\partial}{\partial x_j}\left[\left(\nu + \frac{\nu_t}{\sigma_\varepsilon}\right)\frac{\partial \varepsilon}{\partial x_j}\right] + C_{1\varepsilon}\frac{\varepsilon}{k}G - C_{2\varepsilon}\frac{\varepsilon^2}{k}, \tag{2.12}$$

where

$$\sigma_k = 1.00, \quad \sigma_\varepsilon = 1.30, \quad C_{1\varepsilon} = 1.44, \quad C_{2\varepsilon} = 1.92, \quad c_\mu = 0.09, \tag{2.13}$$

while G represents the generation rate of turbulent kinetic energy and is defined as:

$$G = \nu_t\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)\frac{\partial u_i}{\partial x_j}. \tag{2.14}$$

It has to be pointed out that this model has been developed for high Reynolds number flows, and thus not the in the area immediately neighboring the sail, where viscous damping is dominant. In this areas wall functions have to be used. Two main ones are available: the standard wall function based on the proposal of Launder and Spalding [5] and a nonequilibrium wall function proposed by Kim and Choudhury [33]. The second one is more suitable for this study since it deals better with separation, which is highly predominant in spinnaker flow. The approximations introduced in this method (wall functions, estimate of production and dissipation of ε...) are expected to impact the accuracy of the model significantly.

This turbulence model showed to have some disadvantages [24] when dealing with diffusion in adverse pressure gradients, separation and reattachment, making it unsuitable for downwind sailing flows, but it has been still widely used in industrial applications due to its stability.

2.1.1.2. The standard k-ω model

This model, introduced by Wilcox in 1988 [42] is also a two-equation model but instead of adding an equation for ε it adds it for the specific dissipation rate ω, defined as $\omega = \varepsilon / k$.

The equations needed to determine k and ω are defined as follows:

$$\frac{\partial}{\partial t}(\rho k) + \frac{\partial}{\partial x_j}(\rho u_j k) = \tau_{ij}\frac{\partial u_i}{\partial x_j} - \beta^\star \rho k + \frac{\partial}{\partial x_j}\left[(\mu + \sigma^\star \mu_T)\frac{\partial k}{\partial x_j}\right], \tag{2.15}$$

$$\frac{\partial}{\partial t}(\rho \omega) + \frac{\partial}{\partial x_j}(\rho u_j \omega) = (\gamma\omega/k)\tau_{ij}\frac{\partial u_i}{\partial x_j} - \beta\rho\omega^2 + \frac{\partial}{\partial x_j}\left[(\mu + \sigma \mu_T)\frac{\partial \omega}{\partial x_j}\right], \tag{2.16}$$

where t is time, x_i position vector, u_i velocity vector, ρ density, p pressure, μ molecular viscosity, and τ_{ij} is the Reynolds stress tensor. μ_T is dependent on the quantities k and ω by the relation:

$$\mu_T = \gamma^\star \frac{\rho k}{\omega}. \tag{2.17}$$

The many constants presented in the previous equations can be defined as follows:

$$\beta = 3/40, \quad \beta^\star = 9/100, \quad \gamma = 5/9, \quad \gamma^\star = 1, \quad \sigma = 1/2, \quad \sigma^\star = 1/2. \tag{2.18}$$

This method has some advantages when compared to the k-ε model, the main one being that the ω equation can still be solved in the vicinity of the wall, if a finite value of ω is prescribed at the wall. This diminishes the computation times, being there no need to use wall functions. Moreover, the equation for ω can be integrated accurately in the boundary layer, yielding a better approximation of the separation point, and the whole method is less numerically stiff so it allows the use of larger time steps. The main shortcoming on the other hand is that the needed value of ω at the wall is hard to predict, since its value is rarely known a priori.

2.1.1.3. The Menter k-ω Shear Stress Transport model

Menter [26] in 1994 developed a new model, which blends the k-ε and the k-ω so that they can be used in the areas where they give best results. As it has been said, the k-ω model is superior in the near-wall region, while the k-ε works better in the freestream because of its insensivity to the freestream boundary conditions. For this reason each model will be used where it works best: for the inner 50% of the boundary layer the k-ω will be used, while for the rest the standard k-ε will be used. In order to have a smooth transition from one model to the other some blending functions are introduced, the value of which is zero at the inner edge of a turbulent boundary layer and one at the outer edge of the layer. The other modification the SST model brings is that the eddy viscosity is changed in regions of adverse pressure gradient, so that the shear stress τ in a boundary layer becomes proportional to the kinetic energy:

$$\tau = \rho a_1 k, \tag{2.19}$$

where a_1 is a constant. The closure equations of this method are, for k equation 2.15, and for ω:

$$\frac{\partial}{\partial t}(\rho\omega) + \frac{\partial}{\partial x_j}(\rho u_j \omega) = \frac{\gamma}{\nu_t} P - \beta\rho\omega^2 + \frac{\partial}{\partial x_j}\left[(\mu + \sigma_\omega \mu_T)\frac{\partial \omega}{\partial x_j}\right] + 2(1-F_1)\frac{\rho\sigma_{\omega 2}}{\omega}\frac{\partial k}{\partial x_j}\frac{\partial \omega}{\partial x_j}. \tag{2.20}$$

The introduced variables have the following definitions:

$$P = \tau_{ij}\frac{\partial u_i}{\partial x_j}, \quad \mu_T = \frac{\rho a_1 k}{max(a_1\omega, \Omega F_2)}, \quad \nu_T = \frac{\mu_T}{\rho}, \quad F_1 = tanh(arg_1^4), \quad arg_1 = min\left[max\left(\frac{\sqrt{k}}{\beta^\star \omega y}\right), \frac{4\rho\sigma_{\omega 2} k}{CD_{k\omega} y^2}\right],$$

$$CD_{k\omega} = max\left(2\rho\sigma_{\omega 2}\frac{1}{\omega}\frac{\partial k}{\partial x_j}\frac{\partial \omega}{\partial x_j}, 10^{-20}\right), \quad F_2 = tanh(arg_2^2), \quad arg_2 = max\left(2\frac{\sqrt{k}}{\beta^\star \omega y}, \frac{500\nu}{y^2\omega}\right),$$

$$\tau_{ij} = \mu_T\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3}\frac{\partial u_k}{\partial x_k}\delta_{ij}\right) - \frac{2}{3}\rho k \delta_{ij}$$

where y and Ω are the distance to the next surface and the absolute value of the vorticity, respectively. The introduced constants are:

$$\gamma = 0.15, \quad \beta = 0.075, \quad \sigma_\omega = 0.5, \quad \sigma_{\omega_2} = 0.856, \quad \beta^\star = 0.09, \quad a_1 = 0.31.$$

The key of the model lays in the definition of μ_T: F_2 is defined such that it is 1 in the boundary layer and 0 in free shear areas, therefore the quantity μ_T will have two different definitions when considering boundary layer flow or free shear flow. More specifically, in the adverse pressure gradient BL the production of k is larger than its dissipation, namely $\Omega > a_1\omega$, so the defintion of μ_T guarantees that Eq. 2.19 is satisfied, while the original formulation $\mu_T = \rho k/u$ is used for the rest of the flow. This method combines the strengths of the two methods and for this reason performs well when predicting the separation point in adverse pressure gradient.

2.2. Finite Element Analysis

In chapter 1 the structural properties of the spinnaker have been already summarized, and the main characteristics are: the sail is very thin, presents large deformations and large displacements, has no bending stiffness and therefore does not support compressional loads, only tensional. As it has been stated in the literature study, many studies have been conducted on determining what finite element model was the most suitable for the downwind sails.

The membrane formulation seems the one that has been adopted more often (see [17], [15], [16], [12]), even though it has been reported in [15] that the membrane model is not enough to cover the physics involved, since the sailcloth has a negligible bending stiffness and therefore negligible buckling strength, with compressive loads causing the cloth to wrinkle. The membrane formulation has the same stiffness matrix under compression as well as under tension. The necessity for an extra model able to overcome this lack arose. For this reason an isotropic wrinkling model was added, which allowed to take into account the different behavior under different loads. However, the work that introduced this formulation, named Flexsail, done by Renzsch et al., has still not been made available to the public. Other softwares (such as ARA from k-Epsilon, [32], and RELAX by P. Heppel [36]) are available to the public and have been investigated with the goal of using them in this project, but they both turned out to be very difficult to compile and to use for this specific application.

For this reason, the Matlab code developed by Daniele Trimarchi in his Master Thesis ([35], [12]) has been chosen, for many reasons: it is written in Matlab and therefore easy to use and debug, it is of quite simple understanding for someone who does not have any FEM experience (like the author) and it is designed specifically for sails.

This section describes the governing equations used in this code and the type of element that has been implemented, referring for the theory to [12, 35, 23]. The chapter "FEM Simulations" will then describe what techniques are used to find the solution.

2.2.1. Basics of Finite Elements Method

The principle behind the Finite Element method consists in subdividing the domain in small subdomains, namely the elements, where the differential equations that describe the behavior of the material are left unaltered. In this way the equations that describe the whole domain as a continuum are discretized and solved only on certain points, namely the nodes, that are generally located at the sides of the elements. Once the values at the nodes are known the variables can be defined inside the elements as well through some approximating functions, also known as shape functions.

2.2.1.1. Governing equations

One of the most used principles for the development of the Finite Element Theory is the Principle of Virtual Works, that says:
"*The application of a virtual displacement to a system in equilibrium generates a system of forces (internal and external) for which the work produced by the internal forces is equal to the work produced by the external forces*".

The work of a force is defined as the scalar product of the force with the displacement of the point of application of the force, and this holds also for the virtual work:

$$dL = F \cdot ds, \tag{2.21}$$

and therefore the principle of virtual works becomes:

$$\int_V \varepsilon^T \cdot \sigma dV = \int_V u_B^T \cdot f_B dV + \int_S u_S^T \cdot f_S dS + \sum_i u_i^T \cdot f_i, \tag{2.22}$$

where ε is the deformation vector, σ the stress vector, f_B the vector containing the volume forces, f_S the surface forces vector and f_i the concentrated forces vector. u_B represents the vector of the displacements of the internal nodes, u_s of the surface nodes and u_i of the nodes on which the concentrated forces are applied. This principle is very powerful and helpful in determining the matricial relations necessary for the implementation of the FEM solver. With this in mind, it is possible to express equation 2.22 in discretized form:

$$\sum_{m=1}^{k}\int_{V_m}\varepsilon_m^T\cdot\sigma_m dV=\sum_{m=1}^{k}\int_{V_m}u_{B_m}^T\cdot f_{B_m}dV+\sum_{m=1}^{k}\int_{S_m}u_{S_m}^T\cdot f_{S_m}dS+\sum_{i=1}^{n}u_i^T\cdot f_i, \tag{2.23}$$

where k is the number of elements and n the number of nodes on which concentrated forces are applied. Now introducing some relations:

$$u_m=H_m\cdot\overline{u}, \tag{2.24}$$

where u_m is the displacement vector of the m^{th} element, H_m an extrapolation matrix and $\overline{u}$ the global displacement vector. Moreover

$$\varepsilon_m=B_m\cdot\overline{u}, \tag{2.25}$$

where ε_m is the deformation vector of the m^{th} element and B_m the linear deformation-displacement matrix defined as

$$B_m=\begin{bmatrix}\frac{\partial}{\partial x}&0&0\\0&\frac{\partial}{\partial y}&0\\0&0&\frac{\partial}{\partial z}\\0&\frac{\partial}{\partial z}&\frac{\partial}{\partial y}\\\frac{\partial}{\partial z}&0&\frac{\partial}{\partial x}\\\frac{\partial}{\partial y}&\frac{\partial}{\partial x}&0\end{bmatrix}.$$

Moreover the stress-strain relation can be defined:

$$\sigma_m=C_m\cdot\varepsilon+\sigma^0, \tag{2.26}$$

where σ_m is the stress vector for the m^{th} element, σ^0 the initial stess vector and C_m the compliance matrix defined as

$$C_m=\begin{bmatrix}1&\frac{\nu}{1-\nu}&\frac{\nu}{1-\nu}&0&0&0\\\frac{\nu}{1-\nu}&1&\frac{\nu}{1-\nu}&0&0&0\\\frac{\nu}{1-\nu}&\frac{\nu}{1-\nu}&1&0&0&0\\0&0&0&\frac{1-2\nu}{2(1-\nu)}&0&0\\0&0&0&0&\frac{1-2\nu}{2(1-\nu)}&0\\0&0&0&0&0&\frac{1-2\nu}{2(1-\nu)}\end{bmatrix}.$$

Then the principle of virtual works can be rewritten as

$$\left[\sum_{m=1}^{k}\int_{V_m}B_m^T\cdot C_m\cdot B_m dV\right]\cdot\overline{u}=\left[\left\{\sum_{m=1}^{k}\int_{V_m}H_{B_m}^T\cdot f_{B_m}dV\right\}+\left\{\sum_{m=1}^{k}\int_{S_m}H_{S_m}^T\cdot f_{S_m}dS\right\}+\left\{f_i\right\}-\left\{\sum_{m=1}^{k}\int_{V_m}B_m^T\cdot\sigma_m^0 dV\right\}\right], \tag{2.27}$$

which can be expressed in compact form as:

$$[\mathbf{K}]\cdot\overline{u}=R, \tag{2.28}$$

where **K** is the stiffness matrix and R is the force vector, defined as:

$$[\mathbf{K}]=\sum_{m=1}^{k}\int_{V_m}B_m^T\cdot C_m\cdot B_m dV, \tag{2.29}$$

$$R=\sum_{m=1}^{k}\int_{V_m}H_{B_m}^T\cdot f_{B_m}dV+\sum_{m=1}^{k}\int_{S_m}H_{S_m}^T\cdot f_{S_m}dS+f_i-\sum_{m=1}^{k}\int_{V_m}B_m^T\cdot\sigma_m^0 dV. \tag{2.30}$$

In general the stiffness matrix **K** of an element represents the linear relationship between the applied forces and the consequent displacements. The final goal is to obtain the global displacements, once the stiffness matrix is constructed in function of the geometry and the force vector is defined. This can be done by inverting equation 2.28:

$$\overline{u} = [\mathbf{K}]^{-1} \cdot R. \tag{2.31}$$

In order to construct the global stiffness matrix, it is necessary to sum up all the local stiffness matrices of the single elements. Each entry of the matrix $\mathbf{K}_{ij}$ corresponds to the effect that a force applied to node j has to the displacement of node i. If for a node more elements have effect on its displacements, the entries for the local stiffness matrices will be summed on that node.

2.2.1.2. Shape Functions

With the previous paragraph, the equations to determine the nodal displacements have been outlined. The next step consists in linking the nodal displacement with the field of displacement of the element, finally arriving to obtain deformations and stresses within the element. This will have to be done using some analytic equations that consider the element as a continuum, also known as shape functions.

There are various types of elements, and each one of them has a different stiffness matrix. In order to define the stiffness matrix it is necessary to univocally define the field of displacement within the element. For this reason, the shape functions are useful in two ways: they are necessary to build the stiffness matrix and then to obtain stresses and deformations from the nodal displacements.

The choice for the shape functions N_i, that are usually polynomials, is crucial to obtain a solution that is more or less close to the simulated reality. Each function represents the weight that each component of nodal displacement has in determining the displacement of a generic node inside the element. The field of displacement inside the element can be defined as:

$$u = \sum_{i=1}^{M} N_i(x, y, z) \cdot u_i; \tag{2.32}$$

where u_i is the displacement of the i^{th} node and M the total number of nodes in the element. There are some requirements for the shape functions, in order for them to correctly represent the nodal values:

- They have to yield a deformation equals to zero when the nodal displacement field corresponds to rigid motion (for which the distance between any two given points remains constant in time),
- They have to yield a constant deformation when the nodal displacement field is compatible with said condition,
- They have to yield deformations that are compatible with the interface between different elements,
- They have to guarantee that the displacement of a node on an edge of an element does not depend on the displacements of the opposite node, namely assume value 1 for the considered node and zero for all the other nodes of the element.

Many formulations for the shape functions are available, with the most famous one being the "hat function" for the 1D case: it is 1 for the node it refers to and 0 for all the other ones, and has a linear connection within neighboring nodes, as shown in figure 2.1.

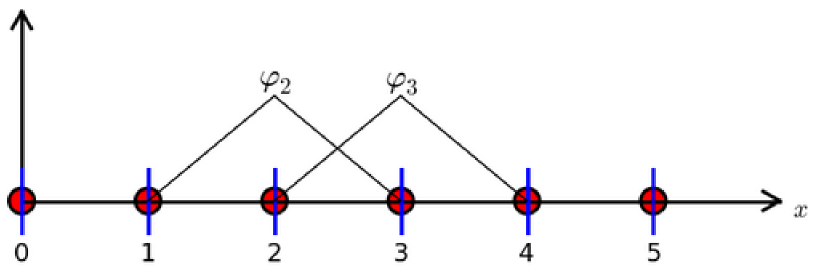


Figure 2.1: Linear shape functions for the 1D case: the function φ_2 is 1 for node 2 and zero everywhere else, and so on.

For a two dimensional triangular element, as shown in figure 2.2 the shape functions can be defined in the following way, still complying with the requirements defined above:

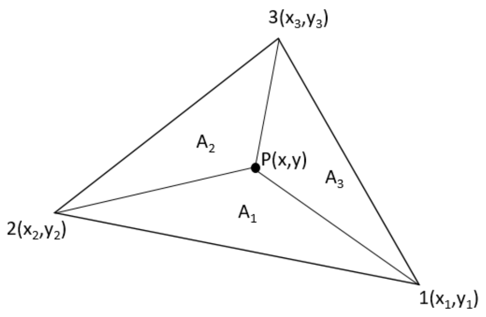


Figure 2.2: Triangular two dimensional element

$$N_1 = \frac{A_1}{A}, \qquad N_2 = \frac{A_2}{A}, \qquad N_3 = \frac{A_3}{A}, \tag{2.33}$$

where A_i are the areas as defined in figure 2.2 and A is the total area of the element. The relation between the shape functions and the Cartesian coordinates of a generalized point internal to the element is:

$$x = N_1 \cdot x_1 + N_2 \cdot x_2 + N_3 \cdot x_3, \tag{2.34}$$

$$y = N_1 \cdot y_1 + N_2 \cdot y_2 + N_3 \cdot y_3, \tag{2.35}$$

that results in

$$\begin{bmatrix} 1 \\ x \\ y \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_2 & y_2 & y_3 \end{bmatrix} \cdot \begin{bmatrix} N_1 \\ N_2 \\ N_3 \end{bmatrix}, \tag{2.36}$$

where the first line derives from the relation $N_1 + N_2 + N_3 = 1$. From this expression it will be sufficient to invert the system to obtain the shape functions as functions of the nodal and generalized coordinates:

$$\begin{bmatrix} N_1 \\ N_2 \\ N_3 \end{bmatrix} = \frac{1}{2A} \begin{bmatrix} x_2 y_3 - x_3 y_2 & y_2 - y_3 & x_2 - x_3 \\ x_3 y_1 - x_1 y_3 & y_3 - y_1 & x_3 - x_1 \\ x_1 y_2 - x_2 y_1 & y_1 - y_2 & x_1 - x_2 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ x \\ y \end{bmatrix}, \tag{2.37}$$

Once the shape functions are defined it is possible to obtain the displacement fields simply by multiplying the shape functions by the known displacements.

2.2.2. Membrane Element Formulation

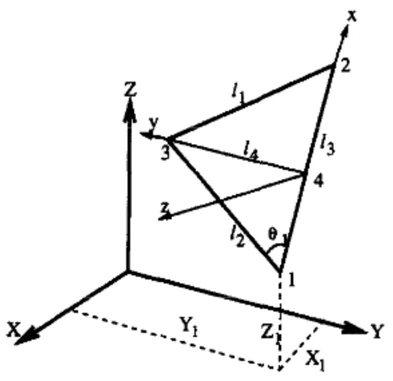


Figure 2.3: Triangular membrane element, from [3]

Now a more detailed explanation of the chosen element will be presented, as reported by Tabarrok [3]. As it has been said before, the membrane element is the one that has been used more extensively for this type of application in the literature. The main resemblance of this element with respect to the real sail is its null resistance to compression or bending. The basic element does not take into account the possibilities of wrinkling or buckling and that is something that can be added if one searches for a more precise representation of the reality.
In figure 2.3 the element is reported in a global reference frame. Another local reference frame is defined on the element plane. When expressing the displacements linearly over the element one can write:

$$u(x,y) = \alpha_1 + \alpha_2 x + \alpha_3 y, \qquad v(x,y) = \alpha_4 + \alpha_5 x + \alpha_6 y, \qquad w(x,y) = \alpha_7 + \alpha_8 x + \alpha_9 y, \tag{2.38}$$

where the coefficients α are not yet specified. They can be expressed as functions of the nodal displacements $u_m = [u_1 \quad v_1 \quad w_1 \quad u_2 \quad v_2 \quad w_2 \quad u_3 \quad v_3 \quad w_3]$, as seen in equation 2.37. When doing so equation 2.38 becomes

$$u(x,y) = (a_1 + b_1 x + c_1 y)u_1 + (a_2 + b_2 x + c_2 y)u_2 + (a_3 + b_3 x + c_3 y)u_3, \tag{2.39}$$
$$v(x,y) = (a_1 + b_1 x + c_1 y)v_1 + (a_2 + b_2 x + c_2 y)v_2 + (a_3 + b_3 x + c_3 y)v_3, \tag{2.40}$$
$$w(x,y) = (a_1 + b_1 x + c_1 y)w_1 + (a_2 + b_2 x + c_2 y)w_2 + (a_3 + b_3 x + c_3 y)w_3, \tag{2.41}$$

where

$$a_1 = \frac{x_2 y_3 - x_3 y_2}{2A}, \qquad b_1 = \frac{y_2 - y_3}{2A}, \qquad c_1 = \frac{x_2 - x_3}{2A}, \tag{2.42}$$
$$a_2 = \frac{x_3 y_1 - x_1 y_3}{2A}, \qquad b_2 = \frac{y_3 - y_1}{2A}, \qquad c_2 = \frac{x_3 - x_1}{2A}, \tag{2.43}$$
$$a_3 = \frac{x_1 y_2 - x_2 y_1}{2A}, \qquad b_3 = \frac{y_1 - y_2}{2A}, \qquad c_3 = \frac{x_1 - x_2}{2A}, \tag{2.44}$$

where A is the area of the triangle and also the determinant of the matrix defined in equation 2.37, and the coordinates x, y and z are defined in figure 2.3.

The membrane element is then a triangular, bi-dimensional, non linear element unable to withstand any bending or torsion. Then the applied load, normal to the surface, is only balanced by the stress tangent to the membrane plane. When the load varies, the shape of the element will also vary to ensure the equilibrium. Hence, big deformations and rotations can be expected within this element and therefore the assumption of small deflection for the linear theory does not withstand. The nonlinear displacement-deformation relation can then be expressed as:

$$\varepsilon_x = \frac{\partial u}{\partial x} + \frac{1}{2}\cdot\left[\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial w}{\partial x}\right)^2\right], \tag{2.45}$$

$$\varepsilon_y = \frac{\partial v}{\partial y} + \frac{1}{2}\cdot\left[\left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2 + \left(\frac{\partial w}{\partial y}\right)^2\right], \tag{2.46}$$

$$\gamma_{xy} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \left[\frac{\partial u}{\partial x}\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\frac{\partial v}{\partial y} + \frac{\partial w}{\partial x}\frac{\partial w}{\partial y}\right]. \tag{2.47}$$

Now, substituting equation 2.39, 2.40, 2.41 in equations 2.45, 2.46, 2.47 and writing in matricial form we obtain:

$$\varepsilon = B_0\cdot u + \frac{1}{2}A\cdot\theta; \tag{2.48}$$

where

$$B_0 = \begin{bmatrix} b_1 & 0 & 0 & b_2 & 0 & 0 & b_3 & 0 & 0 \\ 0 & c_1 & 0 & 0 & c_2 & 0 & 0 & c_3 & 0 \\ c_1 & b_1 & 0 & c_2 & b_2 & 0 & c_3 & b_3 & 0 \end{bmatrix}, \qquad A = \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial v}{\partial x} & \frac{\partial w}{\partial x} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\partial u}{\partial y} & \frac{\partial v}{\partial y} & \frac{\partial w}{\partial y} \\ \frac{\partial u}{\partial y} & \frac{\partial v}{\partial y} & \frac{\partial w}{\partial y} & \frac{\partial u}{\partial x} & \frac{\partial v}{\partial x} & \frac{\partial w}{\partial x} \end{bmatrix},$$

$$\theta = \begin{bmatrix} \frac{\partial u}{\partial y} & \frac{\partial v}{\partial y} & \frac{\partial w}{\partial y} & \frac{\partial u}{\partial x} & \frac{\partial v}{\partial x} & \frac{\partial w}{\partial x} \end{bmatrix}^T.$$

Moreover, $\theta = G\cdot u$, where

$$G = \begin{bmatrix} b_1 & 0 & 0 & b_2 & 0 & 0 & b_3 & 0 & 0 \\ 0 & b_1 & 0 & 0 & b_2 & 0 & 0 & b_3 & 0 \\ 0 & 0 & b_1 & 0 & 0 & b_2 & 0 & 0 & b_3 \\ c_1 & 0 & 0 & c_2 & 0 & 0 & c_3 & 0 & 0 \\ 0 & c_1 & 0 & 0 & c_2 & 0 & 0 & c_3 & 0 \\ 0 & 0 & c_1 & 0 & 0 & c_2 & 0 & 0 & c_3 \end{bmatrix}.$$

Therefore $\varepsilon = B_0\cdot u + \frac{1}{2}A\theta \quad \Rightarrow \quad \varepsilon = B_0\cdot u + \frac{1}{2}A\cdot G\cdot u$.
From this it can be stated that $\delta\varepsilon = (B_0 + A\cdot G)\cdot\delta u$. Now remembering equation 2.26:

$$\sigma = C\cdot\varepsilon + \sigma_0 = C\cdot(B_0\cdot u + \frac{1}{2}A\cdot G\cdot u) + \sigma_0, \tag{2.49}$$

it is possible to rephrase the principle of virtual works in the following way:

$$\int_{V^e}\delta\varepsilon\cdot\sigma dV - \delta u^T\cdot p = 0, \tag{2.50}$$

substituting the expression for $\delta\varepsilon$ and dividing by δu^T:

$$\int_{V^e}(B_0 + A\cdot G)^T\cdot\left[C\cdot(B_0\cdot u + \frac{1}{2}A\cdot G\cdot u) + \sigma_0\right]dV - p = 0. \tag{2.51}$$

Solving this equation iteratively will yield a residual, ϕ_i, instead of the zero on the right hand side:

$$\int_{V^e}(B_0 + A^i\cdot G)^T\cdot\left[C\cdot(B_0\cdot u^i + \frac{1}{2}A^i\cdot G\cdot u^i) + \sigma_0\right]dV - p = \phi^i. \tag{2.52}$$

In the next iteration $i+1$ then,

$$\phi^{i+1} = \phi^i + \frac{\partial\phi^i}{\partial u}\cdot\Delta u^i = 0 \quad \Rightarrow \quad \frac{\partial\phi^i}{\partial u}\cdot\Delta u^i = -\phi^i. \tag{2.53}$$

Noting that $u^{i+1} = u^i + \Delta u^i$ an important observation can be made, namely that equation 2.53 is equivalent to $K^i \cdot u^i = f^i$. Then,

$$K^i = \frac{\partial \phi^i}{\partial u} = \int_{V^e} (B_0 + A^i \cdot G)^T \cdot \frac{\partial}{\partial u} \left[C \cdot (B_0 \cdot u^i + \frac{1}{2} A^i \cdot G \cdot u^i) \right] dV + \int_{V^e} \frac{\partial}{\partial u} (B_0 + A^i \cdot G)^T \cdot \left[C \cdot (B_0 \cdot u^i + \frac{1}{2} A^i \cdot G \cdot u^i) + \sigma_0 \right] dV = K_e^i + K_g^i, \tag{2.54}$$

and finally it is possible to recognize the two stiffness matrices that create the global one, namely the geometric stiffness K_g and the elastic stiffness K_e, defined as:

$$K_e^i = \int_{V^e} (B_0 + A^i \cdot G)^T \cdot C \cdot (B_0 + A^i \cdot G) dV = \int_{V^e} B_0^T \cdot C \cdot B_0 dV + \int_{V^e} (A^i \cdot G)^T \cdot C \cdot (A^i \cdot G) dV, \tag{2.55}$$

$$K_g^i = \int_{V^e} G^T \cdot \frac{\partial (A^i)^T}{\partial u} \cdot [C \cdot (B_0 \cdot u^i + \frac{1}{2} A^i \cdot \theta^i) + \sigma_0] dV = \int_{V^e} G^T \cdot \frac{\partial (A^i)^T}{\partial u} \cdot \sigma^i dV = \int_{V^e} G^T \cdot M^i \cdot G dV, \tag{2.56}$$

where

$$M^i = \begin{bmatrix} \sigma_x^i & 0 & 0 & \tau_{xy}^i & 0 & 0 \\ 0 & \sigma_x^i & 0 & 0 & \tau_{xy}^i & 0 \\ 0 & 0 & \sigma_x^i & 0 & 0 & \tau_{xy}^i \\ \tau_{xy}^i & 0 & 0 & \sigma_y^i & 0 & 0 \\ 0 & \tau_{xy}^i & 0 & 0 & \sigma_y^i & 0 \\ 0 & 0 & \tau_{xy}^i & 0 & 0 & \sigma_y^i \end{bmatrix}.$$

It can be noted that the terms to be integrated are 6x6 matrices, because they are defined in the 2D plane of the element, in order to see them in the global coordinate system they will have to be rotated and will become 9x9 matrices. Being the thickness of the element generally constant, the integral can then be expressed as $A_{el} * t$, where t is the thickness. Now it is possible to linearize the expression for the elastic stiffness, eliminating the mixed terms for ε, see equation 2.47, if the hypothesis of big displacements and small deformations is valid, which in this case can be considered true. The linearization consists in eliminating the term depending on the displacement, namely $\int_{V^e} (A^i \cdot G)^T \cdot C \cdot (A^i \cdot G) dV$. In this way the elastic stiffness matrix, not being function of the displacement, will be constant at each iteration and will not have to be updated, differently from the geometric stiffness matrix that will change at every cycle.

Now, Li and Chan [23] have reformulated the matrices K_e and K_g in the global coordinate system, namely in the three dimensions. The element is made of 3 nodes with 9 degrees of freedom. As seen in figure 2.5, the global coordinate system is made by X1, X2 and X3, while the local coordinate system is bi-dimensional and made by x and y.

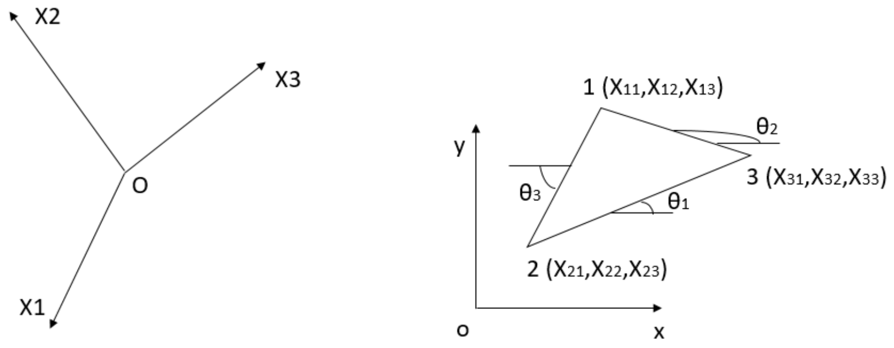


Figure 2.4: Definition of membrane element

After reformulating the two stiffness matrices in the global coordinate system, differently for what was done in the previous paragraph, the matrices have the form:

$$K_e = \int_V T_G^T \cdot T_N^T \cdot C \cdot T_N \cdot T_G dV = A \cdot t \cdot T_G^T \cdot T_N^T \cdot C \cdot T_N \cdot T_G, \tag{2.57}$$

where $A \cdot t$ is the element volume, C the compliance matrix, T_N the rototranslation matrix from local to global coordinate system and T_G the global displacement-deformation matrix. They are defined as:

$$C = \frac{E}{1-\nu^2}\begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix}, \quad T_N = \Psi^{-1} \cdot L_{0d}^{-1}, \quad L_{0d} = \begin{bmatrix} l_{023} & 0 & 0 \\ 0 & l_{031} & 0 \\ 0 & 0 & l_{013} \end{bmatrix}, \quad \Psi = \begin{bmatrix} cos^2\theta_1 & sin^2\theta_1 & sin\theta_1 cos\theta_1 \\ cos^2\theta_2 & sin^2\theta_2 & sin\theta_2 cos\theta_2 \\ cos^2\theta_3 & sin^2\theta_3 & sin\theta_3 cos\theta_3 \end{bmatrix},$$

$$T_G = \begin{bmatrix} 0 & 0 & 0 & \frac{X_{21}-X_{31}}{l_{023}} & \frac{X_{22}-X_{32}}{l_{023}} & \frac{X_{33}-X_{23}}{l_{023}} & \frac{X_{31}-X_{21}}{l_{023}} & \frac{X_{32}-X_{22}}{l_{023}} & \frac{X_{23}-X_{33}}{l_{023}} \\ \frac{X_{11}-X_{31}}{l_{031}} & \frac{X_{12}-X_{32}}{l_{031}} & \frac{X_{13}-X_{23}}{l_{031}} & 0 & 0 & 0 & \frac{X_{31}-X_{11}}{l_{031}} & \frac{X_{32}-X_{12}}{l_{031}} & \frac{X_{33}-X_{13}}{l_{031}} \\ \frac{X_{11}-X_{21}}{l_{012}} & \frac{X_{12}-X_{22}}{l_{012}} & \frac{X_{13}-X_{23}}{l_{012}} & \frac{X_{21}-X_{11}}{l_{012}} & \frac{X_{22}-X_{12}}{l_{012}} & \frac{X_{23}-X_{13}}{l_{012}} & 0 & 0 & 0 \end{bmatrix},$$

where l_0 is the distance between the two vertices of the undeformed element specified by the subscript. Moreover,

$$K_g = A \cdot t \cdot G^T \cdot M \cdot G = \begin{bmatrix} B_{12} + B_{31} & -B_{12} & -B_{31} \\ -B_{12} & B_{12} + B_{23} & -B_{23} \\ -B_{31} & -B_{23} & B_{23} + B_{31} \end{bmatrix}, \qquad B_{i,j} = \frac{P_{ij}}{I_{ij}} \cdot [I_3 - D_{ij} \cdot D_{ij}^T],$$

$$D_{ij} = \frac{1}{l_{0ij}}[(X_{j1} - X_{i1}) \cdot (X_{j2} - X_{i2}) \cdot (X_{j3} - X_{i3})]^T,$$

where P_{ij} is the stress on the edge between nodes i and j, I_3 is the 3x3 eye matrix, and D_{ij} is the rotation matrix for the edge between nodes i and j.

Now that the theoretical basis has been laid out, it is necessary to define the implementation of this theory in a FEM solver. That will be done extensively in chapter 4, where the code will be presented together with the running techniques, convergence criteria and solver setup.

2.3. Fluid-Structure Interaction

Fluid-Structure Interaction is a vast field that deals with many subjects; its main goal is to describe situations in which there are large modifications of the fluid domain due to structural motion. The complexity not only lays in the fluid and structure domains themselves, which can present large non-linearities and therefore represent a big challenge to be solved correctly, but also in the coupling of the solvers and the transmission of information across the two domains. One of the difficulties of FSI lays in the fact that being the phenomena very different in the two domains the parameters to solve both problems are largely different: for example the time step, the mesh size and the order of convergence can vary a lot between the two domains. For this reason a method should be developed, able to take into account the differences without penalizing any of the domains in order to have an accurate and fast as possible solution. The following sections will present how these problems have been tackled and overcome. The main literature sources for this chapter are [7], [18] and [19].

The two main approaches when dealing with a fluid-structure interaction problem are the monolithic and the partitioned approach. The former one consists in developing a single solver to solve both domains, and this always results in complex and hard to maintain codes that are build ad-hoc for a single application (so not reusable). Moreover, being the problem approached in its entirety, the monolithic approach is much more memory consuming and it requires compromises that slow down the solution process. For example, consider that the fluid domain generally needs a finer mesh than the structure one: in this case the whole structural problem would be over resolved, slowing down the process with negligible gains in accuracy. For the application to spinnaker, for example in [37], [25] and [6], the meshes used for the two domains had different number of cells, namely the fluid mesh was finer than the structure one. However in the literature there are some rare cases of a single mesh for both domains, for example the one in the work of Renzsch et al. [16]. The choice of using the same mesh for the two domains was motivated by the fact that being the sail cloth a very flexible and deformable material it required a sufficiently fine mesh in order to capture its behavior. This means that the difference in number of cells between the fluid and structure domain was not so big as it would have been for an aeronautical application, where the material used would have had a much higher stiffness and therefore would have required a coarser mesh.

The partitioned approach on the other hand consists in using two separate solvers for the fluid and structure domain, so that preexisting codes can be used and adapted for various applications. The two solvers work independently but are coupled consistently so that the forces and displacements are transmitted correctly across the boundaries. The gain in accuracy obtained with the fact that the solvers are working individually is counterbalanced by the errors that can occur in the transfer of information, known as the partitioning errors, for example in the reconciling the discordance in discretizations for the two solvers. Another drawback lays in the fact that the solvers are implicitly dependent on one another and therefore they can require sub-iterations for each time step in order to obtain convergence, resulting in a slow process. Anyways the partitioned approach has been the only one used so far in sailing applications ([37, 25, 6, 14, 16]), mainly because it requires less expertise and has more flexibility.

2.3.1. Transmission of Information across the Boundary

One of the issues that make fluid structure interaction a delicate matter is the possibility of having non-matching grids at the interface between the domains: it has been said before that usually the fluid domain requires a finer grid than the structure one, and therefore a method for passing the information through the boundary has to be identified. A typical non-matching mesh configuration is shown in figure 2.5.

There are two ways of approaching the problem: the consistent approach, for which a constant displacement or pressure is recovered across the boundary, or the conservative approach, for which the change in work across the boundary is conserved. In either approach, the conditions to be respected are:

$$\mathbf{u}_f = \mathbf{u}_s \qquad and \qquad p_s\mathbf{n}_s = p_f\mathbf{n}_f, \tag{2.58}$$

where the subscripts f and s are to indicate the fluid and structure domain, respectively, $\mathbf{u}$ is the displacement vector, p is the pressure at the surface and $\mathbf{n}$ is the vector normal to the surface. When discretizing equation 2.58 one obtains:

$$\mathbf{U}_f = H_{sf}\mathbf{U}_s \qquad and \qquad \mathbf{P}_s = H_{fs}\mathbf{P}_f, \tag{2.59}$$

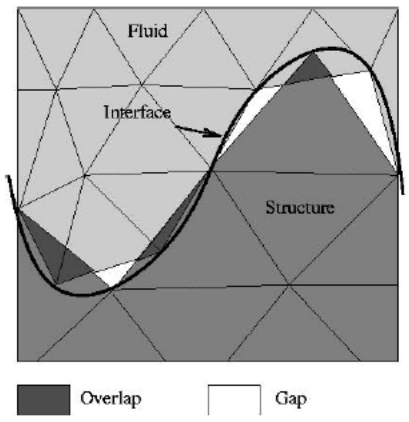


Figure 2.5: Example of nonmatching grids, from [7]

where H_{sf} and H_{fs} are transformation matrices and will be defined later. **U** and **P** are defined by the approximations:

$$\mathbf{u}(x) = \sum_{i=1}^{n_u} N^i(x)\mathbf{U}_i, \qquad p(x)\mathbf{n}(x) = \sum_{j=1}^{n_p} D^j(x)\mathbf{P}_j, \tag{2.60}$$

where $n_{u,p}$ is the number of unknowns at the interface for displacement and pressure, respectively, $N(x)$ a function depending on the spatial discretization used for displacement and $D(x)$ a function depending on the spatial discretization used for pressure. For the consistent approach, the condition that a constant quantity remains unchanged after the transformation translates in the rowsum of H being equal to 1. For the conservative approach on the other hand, the condition is the following:

$$\delta W_f = \delta W_s, \tag{2.61}$$

where $\delta W = \mathbf{F}^T\mathbf{U}$ is the change in work and $\mathbf{F} = M^T\mathbf{P}$. The matrices M_f and M_s are defined as follows:

$$M_f^{ij} = \int_{\Gamma_f} D_f^i N_f^j ds, \qquad M_s^{ij} = \int_{\Gamma_s} D_s^i N_f^j ds. \tag{2.62}$$

With the definition of change in work, equation 2.61 becomes

$$\mathbf{F}_s^T\mathbf{U}_s = \mathbf{F}_f^T\mathbf{U}_f. \tag{2.63}$$

Inserting equation 2.59 in equation 2.63 the following is obtained:

$$\mathbf{F}_s^T\mathbf{U}_s = \mathbf{F}_f^T H_{sf}\mathbf{U}_s, \tag{2.64}$$

which simplifies to

$$\mathbf{F}_s = H_{sf}^T\mathbf{F}_f. \tag{2.65}$$

Finally, inserting the definition of **F** in equation 2.65 the pressure condition is obtained:

$$\mathbf{P}_s = [M_f H_{sf} M_s^{-1}]^T\mathbf{P}_f. \tag{2.66}$$

Note that $[M_f H_{sf} M_s^{-1}]^T = H_{fs}$, and, in this case, the rowsum of the transformation matrix is not necessarily equal to 1.

Consequently, the transformation matrices have to be built: various methods to build them are available, but here a short overview of the ones used in this project will be given.

2.3.1.1. Nearest Neighbor Interpolation

This is the simplest and less accurate way of interpolating quantities across the boundary [30]. Given a point x_A in mesh A, a search algorithm finds the closest point to x_A, x_B, in mesh B and assigns the value of x_A to x_B. The transformation matrix H_{AB} is then a Boolean matrix, with only one non-null (and equal to 1) element per row, thus respecting the consistent approach. However it can be shown that if the conservative approach is used, the method is not consistent for pressure. According to de Boer et al. [8], the consistent method should be preferred over the conservative one. Especially for this application the consistency of the pressure is fundamental. Consider the matter at hand: the displacement in the coarser structure mesh has to be transmitted to the finer fluid mesh. Using this method it could be the case that, for example, three fluid cells are assigned the same displacement value coming from one structure cell. The neighboring fluid cells would be assigned with the value of the neighboring structure cell and so on. In this way a discontinuity in the transmitted displacement would be present, and being the material very flexible and not resistant to compression some unwanted wrinkling could arise, as a result of the low accuracy of the interpolation method.
For this reason, the nearest neighbor method was used in this project only when interpolating the pressure resulting from the CFD simulation to the structure mesh: in that case the fluid mesh was much finer than the structure one and there was no risk of incurring in the situation presented above.

2.3.1.2. Radial Basis Function Interpolation

This coupling method is based on the use of spline functions. More information about this method can be found in [1, 28, 27]. The quantity to be transported across the boundary $\mathbf{w}_i$ from A to B is approximated by a sum of basis functions both at the interface of mesh A and mesh B:

$$\mathbf{w}_i(x) = \sum_{j=1}^{n_A} \gamma_j \phi(||x - x_{A_j}||) + q(x) \qquad i = A, B, \qquad \mathbf{w} = \{\mathbf{u}, p\mathbf{n}\}, \tag{2.67}$$

where x_{A_j} are the points on mesh A where the values are known, q an unknown polynomial, γ_j some unknown coefficients and ϕ a given radial basis function. In order to determine γ_j and q the interpolation condition is used:

$$\mathbf{w}_A(x_{A_j}) = \mathbf{W}_{A_j}, \tag{2.68}$$

wherein $\mathbf{W}_A$ is a vector containing the discrete values at the locations x_A. Another requirement is added:

$$\sum_{j=1}^{n_A} \gamma_j s(x_{A_j}) = 0, \tag{2.69}$$

for all polynomials s with a degree less than or equal to the one of q. The degree of q depends on the choice of ϕ. If the degree of ϕ is less than or equal to 2 and ϕ is positive definite, then q can be a linear polynomial. For the known quantity at the interface of A the two interpolation conditions 2.68,2.69 can be written in matrix form as:

$$\begin{bmatrix} \mathbf{W}_A \\ 0 \end{bmatrix} = \begin{bmatrix} \Phi_{AA} & Q_A \\ Q_A^T & 0 \end{bmatrix} \begin{bmatrix} \gamma \\ \beta \end{bmatrix},$$

with γ containing the coefficients γ_j, β the coefficients of the linear polynomial q, Φ_{AA} an $n_A \times n_A$ matrix that contains the evaluation of the basis function $\phi_{A_i A_j} = \phi(||x_{A_i} - x_{A_j}||)$ and Q_A an $n_A \times 4$ vector with row j given by $[1 \ x_{A_j} \ y_{A_j} \ z_{A_j}]$. To obtain the unknown quantities at mesh B, equation 2.67 in the nodes of the interface of B is written in matrix form as

$$\mathbf{W}_B = \begin{bmatrix} \Phi_{BA} & Q_B \end{bmatrix} \begin{bmatrix} \gamma \\ \beta \end{bmatrix}.$$

Combining the two matricial expression one obtains the final expression for the unknowns:

$$\mathbf{W}_B = \begin{bmatrix} \Phi_{BA} & Q_B \end{bmatrix} \begin{bmatrix} \Phi_{AA} & Q_A \\ Q_A^T & 0 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{W}_A \\ \mathbf{0} \end{bmatrix}.$$

The matrix product between the two matrices that multiply the vector $[\mathbf{W}_A\ 0]^T$ can be called $\tilde{H}$, and the transformation matrix H_{AB} is constituted by the first n_B rows and n_A columns of matrix $\tilde{H}$.

Many radial basis functions are available for this method. The RBF's can be divided in two groups, namely functions with compact support or functions with global support. An example of compact support function, introduced by Beckert and Wendland ([1]), is:

$$\phi(||x||) = \left(1 - \frac{||x||}{r}\right)_+^4 \left(4\frac{||x||}{r} + 1\right), \tag{2.70}$$

where the subscript + means that only positive values will be taken into account. The radius r defines the support of the RBF. In case this value is chosen to be big the support is large and the approximation is more correct, but more costly to resolve as a full matrix system has to be solved. On the other hand, if r is chosen to be small the system is solved more rapidly but the interpolation is less accurate. A lower limit for the choice of r is then the maximum distance of all centres with their nearest neighbours in both meshes.
This function was chosen and implemented for this project, and the RBF method was used to interpolate the displacements from the structure to the fluid mesh, since it seemed the one that yielded the best results in terms of transmitting the displacements from a coarser to a finer mesh without losing accuracy.

3 CFD Simulations

In this chapter all the work performed with CFD softwares will be presented. Given the nature of the investigated problem, much attention had to be paid to preprocessing and meshing, as they are crucial for the correct resolution of the problem. The main difficulty of this problem is due to the fact that the spinnaker sail works mainly as a drag device, as presented in chapter 1, therefore making it a more delicate matter than if it worked as a lift device for example. A major difficulty lays in the fact that the sail has a really low thickness compared to its extension, and that can create problems in the meshing step. Below, for each of these problems, a strategy to overcome them is presented.

3.1. Presentation of the Softwares

The CFD simulations have been performed with two softwares, FINE™/Open and OpenFOAM 4.1, in order to evaluate their performance for this type of CFD problem.

OpenFOAM (Open Field Manipulation and Operation) is an open source CFD software package containing a wide range of features to solve complex fluid flows involving turbulence, chemical reactions and heat transfer. For turbulence in particular, solution methods for RANS, LES and DES are implemented. It contains numerous applications and libraries, and it is nowadays extensively used in the scientific community. For a detailed description, one should refer to the OpenFOAM web-site[1] or the OpenFOAM-extend project website[2]. Due to its high level syntax, solving equations in a fluid domain is made particularly easy with OpenFOAM. Different solving strategies are available, depending on the type of flow, the main ones being:

- *simpleFoam* (Semi-Implicit-Method-Of-Pressure-Linked-Equations), a steady-state solver for incompressible, turbulent flow,
- *pisoFoam* (Pressure-Implicit-Split-Operator), a transient solver for incompressible flow, where a laminar, RANS or LES approach can be chosen. The calculations are bound by a maxiumum Courant number < 1,
- *icoFoam*, a transient solver for incompressible, laminar flow of Newtonian fluids,
- *pimpleFoam*, a large time-step transient solver for incompressible flow using the PIMPLE (merged PISO-SIMPLE) algorithm.

OpenFOAM uses a Semi-projection scheme of the Semi Implicit Method for Pressure Linked Equations (SIMPLE) family. The basic idea of this approach is to split the unknowns describing the pressure and the velocity, and to find opportune corrections for iteratively reaching the final solution.
OpenFOAM has its own meshing and post-processing tools: *snappyHexMesh* and ParaView, respectively. The first one turned out to be not suitable for this application, given the complexity of the geometry, and therefore it will not be described here. On the other hand, ParaView is a data analysis and visualization application, it is independent of OpenFOAM but they are often used together. This tool also allows the manipulation of the

[1] www.openfoam.com
[2] http://www.extend-project.de/

data and has the possibility to export the manipulated data for post-processing or future usages.

On the other hand, FINE™/Open is a commercial CFD tool developed by the software company NUMECA International. FINE™/Open is a powerful CFD Flow Integrated Environment dedicated to complex internal and external flows. It has the capability of dealing with all types of flows and speeds: incompressible, condensable and fully compressible, and low speeds to hypersonic regime are all allowed. It combines completely unstructured hexahedral grids with an efficient preconditioned compressible solver with fast agglomerated multigrid acceleration and adaptation techniques. FINE/Open allows users to freely develop and exchange physical models in CFD: many types of flow conditions can be simulated, for example combustion, heat transfer, radiation, cavitation, multiphase flow and many more physical processes. Moreover, the solver uses one single unsteady RANS code for all types of fluids (incompressible, compressible etc.) and also presents the possibility of speeding up convergence through the tool CPU-Booster. The solver is based on the finite volume method to build the spatial discretisation of the transport equations.

This solver is coupled with two other important tools: HEXPRESS™and CFView™, the first being the meshing tool and the second being the post-processing tool. HEXPRESS™is a powerful software able to generate body-fitted full hexahedral unstructured meshes on complex arbitrary geometries. It allows the insertion of special boundary layer cells and the adaptive refinement of the mesh. It has the advantage of being very user friendly both in the meshing process and in the analysis of the mesh quality.
CFView on the other hand allows to display the results, manipulate and extract them following the user's requirements. It also has the capabilty of recording macros so that a certain routine can be applied automatically to the results. Also this tool has a high user friendliness and serves a fast post-processing.

3.2. 2D Case

In this section the steps taken to run the 2D CFD simulations are presented. The 3D geometry of the yacht, shown in figure 3.1, is the starting point from which the 2D sections of the sails will be extracted. The objective of this analysis is to compare the pressure coefficient distributions along five horizontal sections of the spinnaker sail with the experimental validation data provided in [9]. In fact the 3D CAD model represents the model scale (2.3 m high) model used in the wind tunnels by Viola et al. to measure the pressure on both suction and pressure sides of the sail [9].

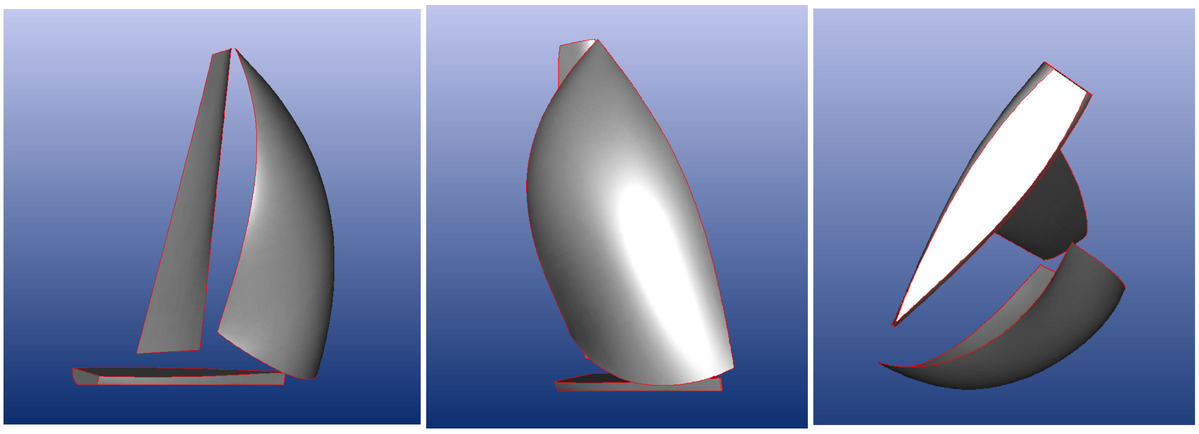


Figure 3.1: X (left), Y (center), Z (right) views of the CAD geometry

The pressure measurements in [9] have been collected at five horizontal sections, expressed as a fraction of the sail mitre, namely the line equidistant from the leading and trailing edges of the sail. The analyzed sections are the ones presented in table 3.1.

In order to obtain the horizontal sections, the CAD software CATIA [11] has been used. Being the sails in the original geometry zero thickness, a 1 mm thickness has been added to the sail. This ensures that the geometry describes a volume and not only a surface. The thicker sail section has been consequently

Section	Z coordinate [m]
1/8	0.35
1/4	0.64
1/2	1.22
3/4	1.8
7/8	2.09

Table 3.1: Z coordinates of the spinnaker horizontal sections

extruded in the z direction for the same reason; being the simulation two-dimensional all the quantities will be constant along the z direction and the length of the extrusion is not relevant to the results.
The final requirement to obtain a geometry accepted by HEXPRESS™is to have the geometry as a Parasolid solid body, and that option is not provided by CATIA so this step has been perfomed through ANSYS's GAMBIT mesher [13]. Once all these steps are performed, the analyzed geometry is the one shown in figure 3.2.

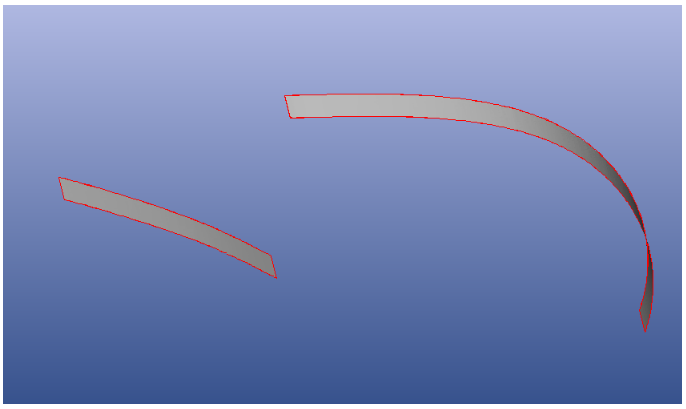


Figure 3.2: 2D section of mainsail and spinnaker at 1/8th of the mitre

3.2.1. Meshing

The meshing, done using HEXPRESS™starts with the definition of the computational domain. The domain is 18.7 m long and 6.2 m wide, as the wind tunnel used in [9]. The extension in the z direction has been assigned to be of 1 m, but the mesh generation is done in two dimensions and therefore there will always be only 1 cell in the z direction. HEXPRESS™recognizes that and does not consider the z direction when evaluating the mesh quality, for example when computing expansion or aspect ratios.

The meshing starts with the definition of the initial mesh, dividing the whole domain in cells, without considering the geometry in it yet. The number of cells is x and y directions is calculated so that the initial cell size is of 0.2 m. Once this step is taken, the "Adapt to geometry" step starts, where the zones of refinement are defined, taking into account the geometry and the physics of the problem. The main idea here is to refine close to the sail surfaces, in order to capture the sail curvature, which is quite high, and moreover to have a refined area behind the sails that will help to have a better glimpse of the wake development.

The surface refinement is done in the following way: consider that each sail has four surfaces, the two that describe the sail curvature and the two that describe the sail thickness. Being the thickness really low in this computation (1 mm), the cells on that side will have to be smaller than 1 mm in order to correctly capture the geometry. For this reason, a target cell size of 0.0003 meters has been imposed, in order to have at least three cells on the thin sides of the sail. On the other hand, the long side of the sail can use a coarser cells and still be captured. Figure 3.3 shows how the mesh is much more refined getting closer to the sail trailing edge, while getting away from it the cells start inflating until they reach the size of 0.002 m.

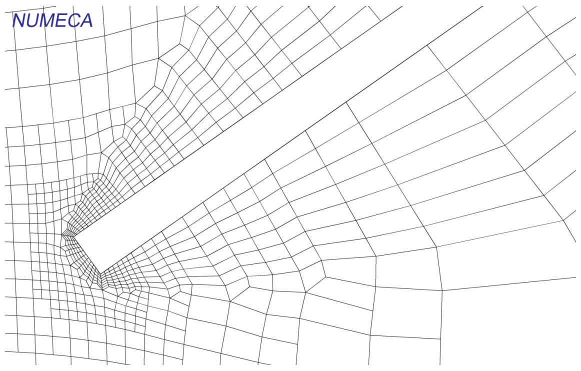


Figure 3.3: Detail of the 2D mesh, with particular attention to the sail surface that describes its trailing edge

Moreover, the mesh has been refined downstream of the sails to capture the wake using a box refinement.

After the adaptation step, the snapping to geometry step is taken. It is an automatic process in which the refined cells that intersect the geometry are split to accommodate the solid body and the ones on the inside of the body are removed. In order for this step to be successful the mesh has to be fine enough to describe the geometry, otherwise the snapping will fail. Indeed the value of 0.002 m for the cell size on the pressure and suction side of the sail has been determined with a trial and error process based on the output of the snapping process.

Once the snapping is completed the optimization is run, with the goal of regularizing the mesh and avoiding very large or very small angles in the cells, trying to orthogonalize as much as possible the cells. The last step is then the insertion of viscous layer cells, fundamental for the simulation of the boundary layer. The number of layers is determined by the software automatically given the reference length, Reynolds number and y+ value.

The final mesh has 96593 cells and a representation of it is shown in figure 3.4. The mesh quality is satisfying: all the requirements are met, except for the expansion ratio that is slightly above what is recommended (7.554 instead of 5). The results of the mesh check are shown in table 3.2.
It is worth mentioning that the flow is coming from the right in all the figures reported in this report. As it can be seen in figure 3.4 the positive x axis is pointing outside of the domain, therefore the freestream velocity will always be negative. The setting of 55 degrees of angle of attack is obtained by rotating the model, so that the freestream velocity can always be parallel to the x-axis.

Criterion	**Value**	**Acceptable?**
Number of negative cells	0	Yes
Number of concave cells	0	Yes
Number of twisted cells	0	Yes
Number of relaxed cells	0	Yes
Minimum skewness [deg]	36.132	Yes
Averaged skewness [deg]	85.949	Yes
Maximum expansion ratio	7.554	No
Maximum aspect ratio	5.231	Yes

Table 3.2: Results of the mesh check

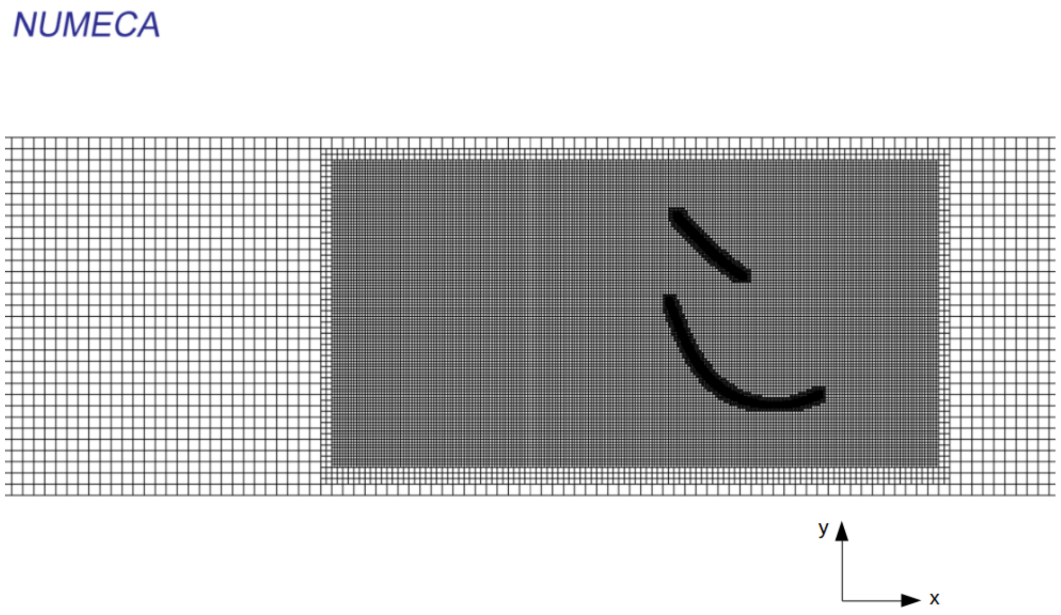


Figure 3.4: The final 2D mesh

3.2.2. Unsteady case - OpenFOAM

3.2.2.1. Running

The first 2D simulation is done in OpenFOAM using an unsteady scheme. The chosen solver is the PIMPLE algorithm. The advantage of this solver is that it allows to have a Courant number larger than one, differently from PISO that only allows Courant number to be as high as one. It also has the possibilty of setting the maximum Courant number and consequently adjust the time step during the simulation, giving the possibility of having higher time steps than with PISO. The Courant number is an indicator of the stability of the simulation, and it depends on the reference velocity U, the time step Δt and the smallest cell size in the mesh Δx, through the relation:

$$Co = \frac{U\Delta t}{\Delta x}. \tag{3.1}$$

A Co< 1 ensures that the information from one cell can reach the next neighboring cell within one time step. If it is larger, the information would skip some neighboring cells, causing numerical instabilities.

The solving algorithm works as follow, as explained in [20]: the steady state solution is searched within one time step with under-relaxation (using SIMPLE algorithm, explained in subsection 3.2.3). Some outer correction loops are used for pressure and momentum, that guarantee that the explicit parts of the equations are converged. Once the steady solution is found, namely some tolerance condition is met, the time step can advance, leaving the outer correction loop.

The simulation for this specific case has been run for 10 seconds of physical time, in order to make sure that the solution could converge to a steady state, after the initial settling time. The turbulence model has been chosen to be the k-ω SST for all the simulations in this work, for the reasons explained in the previous chapter 2.1.1. The physical values of the basic flow properties for all the simulation in this are summarized in table 3.3.

The boundary conditions are supposed to represent the wind tunnel and therefore the inlet will have a prescribed velocity and zero gradient pressure, and the outlet will have a zero gradient velocity and the pressure set to 0. Regarding the top and bottom parts of the domain, in this simulation they are set as "empty" for all the quantities as it is a two dimensional simulation. The "empty" condition is an OpenFOAM boundary condition used for 2D simulations to indicate a patch representing a direction that is not solved. A slip condition is applied to the wall for both pressure and velocity. On the solid walls, velocity is set to zero with a no

Name	**Symbol**	**Quantity**
Velocity in x direction	U	3.5 m/s
Reynolds number	Re	230000
Density	ρ	1.225 kg/m^3
Kinematic viscosity	ν	1.54e-05 m$^2/s$
Dynamic viscosity	μ	1.88e-05 N s/m^2
Reference length	L	1 m
Turbulent length scale	l	0.01 m [9]
Turbulence intensity	I	5% [9]

Table 3.3: Flow properties for the simulations

slip condition, while the zero gradient condition is applied to the pressure.

Regarding the turbulent quantities, their values have been calculated in the following way:

$$\nu_t = \sqrt{\frac{3}{3}} U I l = 0.002143 \frac{m^2}{s}, \quad k = \frac{3}{2}(UI)^2 = 0.0459375 \frac{m^2}{s^2},$$

$$\omega = \frac{\sqrt{k}}{l} = 21.433035 \frac{1}{s}, \quad \varepsilon = C_\mu \frac{k^{\frac{3}{2}}}{l} = 0.088612 \frac{m^2}{s^3},$$

where ν_t is the turbulent viscosity, k the turbulent kinetic energy, ε the dissipation of turbulent kinetic energy, ω the specific dissipation rate, C_μ is a constant of value 0.09, I the turbulence intensity and l the turbulent length scale. The boundary conditions for the turbulence quantities are: k is fixed to the value specified above at the inlet, a zerogradient condition is imposed at the outlet and a slip condition is applied to the side walls. At the solid boundary, its value is also fixed as at the inlet. The turbulent viscosity is set to zero everywhere except at the inlet, where the value mentioned above is assigned. This will allow the turbulence to gradually develop in the domain. The value ω is set to be 21.433 everywhere, except for the side walls where the slip condition is assigned. Using the k-ω SST turbulence model there is no need to specify boundary conditions on ε.

Regarding the initial conditions, the internal field is initialized with velocity of 3.5 m/s in the x direction, a the turbulent kinetic energy of 0.0459375 $\frac{m^2}{s^2}$ and a specific dissipation rate of 21.433 $\frac{1}{s}$. The turbulent viscosity and pressure are set to 0 for the whole internal domain.

The control variables to take into consideration in this case are the time step value and the number of time steps. The number of time steps will be computed based on the 10 seconds of simulation and the time step value. The condition on the time step to respect to ensure stability is:

$$\Delta t \leq \frac{\Delta x}{U}, \tag{3.2}$$

which is equivalent to a Courant number smaller than 1, see equation 3.1. Being the smallest cell size Δx = 0.0001 m and the velocity U = 3.5 m/s, the resulting maximum time step value allowed is 0.00002 s. After a couple of trials the value of the time step that allowed the simulation to run was found to be 0.000001 s. That implies that the number of timesteps would be 10 million. However the possibility of having an adjustable time step with a maximum courant number of 2 allows the time step to grow bigger during the simulation, making the number of time steps smaller.

3.2.2.2. Post-processing

In this paragraph the results of the unsteady simulation done with OpenFOAM are shown. The post processing has been done through ParaView, which allows to manipulate and visualize the relevant data. The expectation would be that after a certain simulation time the quantities would converge to a steady state condition, allowing to have a converged pressure distribution on the sail. Looking at the results, it seems like this prediction has not been met. What is observed is a periodic behavior: the pressure distribution cannot stabilize but keeps on varying periodically. This is due to the periodic vortex shedding that happens at the leading edge of both spinnaker and mainsail. It can be observed that the vortex, represented by and area of lower pressure (blue in the figures) forms at the leading edge and then moves downstream, leaving space for a new vortex to create and to follow its path. This flow behavior is shown in figure 3.5, where the pressure field is captured at various times in the simulation. As it has been said before, the flow is coming from the right in all figures.

Looking at these figures some observations can be made: in figure 3.5a the simulation has just started and the starting vortex can be seen at the trailing edge of both sails. Proceeding in time, in figure 3.5b it can be observed how the starting vortex is shed downstream in the wake of the sails, while new areas of low pressure are appearing at the leading edges: these will be the vortexes that will form periodically and will be also shed in the wake. It can be observed that the vortex stays attached to the sail for a portion of the chord length, and consequently it is shed and separates from the solid surface (figure 3.5d). However, the formation of the vortex forces the flow to separate right away, as can be observed from the streamlines in figures 3.5i and 3.5j. From these figures it can be observed that the streamlines on the suction side of the sail cannot follow the curvature of the sail but detach and start forming circular patterns in correspondence of the vortex. Observing figure 3.5e, it is clear that at some point of the simulation a bigger area of low pressure starts to form. Advancing in time the vortex is convected downstream (figure 3.5f) and separates 1.5 seconds after forming (figure 3.5g). After, this the small vortices start forming again, the situation depicted in figure 3.5b starts over and the whole cycle repeats itself. This behavior is repeated periodically and there is no settling to a pressure distribution on the sails constant in time. This is not in agreement with the reference data from [9], where periodic vortex shedding is not present. The reason for this discrepancy is that the experiment is done in three dimensions, where the vortex formation and convection in the flow is also a three dimensional process. Trying to reproduce this 3D effect with a 2D simulation will not yield satisfying results, as the physics of the problem can not be reproduced correctly, and a steady solution can not be found. However, a steady simulation has been run with the hope of reaching a steady state solution of the two dimensional problem, to at least be able to compare the results with the experimental data. For this specific case, since for each time step there is a different pressure distribution, the only way to compare the data with the experimental values is to average the pressure distribution over the period, namely from when the vortex is formed to when it detaches from the sail. This will be done in the next paragraph.

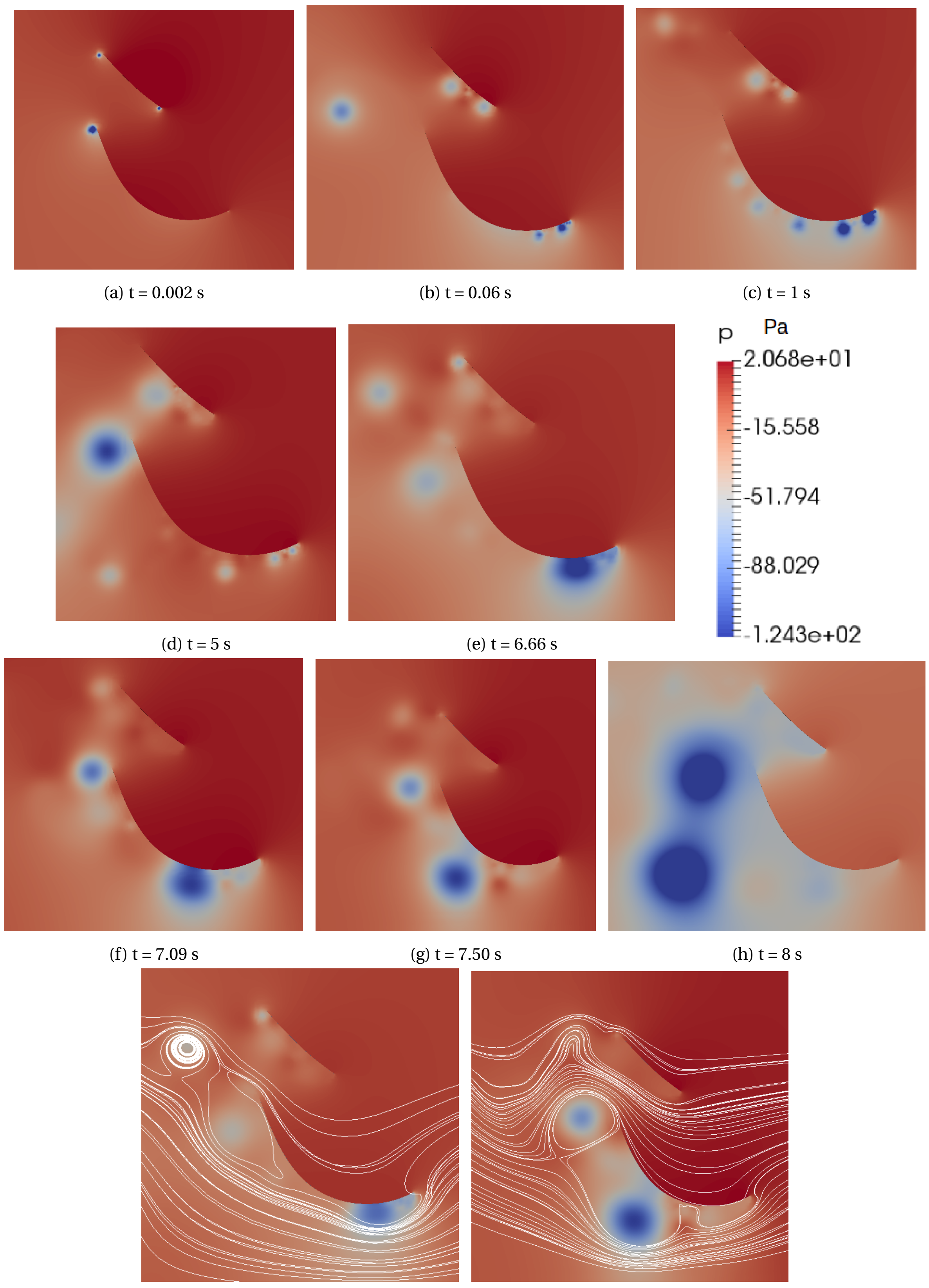
3.2.2.3. Comparison with validation data

As previously explained, in this case it is not possible to directly compare the pressure distributions with the experimental data without averaging the pressures over a period of vortex shedding. This means the time from when the vortex is generated to when it detaches from the surface and is shed in the wake. Observing the flow field, it has been verified that one period of formation and shedding has the duration of 1.5 seconds for the big vortex described in the previous paragraph. For this reason the pressure distribution over the suction side of the spinnaker sail has been saved every 0.02 s from t=6 s to t=7.5 s, and then averaged, plotted as a function of the curve length and compared to experimental data through a Matlab script.

The results of said operation are shown in figure 3.6, where the two distributions are compared in figure 3.6a and only the result of the simulation is shown in figure 3.6b. The plots are done for the lowest section of the spinnaker, at 1/8th of the mitre. The pressure coefficient is computed in the following way:

$$C_p = \frac{p - p_{inf}}{\frac{1}{2}\rho U^2}, \tag{3.3}$$

where p_{inf} is the freestream pressure and has been computed as the average of the pressure over the inlet. The results support the idea expressed in the paragraph before, for which a 2D simulation cannot reproduce three dimensional phenomena like the creation and shedding of a three dimensional vortex. For this reason the results do not resemble each other. Figure 3.6b shows how the result of the simulation does not even resemble the tendency of the validation data. From this figure we can notice a plateau of the pressure starting around x/c = 0.6, which represents the complete detachment of the flow from the sail after that point.



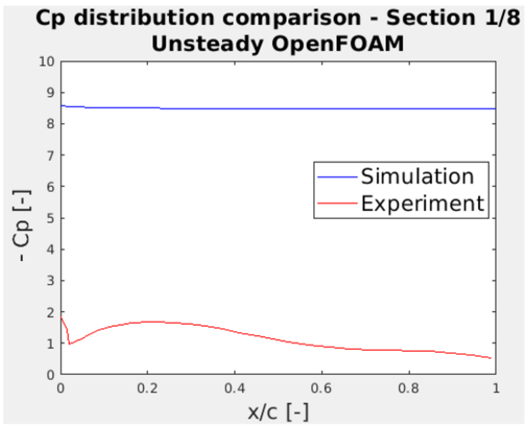
(a) t = 0.002 s (b) t = 0.06 s (c) t = 1 s

(d) t = 5 s (e) t = 6.66 s

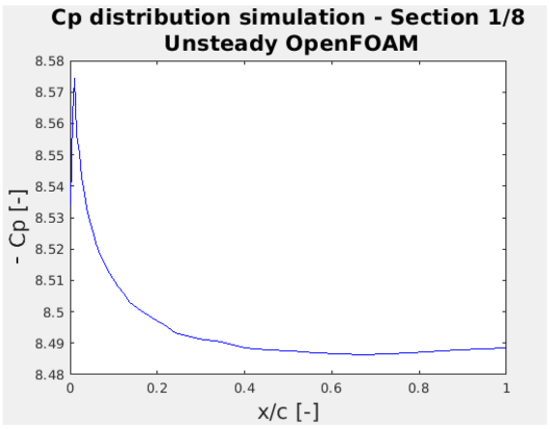
(f) t = 7.09 s (g) t = 7.50 s (h) t = 8 s

(i) t = 6.66 s, with streamlines (j) t = 7.5 s, with streamlines

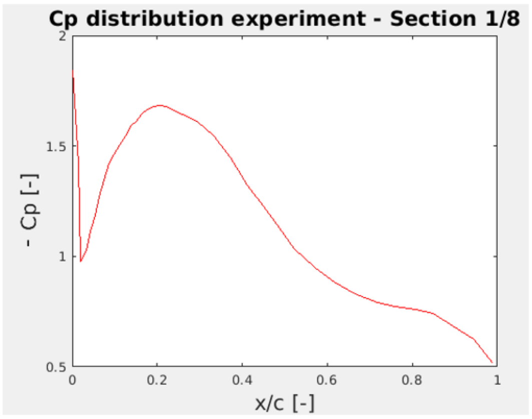
Figure 3.5: Pressure field around sails at different time steps, unsteady 2D simulation OpenFOAM



(a) Simulation vs. Experiment



(b) Simulation only



(c) Experiment only

Figure 3.6: Cp distributions - Section 1/8 - OpenFOAM 2D Unsteady Simulation

Figure 3.6c shows on the other hand how this phenomenon does not happen in reality: after an initial peak the pressure drops but does not flatten: the flow is still attached to the sail.

This results are useful in the sense that now the strategy of the project needs to be changed: the initial idea of having a two dimensional solver cannot be carried on if the CFD results are not satisfying, and the need for 3D simulations is evident. One last try will be made with the steady 2D simulation, but seeing how discouraging the results are so far it is almost certain that the 3D simulation will be needed.

3.2.3. Steady case - OpenFOAM

3.2.3.1. Running

This simulation in OpenFOAM is done using the SIMPLE algorithm. Its aim is to reach a steady state solution so in the equations to solve the time derivatives are set to zero. The algorithm is iterative and it is based on the following scheme:

- Guess a field for the pressure and calculate velocity components u^* and v^* (from previous iteration),
- The Navier Stokes equations are solved using the guessed pressure value. The unknowns are then only u^* and v^*,
- Being the velocity components calculated with the guessed pressure, there will be a residual for the continuity equation. This residual is used to compute a correction for p, and for calculating a new initial pressure guess,
- Repeat process until the pressure correction is smaller than a given tolerance.

Since in the equations that are being solved in this algorithm the time derivatives are not present, the algorithm is not consistent due to the missing term. For this reason the under-relaxation is fundamental for stability, since without the relaxation it would be easy to have divergence due to some quantity divided by 0, as explained in [20].

Regarding the simulation setup, the mesh used is the same as the one in the previous section, as it is presented in section 3.2.1. The initial conditions and boundary conditions are also the same as the previous simulation. The only modified parameter is the time stepping scheme, defined in the file *fvSchemes*: before it was set to Euler while now it is set to steady state. The Euler scheme for the time derivative is a transient, first order implicit method that expresses the time derivative of the velocity u at time $i+1$ as:

$$\frac{\partial u^{i+1}}{\partial t} = \frac{u^{i+1} - u^i}{\Delta t}, \tag{3.4}$$

while for the steady state scheme

$$\frac{\partial u^{i+1}}{\partial t} = 0. \tag{3.5}$$

This is the only difference between this simulation and the one presented previously. The absence of the time derivative term allows the simulation to go much faster, and also it does not require the definition of the time step Δt, as the steady state simulation is not time dependent and the number of time steps actually corresponds to the number of steady state iterations. In this simulation the number of iterations has been set to 10000, in order to have the certainty that the simulation is converged. In the next section the results of this simulation are presented.

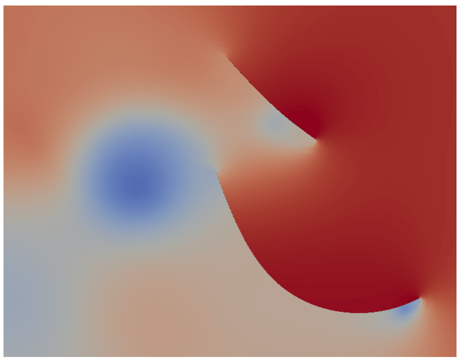
3.2.3.2. Post-processing

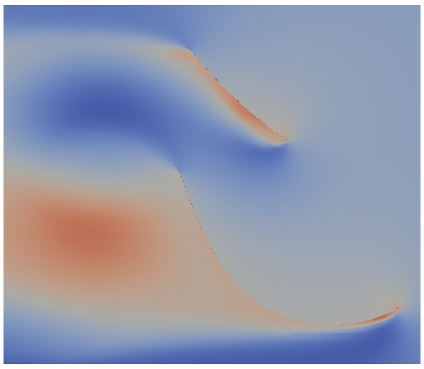
Again for the analysis of the results the software Paraview has been used. Also for the steady state simulation, it was not possible to reach a constant pressure distribution on the suction side of the sail. An oscillating behavior is still visible, because also here the periodic vortex shedding is present. However looking at the representation of the flow field some differences can be pointed out.

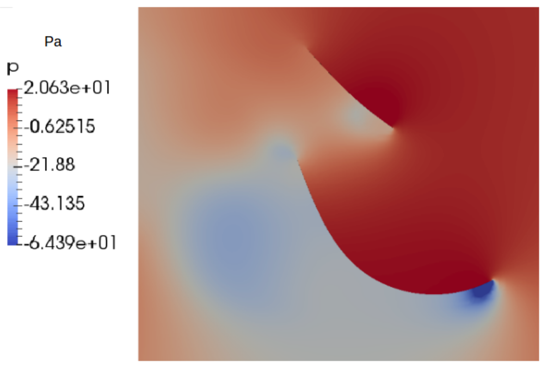
In figure 3.7 the flow field has been captured at the iteration numbers that correspond to one cycle of vortex formation and shedding. More specifically, at iteration 77 the vortex starts to create at the leading edge of the spinnaker, at iteration 83 it reaches its maxiumum intensity, and then it starts diminishing in intensity again until it reaches its minimum, at iteration 88. First of all some evident differences with the flow situation presented in figure 3.5 can be observed: here the flow separates immediately after the leading edge, while in the other simulation it was possible to witness the vortex creation and its movement downstream while it was still attached to the sail. What is happening in this case on the other hand is that the flow separates right away, creating a recirculation zone close to the leading edge (the red color in figure 3.7d represents a velocity opposite to the freestream). The interesting aspect is that this vortex that is created at the leading edge does not travel along the spinnaker chord and then detach like in the previous case, but it stays in the same position, increasing and lowering its intensity within a cycle. On the other hand, vortices are shed in the wake from the trailing edge, as it can be seen in 3.7. The presence of the vortex downstream of the sail, recognizable in the figures by the blue circle in the pressure figures and red circle in the velocity figures, affects the pressure distribution on the sail, making it oscillate. For this reason also in this case it is not possible to find a converged steady pressure distribution, but also here in order to compare with the validation data it will be necessary to average over a period.

3.2.3.3. Comparison with validation data

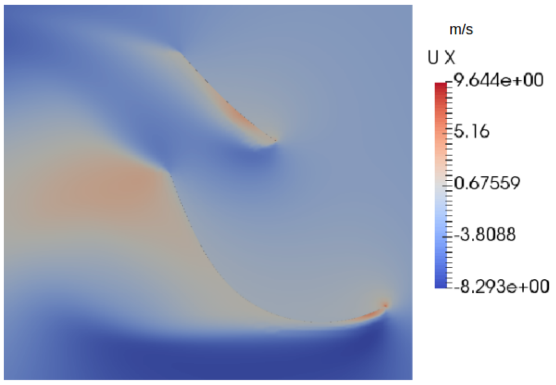
Again here the results of this simulation have been plotted in terms of pressure coefficient in order to compare with the reference data. As it could have been expected, there is no resemblance of the validation data. The main reason is again because the formation of vortices behind a sail is a three dimensional phenomenon and it cannot be correctly captured in a 2D simulation. Another factor that supports the fact that this simulation is not meaningful is that the steady state cannot be reached: also here the values are oscillating periodically, depending on the cycle of vortex formation. Again, in order to compare with the reference data some averaging has been done, over a cycle described in figure 3.7, so over 11 iterations. Looking at the flow field the


(a) Pressure field at iteration 77

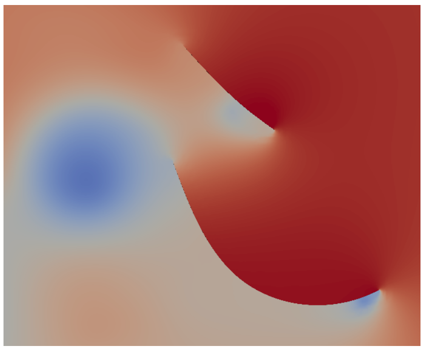

(b) Horizontal velocity field at iteration 77



(c) Pressure field at iteration 83



(d) Horizontal velocity field at iteration 83


(e) Pressure field at iteration 88

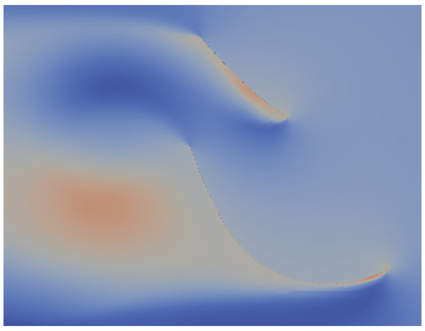
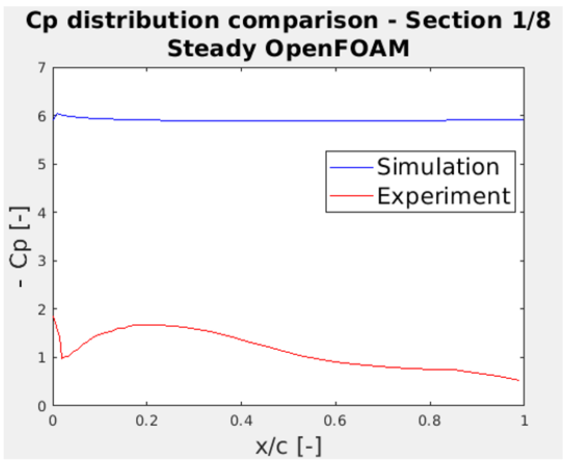
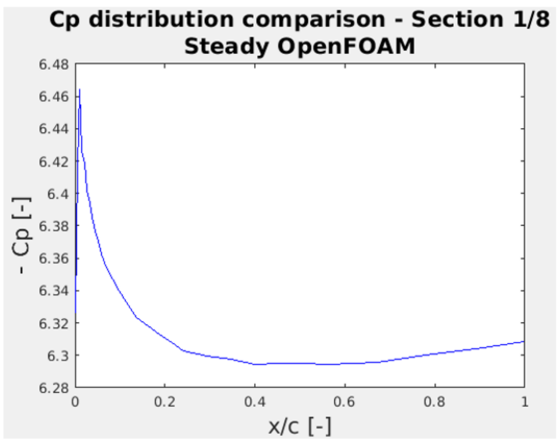

(f) Horizontal velocity field at iteration 88

Figure 3.7: Pressure and horizontal velocity fields around sails at different iterations, steady 2D simulation OpenFOAM

same behavior can be observed periodically every 11 iterations.



(a) Simulation vs. Experiment



(b) Simulation only

Figure 3.8: Cp distributions - Section 1/8 - OpenFOAM 2D Steady Simulation

The result is what is shown in figure 3.8: the results are still quite far off from the reference data, but we can observe a similarity with figure 3.6b: the tendencies are similar, except for the fact that in the steady case separation is reached earlier (around x/c=0.4). This tendency is meant to be interpreted in the following way: as it can be seen from the flow field pictures, the flow is separated as soon as it encounters the sail. The separation creates a recirculation zone, where the pressure is low. That is represented by the peak in the Cp distribution. after that peak, the flow is completely separated (red area behind spinnaker, figure 3.7b). Let us not forget that the plot in figure 3.8b is obtained by averaging over a period: therefore it is a sort of summary of the three figures 3.7b,3.7d and 3.7f. As it is visible in figures 3.7b, 3.7f, there is a vortex of opposite sign coming from the mainsail that also partially affects the pressure distribution on the spinnaker: its effect can be seen between x/c=0.8 and 1, where the Cp rises because the flow is again in the direction of the freestream velocity.

In conclusion, also with this simulation the reference data could not be reproduced. Changing the solver from unsteady to steady did make some changes to the final result, but the outcome is still not satisfying since we get a solution that is not in agreement with the reference data: according to the experiment the flow is attached through the whole chord length while here it separates almost right away, and the pressure difference is much higher than in the experiment. The next step is to try once again this steady simulation with another solver, FINE/OPEN, and assess if that solver is adequate for this test case.

3.2.4. Steady case - FINE/Open

3.2.4.1. Running

For this simulation the solver FINE/Open has been used. Its presentation is done in section 3.1. The simulation setup is done much easier thanks to the user friendly graphical user interface of the software, that allows to set the parameters visually. It also allows to visualize the geometry and mesh together with the project parameters, all in one window.

The mesh used is the same as the one used in the two previous simulations. The initial and boundary conditions are slightly different because the patches representing the bounding box of the domain are all considered as external (meaning that the geometry is in the freestream, and the walls are not present), except for the top and bottom patches that are set to "mirror", for which no boundary or initial condition needs to be set. This is a better representation of the reality, but not of the wind tunnel experiment. Anyways the boundary and initial conditions are the same for all external patches: there the horizontal velocity, pressure, k and ε are prescribed, with the values summarized in subsection 3.2.2 and table 3.2. The initial condition is the same for all patches and the internal domain. Being a steady simulation, it was not necessary to impose a time step value, but only a number of iterations and a convergence criterion. To begin with the number of iterations was set to 1000 and the convergence criterion to -6. The convergence criterion corresponds to the (negative) number of orders of magnitude the norm of the residuals must decrease before stopping the calculation, as explained in [29]. The solution converged in about 1 hour of running on a single processor.

3.2.4.2. Post-processing

The data is post processed through CFView, a tool of NUMECA that allows an easy visualization of the obtained results. The difference between this and the previous simulations is that here a single state is reached and only one flow field is visible: that means the steady state solution has been reached.

Taking a look at the flow field the topology looks similar to the one presented in the steady OpenFOAM simulation: the flow is separated right after the leading edge of the spinnaker and there is a big area of recirculation and negative velocity behind the sail. Looking at the pressure field in figure 3.10 it is possible to recognize the low pressure area in correspondence of the leading edge, that helps to accelerate the flow that is separated (blue area below spinnaker in figure 3.9), but also to create an area of recirculation close to the spinnaker. When looking at the figures with the velocity field one must not forget that the freestream velocity is negative in the x directions with a value of 3.5 m/s, therefore the positive velocity values will correspond to the recirculation areas. It is quite obvious that also in this simulation separation was reached much earlier than in the experiments, similarly to the two previous simulations. The difference in this simulation was that at least a steady state was reached and the solution stabilized to one configuration, but already looking at the flow field it is possible to state that the solution will not resemble the validation data.

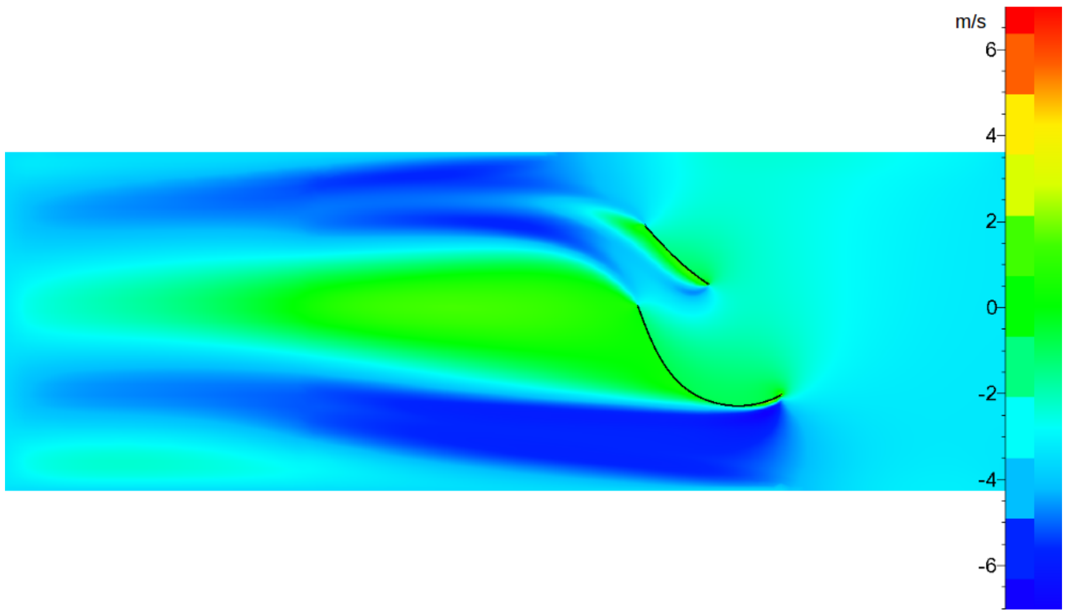


Figure 3.9: Horizontal velocity field - Section 1/8 - FINE/Open 2D Steady Simulation

3.2.4.3. Comparison with validation data

The last step consists in analyzing the solution by comparing it with the experimental data given in [9]. From CFview, it is possible to export the data to `.txt` files in order to be able to manipulate them in Matlab. A script that reads the data, computes the coefficient of pressure following equation 3.3 and plots it against the validation data has been implemented. The result is shown in figure 3.12.

In this figure the prediction about the early separation is evident: the Cp reaches a plateau around x/c=0.3 and stays constant with a value of 1 from there. That represents the separated area visible in figure 3.9 as the green area downstream of the spinnaker. Also the initial suction peak is visible, represented by the blue dot

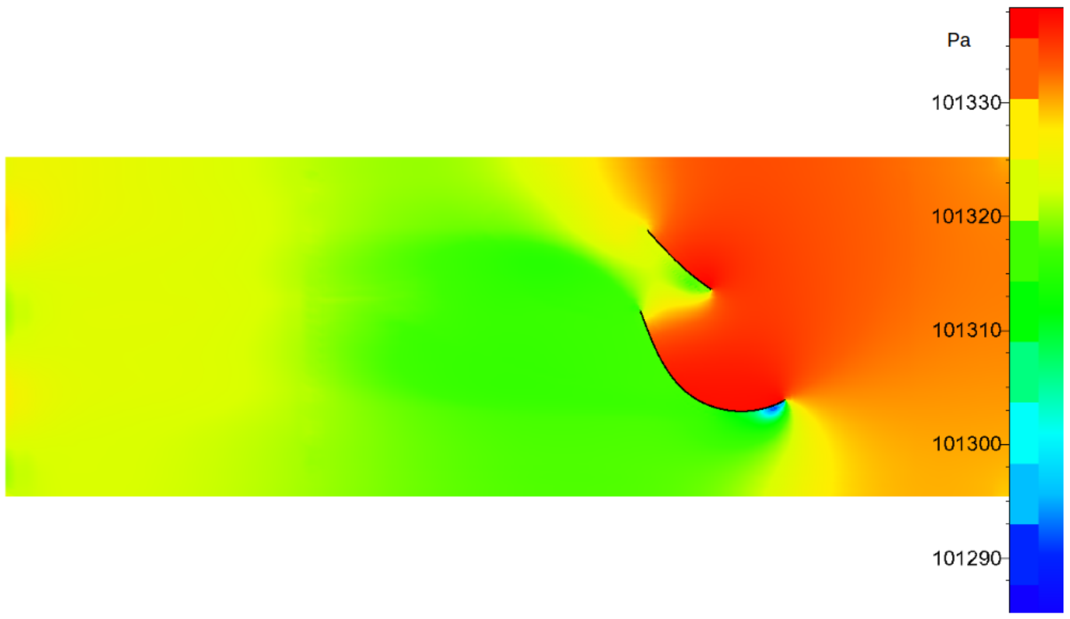


Figure 3.10: Pressure field - Section 1/8 - FINE/Open 2D Steady Simulation

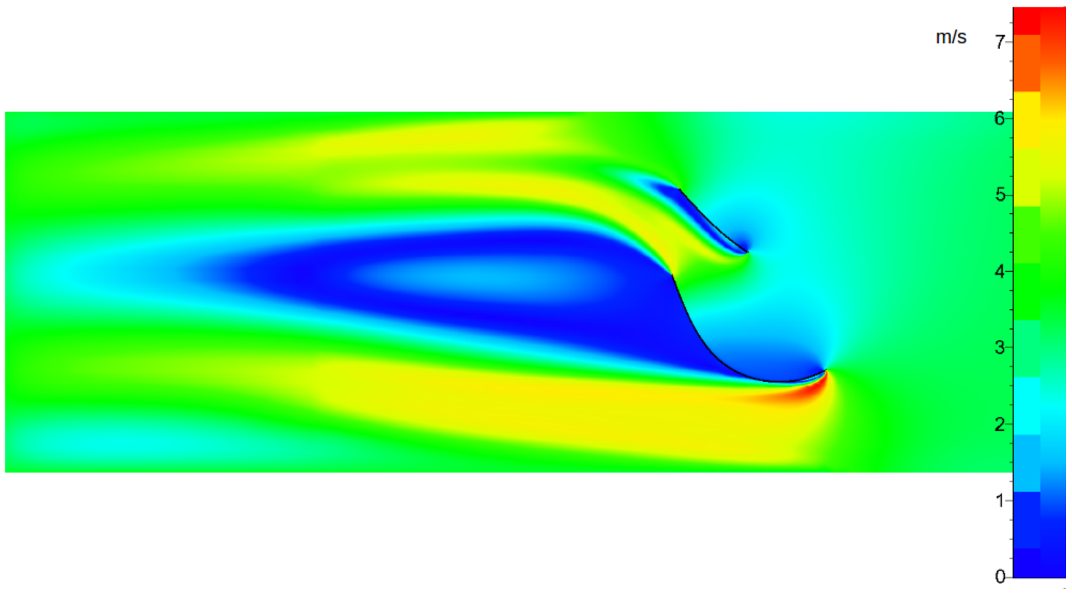


Figure 3.11: Velocity Magnitude field - Section 1/8 - FINE/Open 2D Steady Simulation

in figure 3.10. However it is possible to notice an improvement with respect to the OpenFOAM results: here the range where the Cp lies is more similar to the one of the validation data, with a difference in amplitude of the suction peak, that here is much higher than in reality. Given this result, it is possible to say that the resemblance of the solution is more close to the experiments, but still not similar enough to be satisfied with the results and proceed with the FSI coupling. Again the hope is that with a three dimensional simulation the creation of the leading edge vortex will be better predicted, as well as its point of separation, in a way that allows the FSI simulation to be meaningful.

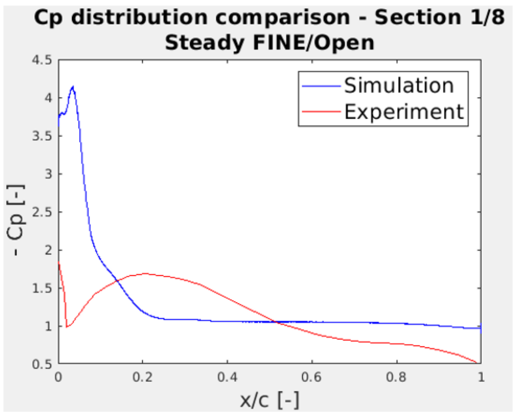


Figure 3.12: Cp distributions - Section 1/8 - FINE/Open 2D Steady Simulation

3.3. 3D Case

Given the discouraging results from the previous simulations, the need for a three dimensional flow analysis became evident. The initial plan of having a 2D FSI solver can not be followed anymore, because only with a 3D simulation it is possible to obtain reasonable CFD results and therefore the whole solver will have to work in 3D. This means much more complexity and longer simulation times, and less possibility of automatizing (much more time than the thesis time would be required for that). For now the aim will be to have a working solver: its high performance and short run times are secondary in the present work.

For the 3D case, the whole geometry presented in figure 3.1 is used. The configuration of the boat for the simulation is at 10° heel (angle of the mast with respect to the vertical line, see figure 3.13b) and at 55° AWA. The apparent wind angle is the angle obtained by subtracting the boat velocity vector to the true wind velocity vector, as shown in figure 3.13a.

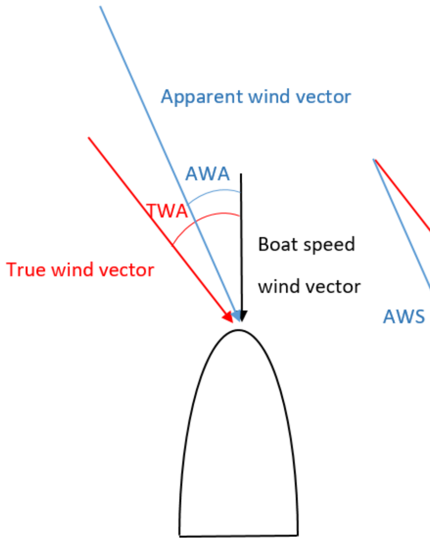
The aim is to represent the experiment presented in [9]: the 2.3 m high model is placed in a wind tunnel (6.2 m wide, 3 m high and 18.7 m long) and a horizontal velocity of 3.5 m/s is imposed at the inlet. The model is placed 5.6 meters downstream of the inlet so that there is a lot of space downstream for the investigation of the wake. The computational domain, created in HEXPRESS, is shown in figure 3.14a. Similarly to the 2D case, also here the inflow velocity is in the negative x direction, as shown in figure 3.14a. That means that in all the plots the freestream velocity will be coming from the right side and the negative velocity is flowing from right to left. As previously explained, the freestream flow is always parallel to the x axis and the model is rotated to simulate the angle of attack of 55 degrees. The steps taken to create the mesh are presented below.

3.3.1. Meshing

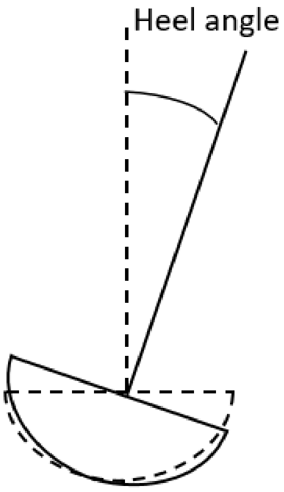
This part of the project turned out to be more complicated than expected. While in the two dimensional mesh the final cell count was of more or less 100000 cells, here it is reasonable to expect many more cells as it will be necessary due to the additional dimension added. The main problem that arises when meshing in 3D is the very low thickness of the sail: that would entail an excessive cell count to be resolved. Therefore, the first approach employed is to impose the cell sizes as high as possible, in order to minimize the final cell count. This cannot be done so easily though, since the snapping step is possible only if the cells are fine enough to describe the geometry. While trying with this approach the minimum number of cells obtained was 14 million. That is already a quite big number of cells, which would slow down the whole FSI cycle by a lot.

After some investigations, another solution was found. The sails can be represented and meshed as zero thickness surfaces, eliminating the problem of meshing the thin side of the sail and saving many cells for the final count. In this case two meshes have been created, a coarser one and a refined one obtained by doubling the amount of cells, in order to be able to perform a mesh convergence study. The mesh setup is done in the following way (for the coarser mesh):

1. The initial cell size is set to 0.2 m,

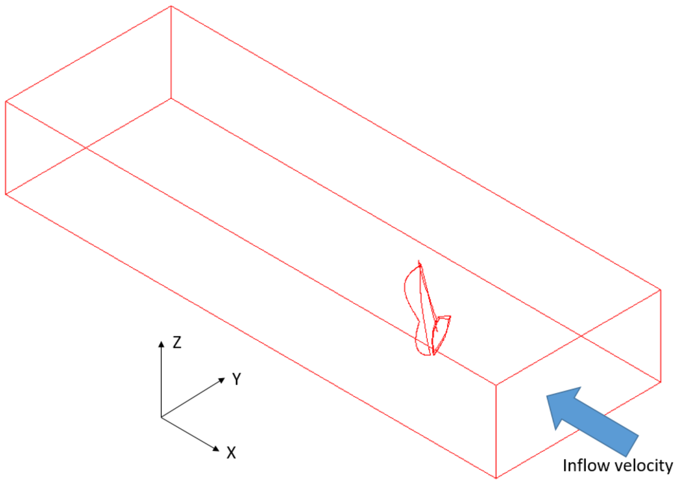


(a) Apparent wind angle (AWA)

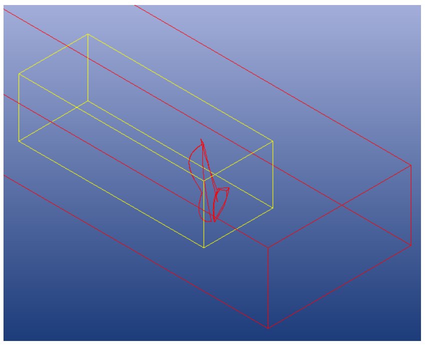


(b) Heel angle

Figure 3.13: Definition of apparent wind angle and heel angle



(a) 3D Computational domain



(b) Refinement box

Figure 3.14: 3D domain

2. Adapt to geometry:

    (a) Curve refinement: target cell size set to 0.002 m for the curves describing the sails.

    (b) Surface refinement: target cell size set to 0.02 m for all the solid surfaces (sail and hull).

    (c) Box refinement: box enclosing the yacht and extending downstream (see figure 3.14b) with target cell size 0.05.

3. Snap to geometry: in this step the cells intersecting the geometry are split and the ones insider the geometry are removed from the mesh.

4. Optimization: the quality of the mesh is automatically improved, focusing especially on the cells orthogonality, that has to be as close as possible to 90°.

5. Viscous layers: a refinement to account for the viscous layers are inserted on the specified surfaces (in

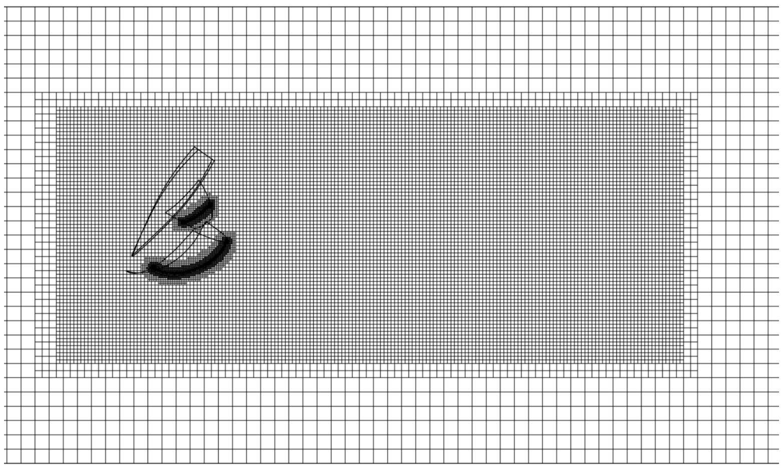
this case the two sails) and the first layer thickness and number of layers are computed by the software given reference length, Reynolds number and y+ value.

It is worth to discuss a bit further the viscous layer refinement: it consists in layers of high aspect ratio cells inserted tangentially to the wall in order to correctly resolve boundary layers. HEXPRESS has two ways of creating the viscous layer: with fixed first layer thickness or variable first layer thickness. For this project the first method has been chosen, because it is faster even though a little less flexible. Being the first layer thickness fixed, it must be computed, and the software uses the following formula:

$$y_{wall} = 6\left(\frac{V_{ref}}{\nu}\right)^{\frac{7}{8}}\left(\frac{L_{ref}}{2}\right)^{\frac{1}{2}} y^+, \tag{3.6}$$

where y_{wall} is the first layer thickness, V_{ref} is the reference velocity, ν the kinematic viscosity, L_{ref} the reference length and y^+ the parietal coordinate. This dimensionless value is an estimate of how fine the mesh is with respect to the current CFD problem. When using a turbulence model for the close to the wall region, it is advisable that this value is close to one: in that way there is the certainty of having a good representation of the boundary layer. In this case, setting y^+ to 1 was very restrictive and for this reason it has been set to 5.

In total, 5 layers were inserted, and the first layer thickness was of 5.6e-4 m. The final cell count is of 5.5 million cells, that can be satisfying in the sense that it is fine enough to describe the geometry and the physical processes but also not too fine, desirable for the reasons explained earlier. The quality of this mesh is obviously much lower than the 2D one, because in order to satisfy all the criteria it would have been necessary to have many more cells and again that is not desirable. Hence, some criteria fail the mesh check. Specifically for the cell skewness (a measure of how close to ideal (i.e., equilateral or equiangular) a cell is),for hexahedral meshes the suggested maximum is 0.85, while the obtained value is 0.92. Also the expansion ratio, a measure of the size variation of two adjacent cell, maximum recommended 5, here is 24. Moreover, some relaxed cells are still present after the optimization. However, while running the simulations it has been assessed that these failed mesh checks do not influence the solution, namely, in correspondence of the "bad" cells locations the solution is still smooth and does not present unwanted jumps or non physical values. An attempt to further enhance the mesh quality has been done, by increasing the number of optimization loops, but after a couple of trials the result did not change anymore, so this is the best combination between quality and number of cells that could be found by the author. An overview of the mesh is reported in figure 3.15. In figure 3.15a a cutting plane at z=1.5 m has been applied in order to see the internal part of the mesh: the box refinement of figure 3.14b can be observed, as well as the initial mesh outside of it. Also the much finer refinement of the sail can be seen. In figure 3.15b the meshed yacht is visible: note how the refinement is higher close to the sail boundaries, while in the center the cells get larger.


(a) View of the mesh through cutting plane at z=1.5 m

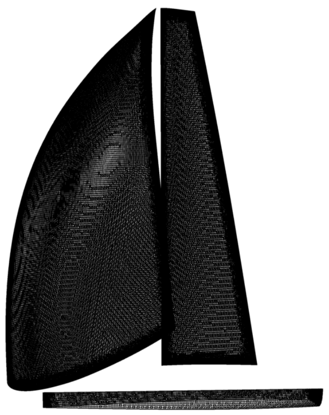

(b) Meshed yacht

Figure 3.15: Details of the 3D mesh (5.5 million cells)

Regarding the finer mesh, it has been obtained simply by dividing the target cell sizes by a factor of $\sqrt[3]{2}$, so that the final cell size would be 2 times smaller than the one in the coarser mesh. The result of this operation is a mesh with 9.8 million cells. The same output as the previous mesh was given when running the mesh check, so the problems could not be eliminated by refining by a factor 2. The idea is that this finer mesh will be used for the mesh independecy study but hopefully the coarser one will be used for the FSI.

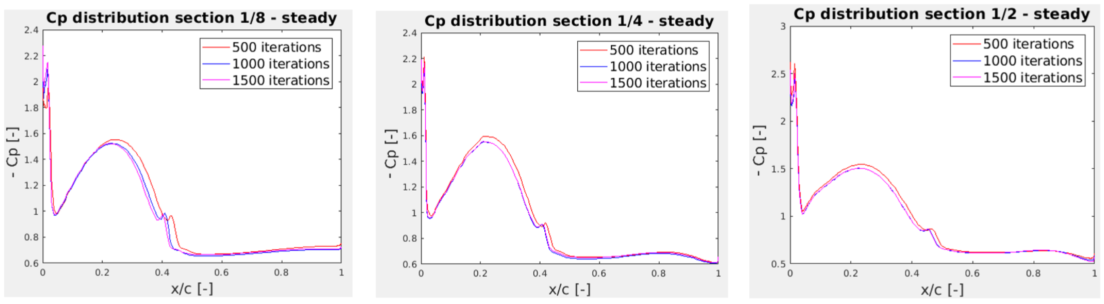
3.3.2. OpenFOAM Simulation

3.3.2.1. Running

The simulations were run for both OpenFOAM and FINE/Open for both meshes, the fine and coarse one. Again similar settings to the ones used for the 2D simulations are used for the OpenFOAM simulations: all the parameters defined in section 3.2.2 are set to the same values. The only difference would be that the top and bottom parts of the domain, set to "empty" for the 2D simulations, here are treated like the side walls, and therefore their boundary condition is slip for k, ω, p and U, while it is set to calculated with value 0 for ν_t. This boundary condition is not designed to be evaluated; it is assumed that the value is assigned (in this case 0). The solver is SIMPLE, therefore steady state solver, given the results from the unsteady simulations run for the 2D case. The hope here is that the solution converges to a steady state configuration and that the pressure distribution on the 5 sections of the sail resemble more the experimental data. The simulation that gave best results was run for 1000 steady iterations, and the run time was of 1 hour on 20 processors for the 5.5 million cells mesh, and of 5 hours on 20 processors for the 9.8 million cells mesh.

3.3.2.2. Post-processing

In order to assess what was the optimal number of steady iterations a convergence study has been performed on the coarser mesh. The number of iterations investigated were 500, 1000 and 1500. The quantity investigated for this study was the pressure distribution at the sections that will be also compared to the validation data. Figure 3.16 shows the results of this study.

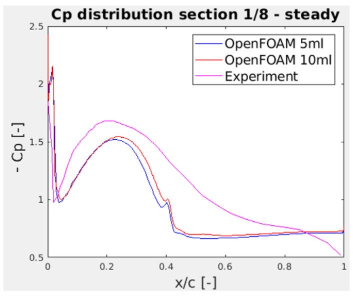


(a) Section 1/8 (b) Section 1/4 (c) Section 1/2

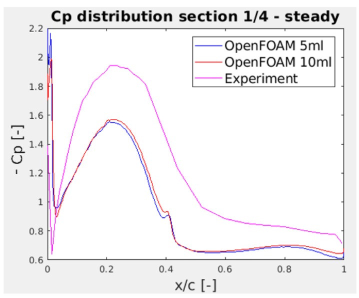
Figure 3.16: Steady iteration study for the 5 ml mesh

From these figures it can be argued that the 500 iteration pressure distribution is slightly off with respect to the other quantities plotted in the graph. This suggests that 500 iterations are too little for this simulation to converge. While looking at the distributions for 1000 and 1500 iterations however, we do not notice much difference: especially for figures 3.16b and 3.16c the two pressure distributions are basically overlapping. That means that convergence is reached and iterating further would not change the results anymore. For this reason, the simulation with 1000 iterations can be considered the best one in terms of results and convergence time. That being said, from now on only the results obtained with 1000 iteration will be shown and taken into consideration.

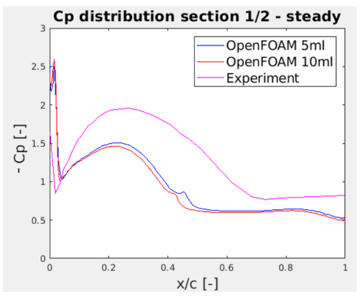
A mesh independence study has been performed, this time investigating the Cp distribution as a function of the level of refinement of the mesh. The simulations, run on the coarse and fine mesh with 1000 iterations, yielded the results plotted in figure 3.17, together with the validation data. First, looking at the results for the 5.5 million mesh it can be argued that the solution is quite far from the validation data: the Cp amplitude is lower for all sections. When refining the mesh one would expect that the solution would grow closer to the



(a) Section 1/8



(b) Section 1/4



(c) Section 1/2

Figure 3.17: Mesh study

experimental one, as it would be more precise in describing the physics of the problem. In figure 3.17 it is clear that this is not the case: the solution is very similar for the two meshes and refining the mesh does not seem to help bringing the solution closer to the experimental one. Being the results very similar, it seems reasonable to use the coarser mesh to have shorter simulation times in the rest of the project, without losing accuracy in the results.

Now that the best combination of number of cells and number of iterations is found, the solution can be analyzed more specifically for the chosen case. Figures 3.18, 3.19 and 3.20 show some processed images of the flow field that can help understand the solution better.

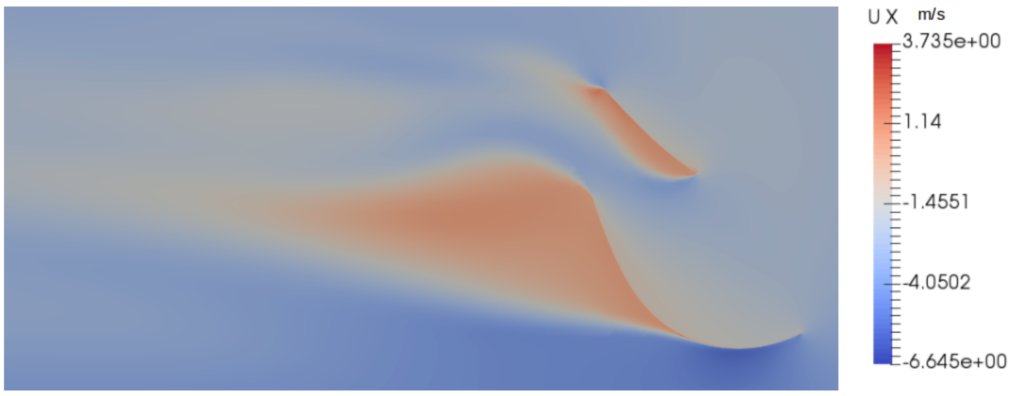


Figure 3.18: Horizontal velocity field for the section 1/8

Focusing on figure 3.18, which depicts the horizontal velocity field at the 1/8 section, two very easy observations can be made: in the blue area the velocity is following the freestream and therefore the flow is moving to the left, while the red area is the area of recirculation, where the flow goes against the freestream direction. A big difference between this flow situation and the one depicted for the 2D case in figures 3.7 and 3.9 can be noticed: here the flow stays attached for a longer portion of the chord length (until more or less half of the chord length), while in the 2D case the separation occurred immediately. This result is reassuring, and it can be expected that this solution will resemble more the validation data. Another difference with the 2D solution is that here the range where the horizontal velocity lies is smaller, meaning that we do not reach such high velocities as in figure 3.7, both in the negative and positive range. The velocity range here is more similar to the one found with the 2D Fine/OPEN simulation (figure 3.9).

Regarding the pressure, figure 3.19 shows a completely different pressure distribution than the one obtained in the 2D case: here there is no periodic vortex shedding. The solution stabilized to this constant pressure distribution on the sail, and no vortices are formed behind the leading edge of the spinnaker, as it was the case for the 2D simulations (see figures 3.5, 3.7 and 3.10). In figure 3.19 it can be seen that there is an area of lower pressure behind the sails, having a minimum around 0.2 x/c, which can be recognized as

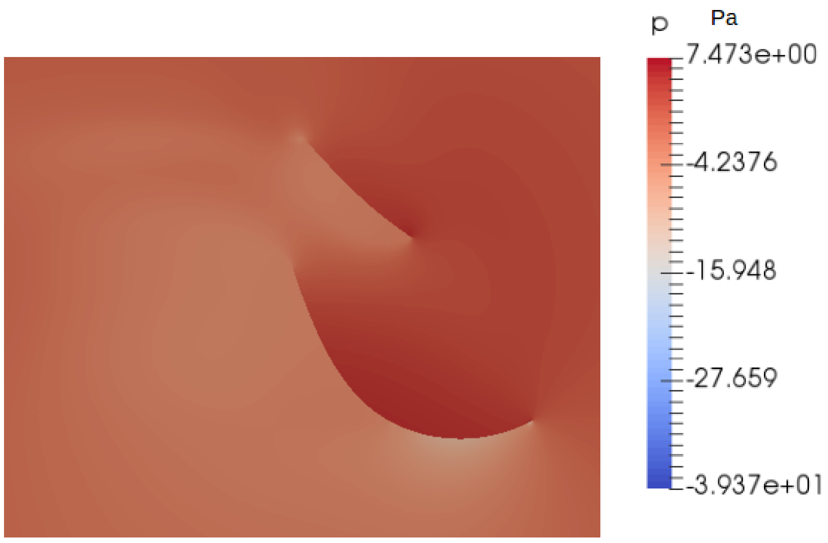


Figure 3.19: Pressure field for the section 1/8

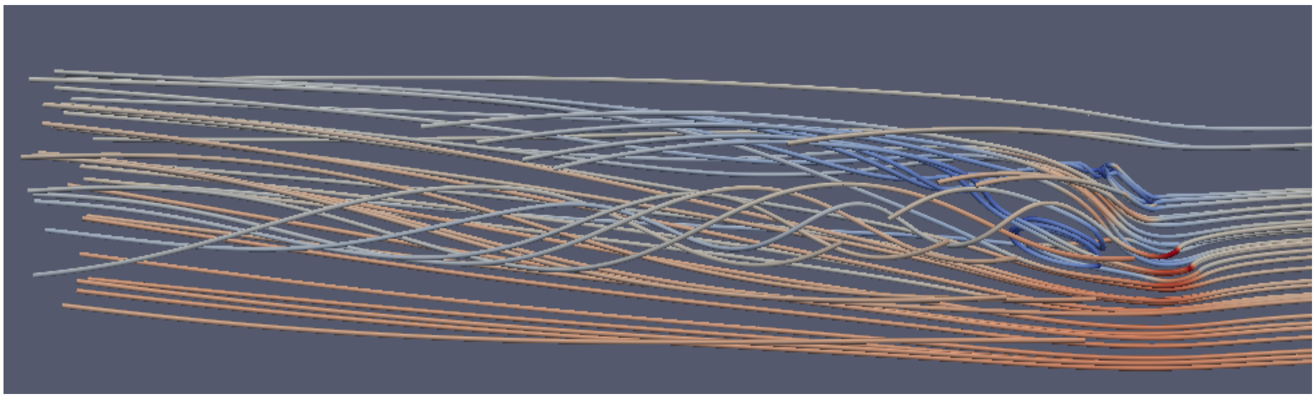


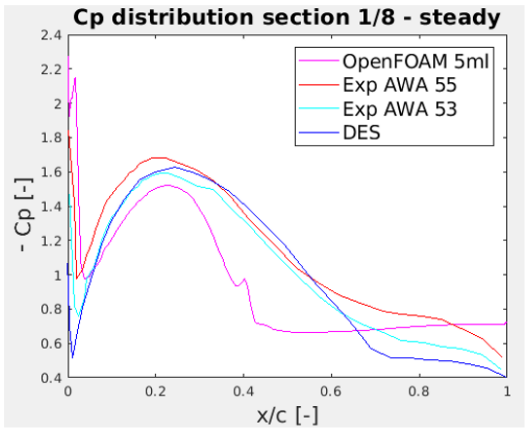
Figure 3.20: Streamlines (Z view)

the suction peak. This suction peak accelerates the flow (blue area below spinnaker in figure 3.18), and then the flow separates so the pressure is restored to a constant value for the rest of the chord length. The pressure distribution will be compared with the experimental data and commented further in the next paragraph.

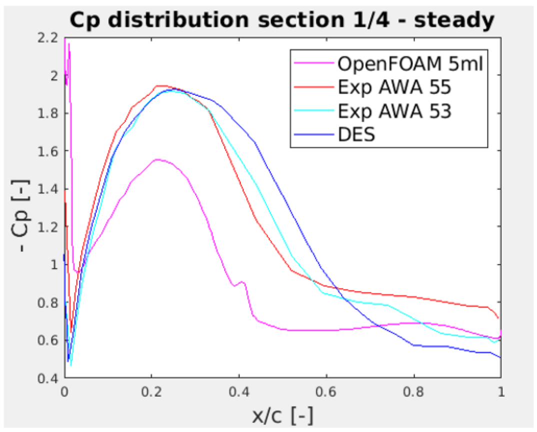
Lastly, the streamlines of the flow have been reported in figure 3.20, to give an idea of how the sails affect the flow. The streamlines that are shown are obtained by visualizing the computational domain from just above the mast (otherwise undisturbed streamlines would have been more evident in foreground). The streamlines are colored by velocity magnitude, the red color corresponding to a higher velocity magnitude and the blue to a lower one. The effect of the sails is evident: the flow, at first parallel to the walls, is deflected and accelerated by the sails. Some streamlines, after being deflected, continue their path in the new direction. Others enter the recirculation area and are slowed down and their direction is inverted. An area of high turbulence is created behind the spinnaker and also convected in the flow, visible as the streamlines that follow a twisted pattern until the end of the domain.

3.3.2.3. Comparison with validation data

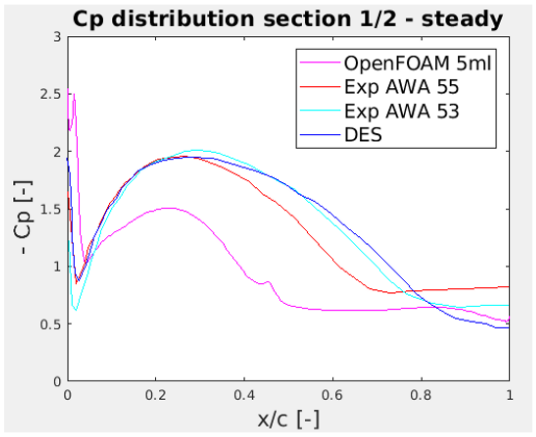
Eventually the obtained data can be compared with the validation data. The results of this operation are shown in figure 3.21. It is worth mentioning that in the reference paper [9] also the results for a different apparent wind angle were reported, namely 53°. The current simulation was done for 55°but both are reported



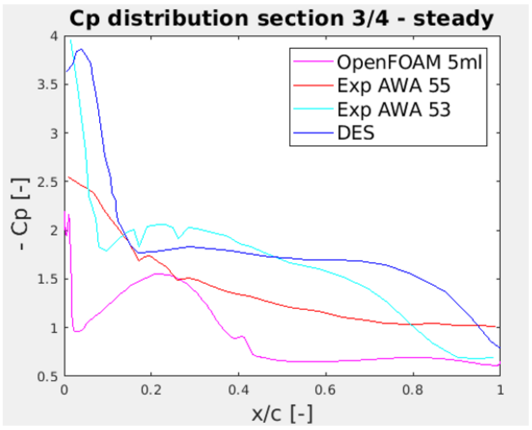
(a) Section 1/8



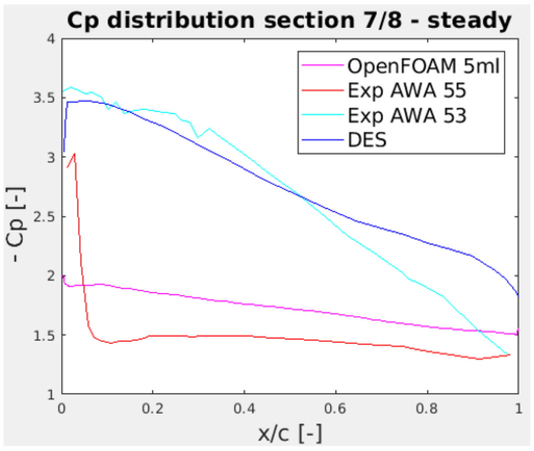
(b) Section 1/4



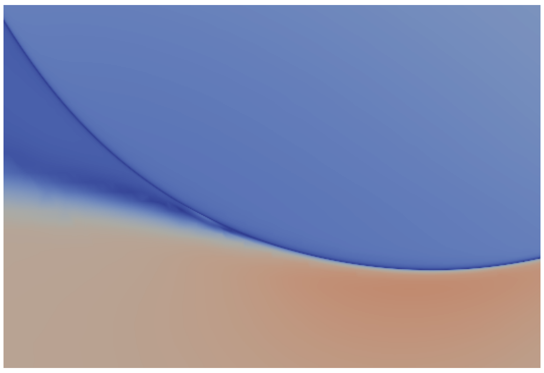
(c) Section 1/2



(d) Section 3/4



(e) Section 7/8



(f) Zoom of the velocity magnitude at the 1/8 section

Figure 3.21: Cp distribution at various sections - compared to validation data

in these figures. Moreover, also the solution reported by Viola et al. [38] obtained by running a DES simulation is reported.

The first remark that can be made observing the figures is that the computed solution does not coincide with the experimental one. The Cp value is always lower than the experimental one for all sections. While there is more resemblance to the reference with respect to the 2D solution, these results still can not be considered satisfying. The main differences can be noticed in the suction peak and the separation point.

Observing the first four subfigures of figure 3.21, it is clear how the amplitude of the peak is not the same as the one suggested by the validation data, and also its location is not computed exactly.

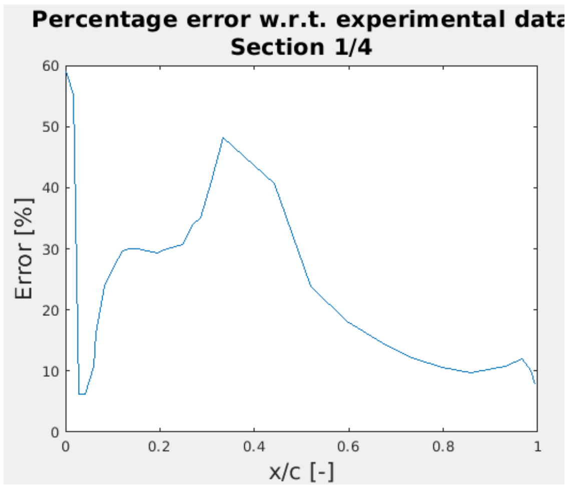


Figure 3.22: Percentage error plot - Section 1/8

On the other hand, the two highest sections have the most variable pressure distributions and also the reference data changes a lot when rotating the model by only two degrees: for example in figure 3.21e the difference between the two experimental results is very large. This can be justified by the fact that in that area the cloth is very short, being it the tip of the triangular sail, and therefore its shape is very variable and the distribution can vary a lot when inputting a small difference in the intial conditions. This can be also verified in real life: the top of the sails is always the area where the most flapping and least constant behavior can be observed. Therefore, when analyzing the accuracy of the simulation data, it is safer to rely on the lower sections.

A study on the error has been conducted for the OpenFOAM solution, with the intention of quantifying the error. For each section the error has been calculated for each validation point in the following way:

$$err_{\%} = \frac{Cp_{exp} - Cp_{sim}}{max(Cp_{exp}) - min(Cp_{exp})} \times 100, \tag{3.7}$$

The root mean square of the error computed with formula 3.7 has been calculated for all sections in the points corresponding to the ones reported in the reference. The root mean square of the error was of 23.2% for the 1/8 section, 28% for the section 1/4, 43% for the section 1/2, around 42% for the 3/4 section and finally 23% for the highest section. All the errors are higher in the first portion of the chord length, due to the location of the peak that can be slightly shifted and therefore yield a high error. The error distribution for section 1/4 has been shown in figure 3.22, where the tendency to have the highest error at the beginning of the chord length is clearly visible.

When observing the computed pressure distribution a behavior common to the first three sections can be observed: separation is always reached earlier than in the validation data. The separation point is identified by the point after which the Cp reaches a plateau. Another common feature is that before reaching the separation, all the computed pressure distributions present a small bump. This could be interpreted as the point where the flow regime transitions to turbulent, and when that happens it cannot stay attached to the profile and therefore separates. When looking closely at the velocity magnitude distribution, for example at the section 1/8 (figure 3.21f), the little bump present in the Cp distribution can be observed. The dark blue color represents a low velocity magnitude while the red one a higher velocity, and the white is in between the two. When beginning the separation, an area of low velocity develops behind the sail (blue zone), while in the little white bubble the velocity is a bit higher. That is in agreement with the pressure distribution, that presents another small suction peak in correspondence of that point. Finally it can be interpreted in the following way: the pressure, after the suction peak, is rising again, but then it undergoes a sudden drop that allows the flow to slightly accelerate, transitioning to turbulence, and leading to separation.

In conclusions these results showed that the 3D simulation can definitely reproduce the experimental data better than the 2D one, but the similarity is still quite low to be considered acceptable. For this reason the simulation will be run also with the other CFD software, with the hope of getting more accurate results.

3.3.3. FINE/Open Simulation

3.3.3.1. Running

The steady 3D simulations were run on FINE/Open for three different cases: the 10 million mesh, the 5 million mesh and then 5 million mesh with different boundary conditions. The first two simulations were run to perform a convergence study for this solver as well, with the hope that the conclusions will be the same as the ones drawn in the previous section. The last simulation has the boundary conditions similar to the ones from the OpenFOAM simulations, while the first two consider all the bounding box patches as external. This will be explained more in detail below.

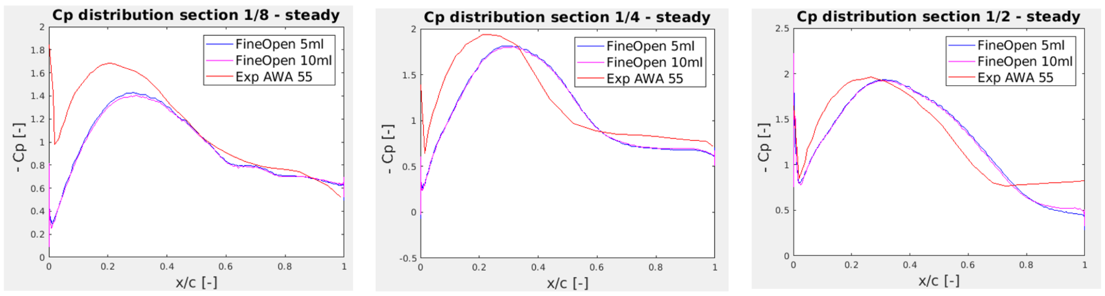
For the first two simulations, the boundary and initial conditions are the same as the ones defined in section 3.2.4, except for the fact that the top and bottom patches of the bounding box are treated as the other patches (inlet, outlet and side walls). Exactly like before, the walls are considered "external": meaning that all the freestream quantites are prescribed there and therefore the model yacht is modeled to be in the freestream and not in the wind tunnel.

In the last simulation, on the other hand, at the inlet velocity Ux, k and ε are prescribed, while at the outlet only the pressure p is prescribed. On the other external patches, namely the side walls and floor and ceiling, all the quantities (Ux, p, k, ε) are prescribed.

For all simulations, 1000 steady iterations were performed (seen the results from the iteration study done before), and the convergence criterion, defined in section 3.2.4 was -6. The run times were: a total time of 20 hours on one processor for the coarse mesh, while 4 hours on 20 processors for the fine mesh.

3.3.3.2. Post-processing

A mesh study has been conducted for this solver as well, in order to investigate the effect of the mesh resolution on the final result. The results of this investigation is shown in figure 3.23. As it has been explained before, for this type of investigation only the lower sections are shown because they are more explicative of the behavior of the solver, while for the higher sections it has been shown (see previous result section) that also the experimental data cannot be considered as the correct solution, given the high variability of the flow conditions in that area of the sail.



(a) Section 1/8 (b) Section 1/4 (c) Section 1/2

Figure 3.23: Mesh study

Figure 3.23 clearly shows that a refinement in the mesh does not improve the solutions: for all sections it is possible to observe that the solution for the coarse and fine mesh are overlapping. This result is encouraging because it allows to run the simulations in the FSI cycle with the 5 million mesh, shortening the run times and not losing accuracy. For this reason from now on only the 5 million solution will be shown and commented.
]

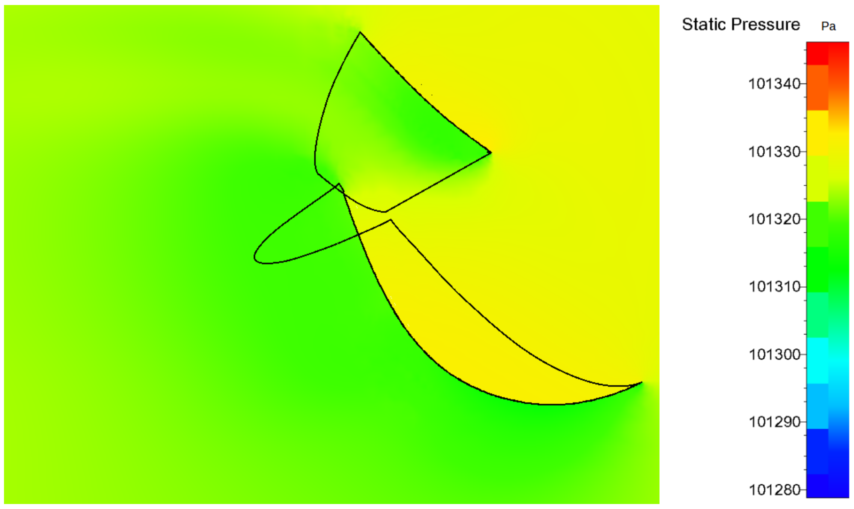


Figure 3.24: Pressure field for the section 1/8

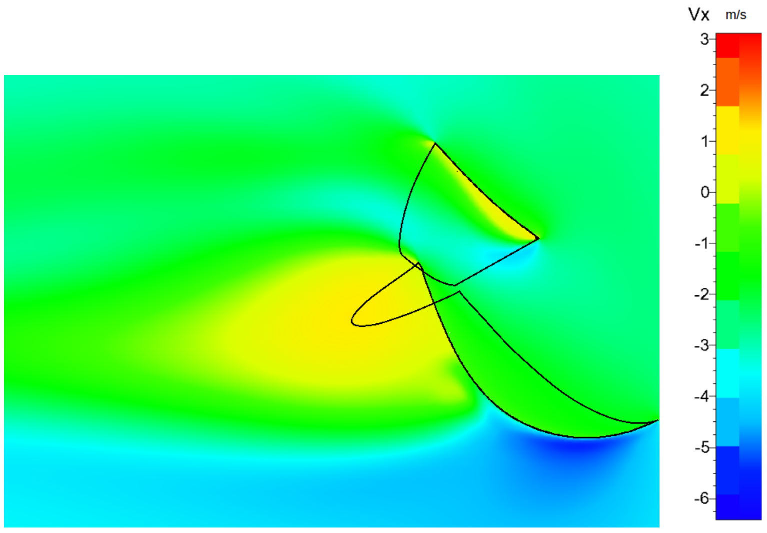


Figure 3.25: Horizontal velocity field for the section 1/8

In figures 3.24 and 3.25 the pressure and velocity fields at the section 1/8 are shown. The goal is to compare these results with the ones obtained for the same section with OpenFOAM. While the pressure distribution looks quite similar, with an area of lower pressure in the first half of the chord length and an area of pressure recovery after that, some differences can be observed in the horizontal velocity contour plots: while in figure 3.18 it seemed like the flow passed smoothly from attached to separated and from an area of freestream velocity to an area of recirculation, here it seems like the area of recirculation is influencing the flow accelerated by the spinnaker. The flow at first follows the sail curvature and then, when it separates, it is forced to slow down and to deflect by the area of backflow (the yellow area in figure 3.25). Another observation that can be made is that in comparison with the OpenFOAM solution the point of separation here is reached later, as it will also be analyzed later. On the other hand, one similarity with the OpenFOAM solution is the range where Ux lies: they are very similar for this case, differently from what was found when comparing the 2D solutions.

The postprocessing tool CFView allowed to create images of the streamlines in the field. Figure 3.26 is a view from above of the computational domain and it is comparable with figure 3.20. It is possible to recognize a similar behavior: also here the streamlines are curved and accelerated by the presence of the sails, and also in this figure it is possible to visualize the recirculation area, together with the are of high turbulence that is

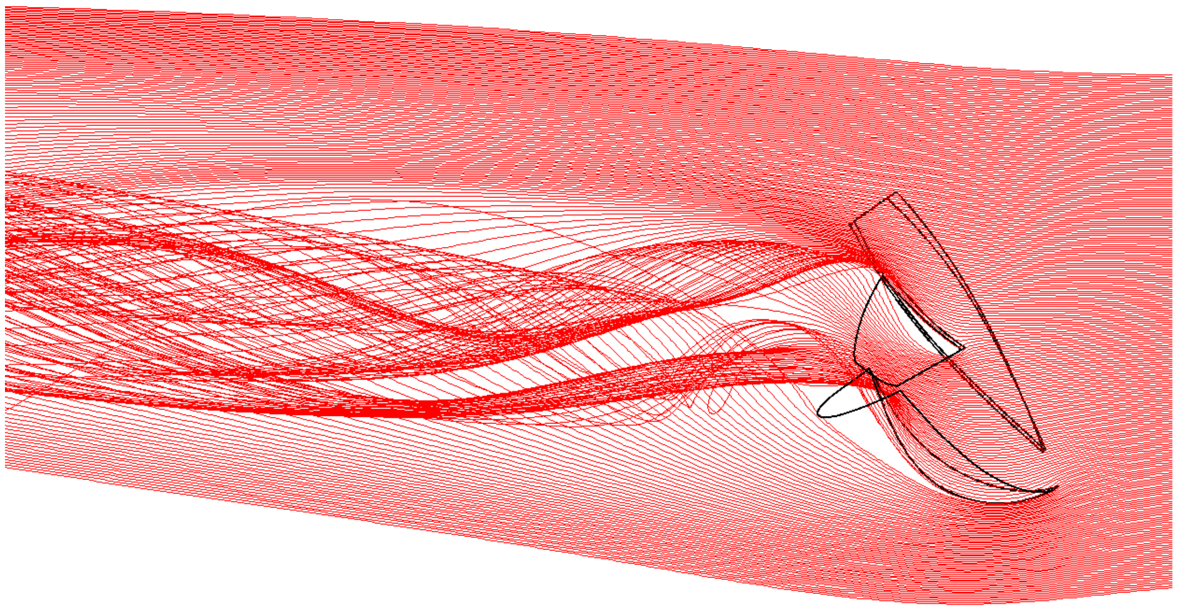


Figure 3.26: Streamlines (Z view)

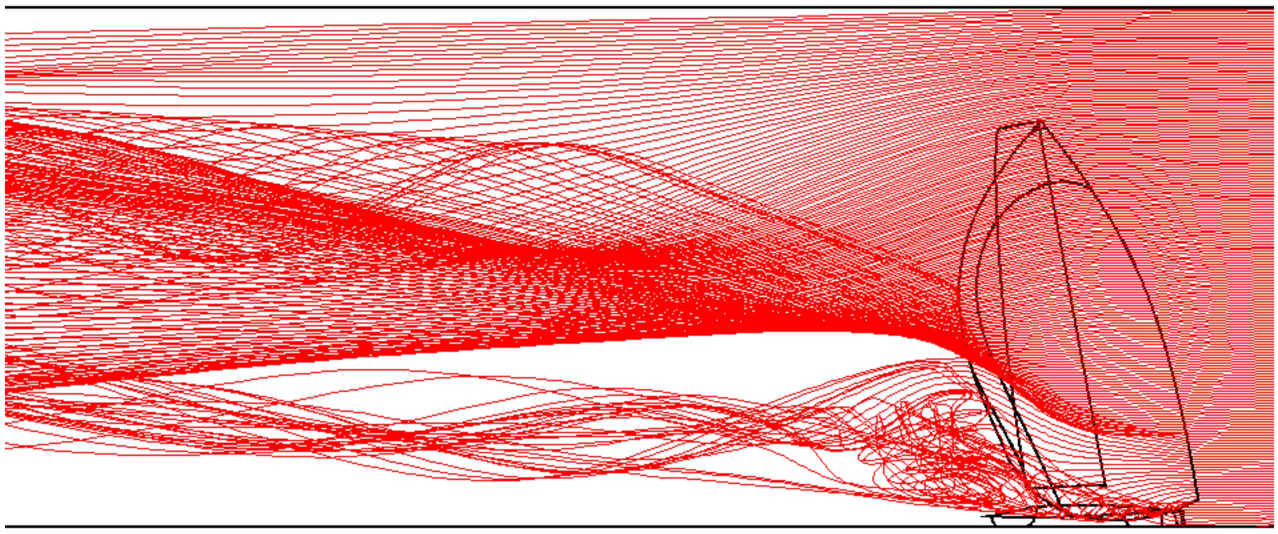
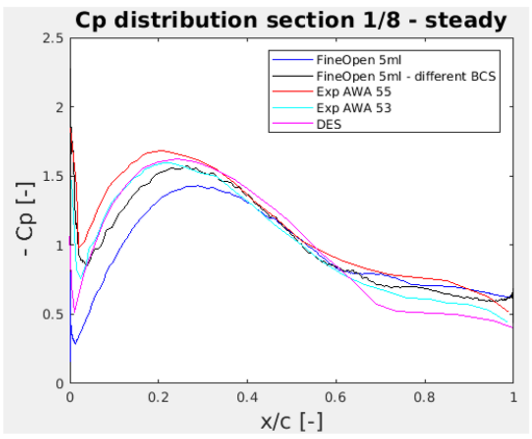


Figure 3.27: Streamlines (Y view)

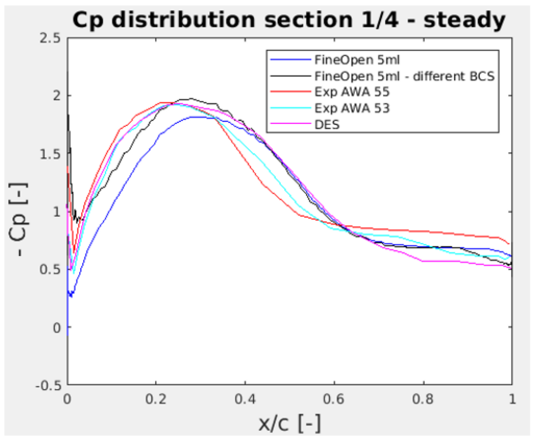
created behind the sail and convected in the flow. An interesting observation can be made when observing figure 3.27: this figure exemplifies why the simulation in 3D was necessary. Consider the streamlines starting in the lower section of the inlet in figure 3.27. At first they are parallel to the floor but then, when the sail is encountered, they are deflected and moved up by the sail. In figure 3.26 it can be observed how the streamlines are deflected in the X and Y direction, but here in figure 3.27 it is clear how also in the Z direction there will be a variation of position of the streamlines. This could have not been captured with a 2D simulation, and now the fact that the 2D simulations were completely not accurate is explained.

3.3.3.3. Comparison with validation data

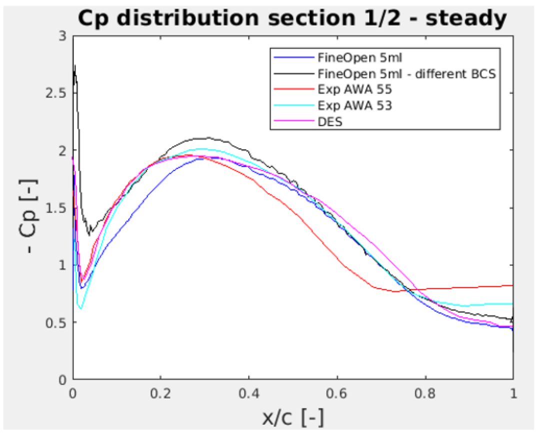
Now that the flow field has been examined it is necessary to compare the obtained results to the experimental data. In figure 3.28 the results for both the 5 million cells simulations are reported, plotted against the experimental data for two angles of attack and the results from the DES simulation done by Viola et al. [38].



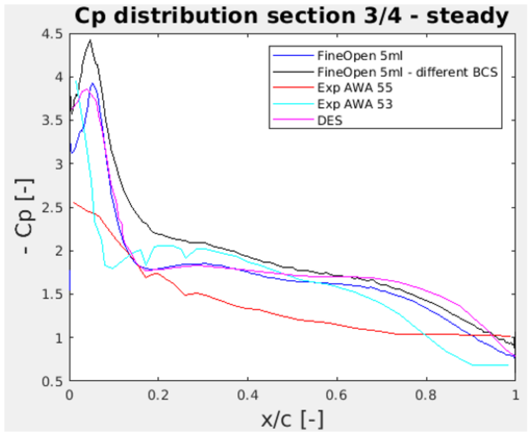
(a) Section 1/8



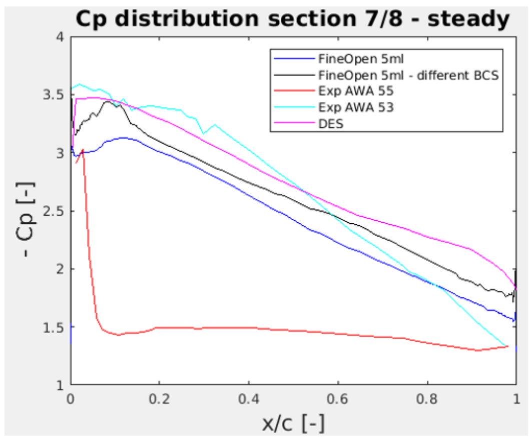
(b) Section 1/4



(c) Section 1/2



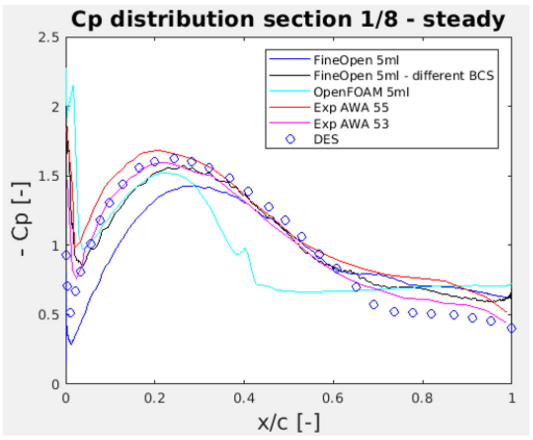
(d) Section 3/4



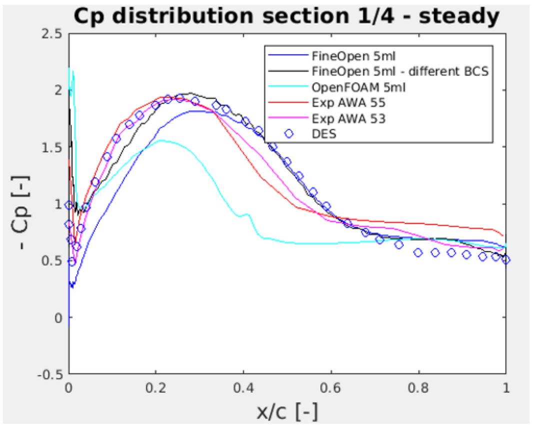
(e) Section 7/8

Figure 3.28: Cp distribution at various sections - compared to validation data

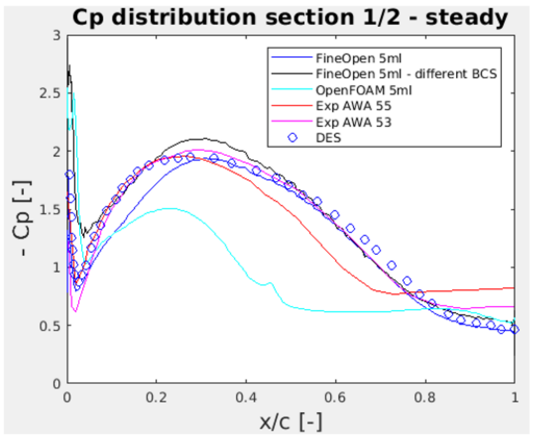
The results seem quite satisfying: in figure 3.28 it is possible to notice a close agreement between the computed and experimental data, especially for the lower sections. The simulation with the different boundary conditions (inlet and outlet prescribed) seems to resemble the experimental data the most. Differently from what was observed for the OpenFOAM simulations here it can be stated that the suction peak location and amplitude is well represented by this simulation. Regarding the distribution at the 7/8 location, it can be observed that the result of the simulations, as well as the results from DES resemble more the experimental data for a slightly different AWA. This is reasonable if one thinks of the variables that are involved in the ex-



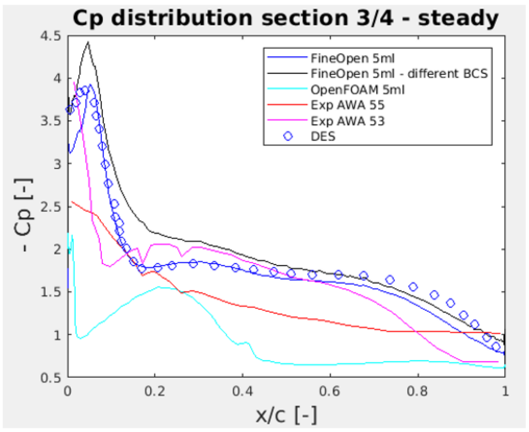
(a) Section 1/8



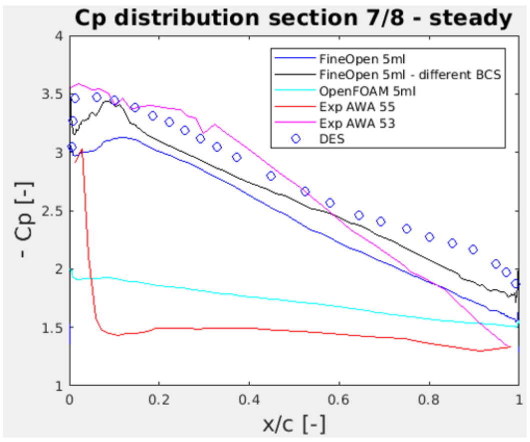
(b) Section 1/4



(c) Section 1/2



(d) Section 3/4



(e) Section 7/8

Figure 3.29: Cp distribution at various sections - compared to OpenFOAM

periment in the tunnel: a small variation of the inclination of the model with respect to the freestream can change a lot the output, and therefore it can be said that the experimental data might not be so reliable as reference data for the simulations in this location. This is also justified by the fact that as stated before also in real life sailing the top of sail is the first part to encounter wrinkling, flapping and instability due to short chord length and variability of wind conditions of that area. However, seeing that the behavior for a slightly lower angle of attack is well reproduced by the Fine/OPEN simulation is reassuring.

A study on the error has been conducted for the solution that seemed to yield the best results: the one with the inlet and outlet boundary conditions. For each section the error has been calculated for each validation point as done before. As it can be already seen in the pictures, the error is lower for the lower sections and it increases as the z coordinate increases. The root mean square of the error computed with formula **??** has been calculated for all sections. The root mean square of the error was of 10% for the 1/8 section, 14% for the section 1/4, 19% for the section 1/2, around 40% for the 3/4 section and finally really high (72%) for the highest section. All the errors are higher in the first portion of the chord length, due to the location of the peak that can be slightly shifted and therefore yield a high error. However visually it can be recognized from figure 3.28 that the amplitude of the initial peak is approximated quite well, except for the highest sections.

One final analysis was conducted: the comparison of the results of the two solvers. The result of that analysis is shown in figure 3.29, where the results of all the 5 million cells simulations are reported. First characteristic that comes to the eye is that FINE/Open definitely performs better than OpenFOAM in reproducing the experimental data. While the FINE/Open results are comparable both to experimental and DES data, the OpenFOAM solution is way off both. In terms of separation location, OpenFOAM underpredicts it for all sections, while FINE/Open is able to predict it quite correctly. The only one section where OpenFOAM seems to have done a better job is the highest one: the experimental data suggests a sudden separation almost right after the leading edge, and OpenFOAM predicts separation right away as soon as the flow encounters the sail. Morevoer, the wiggle in the Cp distribution that was present in all the OpenFOAM results except for the last one, cannot be observed in the FINE/Open data, and neither in the experimental results. That suggests that OpenFOAM is predicting something different. This result is quite surprising because the same simulation settings have been used for both softwares, especially the second FINE/Open simulation was done to resemble the OpenFOAM boundary conditions. The solver was steady state for both, the turbulence model was k-ω SST for both and the turbulent quantites were initialized to the same values. The only explanation that comes to mind can be that the two solvers work in different ways, maybe using different relaxation techniques or different initial guesses, that lead to such different final solutions.

It can be concluded that for the FINE/Open simulations the one conducted with 5 million cells with BC's that impose different values at inlet and outlet was the most successful in reproducing the reference data, and was the most efficient in terms of iterations and number of cells.

3.4. Conclusions

An investigation on the best CFD solver to reproduce the experimental data from [9] has been conducted. 2D simulations were run on both OpenFOAM and FINE/Open, but they turned out to give poor results, showing inability to reproduce the experimental data. The need for 3D simulations was then identified, and it emerged that it was necessary to shift to this type of analysis given the physics of the problem, which can only be correctly captured in 3D. Three dimensional simulations were then run investigating the effect on the solution of the number of cells, number of steady iterations and boundary conditions. Aside from this, also the performances of the two solvers were evaluated and the best combination between all these variables has been found for this problem: the FINE/Open simulation, with 5 million cells, 1000 steady iterations and the boundary conditions prescribed differently for inlet and outlet was the one that gave the most satisfactory results in terms of accuracy and run time.

For all these reasons the results of the best simulation will be used in the rest of the FSI cycles and all the other CFD simulations will be run with these settings. It can be said that the first of the research questions reported in the Literature study has been answered and validated in this chapter:

What are the appropriate techniques to compute the correct pressure distribution around the spinnaker sail through a CFD solver?

The answer has been found, discussed and motivated throughout all this chapter.

4 FEM Simulations

In this chapter the simulations run with the structure solver are presented. First, a presentation of the code is given, with an overview of the flowchart. In order to assert its validity, a quantitative validation has been performed on two simple cases of which the analytic solution is available. Being the results of this first analysis satisfying, the solver is considered valid and the simulations on the spinnaker can be run.
Consequently the pre-processing steps are presented, then the running techniques (convergence, relaxation) are shown and finally the results of the simulations are reported and their validity is assessed. For this part validation data is not available, so only a qualitative analysis of the results is performed, comparing the results with a real spinnaker flying shape.

4.1. Presentation of the solver

The structure solver, SailFEM, is a Matlab code developed by Daniele Trimarchi in his Master's Thesis at the Università di Genova [35]. Trimarchi extended his project in his PhD [34], comparing SailFEM also with other finite element methods for the same application. Those other methods showed to be more accurate but they will not be presented in this report.
The solver is based on a simple FEM solver developed at the Università di Genova that was able to resolve a system of beams. Trimarchi modified that code in order to include the membrane elements that describe the sail. The equations and theory on which this solver is based are all reported and explained in detail in section 2.2.1. The flowchart of the code, taken from [35], is shown in figure 4.1.

The first part of the code deals with the import of the geometry and is done through a function named `importa.m`, whose work will be better detailed in the pre-processing section. Once the geometry is defined and imported the boundary conditions are defined. As stated before, the membrane element is composed of 3 nodes with 3 translational degrees of freedom each, resulting in a total of 9 DOF for each element. As boundary conditions, it will be possible to block one or more directions of translation for each node. The matrix D specifies which directions are clamped for which node. A typical form of the D matrix is:

$$D = \begin{bmatrix} 5 & 1 \\ 5 & 2 \\ 5 & 3 \\ 10 & 1 \end{bmatrix}.$$

This matrix specifies that node 5 is clamped in all three directions, while node 10 only in the x direction. From this matrix it is then possible to identify the total number of free and bound degrees of freedom, which will help determine of which nodes it will be necessary to compute the displacement.

Once the BC's are defined, the pressure needs to be imported from the CFD solver. It will be read from a file that has a pressure value for each node of the structure mesh. That file is obtained through an interpolation routine that will be detailed in chapter 5. All the nodes are loaded with the pressure specified in that

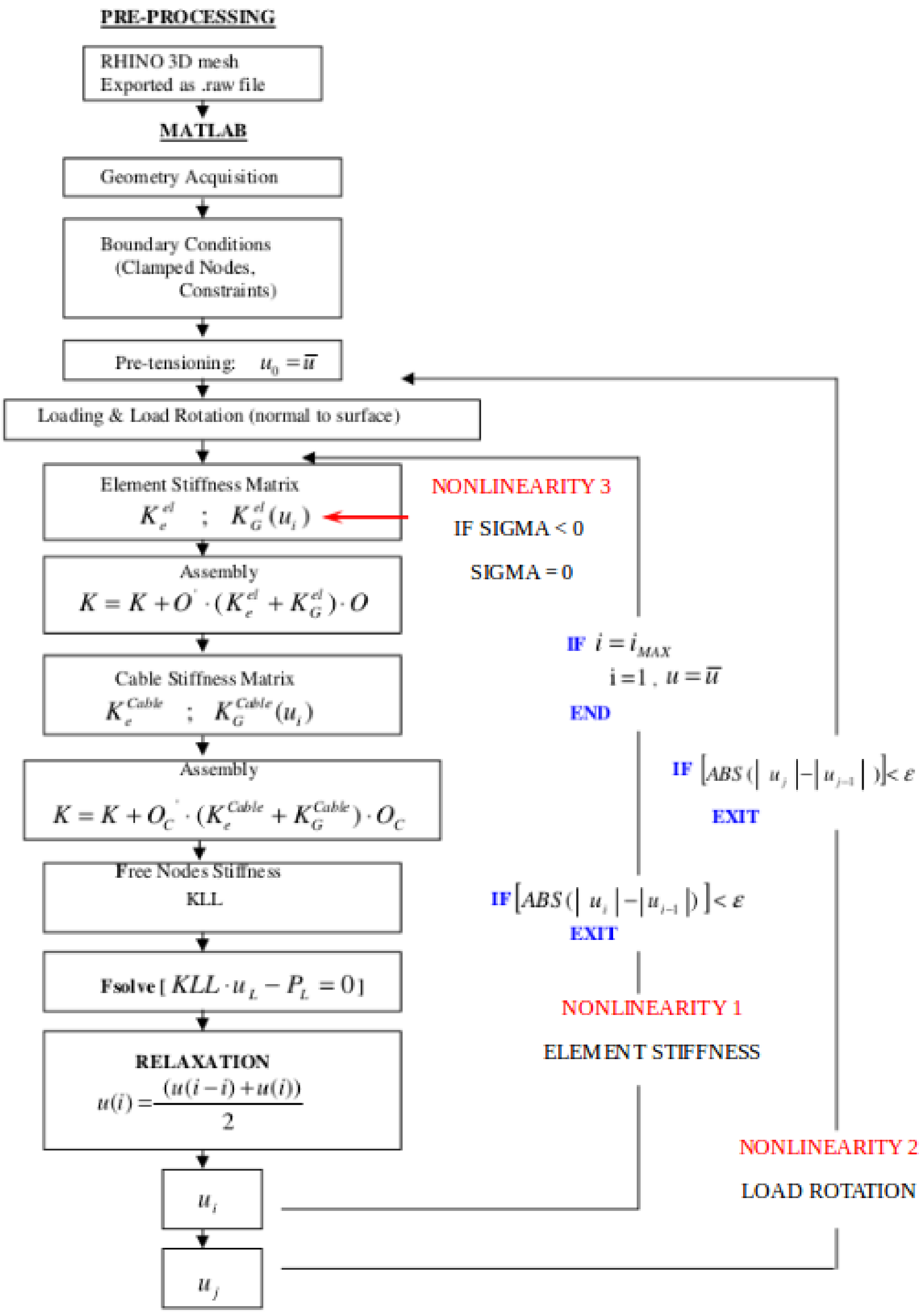

PRE-PROCESSING
RHINO 3D mesh Exported as .raw file
MATLAB
Geometry Acquisition
Boundary Conditions (Clamped Nodes, Constraints)
Pre-tensioning: $u_0 = \bar{u}$
Loading & Load Rotation (normal to surface)
Element Stiffness Matrix K_e^{el} ; $K_G^{el}(u_i)$
Assembly $K = K + O' \cdot (K_e^{el} + K_G^{el}) \cdot O$
Cable Stiffness Matrix K_e^{Cable} ; $K_G^{Cable}(u_i)$
Assembly $K = K + O_C' \cdot (K_e^{Cable} + K_G^{Cable}) \cdot O_C$
Free Nodes Stiffness KLL
Fsolve [$KLL \cdot u_L - P_L = 0$]
RELAXATION $u(i) = \frac{(u(i-i) + u(i))}{2}$
u_i
u_j
NONLINEARITY 3
IF SIGMA < 0
SIGMA = 0
IF $i = i_{MAX}$
$i = 1$, $u = \bar{u}$
END
IF $[ABS(|u_j| - |u_{j-1}|)] < \varepsilon$
EXIT
IF $[ABS(|u_i| - |u_{i-1}|)] < \varepsilon$
EXIT
NONLINEARITY 1
ELEMENT STIFFNESS
NONLINEARITY 2
LOAD ROTATION

Figure 4.1: Flowchart of SailFEM, adapted from [35]

file. Now the pressure is defined the nodes as a distributed pressure (units $\frac{N}{mm^2}$), while the structure solver needs a nodal load in N, namely an integration is needed. For this reason, a routine named `pr_ess.m` finds the normal for each node by averaging the normals of all the elements that share that node and applies the pressure in that direction. Consider the situation depicted in figure 4.2: in order to find the normal for point P, all the normals of the neighboring elements are summed and divided by the number of adjacent elements, obtaining an averaged normal.

$$\overrightarrow{n_P} = \frac{1}{n_e}\sum_{i=1}^{n_e}\overrightarrow{n_i}, \tag{4.1}$$

where n_e is the total number of elements sharing point P and $\vec{n}$ are the normal vectors. Once the normal is

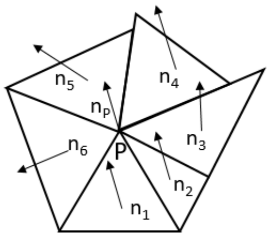


Figure 4.2: Normal of point P as average of adjacent normals

defined, the areas of the adjacent elements are computed, and with the hypothesis that the pressure is equally distributed on the three nodes of the element the equivalent load on each node is defined as

$$f_P = \frac{1}{3}P_P\sum_{i=1}^{n_e}A_i, \tag{4.2}$$

where f is the load in N, P is the load in $\frac{N}{mm^2}$ and A_i is the area of the i^{th} element. Moreover, another routine named `Load_Rot.m` is recomputing the node normal at each iteration, taking in account the deformations computed at the previous iteration. This will also be a way of checking the convergence: namely if the normals stay the same in different iterations the structure has stopped deforming and an equilibrium condition in the displacements has been reached.

Finished this step, the stiffness matrices have to be built. As specified in the previous chapter, the global stiffness matrix is composed of two matrices: the elastic and geometric stiffness. Through the formulation of Li and Chan [23], the elastic stiffness matrix has been linearized and is therefore constant at each iteration. On the other hand the geometric stiffness matrix will depend on the stress generated on the edges by the node displacement, and will therefore need to be recomputed at each iteration. Referring to section 2.2.2 for a more detailed explanation, it is worth reminding how the two matrices are defined:

$$K_e = A\cdot t\cdot T_G^T\cdot T_N^T\cdot C\cdot T_N\cdot T_G, \qquad K_g = A\cdot t\cdot G^T\cdot M\cdot G. \tag{4.3}$$

The matrix T_G is defined by the coordinates of the three nodes of the element with respect to the global coordinate system:

$$T_G = \begin{bmatrix} 0 & 0 & 0 & \frac{X_{21}-X_{31}}{l_{023}} & \frac{X_{22}-X_{32}}{l_{023}} & \frac{X_{33}-X_{23}}{l_{023}} & \frac{X_{31}-X_{21}}{l_{023}} & \frac{X_{32}-X_{22}}{l_{023}} & \frac{X_{23}-X_{33}}{l_{023}} \\ \frac{X_{11}-X_{31}}{l_{031}} & \frac{X_{12}-X_{32}}{l_{031}} & \frac{X_{13}-X_{23}}{l_{031}} & 0 & 0 & 0 & \frac{X_{31}-X_{11}}{l_{031}} & \frac{X_{32}-X_{12}}{l_{031}} & \frac{X_{33}-X_{13}}{l_{031}} \\ \frac{X_{11}-X_{21}}{l_{012}} & \frac{X_{12}-X_{22}}{l_{012}} & \frac{X_{13}-X_{23}}{l_{012}} & \frac{X_{21}-X_{11}}{l_{012}} & \frac{X_{22}-X_{12}}{l_{012}} & \frac{X_{23}-X_{13}}{l_{012}} & 0 & 0 & 0 \end{bmatrix}.$$

The matrix T_N is the rotation matrix from the local to the global coordinate system and can be defined as:

$$T_N = \Psi^{-1}\cdot L_{0d}^{-1} = \begin{bmatrix} l_{023} & 0 & 0 \\ 0 & l_{031} & 0 \\ 0 & 0 & l_{013} \end{bmatrix}^{-1} \cdot \begin{bmatrix} cos^2\theta_1 & sin^2\theta_1 & sin\theta_1 cos\theta_1 \\ cos^2\theta_2 & sin^2\theta_2 & sin\theta_2 cos\theta_2 \\ cos^2\theta_3 & sin^2\theta_3 & sin\theta_3 cos\theta_3 \end{bmatrix}.$$

The geometric stiffness matrix is defined as:

$$K_g = A \cdot t \cdot G^T \cdot M \cdot G = \begin{bmatrix} B_{12}+B_{31} & -B_{12} & -B_{31} \\ -B_{12} & B_{12}+B_{23} & -B_{23} \\ -B_{31} & -B_{23} & B_{23}+B_{31} \end{bmatrix}, \qquad B_{i,j} = \frac{P_{ij}}{I_{ij}} \cdot [I_3 - D_{ij} \cdot D_{ij}^T],$$

$$D_{ij} = \frac{1}{l_{0ij}} [(X_{j1} - X_{i1}) \cdot (X_{j2} - X_{i2}) \cdot (X_{j3} - X_{i3})]^T,$$

The definition of all the parameters introduced in these formulations can be found in subsection 2.2.2. The calculations start with the definition of the element edges, before and after imposing the displacements computed in the previous iterations (or guessed if it is the first iteration). When the nodes coordinates are known, as well as the edge lengths (in the undeformed configuration), it is possible to compute the rotation matrix D_{ij}, defined above. Once the edge lengths after the deformations are known it will be possible to compute the stress in the three element edges through the formulation:

$$\sigma_i = E \cdot \varepsilon_i = E \cdot \frac{l_i - l_{i0}}{l_{i0}}, \tag{4.4}$$

where E is the Young Modulus. In order to avoid numerical instabilities it is necessary to impose that the negative stresses are 0. This reflects behavior of membranes, that do not work under compression. In terms of numerical analysis this stabilizes the computation, because it imposes that the only internal forces are the ones generated by the deformation. Therefore, an `if` statement checks if the stresses are negative, and if yes it sets them to zero.

The stiffness matrices will be assembled inside the global stiffness matrix through extraction matrices, defined as following: their size is [3xNn], where 3 is the number of nodes per element and Nn the total number of elements. They are zero everywhere except in the positions occupied by the element nodes, where it will be one. This matrix will be the base for the Kronecker matrix, namely a matrix where instead of the ones there will be a eye matrix with dimensions equals to the DOF of the node they refer to, 3 in this project. The matlab formulation for these sorts of matrices is the following:

```
O=zeros(3,Nn);
O(1,el(e,1))=1;
O(2,el(e,2))=1;
O(3,el(e,3))=1;
O=kron(O,eye(3));
```

where e is the single element and el the connectivity matrix, containing information about the vertices of each element. The matrix defined in this way will be used to assemble the global stiffness matrix:

$$K_{GLOB} = \sum_{i=1}^{N_e} O_i^T \cdot (K_E^i + K_G^i) \cdot O_i, \tag{4.5}$$

where N_e is the total number of elements. In order to be able to solve a system of the type $Ku = f$, it is necessary to extract from K_{GLOB} the information relative to the free nodes only. The matrix can in fact be seen as:

$$K_{GLOB} = \begin{bmatrix} K_{LL} & K_{LV} \\ K_{VL} & K_{VV} \end{bmatrix},$$

where K_{LL} is the stiffness submatrix regarding the non bounded nodes, K_{LV} and K_{VL} describe the force, on a bounded node, generated by the unit displacement of a non bounded node, and K_{VV} respresents the stiffness submatrix of the bounded nodes. In order to get K_{LL} then it is necessary to impose that $K_{LL} = K_{GLOB}(gdl\_L, gdl\_L)$, where gdl_L is a vector whose values correspond to the degrees of freedom of the structure. For example, if $gdl\_L = [1 \quad 5 \quad 24 \quad 55]$ the DOF of the structure will be:

```
node 1  : translation x
node 2  : translation y
node 8  : translation z
node 19 : translation x
```

because indeed each node has 3 DOF. Therefore node 1 will have DOF 1, 2, 3, node 2 will have DOF 4, 5, 6 and so on.

It is worth pointing out that K_{LL} is made of two parts: K_E only depends on the initial configuration x_0, namely on the undeformed coordinates, while K_G depends also on the deformed coordinates x_S, equivalent to $x_0 + q$, where q is the computed displacement,

$$K_{LL} = K_E(x_0) + K_G(x_0 + q). \tag{4.6}$$

4.2. Verification cases

Due to the lack of validation data for this specific case, the software has been tested for some simple test cases for which the analytic solutions are available. In this way the solver's ability to deal with some simple yet relevant cases can be assessed.

4.2.1. Cylinder case

In this case a cylinder is subject to internal pressure and its circular sides are clamped, see figure 4.3.

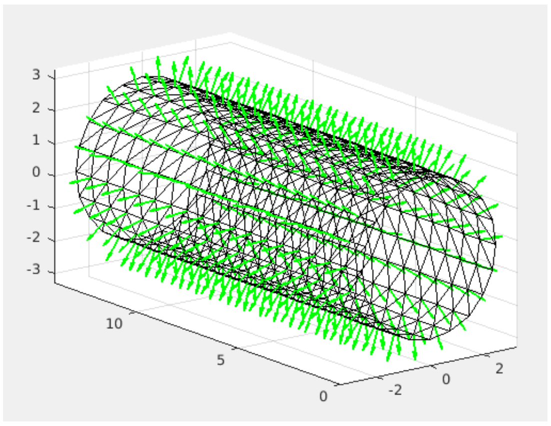


Figure 4.3: Cylinder with triangulation and pressure forces

The quantities to take into account are:

$$R = 2.5mm \qquad P = 100\frac{N}{mm^2} \qquad E = 1000\frac{N}{mm^2} \qquad t = 1mm.$$

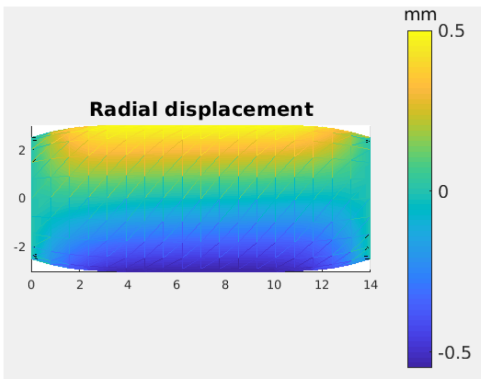
From the analytic solution the maximum radial displacement should be:

$$u_R = \frac{P \cdot R^2 \cdot (1-\nu^2)}{E \cdot t} = 0.568mm, \tag{4.7}$$

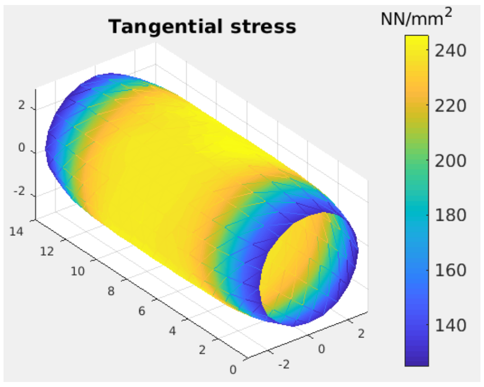
and in correspondence of that location the maximum tangential stress should be:

$$\sigma_\theta = R \cdot P = 250\frac{N}{mm^2} \tag{4.8}$$

The solver yields a maximum displacement of 0.541 mm, resulting in an error of 4%. Regarding the stress, as it can be observed in figure 4.4 the maximum is reached in correspondence of the maximum displacement, and it is there 240 NN/mm^2, also yielding an error of 4%. It can be argued that these results are satisfying and the solver is validated for this case.



(a) Displacement



(b) Tangential stress

Figure 4.4: Results by SailFEM on the cylinder case

4.2.2. Sphere case

Now that the solver is validated for a general case, in this test case values of pressure and Young modulus will resemble the ones used in the simulations of the spinnaker, in order to check if even in these pressure ranges the results agree with the analytic solution. This case is a sphere, subject to internal pressure and clamped at one single point (for the minimum value of z). The situation is depicted in figure 4.5.

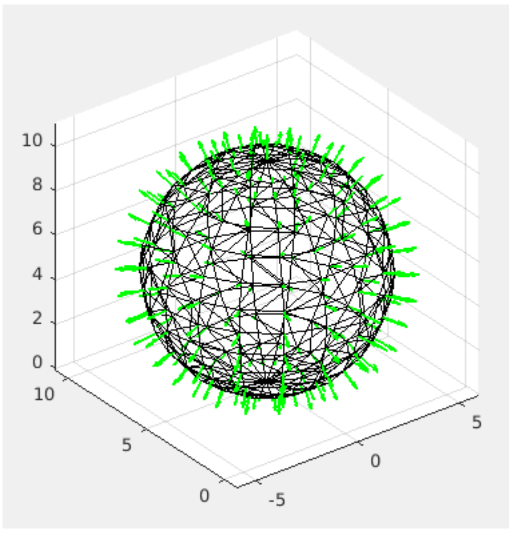


Figure 4.5: Sphere with triangulation and pressure forces

The used values are then the following:

$$R = 5mm \qquad P = 4\times 10^{-5}\frac{N}{mm^2} \qquad E = 375\frac{N}{mm^2} \qquad t = 0.3mm.$$

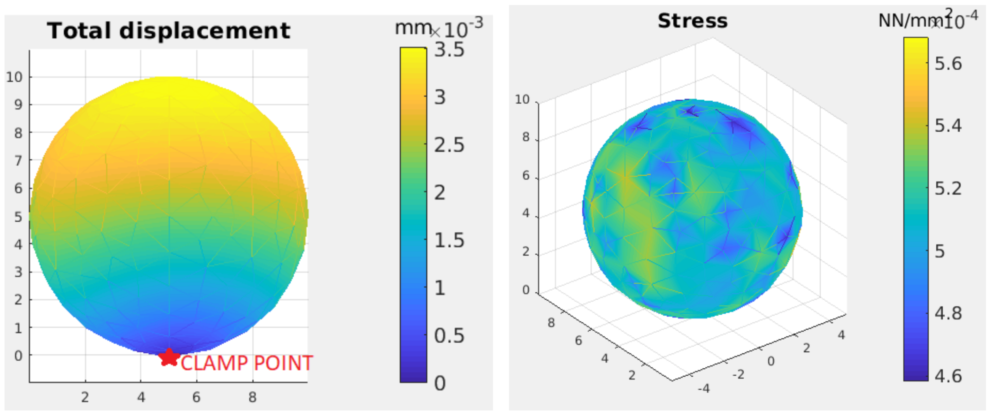
The value of pressure has been chosen to be similar to the values of Δp applied to the spinnaker. The thickness and Young modulus are the same as the spinnaker's ones. For this case the analytic solutions are:

$$u_R = \frac{P\cdot R^2\cdot(1-\nu)}{2Et} = 3.111\times 10^{-6}mm, \qquad \sigma_\theta = \sigma_\varphi = \frac{P\cdot R}{2t}. \tag{4.9}$$

The stress to be compared will then be the norm of the tangential and circumferential stresses, namely

$$\sigma = \sqrt{\sigma_\theta^2+\sigma_\phi^2} = 4.714\times 10^{-4}\frac{N}{mm^2}.$$

The results are reported in figure 4.6. SailFEM computes a radial displacement of 3.34×10^{-6} mm, resulting in



(a) Displacement magnitude (b) Stress

Figure 4.6: Results by SailFEM on the sphere case

an error of 7%, which is satisfying. The solver can compute the displacement correctly even when the loads are small. Regarding the stress, the error is also around 7%, being the mean computed stress 5.063×10^{-4} NN/mm^2. It can be stated that the solver is validated also for a case with the same material properties as the ones used for the spinnaker, even though in this case the error is slightly higher.

4.3. Pre-processing

The first aspect to deal with is the mesh creation. It is not advisable to use the mesh created for the flow simulation, since it is way finer than needed and it would only slow down the simulation. In fact, the fluid mesh has 120000 points only on the spinnaker, while for the structure solver a much coarser mesh can be used without losing accuracy. For this reason a new mesh has been created, only on the spinnaker, with the software Gmsh [10]. This software allows to create triangular surface meshes for the imported geometry and the level of refinement can be easily specified. In order to assess how fine the structure mesh has to be, a mesh study has been carried out on the spinnaker, using a constant pressure as external loading in order to have the same exact starting conditions for all meshes (if the interpolated pressure had been used, the loading could have been slightly different). From what has been observed in literature [12, 35, 17], it seems that it is advisable to have a uniform structure mesh that is not very fine. That is because the membrane element formulation requires that the thickness of the element is low with respect to its extension, namely, that the aspect ratio is quite high. The aspect ratio is defined as

$$AR = \frac{L}{t},$$

where L is the largest distance between two corners of an element and t its thickness. Of course there is an upper limit for this value, because the elements still have to be able to describe the geometry and its displacements correctly. Therefore the mesh study will yield the upper and lower limit for this value. The value for the thickness is the one found in literature typical of a spinnaker [17], 0.3 mm. With this value the mesh has been refined and coarsened and the convergence has been studied in terms of maximum displacement. The result of this analysis is shown in figure 4.7, where the value of the maximum displacement has been plotted as a function of the number of cells. The different studied meshes contained 115, 268, 397, 580, 731, 1023, 2264, 3137 and 4587 nodes, respectively. From this image it can be argued that the displacement is overestimated when using a very coarse mesh, while if the mesh is very fine the displacement is computed to be smaller than in reality. There is an area where the displacement is close to constant, with a value of 17 mm and a variation of ±4%, namely for the meshes with 580, 731, 1023 and 2264 nodes, corresponding to AR values between 146 and 294. That area is identifiable as the one between the two red lines in figure 4.7. It can be observed that if the mesh is coarser than 580 nodes the maximum displacement is widely overestimated, while refining the mesh makes the displacement tend to zero. Given the result of this mesh study the 580 nodes mesh has been selected as the one to use in the simulations, in order to obtain correct results and minimize the CPU time.

Now, once the mesh is exported from Gmsh as an `.stl` file a script in Matlab reads it and creates a matrix with all the node coordinates and another matrix with the connectivity information (its size is $[ne\times3]$, where

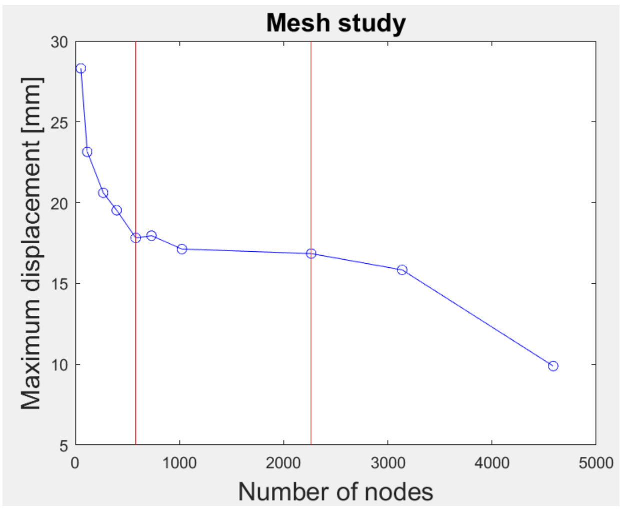


Figure 4.7: Mesh study on spinnaker with constant pressure=1.5×10^{-5}

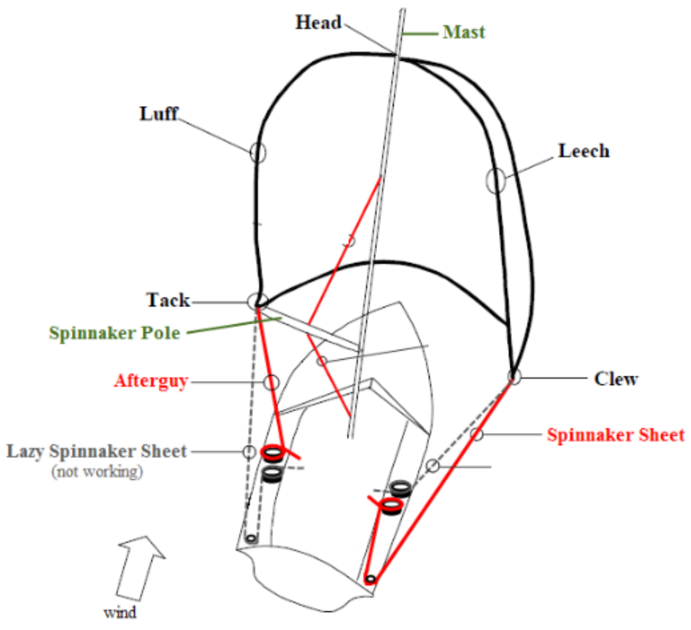


Figure 4.8: Technical names of parts of the spinnaker. The tack is the upwind bottom corner and the clew the downwind bottom corner. [1]

ne is the number of elements). The connectivity matrix has for each row, that represents the element, the index of the vertices that create the element.
The structure solver however requires as a geometry file of type `.raw`, which consists of as many lines as the number of elements, and for each line the x, y, z coordinates of the three vertices are reported. The format of each line will then be

$$x_1 \quad y_1 \quad z_1 \quad x_2 \quad y_2 \quad z_2 \quad x_3 \quad y_3 \quad z_3.$$

This format is easy to obtain with a Matlab script that uses the information of the triangulation to create the file. Once this file is correctly created it will be read and rearranged by the function `importa.m`, where the file is read and the nodes are rearranged so that there are no duplicate nodes. Moreover, the function returns another matrix, `nodel`, that defines the elements insisting on every node, rearranged in lines. Once this process is done the geometry is correctly defined.

[1]Adapted from http://www.sailberkeley.com/student-tips/spinnaker

Next is the definition of the boundary conditions. In section 4.1 the matrix D is defined. In order to decide which nodes to apply the clamps to it is necessary to find the coordinates of the vertex nodes, and then with an if statement it is possible to assign their values in the D matrix. The clamp has been imposed on the three nodes that are at the three vertices of the sail, in order to represent the presence of the halyard at the head and of the sheets at clew and tack (the two lower vertices). For a definitions of these technical terms see figure 4.8 The coordinates of the vertex points in millimeters are:

$$head = \begin{bmatrix} -271.5 \\ -444.7 \\ 2333 \end{bmatrix}, \qquad clew = \begin{bmatrix} -494 \\ -342.5 \\ 454.5 \end{bmatrix}, \qquad tack = \begin{bmatrix} 627.6 \\ -987.7 \\ 182.3 \end{bmatrix}.$$

For these vertex nodes all translations in x, y and z directions are clamped. Ideally it would be more accurate to impose a different boundary condition on the clew and tack, but the implementation of the spinnaker sheets would require a more complex procedure, either modifying the boundary conditions in a more elaborate way or even implementing the ropes as elements with different material properties, but that is beyond the scope of this thesis.

The last things to define before being able to run the simulation are the material constants. With the help of some literature and some sail designer friends it was possible to come up with the following values:

$$\mathrm{E} = 375\ \mathrm{N/mm^2}$$
$$\nu = 0.3$$
$$\mathrm{t} = 0.3\ \mathrm{mm}$$

where E is the Young modulus, whose value is taken from [35], ν the Poisson coefficient and t the sail thickness. The thickness value is the one of the non-scaled boat, as suggested by Giovanni Sanfelice (North Sails sail designer). The value of 0.3 for the Poisson coefficient is the one found online for nylon. Now all the quantities necessary for a successful simulation are defined and it is possible to analyze the running techniques.

4.4. Running

Once the stiffness matrix is created following the procedure shown in section 4.1, the system to solve is

$$K_{LL} \cdot q_L = p_L, \tag{4.10}$$

where p_L is the vector with the loads applied on the free nodes. The system can be solved with an iterative method, such as the Newton-Rhapson algorithm, or with the backslash operator of Matlab. The second choice has been preferred because it is less costly in terms of computational time. The formula to solve is then

$$q_L = K_{LL}^{-1} p_L. \tag{4.11}$$

In order to avoid big instability problem a relaxation routine has been implemented, to damp the peaks and enhance convergence. It is worth noticing that at the first iteration there is no information on the deformed geometry, namely K_G is very small; that is equivalent of saying that the system has a very low stiffness. As a consequence, also the K_{LL} matrix entries are very small and the computed displacements are very high. Therefore, in the second iteration the matrix will have high entries since the deformations will be big. In order to decrease this initial peak quickly a relaxation factor is necessary. Figure 4.9 shows the convergence plot of the norm of the displacement with different damping values. From this figure it is possible to observe that the oscillations are quite symmetric around the convergence value. Therefore the relaxation is imposed so that the displacements at the i^{th} iteration are forced to be an average between the computed displacements at the i^{th} iteration and the ones computed at the $\mathrm{(i\text{-}1)}^{\mathrm{th}}$ iteration. It is important to notice that the relaxation routine starts working only after the 4^{th} iteration.

Regarding the convergence criteria, there are two that need to be applied: one to stop the iterations on the stiffness matrix (nonlinearity 1) and another one to stop the load rotations (nonlinearity 2). Regarding the first criterion, the idea is to stop the iteration when the displacements computed at two consecutive iteration

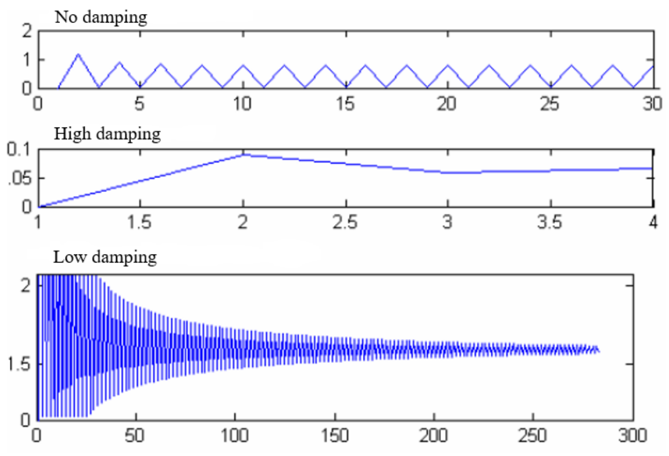


Figure 4.9: Convergence plot with different values of damping

are very similar. A measure of this similarity can be decided arbitrarily, and in this case the condition has been defined as:

$$||q_L(i) - q_L(i-1)|| < \frac{|q_L(i)|}{1000}. \tag{4.12}$$

That is equivalent to saying that the two consecutive displacements should not differ of more than 0.1%.

The second convergence criterion, that stops the iterations on the load rotation (outer loop), is conceptually different. At every iteration the deformed configuration is computed, and it satisfies the equilibrium given the load. Once that is converged (nonlinearity 1), the load is rotated taking into consideration the new geometry and the iteration on the K matrix starts again. When thinking about the equilibrium configuration, it would be safe to say that the loads would not rotate anymore because the geometry is not varying anymore and the solution is converged. Even though the concept behind this criterion is different, the condition will be expressed in the same way, see equation 4.12.

The computation starts then with the undeformed geometry. Once convergence is reached, the loads are rotated. The first displacement guess vector is updated with the new displacement that made the computation converge in the previous iteration. As a consequence of this, also the geometric stiffness part of the K_{LL} matrix is updated with the computed displacements. The loads are rotated again until also the second convergence criterion is satisfied. The maximum number of iterations on the stiffness matrix is set to 100, but usually the simulations converge after 15-17 iterations.

Once convergence is reached, the displacements are available, but it still necessary to use the shape functions to obtain the deformations and stresses. That is done by a function called `stress.m`. The routine starts with the definition of the length of the sides of the element as well as the amplitudes of their internal angles. Then, using a bidimensional coordinate system on the element plane, the origin is placed on one of the nodes and one of the axes is parallel to the element edge. The coordinate of the neighboring node in this coordinate system will then be (L,0). The coordinates of the third node will then be obtained as functions of L and the triangle angles. The situation is depicted in figure 4.10.

The relations will then be:

$$x_0 \cdot tan\theta_2 = (L_3 - x_0) \cdot tan\theta_1^* = -(L_3 - x_0) \cdot tan\theta_1 \quad \Rightarrow \quad x_0 = \frac{tan\theta_1 \cdot L_3}{tan\theta_1 - tan\theta_2} \qquad y_0 = tan\theta_2 \cdot x_0. \tag{4.13}$$

Now that the node coordinates are known in this coordinate system, in deformed and undeformed configurations, immediately it is possible to obtain the nodal displacement vector:

$$a = [0 \quad 0 \quad L_3 - L_{03} \quad 0 \quad x_{0def} - x_0 \quad y_{0def} - y_0].$$

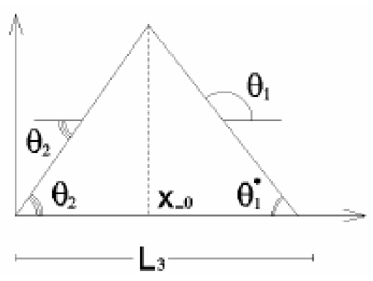


Figure 4.10: Element in the local coordinate system, from [35]

Remembering then equation 4.14:

$$\varepsilon = Ba^e = \begin{bmatrix} B_i & B_j & B_k \end{bmatrix} \begin{bmatrix} a_i \\ a_j \\ a_k \end{bmatrix}, \tag{4.14}$$

the deformation ε can be computed. In order to compute the stresses the deformation vector will have to be multiplied by the compliance matrix C:

$$\sigma = \begin{bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{bmatrix} = C \cdot \varepsilon = \frac{E}{1-\nu^2} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix} \cdot \begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{bmatrix}. \tag{4.15}$$

By diagonalizing the matrix it is be possible to obtain the principal stresses components in the bidimensional coordinate system. That is done with the following command lines:

```
epsilon1=[epsi(1,1) epsi(3,1)
epsi(3,1) epsi(2,1)];
sigma=E/(1-ni^2)*C2*epsilon;
[D]=eig(sigma);
sigma1=(D(1,1)^2+D(2,1)^2)^0.5;
```

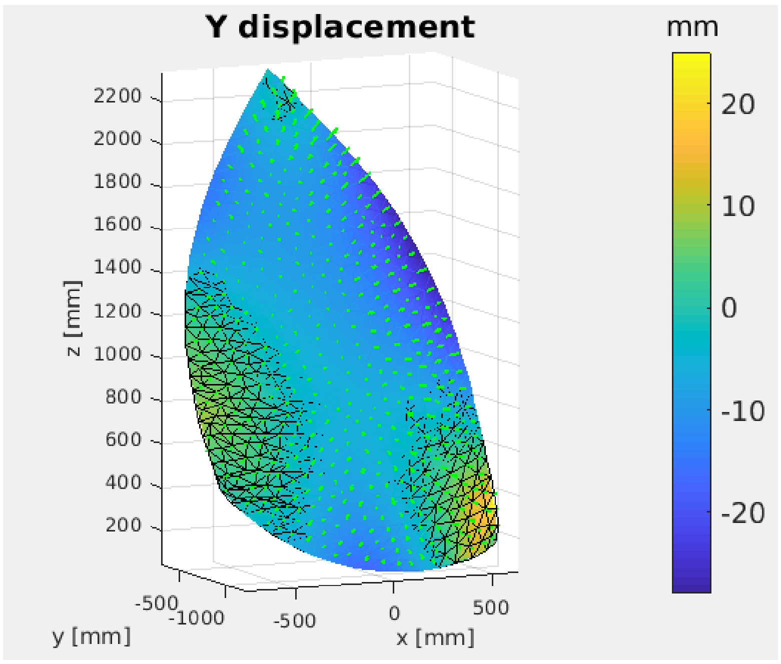
Once the stress on the element is known it should be possible to compute the stress in the nodes by averaging over the neighboring elements (with the information given by matrix `nodel`). The code is not able to compute the directions of the principal stresses.

4.5. Post-processing

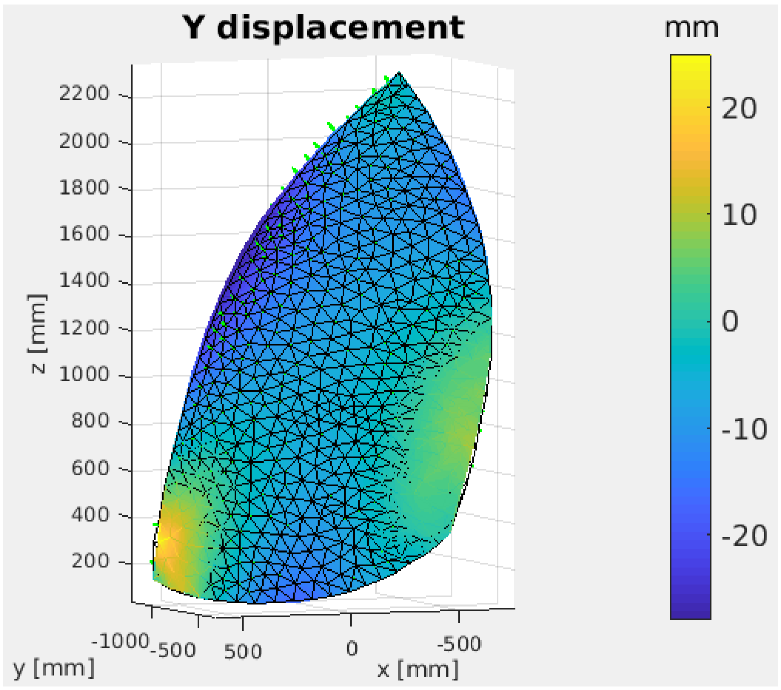
Once the computation is finished and converged it is handy to be able to represent the results in order to assess their validity. As said previously, there is no validation data for the deformations and therefore the evaluation will have to be quantitative. In Matlab the best way to represent meshes made of triangular elements is the command `trimesh`, for which the connectivity matrix, the three vectors of coordinates and some scalar quantity to be represented are required. It will be then possible to represent displacement in x, y and z direction, as well as the stress. For this case the biggest change in geometry happens in the y direction, and therefore the y displacement will be plotted. The command `quiver` allows to represent the nodal loads in form of vectors, so in the final plot there will be: the initial geometry, shown in black, with at each node a small arrow representing the applied load, and the deformed geometry, colored following the magnitude of the displacement. An example for this figure is shown in figure 4.11.

Figure 4.11 is obtained with the following commands:

```
quiver3(x,y,z,P1x,P1y,P1z,1,'LineWidth',2,'color','g')
hold on
trimesh(el,x,y,z,'edgecolor','black');
alpha(0) %The undeformed mesh is transparent
```



(a) Suction side



(b) Pressure side

Figure 4.11: Results by SailFEM on the spinnaker

```
hold on
trimesh(el,xs,ys,zs,vn,'FaceColor','interp');
% "vn" are the y displacement values, represented as a colormap
grid on
axis equal
```

It is not possible to directly validate this solution due to the lack of experimental data for this specific case. Therefore the validation will have to be qualitative. First of all, intuitively when looking at the direction of the

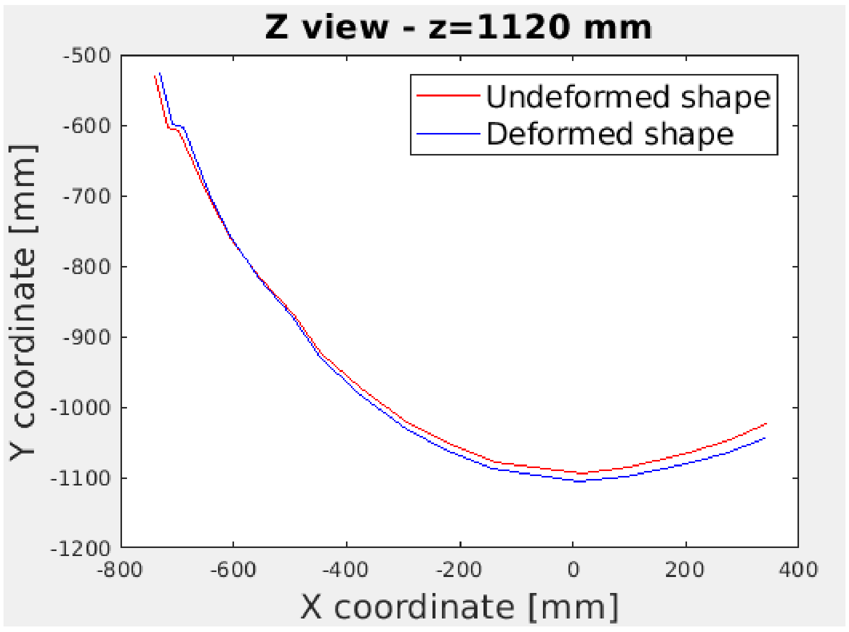


Figure 4.12: Cross section of the undeformed and deformed geometries

loads in figure 4.11a one would expect that the sail deforms towards the suction side. The presence of the black undeformed mesh helps to see in what direction the deformation took place: in figure 4.11b it can be observed that the deformation takes place in the negative y direction for most part of the sail. Obviously, the presence of the clamps does not allow for the whole structure to move in the negative y direction: the points close to the clew and tack vertices (the lower vertices) show displacement in the positive y direction. That can be explained thinking that the material is not elastic and therefore cannot inflate completely towards the suction side as a balloon would do, but given the presence of the clamps at the vertices there is a higher tension on the sides, that results in a deformation towards the pressure sides in these areas. This phenomenon can be observed more closely in figure 4.12, where it is clear how for most part of the sail the deformation is towards the suction side, but at one side it takes place in the opposite direction, given the presence of the clamps and the material stiffness.
This result is meaningful and can be observed in reality, for example in the situation depicted in figure 4.13. The red circle highlights the areas where the sail is deforming towards the pressure side, and the areas are the ones close to the points where the sheets are attached. It can be seen that while the rest of the sail is inflated and well taut, in those areas wrinkling arises, because the cloth tends to move towards the wind direction. In the structural model the wrinkling is not modeled and will not arise, but the displacement in the opposite direction is well captured.

4.6. Conclusions

In this chapter the FEM solver has been presented: the structure of the code and the equations it solves have been reported. In order to confirm that the solver is working correctly some validation tests have been run, on test cases for which the analytic solution is known. The validation was successful for all cases, having an error of maximum 5% for the cylinder case and of 7% for the sphere case. The sphere case used the same material properties as the ones used in the simulations on the spinnaker. Therefore, it can be expected to have a similar range of inaccuracy for the spinnaker case. However, it can be stated that the solver is correctly reproducing the analytic data and it is therefore verified.

Consequently, the strategies adopted to have the solver correctly working are presented: a mesh study has been conducted and the optimal number of cells has been chosen. Once the setup of the solver is done the simulation has been run to find the deformed shape of the spinnaker given the pressure load from CFD. No validation data was available for this specific case and therefore a qualitative comparison has been done with a real-life sail, see figure 4.13. The resemblance of the solution to the real life condition is very satisfactory.

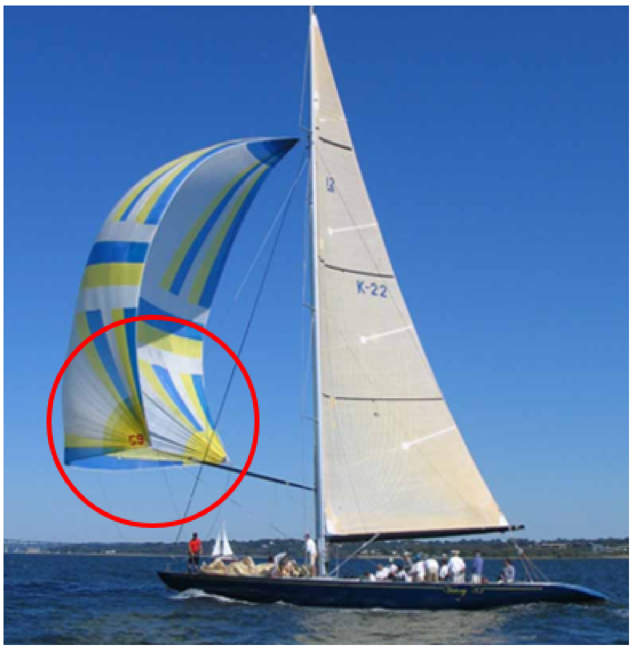


Figure 4.13: Deformation of a real spinnaker sail

5
FSI Simulations

In the previous chapters the simulations for the fluid and structure domains have been analyzed in depth and their validity for this application has been asserted. The final step is to couple the two solvers, obtaining a solution that takes into account the information given by both domains. There are many techniques to perform this coupling and they have been extensively analyzed in the literature study, while the ones chosen to be the best for this application have been presented in detail in chapter 2.3. In this chapter an explanation of how the presented techniques were implemented will be reported, together with an evaluation of their performance.

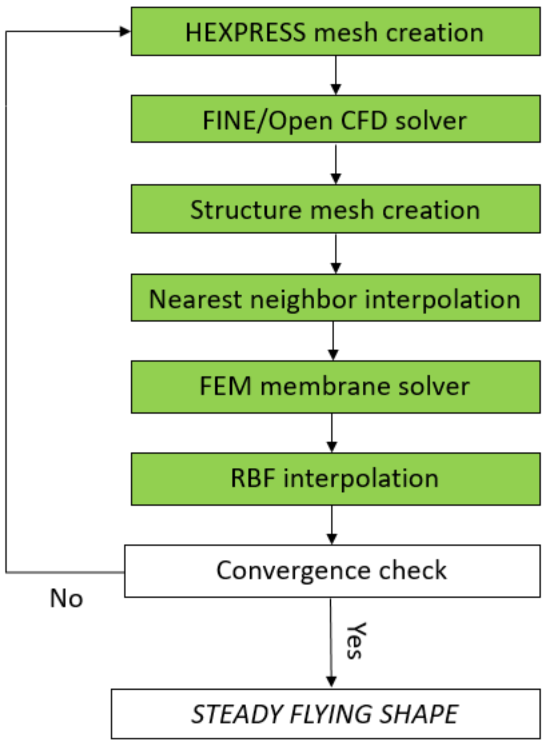


Figure 5.1: Fluid Structure Interaction loop

5.1. FSI Loop

The fluid structure interaction loop is the one summarized in figure 5.1. It is worth mentioning that being a steady simulation, there is no time dependence on the previous solutions within the FSI loop and hence thr remeshing can be performed easily as no projection between meshes is necessary. The CFD solver converges to a steady state solution and that is used as an input for the structure solver until an equilibrium on the sail shape and pressure forces is reached.

The CAD geometry that has been used for the spinnaker is the one obtained through photogrammetry in the wind tunnel by Viola et al. [9]. This represents then the shape of the sail when it is already inflated by the wind, namely the flying shape. In order to have a significant analysis, it is necessary to have an initial shape to start the iterations from, with the final goal of obtaining the flying shape that is given. The quantities in play in this situation are then:

$$x_0 = \text{Initial shape}, \qquad q = \text{Displacement}, \qquad x_f = \text{Flying shape}, \qquad p_f = \text{Flying shape load}.$$

For the flying shape, the following should hold:

$$K_{LL}(x_0, q) \cdot q = p_f \qquad \Rightarrow \qquad \left[K_E(x_0) + K_G(x_0 + q)\right] \cdot q = p_f. \tag{5.1}$$

For this case then the unknowns are two: x_0 and q. Equation 5.1 contains both unknowns and therefore another relation is needed to solve the problem. That relation will then be:

$$x_f = x_0 + q. \tag{5.2}$$

With equations 5.1 and 5.2 the system can be solved and the initial shape can be determined. It will however not be possible to obtain it directly, for this reason an iterative process has been implemented in Matlab. The script works in the following way:

1. Runs the structure solver on the flying shape with the pressure distribution p_f obtained from CFD and obtains a displacement,
2. Subtracts that displacement from the flying shape obtaining the first guess for the initial shape,
3. Reruns structure solver on the obtained initial shape, with the same pressure as point 1,
4. Add the displacement to the initial shape and obtain a new guess for the flying shape,
5. Computes maximum difference between obtained and given flying shape ε,
6. Defines a new guess for the initial shape:

   $$x_0^{k+1} = x_0^k + 0.5 * \varepsilon,$$

   where 0.5 is a relaxation factor and k is the iteration number.
7. Restarts from point 3 with the new initial shape x_o^{k+1}.

The whole process is repeated until a certain condition on the error is satisfied. In this case the minimum possible error found was of 0.2 mm. That represents the maximum difference in coordinates between the given flying shape and the one obtained with the script. Being the thickness of the spinnaker 0.3 mm, the difference can be considered satisfying, since it is still lower than the minimum characteristic length of the problem. Once the initial shape is obtained, it is possible to start the FSI loop as described below.

The loop starts with the generation of the mesh with HEXPRESS on the initial geometry. Then the CFD simulation is run in FINE/Open and with the help of the postprocessor CFview the pressure distribution on the spinnaker is obtained as a `.txt` file. This pressure will have to be interpolated to the structure mesh, so before performing the interpolation it is necessary to create the structure mesh. That will be done using Gmsh, as explained in chapter 4.3. Once the structure mesh is ready the interpolation of the pressure can be performed. That is done in Matlab by a script that performs the nearest neighbor interpolation from the fine fluid mesh to the coarse structure one. Once the pressure is defined on the structure nodes the FEM solver can be run until convergence. The resulting displacement field is then interpolated again to the fluid domain, by a script that uses RBF functions to modify the fluid domain file. Then a convergence check is performed: the monitored value is the maximum displacement obtained in the structure solver. At each iteration k the following check is performed:

$$max(q_k) - max(q_{k-1}) < \frac{max(q_k)}{250},$$

where maximum displacement is compared to the one obtained in the previous operation. In the next sections the interpolation operations will be analyzed and some intermediate results will be shown.

5.2. Interpolating pressure

The step of interpolating the pressure distribution from the fluid to the structure mesh is fundamental in order to run the FEM solver with the information obtained from the CFD solver. First of all it is necessary to obtain a file that contains the values of the pressure difference across the sail for every fluid mesh node. From CFView it is possible to obtain separate files with the pressure defined at the nodes for the front and back of the sail. Unfortunately the mesh points do not coincide and therefore another interpolation will have to be done in order to perform the subtraction of the pressures. This will be done with a Matlab function named `griddata`, that simply interpolates the pressure values from the pressure side to the suction side of the sail. Now the file with the correct pressure to interpolate is ready and the transfer to the structure mesh can be performed.

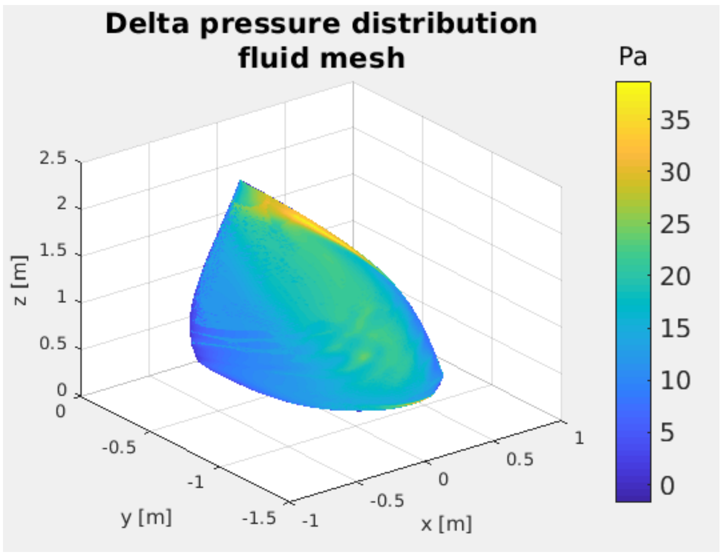


Figure 5.2: Delta pressure distribution on the fluid mesh

The (delta) pressure distribution is the one shown in figure 5.2: it is clear that the biggest pressure variations happen in the part of the sail close to its leading edge, where the suction peak is present. When interpolating it will be important to make sure that the pressure distribution is well captured, so that the pressure imposed on the structure can be as similar as possible as the one computed with CFD.

The theory behind the nearest neighbor interpolation has already been presented in chapter 2.3, so it will not be repeated here. The script that has been written in Matlab reads the two files with the lists of nodes, and then starts a loop over all the structure nodes. For each node it computes the distance with all the fluid nodes and then finds the minimum distance. In correspondence of the index of the closest node a 1 will be put in the H matrix, and all the rest of the row corresponding to the structure node will be 0. Doing so the matrix H is created: this sparse matrix has size $ns \times nf$, where ns is the number of structure points and nf the number of fluid points. For each row then, corresponding to a structure point, there will be only one nonzero entry corresponding to the closest fluid node to that structure node. When this matrix is built the pressure can be easily interpolated by performing the operation

$$p_s = H \cdot p_f.$$

The resulting pressure distribution on the structure mesh is the one shown in figure 5.3. There are some differences that can be observed visually: the biggest difference is observed at the sail's free sides, there the pressure in the structure is zero while in the fluid it is not. This error is due to the fact that there are some very small cells at the sides of the spinnaker in the fluid mesh in which the pressure is the same in the back and front of the sail, due to the stagnation point, resulting in a zero Δp. They cannot be seen in figure 5.2, because their size is really low, but a zoom of the same figure is reported in figure 5.4. In the figure 5.4 it is possible to visualize these elements at the side of the sail as the darker line on the side of the sail. These cells where the pressure is zero or very low are the ones that result to be closer to the structure elements laying on the spinnaker's free sides, and for this reason the behavior observed in figure 5.3 shows, for which in all free sides there is zero pressure. This error could be minimized by refining the structure mesh on the free sides, but it would still not eliminate the error. Another strategy would be to change interpolation method, for example a radial basis function (RBF) interpolation, that would make sure to take into consideration also other neighboring cells. These strategies have not been adopted because the structure solver works well only with a uniform mesh and because the RBF interpolation was too computationally costly (the fluid mesh has 120000 points and for RBF a full 120000x120000 matrix would have had to be constructed). Therefore, it is possible that some dissimilarities will be found in the final solution due to this interpolation error.

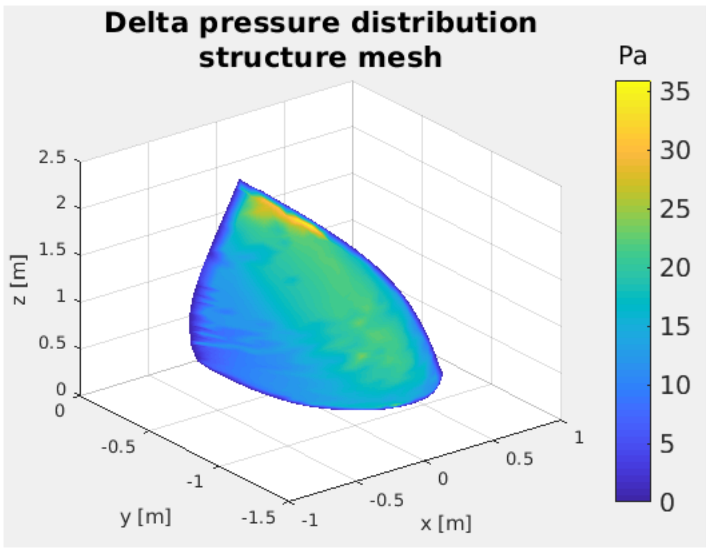


Figure 5.3: Delta pressure distribution on the structure mesh at the first iteration

Another analysis has been conducted to assure that the interpolation has been performed correctly: the total force on the sail has been computed for both meshes and the results compared. In order to compute the force the following operations have been performed on both meshes:

- For each element, the pressure in the center of the element is computed by averaging the pressures of the 3 nodes that compose the element,
- For each element, the area of the element is computed using Heron's formula, where a, b and c are the element edges lengths, previously computed from the node coordinates:

$$S = \frac{a+b+c}{2} \qquad \Rightarrow \qquad Area = \sqrt{S(S-a)(S-b)(S-c)},$$

- For each element the force is computed by multiplying the pressure by the area.

Once the forces in all the elements are defined it is possible to add them all to find the total force. The resulting total force for the fluid is 46.6 N, while for the structure 43.02 N, resulting in a relative error of 7%, which can lead to some dissimilarities in the final result. The new interpolated distribution of pressure can then be used as an input for the structure solver.

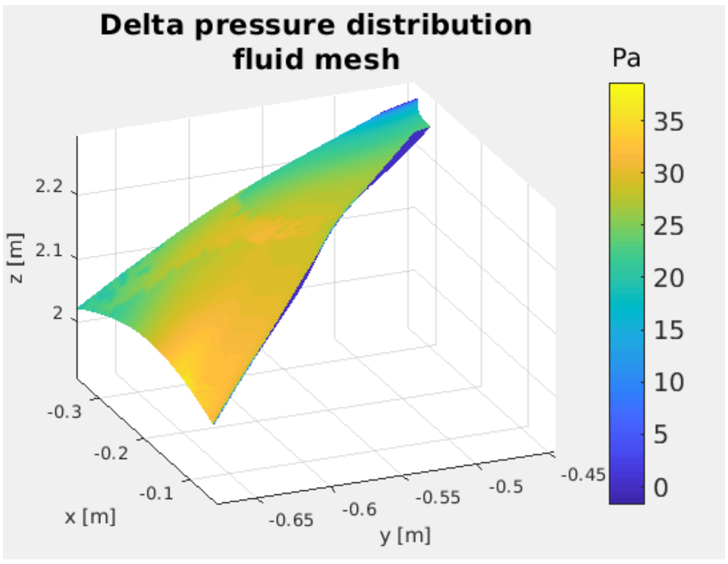


Figure 5.4: Zoom of figure 5.2 on the spinnaker leading edge

5.3. Interpolating Displacement

The interpolation of the displacement is a slightly more delicate matter because in this case the passage is from the coarse structure mesh to the fine fluid mesh. For this reason, an approach of the nearest neighbor type would lower very much the accuracy of the process, since it would assign the same value of displacement to more adjacent fluid cells, resulting in a step like behavior in the displacement distribution. In order to avoid this phenomenon the Radial Basis Function approach has been used, a method that guarantees a more gradual interpolation of the displacement, considering for each node the influence of many cells, each one with its appropriate weight. The theory of the radial basis function technique for interpolation has been already reported in chapter 2.3, so here attention will only be given to the implementation of the method.

The problem to solve is then the following: the nodal displacements have been obtained by the structure solver on the structure mesh, and they have to be transferred to the fluid mesh with a minimum loss of accuracy. The implementation of the RBF method would be quite simple to implement in Matlab in matrix form, as explained in chapter 2.3, but unfortunately here it was not possible because the number of points on which the distances have to be computed to create the matrix Φ_{BA} was too big and the software ran out of memory. An alternative strategy was then required, for which the creation of such a big matrix would not be necessary.

The idea behind this alternative method is based on applying the RBF interpolation point by point, without using the matrix formulation presented in chapter 2.3. That is done working directly on the file describing the fluid domain. Said file contains information about the topological vertices, the curves and the surfaces of the domain. The idea is to modify only the part of this file that is relative to the spinnaker, applying the RBF interpolation to the points in the file and modifying them with the found result. The rest of the file can remain untouched. Information regarding the spinnaker is present in the part regarding the curves and the surfaces. The first step is to identify exactly what lines describe the spinnaker and what do not. That has been achieved by manually checking the file and comparing the coordinates with the graphical representation of the domain in HEXPRESS. A set of lines that were relevant was then obtained.

The RBF script then works like this: first of all the matrix Φ_{AA} is created, which only depends on the structure mesh coordinates. With that the matrix M can be created, defined as

$$M = \begin{bmatrix} \Phi_{AA} & Q_A \\ Q_A^T & 0 \end{bmatrix}.$$

Consequently the coefficients β and γ can be computed, by inverting the matrix M and multiplying by the known displacement values on the structure nodes. Once these coefficients are known it will be possible to modify each set of coordinates with a simple expression (reported below) and without matrix operations. In appendix A a detailed explanation on how the parts relative to the spinnaker have been found in the domain file is reported. Knowing the number of lines it has to ignore and the ones it has to modify, the code applies two different operations depending on the line number it is dealing with: if the line is not relative to the spinnaker it simply reads the line and copies it as it is to a new file, while if the line is relative to the spinnaker it applies the RBF interpolation to the coordinates reported in that line and then writes the new coordinates to the new domain file. The lines of code describing this operation are the following:

```
a=fgetl(fid);
b=str2num(a);
ind=b(1); X=b(2); Y=b(3); Z=b(4);
coords = [X Y Z];
r=1;
qx = 0; qy = 0; qz = 0;
for i=1:ns
    if (1-norm(coords-coords_s(i,:))/r)>0
        phi=(1-norm(coords-coords_s(i,:))/r)^4*(4*(norm(coords-coords_s(i,:)))/r+1);
    else
        phi=0;
    end
    qx = qx + Gammax(i)*phi;
    qy = qy + Gammay(i)*phi;
    qz = qz + Gammaz(i)*phi;
end
qx = qx + Bx(1) + Bx(2)*X + Bx(3)*Y + Bx(4)*Z;
qy = qy + By(1) + By(2)*X + By(3)*Y + By(4)*Z;
qz = qz + Bz(1) + Bz(2)*X + Bz(3)*Y + Bz(4)*Z;

X1=X+qx;
Y1=Y+qy;
Z1=Z+qz;

fprintf(f2,'%d␣%.13f␣%.13f␣%.13f␣\n',ind,X1,Y1,Z1);
```

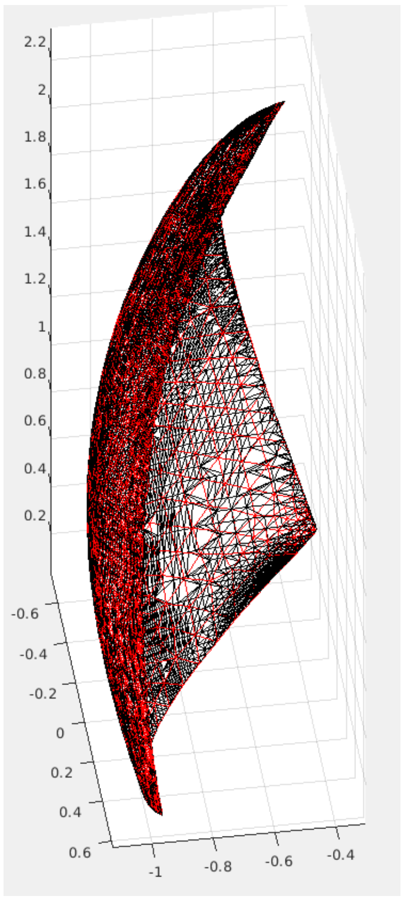
The value of the compact radius has been chosen to be 1 m, being the sail roughly 2.5 m high and 2 m long.

The script works successfully and is also relatively fast. The result is a new domain file with the modified coordinates, that if imported in HEXPRESS describes a spinnaker geometry that is deformed following the results from the structure solver. In figure 5.5b the deformed and undeformed domains have been superimposed in HEXPRESS to show the differences. The blue line represents the deformed spinnaker shape while the red one the undeformed one. It is worth noting that also the hull and mainsail are superimposed in this figure, but being them the same the differences can not be noted.

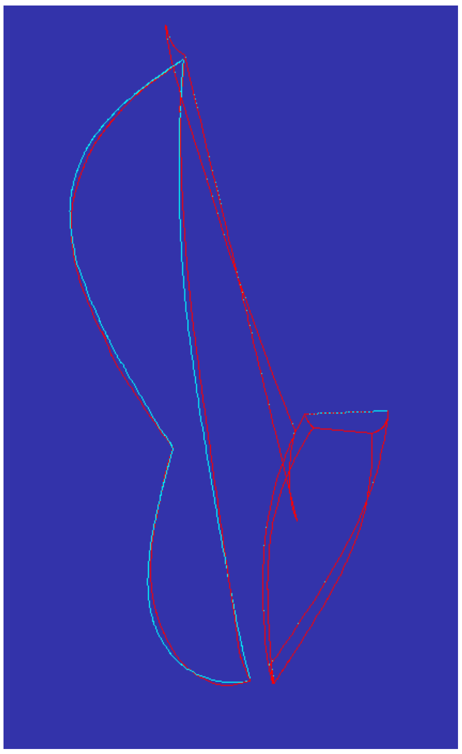
In order to assess if the interpolation went right another image has been created, representing the deformed structure mesh together with the deformed fluid one. The result is reported in figure 5.5a, where the red lines represent the structure while the black ones represent the fluid mesh. The chosen view allows to check if the displacement of internal points has been represented correctly, as well as the one of the points laying on the spinnaker edges. Being the two meshes completely overlapping it is safe to say that the interpolation went well. The author could not think of other ways of assessing the validity of the interpolation for the displacements.

5.4. Conclusions

In this chapter the scripts that perform the transmission of information across non-matching boundaries have been presented. For the interpolation of the pressure from the fluid to the structure mesh the nearest neighbor interpolation has been used, while the radial basis function has been chosen to interpolate the displacements. The nearest neighbor interpolation is very fast but not so accurate, in fact the difference in total



(a) Deformed structure (red) and fluid (black) meshes, before and after RBF interpolation



(b) Undeformed (red) and deformed (blue) sail representation in HEXPRESS

Figure 5.5: Two visual ways of checking the success of the RBF method

force on the sail between the fluid and the structure is of approximately 7%. That is due to the fact that the fluid mesh is very refined at the sail free sides while the structure mesh is coarse everywhere. For that reason in areas where there are big changes of pressure (near the leading edge) some structure cells can acquire incorrect pressure values. This error will be taken into account when assessing the final results.
The transmission of the displacement from the structure to the fluid mesh is done through the RBF interpolation directly on the coordinates of the fluid domain file. The lines of the domain file regarding the spinnaker are recognized and the interpolation is applied to each point reported in the file. The script is not as fast as the nearest neighbor one but it gives satisfactory results in terms of accuracy of the displacement. It is expected that possible errors will depend on the interpolation of the pressure and not of the displacements.

6
Results

In this chapter the results of the Fluid structure interaction cycle will be shown for two testcases: the first one uses the geometry given by Viola [9], of which the flying shape is available and the initial shape has been computed with the algorithm presented in section 5.1. The second one is done using the initial shape of the spinnaker provided by Daniele Trimarchi and used in his thesis [35]. For this testcase the flying shape is not available and therefore only qualitative conclusions will be drawn.

6.1. Viola Testcase

The results will be shown in terms of displacement of the sail with respect to the initial shape. Then, the obtained flying shape will be compared with the reference one, showing differences in terms of coordinates in a contour plot. The last check will consist in comparing the CFD pressure distributions on the reference and computed flying shape.

First of all, the maximum displacement over the FSI iterations has been reported. This plot has been used

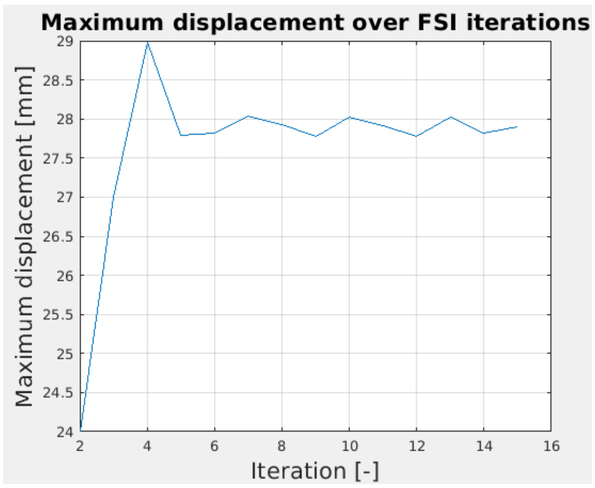


Figure 6.1: Maximum displacement over FSI iterations

to monitor the convergence iteration by iteration: the maximum displacement has been plotted for each iteration and the result is shown in figure 6.1. The criterion used to stop the iteration was the following:

$$max(q_i) - max(q_{i-1}) < \frac{max(q_i)}{250},$$

where q is the displacement at the iteration i or $i-1$. That is equivalent to saying that the two consecutive displacements should not differ of more than 2.5%. Convergence following this criterion was reached after 15 iterations. No relaxation techniques were used in the FSI cycle since it seemed like the maximum displacement was already converging at the first iterations. The value of maximum displacement to which the iterations converged is 27.9 mm, while for the reference testcase the value is 28.26 mm, resulting in a difference of 0.36 mm, that corresponds to an error of 1%. This value can be considered widely satisfying, especially when thinking that the imposed sail thickness is of 0.3 mm: the error is of the same magnitude of the sail thickness, and that can be considered as an upper limit for the obtained error. Moreover, when comparing the global displacements with respect to the initial shapes with the formula

$$q_{GLOB} = \sqrt{q_x^2 + q_y^2 + q_z^2}, \tag{6.1}$$

for the reference case a value of 309 mm is obtained, while for the computed case the global displacement is of 300 mm, resulting in a relative error of 2%, which can also be considered more than satisfying.

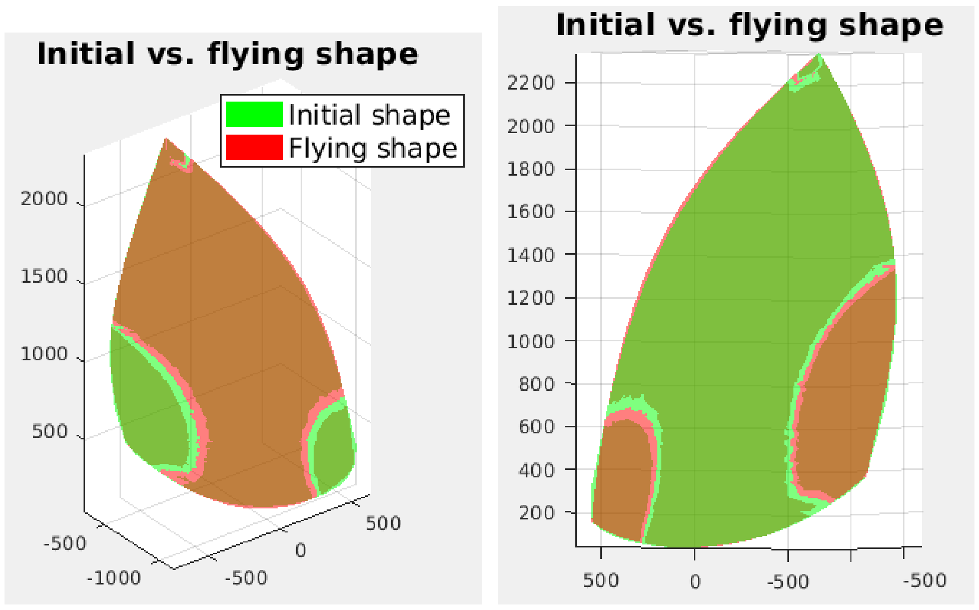
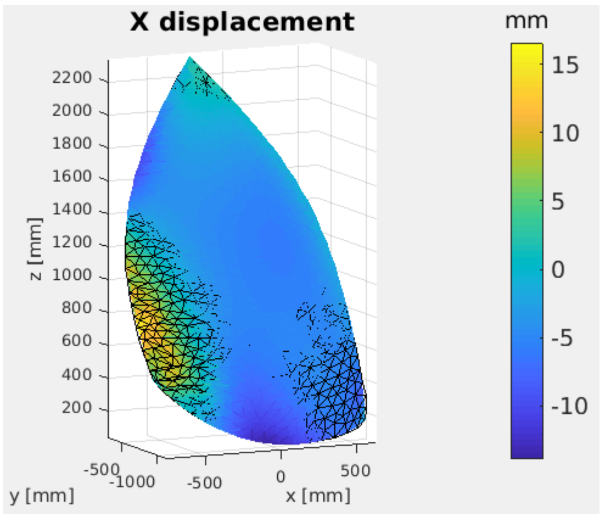


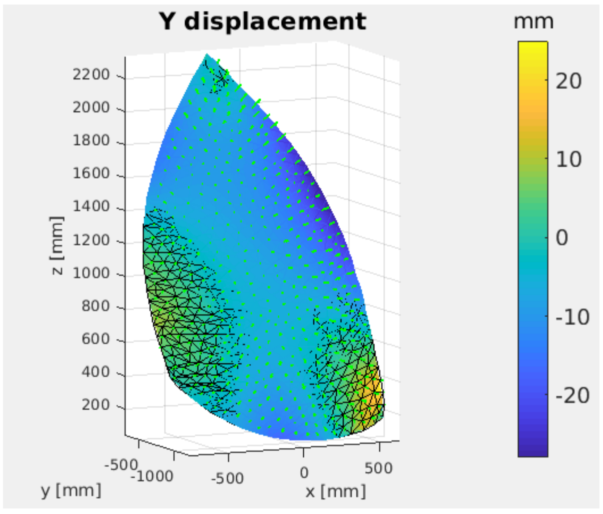
Figure 6.2: Initial shape and computed flying shape

The next figure 6.2 shows the initial and final shapes of the spinnaker. The red sail represents the flying shape and the green one the initial shape. The differences are quite evident: it is possible to notice that the flying shape is moving in its center towards its suction side, while in the points close to the corners it is deformed towards the pressure side due to the presence of the clamps, as it has been already shown in section 4.5. Figure 6.3 shows the displacements in the sail in the x, y and z direction. As it has been observed already in chapter 4, the main displacement takes place in the negative y direction, namely the direction towards which the sail inflates. In the figures 6.3 it is possible to observe how inflating the sail towards the suction side results also in a displacement in the z direction, bringing the nodes in the upper part of the sail higher and the ones in the lower part lower. Considering the displacement in x direction, it is possible to observe that the maximum values appear in the areas close to the bottom corners of the sail, where the presence of the clamps and the material stiffness forces the sail to deform towards the pressure side, resulting also in a movement of the sail in its longitudinal direction. Being them sucked towards the central part of the sail, it is possible to observe a more or less symmetrical behavior on both corners being the displacement negative on the tack side (upwind point) and positive on the clew side (downwind point).

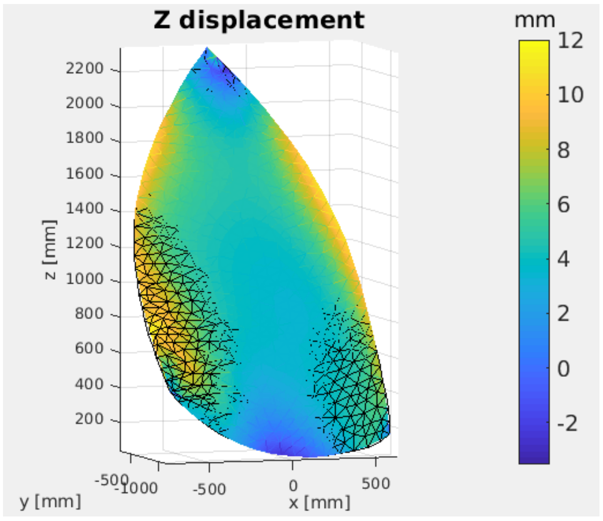
Now, the computed flying shape is compared with the flying shape given by Viola et al. [9]. In order to make this comparison, first of all the two shapes have been plotted in the same graph and visually they look like they are overlapping, see figure 6.4. In the figure only the edge of the cells are plotted in order to be able to visualize the points where there might be some differences, but from this plot it seems very much like the edges are overlapping in the whole geometry. It might not be very clear from figure 6.4, but when interactively rotating the figure it is possible to see that the color of the edge is interchangeable between red and green,



(a) Displacement in x



(b) Displacement in y



(c) Displacement in z

Figure 6.3: Displacement contour plots in the three directions

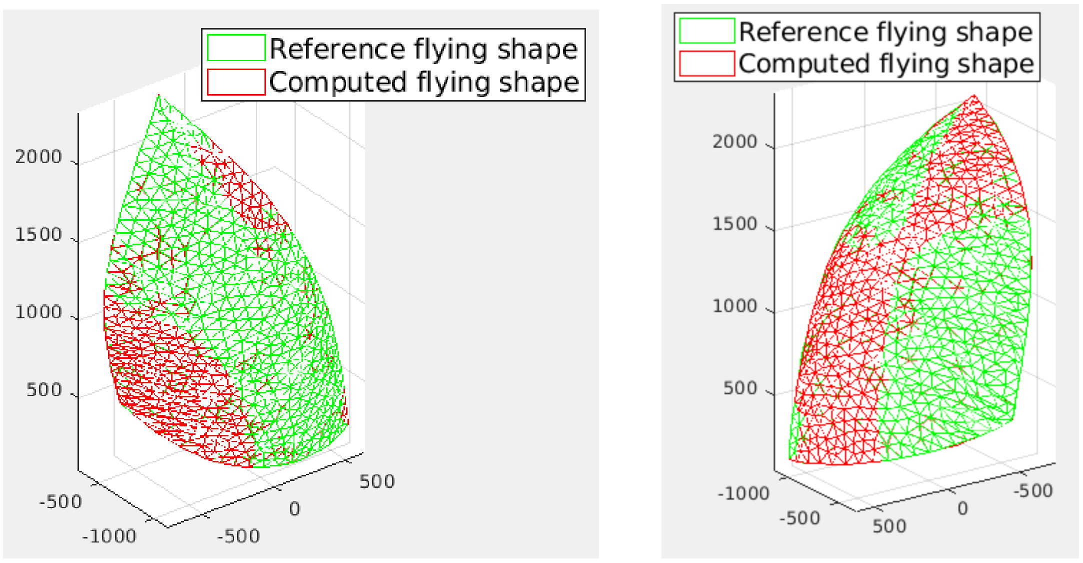


Figure 6.4: Reference (green) and computed (red) shapes of the spinnaker

depending on the chosen view angle.

But evaluating this difference only visually is not enough, and therefore another plot is reported, for which the maximum difference between the y coordinate of the reference and the computed flying shape is plotted at each iteration. The result is shown in figure 6.5. In order for the two shapes to be identical the plotted value should tend to zero, while it is observed that here it stabilizes to a value around 2.8 mm. That means that the converged shape is not exactly the same as the reference one but the maximum differences are of more or less 3 mm, which represents an error of 0.2% on the coordinate value.

Another check that can be performed is the comparison of the total force generated by the reference and computed shape. By integrating the pressure value on each element the total force obtained on the computed flying shape is of 46.6 N, while for the computed one it is slightly lower, namely 42.6 N, resulting in an error of 8%. It can be argued that the simulation converged to a equilibrium shape that is slightly different from the reference one, and consequently it provides a different force. In the next paragraph a more detailed comparison between the two shapes is reported, together with a possible explanation of the differences.

When wanting to quantify the differences between the two shapes, the following operation is made: the difference between the x, y and z coordinates of the two shapes is computed and plotted on the sail, in order to assess which ones are the areas where the error is larger. The resulting plots are shown in figure 6.6. Looking at these figures, first of all it is clear that most of the error is created in the y direction, the one where the biggest displacements take place. For the z error it is possible to see that the error is zero in most part of the sail, however there are some critical points where the error is not zero that can be seen in all three figures: the area close to the bottom left corner and the area in the center of the bottom edge. These are most critical areas indeed because two phenomena are happening simultaneously: the sail is being inflated in its center part and also being brought back upwind close to the corner points, due to the presence of the corner points modeled as clamps, as it has been explained previously. Those areas are where the biggest error is present and also where the highest displacement happens, see figures 6.3a and 6.3b. Another possible explanation for the fact that the error is higher in the areas close to the sail's free sides can be found by looking at the interpolated pressure difference distribution on the structure mesh, shown in figure 5.3. It is clear that close to the free edges the pressure difference is often zero, therefore there is no force applied on the free sides element. That can result in an unpredictable behavior, as a small variation in displacement of the neighboring cells can result in a different behavior in the unloaded cells at each iteration. It is worth mentioning that the fact that the forcing is zero on these elements does not correctly represent reality, in fact the pressure is not zero on the fluid mesh (see fig. 5.2). The error is due to the interpolation process, as it has been extensively

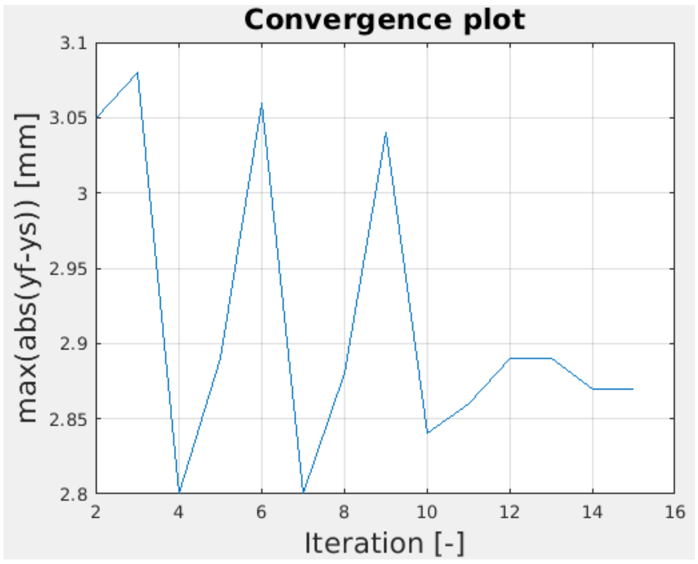
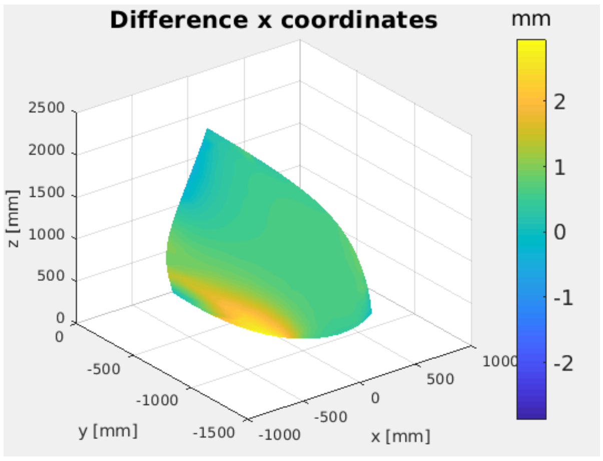


Figure 6.5: Convergence plot, yf is the reference and ys the computed coordinate

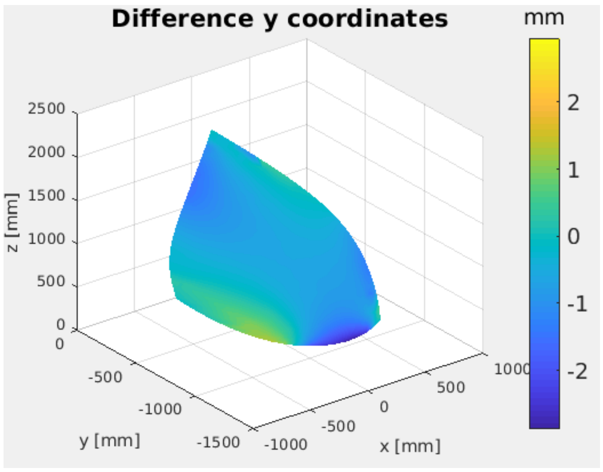
explained in section 5.2.

Another comparison is done by plotting the displacement at the 5 horizontal sections for both reference and computed sail shape and evaluating their differences, as shown in figure 6.7. Being the y direction the one where the highest errors were recorded (see fig. 6.6) it will be the only one plotted in the following figures. The plots for the displacements in x and y are shown in appendix B. In order to obtain the displacement distributions at certain sections a script that identifies the coordinates close to the wanted sections has been implemented. This script creates a straight line at the height of the wanted z coordinate and then uses nearest neighbor interpolation in order to identify which points of the sail to consider. This is done for 5 horizontal sections and the result is shown in figure 6.7. What is evident from these figures is that the computed flying shape has an offset with respect to the reference one but follows the tendencies in the displacement very well. The fact that the converged shape is not the same as the reference one has already been established, and with these plots it is confirmed. The extra information obtainable from these plots is that the obtained shape follows very well the behavior of the reference shape, and the errors observed in figure 6.6 represent an offset and not a change in behavior. Moreover, there is agreement between the figures 6.7 and 6.6: in 6.7 it can be seen how the central section is the one where the error is lowest, and the two lines are basically overlapping. The central area is in fact the one where the error in the coordinates results zero in figure 6.6. The lower and upper areas however show some differences in coordinates and therefore a difference in the displacement distribution can be expected. The positive aspect is that the difference in displacement is never higher than 1.62 mm, resulting in a maximum error of 4%, for the section 1/8. It can be concluded that even though the converged shape is not the same as the reference one, it is very similar and shows the same types of behaviors.

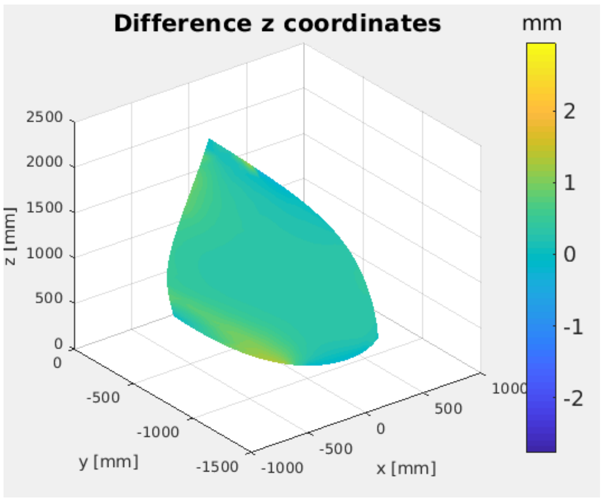
The last check to be performed in order to assess the validity of the solution is to compare the pressure distributions on the two flying shapes. The result of that operation is shown in figure 6.8. This figures confirms the statements obtained in the rest of the chapter: the converged shape is not the same as the flying shape, in fact there are differences in the pressure distributions. Once again, the biggest differences are observed at the lower and higher sections, where the differences in the geometries are. It seems that as a general tendency the computed shape reports a lower pressure in all sections, also explaining the underestimation of the total force mentioned before. Just like the displacement distribution, it can be observed that the tendencies are the same, and the error lays in the offset. The explanation for this offset can lay in the fact that the FSI problem has multiple solutions, and the obtained one is one of them, as valid as the reference one.
The reason why the cycle converges to a different solution can be attributed to the fact that when interpolating the displacement from the structure to the fluid mesh some information is lost: in fact, the structure mesh has 580 points while the fluid domain file contains approximately 100000 points. It might be that when the displacement is interpolated the obtained shape is not exactly the same as the one computed by the struc-



(a) Difference in x

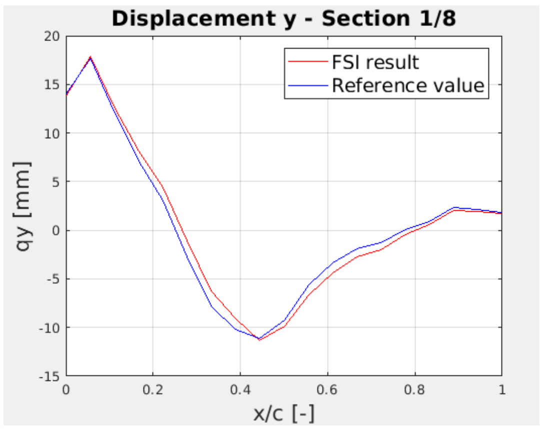


(b) Difference in y

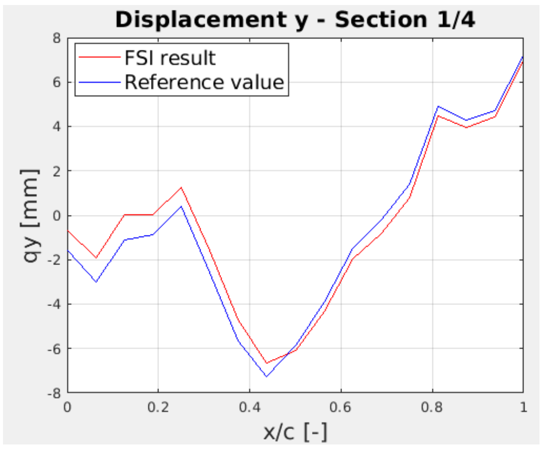


(c) Difference in z

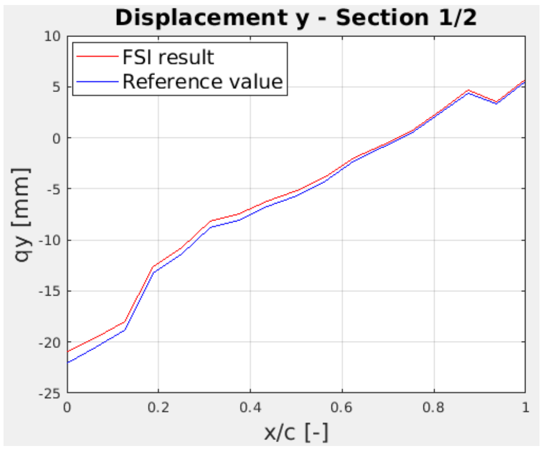
Figure 6.6: Difference in coordinates between reference and obtained flying shapes



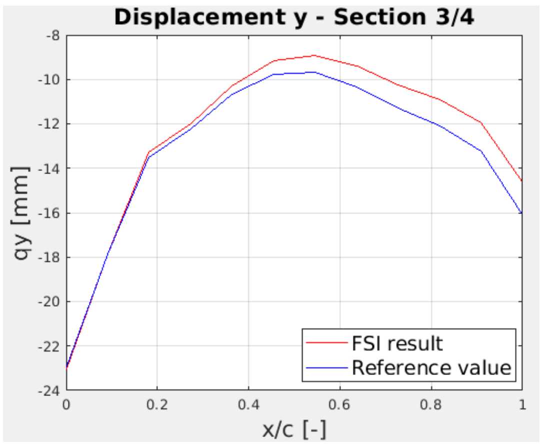
(a) Section 1/8



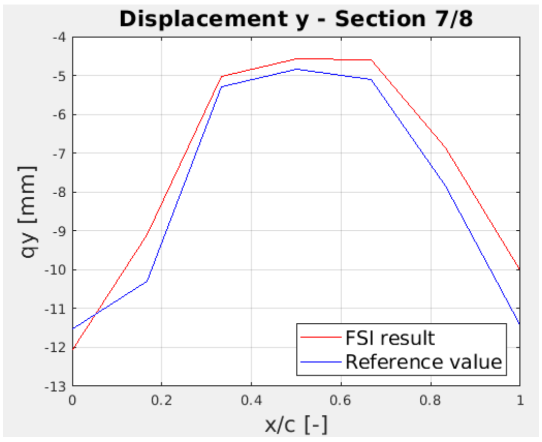
(b) Section 1/4



(c) Section 1/2



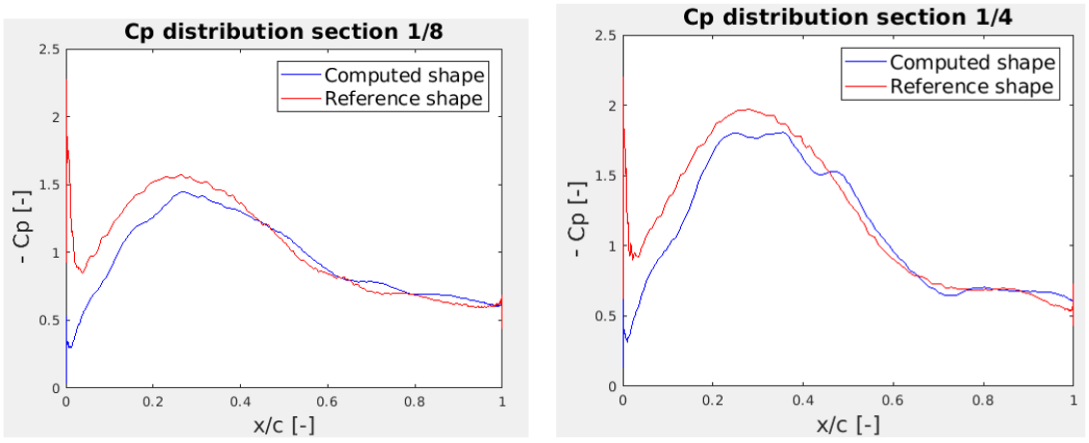
(d) Section 3/4



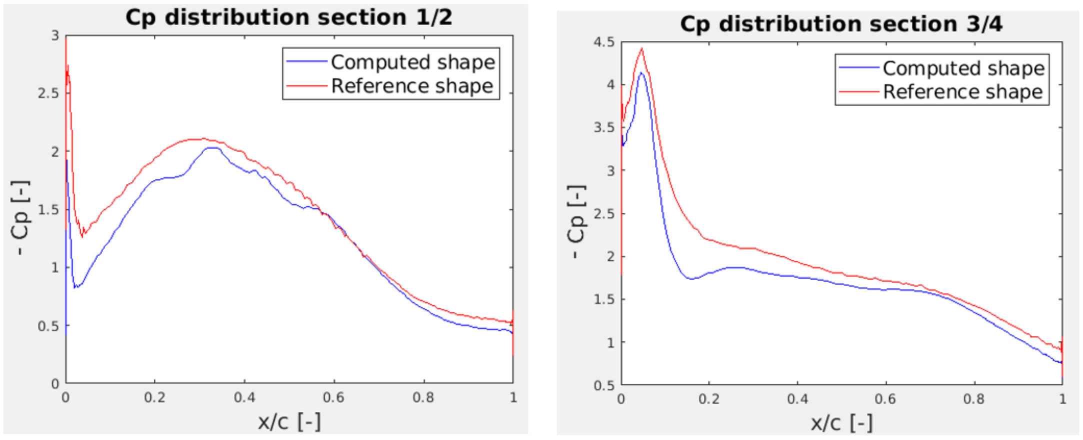
(e) Section 7/8

Figure 6.7: Computed and reference y displacement at 5 horizontal sections

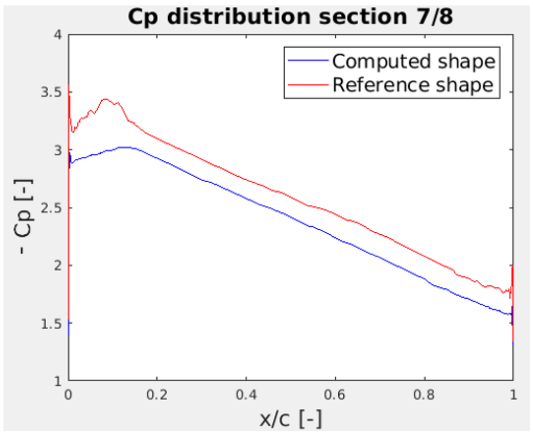
ture solver, and that yields a difference in the CFD results that are then used a starting point for another FEM simulation. In this way the error propagates and creates a solution that differs from the reference one. However, it is possible to conclude that the solver is working correctly and can reproduce a given testcase with an overall error of around 8%, which could be considered acceptable given the complexity of the project.



(a) Section 1/8 (b) Section 1/4



(c) Section 1/2 (d) Section 3/4



(e) Section 7/8

Figure 6.8: Computed and reference pressure distribution at 5 horizontal sections

6.2. Trimarchi Testcase

The second testcase is based on Daniele Trimarchi's masters thesis [35]. Trimarchi used a spinnaker design shape created by using a portion of a sphere, see figure 6.9. This shape does not resemble the one of the inflated spinnaker. The relevance of this testcase is that indeed here only the design shape is known, so it can be expected to find some big differences between initial and flying shape. That could not be observed in the previous testcase, for which only the final shape was available. The design shape is 6 m high and 3.75 m wide.

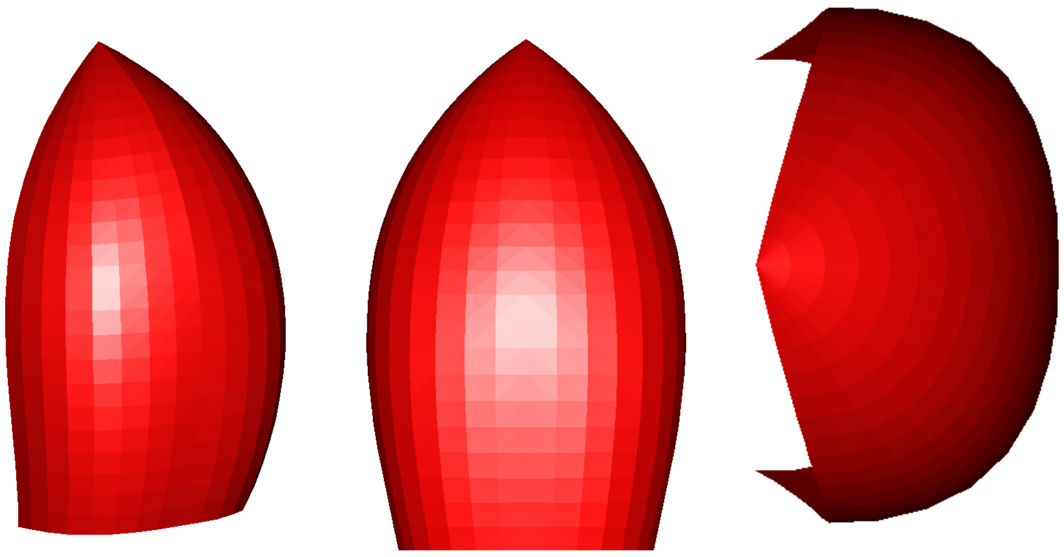


Figure 6.9: Spinnaker design shape

In Trimarchi's work, he applied to the spinnaker a constant pressure equivalent to the aerodynamic load of 15 knots of wind, corresponding to a wind velocity of 7.71 m/s. The goal of this work is to show the differences in flying shape when applying a constant pressure or an interpolated FSI pressure. First of all, a CFD simulation has been run on the design shape and the equivalent constant pressure has been computed by dividing the total force acting on the spinnaker by the total area. The total force is obtained by summing the contribution of each cell, obtained by multiplying the element area by the pressure value. The obtained values are:

$$F_{tot} = 2143N, \qquad A_{tot} = 30m, \qquad p = \frac{2143N}{30m} = 71.43Pa = 7.143 \times 10^{-5}\frac{N}{mm^2}. \tag{6.2}$$

The constant value in $\frac{N}{mm^2}$ is then used as constant loading on SailFEM. A difference can already be observed with Trimarchi's work, for which the constant pressure equivalent to a wind velocity of 15 knots is $3 \times 10^{-4}\frac{N}{mm^2}$. The author can not explain this difference of 1 order of magnitude, and suspects that Trimarchi might have had some typing error because a pressure in the order 10^{-5} is correct in this case. Anyways, the obtained pressure value is applied in SailFEM and a first flying shape is obtained.

The other strategy is then to interpolate the pressure information obtained in CFD using the scripts described before, applying then a non-constant pressure on the sail and running the structure solver. The interpolated pressure on the structure mesh looks like the one reported in figure 6.10. Similarly to the other testcase, also here the areas of zero pressure are present at the sides of the sail: their presence is due to the nearest neighbor interpolation routine for which the values at the boundary of the fluid mesh are assigned to the structure elements that are much coarser, resulting in an inaccuracy.

The FEM simulations are then run using this interpolated pressure difference as a non constant loading. Consequently, the obtained displacement is transmitted through RBF interpolation acting on the fluid domain file, with the same technique as the one described in section 5.3. The FSI iterations are then run until convergence, using the same convergence criterion as the Viola testcase, reported here for clarity:

$$max(q_i) - max(q_{i-1}) < \frac{max(q_i)}{250},$$

where q is the computed displacement at the ith iteration. Convergence is reached at the 14th iteration. Figure 6.11 shows the two shapes obtained with the two different approaches: the constant pressure, figure 6.11a and 6.11b, and the FSI pressure, figure 6.11c and 6.11d.

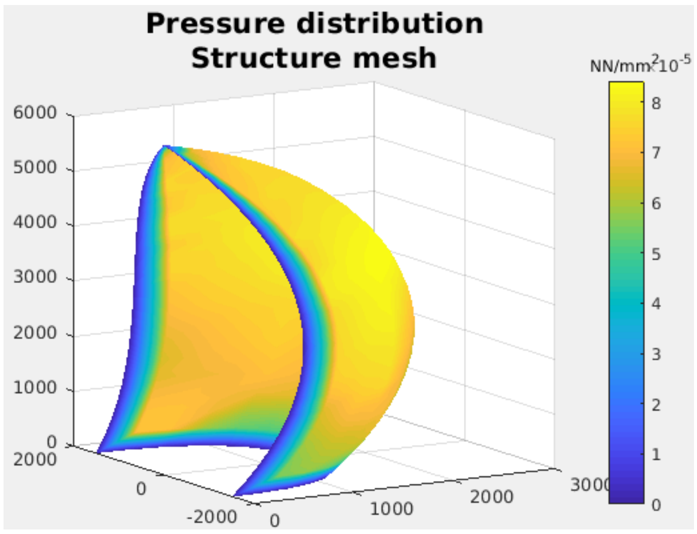


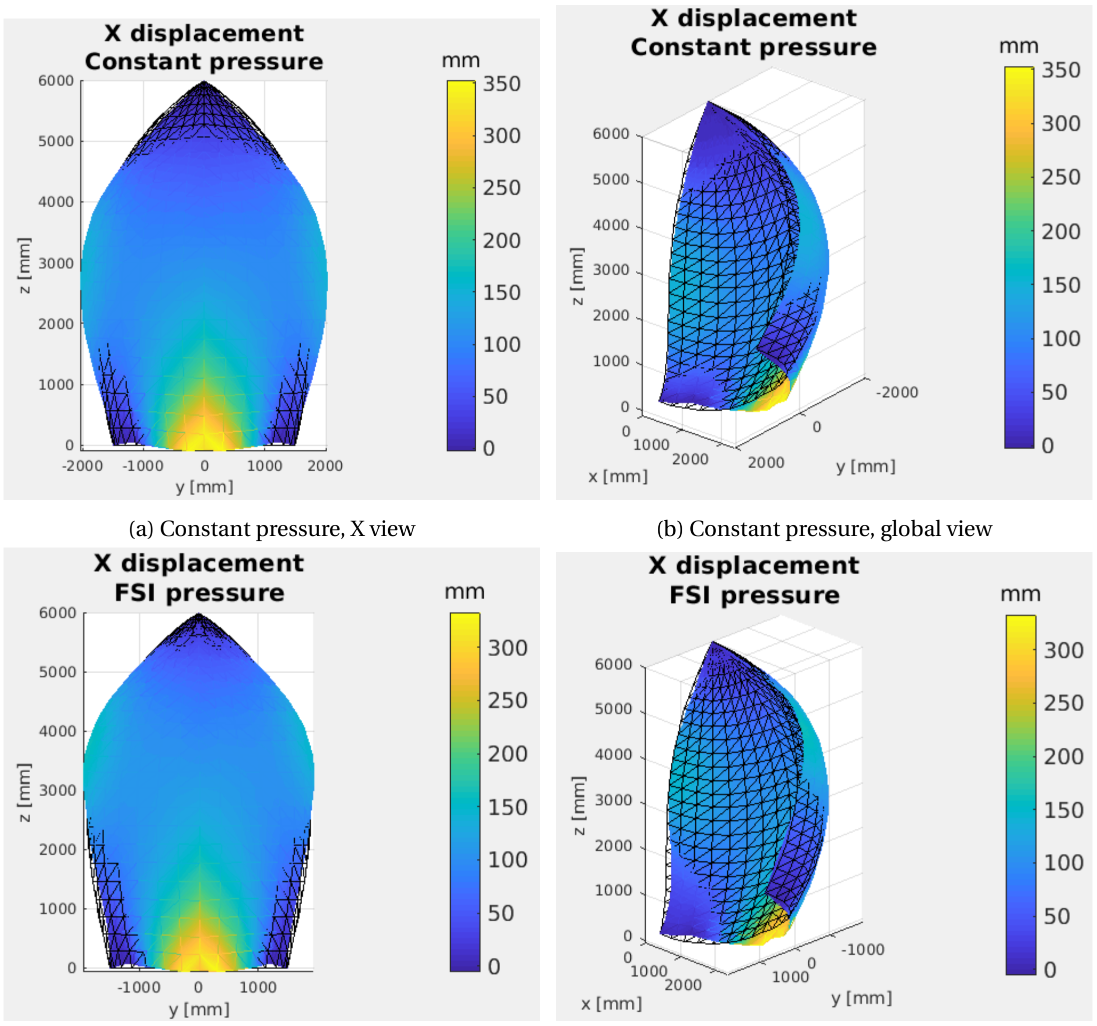
Figure 6.10: Interpolated delta pressure load on the structure mesh

First of all, it is worth mentioning that the obtained results cannot be compared to the ones obtained by Trimarchi [35], because of the difference in applied pressure load, as explained before. The main difference that can be observed between the two shapes in figure 6.11 is that the spi with constant pressure deforms more on its free sides, both on the bottom and the side edges. That is because on those elements the same pressure is applied as the rest of the sail, while in the FSI case the side elements do not get any forcing, see figure 6.10. Another figure that compares the outline of the two obtained shapes is reported 6.12: from this figure it is even more clear how the sides of the spinnaker are more deformed for the constant pressure case (blue line). The FSI case (red line) on the other hand shows a less regular shape, with the part next to the corners wearing thin and then inflating more in the center and then wearing thin again next to the sail head.
When wanting to assess which one of the two deformed shapes more represents the reality, one can compare to the situation depicted in figure 6.13, where a real-life symmetrical spinnaker is depicted. When looking at the side edge of the spinnaker in figure 6.13, it seems like the shape resembles more the one obtained imposing the FSI pressure, in fact the sail does not have a regular, circular shape like in the constant pressure case, but more a wave-like shape, like the red line reported in figure 6.12. This suggests that indeed an FSI investigation will yield more relevant results than an investigation with a constant pressure.
The last figure reported 6.14 compares the design shape and the flying shape obtained with FSI. In this case the differences are more noticeable than in the Viola testcase, due to the higher velocity and pressure load. From this image it can be argued that indeed the two shapes are widely different and that is something to take into consideration in the design process. Regarding the generated total forces, for the design shape the generated force is equal to 2143 N, while for the flying shape the force is 1981 N, resulting in a difference of 7%. This is representative of how using an FSI cycle results in a more accurate prediction of the force provided by the sail, thanks to the more accurate prediction of the deformed shape. The difference of 7% has to be taken in consideration when designing the sail and the rigging. In fact, the flying shape provides less thrust than the initial shape, and therefore using the estimation of the force from the initial shape would result in an over-dimensioning of the rigging and mast, for example.

6.3. Conclusions

Two test cases have been presented in this chapter. The first case was done using the geometry from Viola [9], which corresponded to the flying shape obtained in the wind tunnel. An iterative routine was implemented to obtain an initial shape to start the FSI simulations from. The results were very satisfying as a very similar shape to the reference one was obtained from the FSI iterations, with maximum differences in coordinates of approximately 2.5 mm. The result can be considered quite good given the complexity of the problem.

The second testcase, taken from Trimarchi [35], only provided the initial shape of the sail. For this case then it was not possible to evaluate quantitatively the solution, but the results have been compared qualita-



(a) Constant pressure, X view

(b) Constant pressure, global view

(c) FSI pressure, X view

(d) FSI pressure, global view

Figure 6.11: Computed displacement for the constant pressure case (6.11a,6.11b) and FSI pressure case (6.11c,6.11d)

tively to a real life spinnaker sail and the results were reassuring. An investigation has been performed on how the results change when using a constant pressure load or an interpolated pressure distribution. The analysis shows that using a pressure obtained with FSI yields more reliable results.

In conclusion, it can be argued that the solver gives reasonable results for both test cases and if it could replicate test cases of which both initial and flying shapes are known. Unfortunately it was not possible to find such test cases to use and compare in this work, since the field of sail simulations is not very well explored yet.

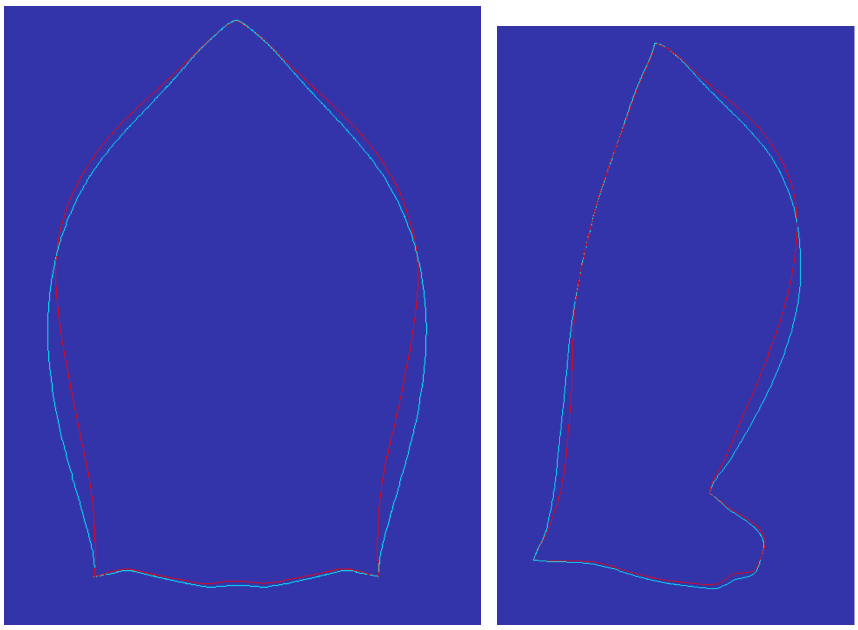


Figure 6.12: Solution with constant pressure (blue) and FSI pressure (red)

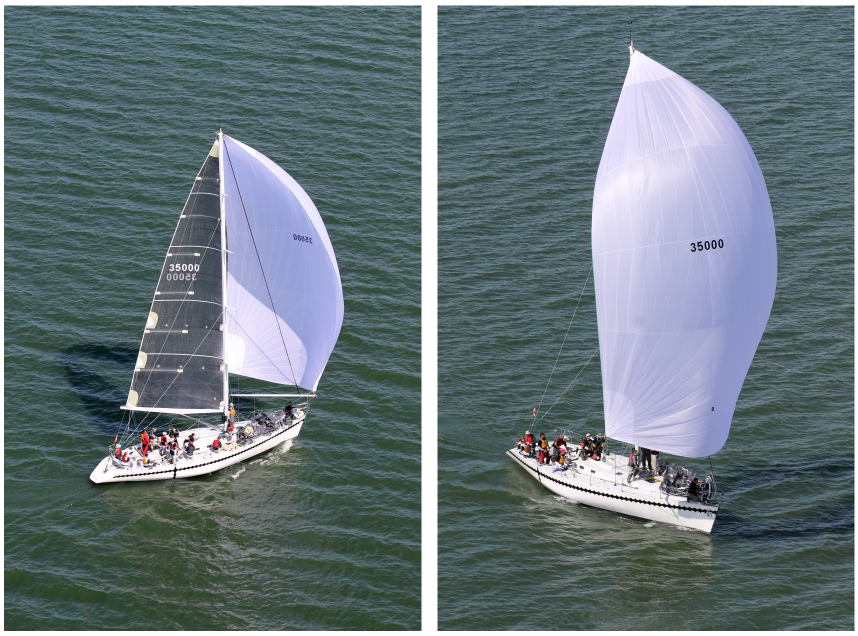


Figure 6.13: Real-life symmetrical spinnaker

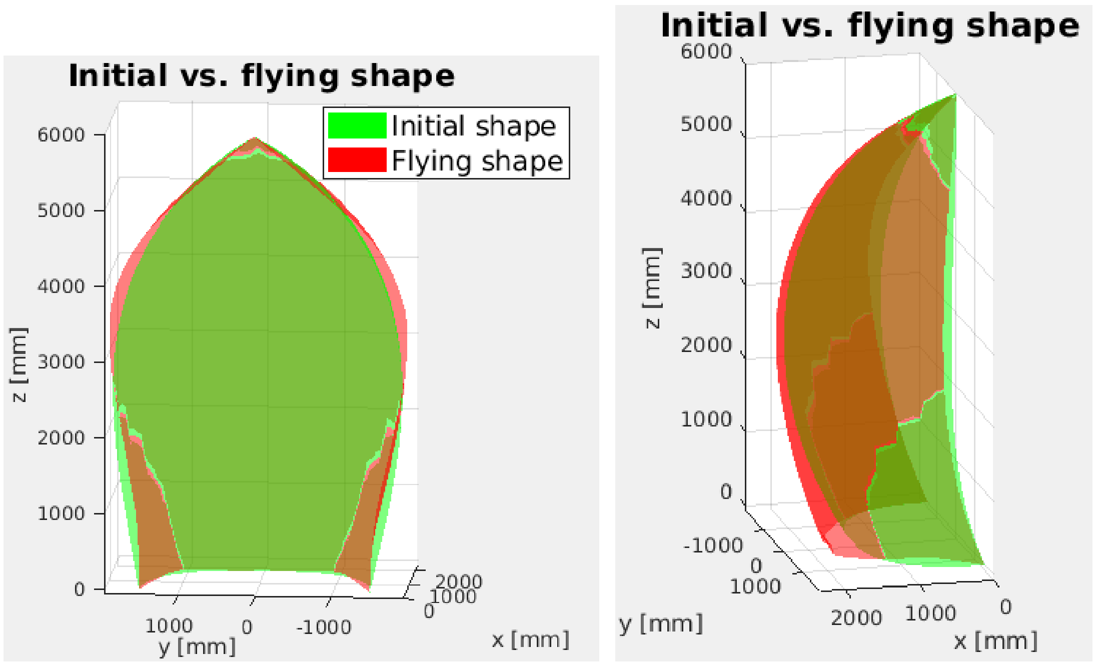


Figure 6.14: Initial shape (green) and flying shape (red)

7 Conclusions

In this thesis work, a fluid structure interaction solver has been implemented and validated. The intermediate steps to be taken consist in the CFD simulations, the FEM simulations and the coupling of the two solvers. The CFD simulations have been run with two solvers, FINE/Open and OpenFOAM, and the results compared to the experimental data obtained by Viola et al. [9]. The results were reassuring for the FINE/Open simulations, a good agreement with the experimental data was found. For the OpenFOAM simulations the differences were quite big and therefore the FINE/Open results have been used in the rest of the project.

Next, the FEM solver developed by Daniele Trimarchi [35] has been validated against some cases for which the analytical solution was available, namely the cylinder and sphere case, see section 4.2. These cases were chosen because no reference data was available for the sail testcase. However, the deformation of the sail has been qualitatively compared with some real life sailing condition and the main tendencies were well captured.

Consequently the interpolation techniques have been implemented and their validity asserted: the nearest neighbor interpolation has been used to transfer the pressure obtained with the CFD solver to the structure mesh, and the RBF method has been used to interpolate the displacements of the structure nodes to the fluid domain file. In the nearest neighbor interpolation an error of around 8% arose due to the coarseness of the structure grid, while the RBF was not quantitatively estimated but showed to work really well.

Finally, the result of the FSI simulation has been compared to the flying shape obtained by Viola et al. [9] and a small offset has been observed between the two shapes. However, the tendencies, displacement and pressure distributions are well reproduced, always with an error lower than 8%. Considering this work as a basis for a more elaborate and accurate project this is a quite satisfying starting point.

The aim of the project was not only to be able to reproduce some reference data but also to provide a tool for the sailmakers that allows to get an estimate of the sail thrust under certain wind conditions, in order to help with the sail design process. This objective has not been fully reached, since the solver is not totally automatized and has quite long run times, however with some more work put into it it can result in a very useful and powerful tool.

In conclusion, it is worth mentioning the research questions proposed in the literature study, and their answers, if they were found.

1. *Question: What are the appropriate techniques to compute the correct pressure distribution around the spinnaker sail through a CFD solver?*
   Answer: In order to obtain a pressure distribution that resembles the reference data, steady RANS simulations have been run with the k-ω turbulence model, using BC's that resemble the wind tunnel experiment. The simulations were three dimensional and used 5 million cells and 1000 steady iterations.

2. *Question: What techniques are able to capture the spinnaker's flying shape by coupling the CFD solver with a structural solver in an efficient manner?*

Answer: The FSI cycle implemented in this thesis uses the nearest neighbor interpolation to transfer the pressure data from the fluid grid to the structure one. The displacement on the structure grid is then transferred to the fluid mesh through radial basis function interpolation. No relaxation scheme has been used because the simulation converged quite quickly.

3. *Question: With the results from the two previous points, can the force generated by the spinnaker be computed correctly and, possibly, optimized using various trim settings?*
   Answer: The force generated by the sail has been computed by integrating the pressure information on the mesh, and it has been observed that the force produced by the flying shape is different than the one of the design shape. That proves the point that this type of analysis is necessary for a correct prediction of the forces in play, also thinking of the dimensioning of the other components of the yacht. The employment of this solver for optimization studies has not been performed but it can be a good starting point for another master thesis.

8 Recommendations

The obtained solver mainly has four weak points:

1. It is not fully automatized, still requires user action from iteration to iteration,
2. Its run times are quite long, namely 12 hours for FSI iteration on 6 cores,
3. The interpolation of the pressure to the structure mesh is not completely accurate and can lead to some error,
4. In the structure solver, the sheets are currently modeled only as clamps. In reality, they can be trimmed and their length can be shortened or extended. If that was modeled it would allow to simulate many more conditions.

These flaws can be fixed with some work and that could be the starting point for another thesis project. The steps to improve the solver have not been implemented not for lack of ability but for lack of time. However the ideas on how to solve the problems have been though of:

1. Automatizing the whole process by using HEXPRESS and FINE/Open in batch mode, and using one single script for the interpolations and FEM simulations. If using Matlab (like it was the case in this project) some Python script that runs the CFD software would have to be written in Matlab and run in Python.
2. Shortening the run times of the CFD simulation by imposing a stopping criterion for the testcase, maybe defined experimentally (see Appendix C).
3. Change interpolation technique for the pressure or use a more refined structure mesh in order to lower the interpolation error (also this could depend on the testcase and the pressure distribution).
4. The sheets can be modeled differently: either adding new elements with different properties in the structure or modifying the imposable boundary conditions, for example imposing that a certain point can only lie in a volume defined by a sphere of radius L, length of the sheet from its attachment point. The sphere would represent all the possible positions of the sheet, and therefore of the sail vertex.

If these steps were to be taken the solver would improve its performance and accuracy significantly. Once the solver is improved it can be used in the sail design process, maybe trying different geometries under same conditions and estimating which is the best one, or simulating various trim settings for the same geometry in order to find the optimal trim.

Acknowledgements

This project has been a lot harder than I thought it would be. It took a lot of effort from my side but I have to recognize that many people helped get through the process, being by my side in the most difficult moments and giving me the strength to push forward.

First of all I want to thank my parents. They have supported me throughout all the years of my education, helping me and motivating me when I was down. During the thesis, even if they were far away the followed me and gave me suggestions when I needed them. I hope they know how grateful I am to them for everything they have done and keep doing for me. I hope I will be able to pay them back one day.

Next I owe a big thanks to my big Delft family: Alex, Omar, Oriol, Simone, Victor and Edoardo thank you for sharing all the times with me, the good and the bad ones, for giving each other strength when we were down and for celebrating together when we achieved something. Life in Delft would have been so sad without you and I am thankful to have met you and intend to keep nourishing our friendship in the future.

A special thanks goes also to my bestie Carlos, with whom I shared so much of these years and hopefully will share much more in the future. Without you my experience in Delft would have been half the fun. Thank you for always being next to me and being weird with me and sharing every single thing with me. It is rare to find a person as special and incredible as you and now that I found you I will not let you go.

Another person I am grateful to is my boyfriend Giovanni. We shared a lot during the last year and you supported me every night when I was coming home tired from the basement, even if you were more tired than me you always had some reassuring words for me, and having you there in the morning to say "have a good day" was more important to me than you think. You gave me so much strength and I know you will keep giving me that in the future.

Last but not least, thank you to my friends from Milan: Carlo, Lucia, Virginia, Giacomo, Filippo, Silvia and Francesca thank you for always having been with me for so many years. You are a constant in my life and no matter what happens to us, wherever we go we will always have each other because our bond is too strong to disappear, even in thousands of kilometers of distance. I know you are just one phone call away and you know I rely on that.

Now more on the academic side: thank you to my supervisor Sander, who helped me throughout the whole project and put up with all my little and big problems, was always available and gave me some great ideas when I was stuck in whatever part of my thesis. I hope you enjoyed working with me as much as I did with you.

And thank you to Daniele Trimarchi, who helped me even though he didn't have to, and answered all my questions even if they were stupid, was more helpful than what I could have hoped and saved me from a "computational crisis".

Another special thanks goes to the basement students, that were there all throughout the thesis and shared with me the good and the bad times. Especially thank you Derek for the amazing help on all my little problems and for the laughs we shared.

Finally, thank you Delft for being my home for the past three years, you have given me so much and I will never forget all the life experiences and the new things I discovered here. You made me grow up on my own by showing me the good and bad parts of life. I will now leave you but you will always have a space in my (blue) heart.

A
Appendix

A.1. Structure of .dom file

This part is taken from the NUMECA documentation [29]. The domain file describes the computational domain for the simulation and it is made of two parts:

- The topology part describes the vertices of the model and defines a closed volume in which the domain lies.
- The geometry part defines the actual geomety of the model. Each model surface is described by a triangulation, each curve by a list of points connected by segments and the corners are defined by a single point.

The first three lines of the file are header lines which specify the software version, date of creation and the number of blocks the domain in composed of. In this case the domain is made by a single block. Next, some parameters for the triangulations are set, then the list of seed points is reported. After that a list of topological vertices is reported, which are the corners of the computational domain as explained earlier. In the next line starts the definition of the curves. Each curve is defined by a list of points and a list of segments connected to these points, and the expression in the domain file is:

```
    curve_id_1 (starting from 0)
number_of_points
point_1 x y z
point_2 x y z
point_3 x y z
...
number_of_segments
segment_1 point_id_O point_id_D
segment_2 point_id_O point_id_D
segment_3 point_id_O point_id_D
...
```

After the description of all the curves the definition of the surfaces is reported. Similarly to the curve, each surface is defined by a set of points and a list of triangles connected to these points:

```
    surface_id_1 (starting from 0)
number_of_points 1 (1 is not used but must be present)
point_1 x y z
point_2 x y z
point_3 x y z
...
number_of_triangles
```

triangle_1 3 point_id_0 point_id_1 point_id_2
triangle_2 3 point_id_0 point_id_1 point_id_2
triangle_3 3 point_id_0 point_id_1 point_id_2
...
surface_id_2

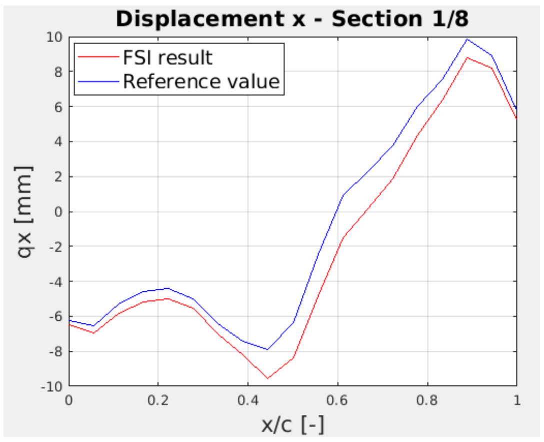
After these lines the topological edges are defined. For each topological edge, the IDs of the topological vertices used as extremities are given. For a cyclic topological edge, these IDs are (-1, -1). The type of the curve is specified, 1 for a convex edge, 2 for a concave edge. In addition, the curve ID defining the topological edge is given, it usually equals to the topological edge ID. This part will not need to be modified and therefore no details on the way the lines are written will be given.

A.2. Recognizing the useful lines

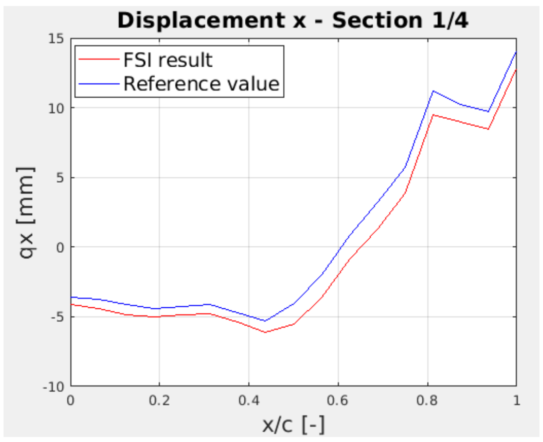
From the structure of the domain file described above, it can be argued that the lines to modify will be the ones regarding the topological edges, the curves and the surfaces. The strategy adopted to find out if the points in the file are relative to the spinnaker or not is based on taking the coordinate from the file and inputting it in the HEXPRESS GUI, where the point can be visualized in the domain. Only the points that lie on the spinnaker will be considered for the interpolation. Obviously this operation is not done for all the curves and surfaces but a preliminary selection is done. For example, in the HEXPRESS GUI it is possible to visualize the curve numbers: that showed that the spinnaker curve was curve zero, so a check has been performed with the technique described above to verify that the numbering was coinciding between the domain file and the GUI. It turned out that they were, for the curve but not for the surfaces. The spinnaker curve has then been identified and the lines describing it have been noted (lines 353 to 1448).
Regarding the topological vertices, after checking the position of the points in the domain it resulted that no points were lying on the spinnaker. This can be explained with the fact that being the spinnaker a zero thickness surface it does not really describe a volume in the domain and therefore the corners are not treated as topological vertices.
Regarding the surfaces, from the GUI the two surfaces describing the spinnaker were `face.1085_duplicate` and `face.1085`. These two names were not present in the domain file and therefore a check has been performed on one point for each surface present in the file until the two surfaces describing the spinnaker have been found, namely surface 11 and 13. The relevant lines were 17208-37973 and 82623-103388, respectively. With this information the RBF script can then skip the lines that do not lie in the relevant ranges and apply the interpolation to the lines describing the spinnaker geometry.

B
Appendix

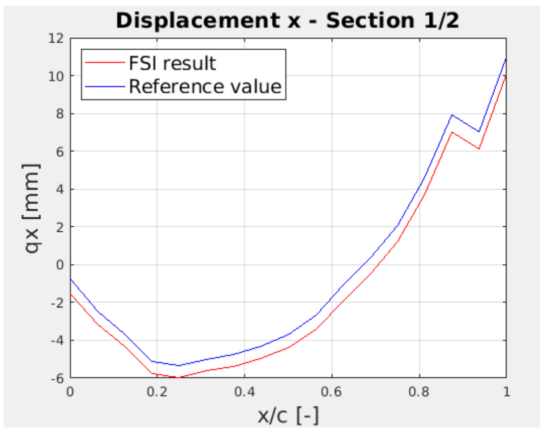
In this appendix the computed and reference displacements in the x and z direction for the Viola testcase are shown, in order to extend what was presented in chapter 6, figure 6.7.



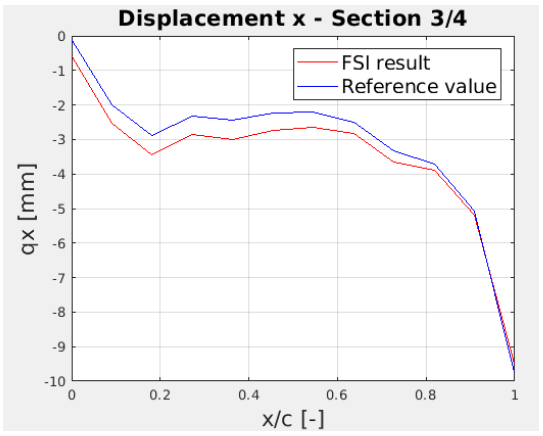
(a) Section 1/8



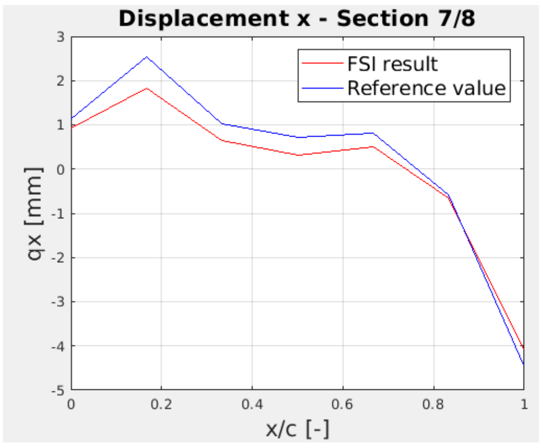
(b) Section 1/4



(c) Section 1/2

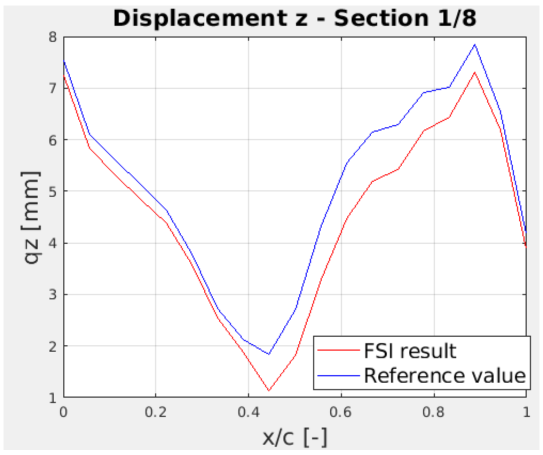


(d) Section 3/4

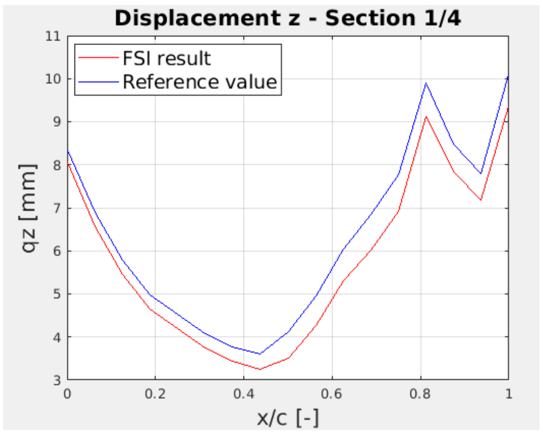


(e) Section 7/8

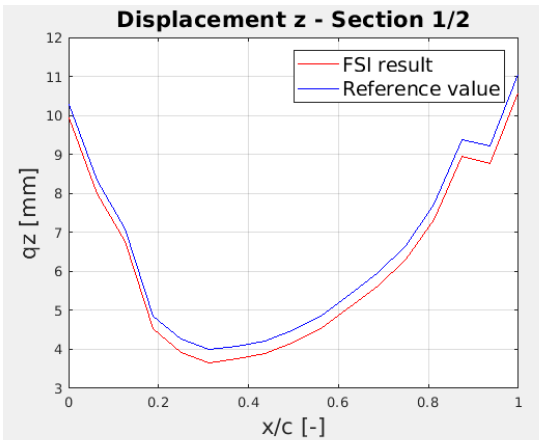
Figure B.1: Computed and reference x displacement at 5 horizontal sections



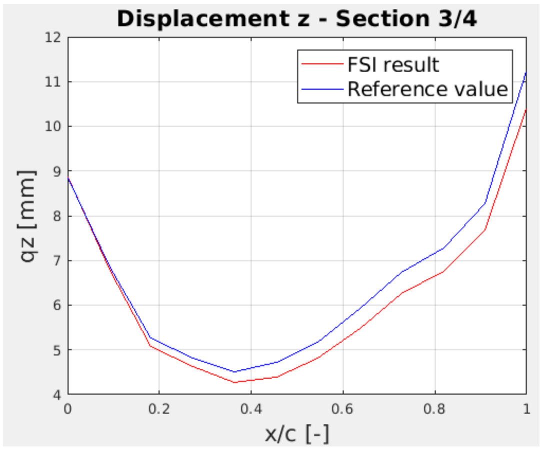
(a) Section 1/8



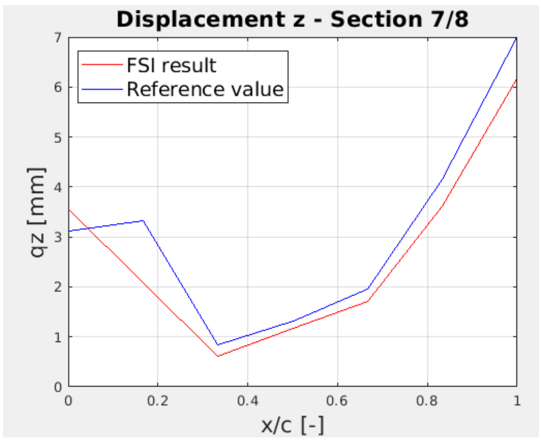
(b) Section 1/4



(c) Section 1/2



(d) Section 3/4



(e) Section 7/8

Figure B.2: Computed and reference z displacement at 5 horizontal sections

C
Appendix

This appendix clarifies what is expressed in point 2 of the recommendations. The goal is to shorten the run times of the CFD simulations by determining experimentally when to stop the simulation, monitoring the residuals and the convergence of lift and drag for example.
Figures C.1 and C.2 show the typical convergence plots obtained in the CFD simulations for the Viola testcase. The same tendencies were observed for all FSI iterations, so only one iteration is reported here.

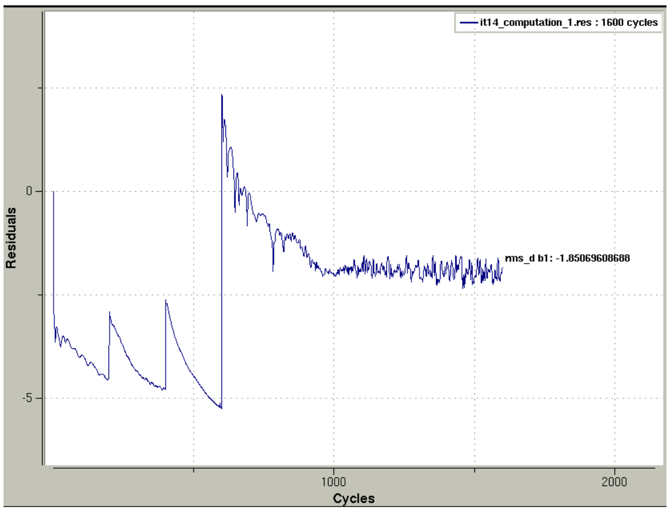


Figure C.1: Density residuals

From the density residuals plot it is clear that their values stabilize around a value of -1.9 after the 1000th iteration, meaning that adding more iterations will not make them decrease. From the lift convergence history however, it is possible to recognize that the value for the lift completely stabilizes only after the 1200th iteration. A zoom of figure C.2 is reported in figure C.3, where this tendency can be observed more clearly.

It is possible to argue that seeing these results one could stop the simulation at the 1200th iteration, maybe at the 1250th for safety, in order to reduce the simulation time. In fact the number of iteration on the fine grid for the case reported in figure C.2 was of 1000, while here it could be of 650, reducing the time by approximately 30%.

This strategy can be used in other simulations as well, but it is necessary to first run some simulations with the high number of iterations, in order to determine the tendency and if the same behavior is present in all the FSI iterations. It can however help to reduce run times quite significantly.

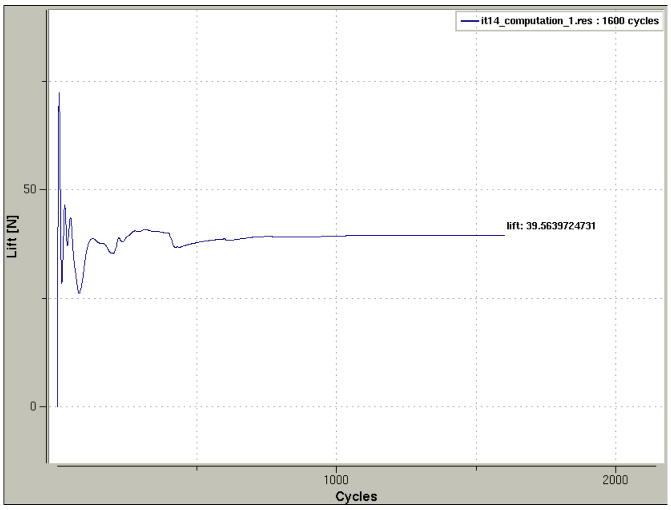


Figure C.2: Lift convergence history

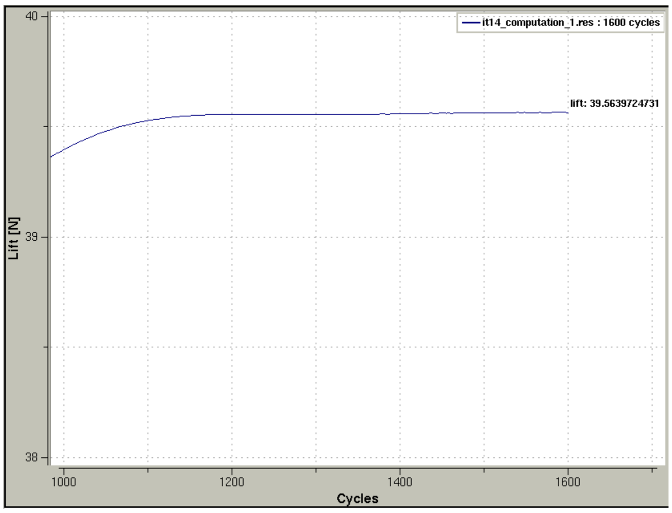


Figure C.3: Zoom of figure C.2

Bibliography

[1] H. Wendland A. Beckert. “Multivariate interpolation for fluid-structure interaction problems using radial basis functions”. In: *Aerospace Science and Technology 5* (2001), pp. 125–134.

[2] O.C. Torrey A. Loory. *UK Sailmakers’ Encyclopedia of Sails.*

[3] Z. Qin B. Tabarrok. “Nonlinear analysis of tension structures”. In: *Computer and Structures 45* (1992), pp. 973–984.

[4] D.B. Spalding B.E. Launder. *Lectures in Mathematical Models of Turbulence.* 1972.

[5] D.B. Spalding B.E. Launder. “The numerical computation of turbulent flows”. In: *Computer Methods in Applied Mechanics and Engineering 3* (1974), pp. 269–289.

[6] F.M.J. Bergsma et al. “Development Of Computational Fluid-Structure Interaction Method For Yacht Sails”. In: *The Third International Conference on Innovation in High Performance Sailing Yachts.* 2013.

[7] H. Bijl et al. *Fluid-Structure Interaction - An introduction to numerical coupled simulation.* 2008.

[8] A. de Boer, A. van Zuijlen, and H. Bijl. “Review of coupling methods for nonmatching meshes”. In: *Computer Methods in Applied Mechanics and Engineering 196* (2007), pp. 1515–1525.

[9] P. Bot et al. “Wind-tunnel pressure measurements on model-scale rigid downwind sails”. In: *Ocean Engineering 90* (2014), p. 84.

[10] J.F. Remacle C. Geuzaine. *Gmsh Reference Manual.* Version 4.0. 26 September 2018.

[11] *CATIA Version 5 Release 19 User’s Documentation.* English. Version 5.19. Dassault Systèmes. 2008.

[12] C.M. Rizzo D. Trimarchi. “A FEM-Matlab code for Fluid-Structure interaction coupling with application to sail aerodynamics of yachts”. In: *13th Congress of International Maritime Association of Mediterranean.* 2009.

[13] *GAMBIT 2.2 Tutorial Guide.* English. Version 2.2. ANSYS Fluent. September 2004.

[14] K. Graf H. Renzsch. “An experimental validation case for fluid-structure-interaction simulations of downwind sails”. In: *The 21st Chesapeake Sailing Yacht Symposium.* 2013.

[15] K. Graf H. Renzsch. “Fluid Structure Interaction Simulation of Spinnakers - getting closer to reality”. In: *The International Journal of Small Craft Technology* (2011).

[16] K. Graf H. Renzsch. “Fluid Structure Interaction Simulation of Spinnakers – Towards Simulation Driven Sail Design”. In: *21st International HISWA Symposium on Yacht Design and Yacht Construction.* 2010.

[17] K. Graf H. Renzsch O. Muller. “Flexsail – A Fluid Structure Interaction Program For The Investigation Of Spinnakers”. In: *Proceedings in the International Conference on Innovations in High Performance Sailing Yachts.* 2008.

[18] M. Schafer H.J. Bungartz. *Fluid-Structure Interaction.* Lecture Notes in Computational Science and Engineering. 2006.

[19] M. Schafer H.J. Bungartz M. Mehl. *Fluid Structure Interaction II: Modelling, Simulation, Optimization.* Lecture Notes in Computational Science and Engineering. 2010.

[20] Tobias Holzmann. *Mathematics, Numerics, Derivations and OpenFOAM(R).* Fourth edition. Leoben: Holzmann CFD, Feb. 2017.

[21] M. Riotte I.M. Viola P. Bot. “Upwind Sail Aerodynamics: a RANS numerical investigation validated with wind tunnel pressure measurements”. In: *International Journal of Heat and Fluid Flow 39* (2013), pp. 90–101.

[22] R.G.J. Flay I.M. Viola. “Force and Pressure Investigation of Modern Asymmetric spinnaker”. In: *The International Journal of Small Craft Technology* (2009).

[23] S. Chan J. Li. “An integrated analysis of membrane structures with flexible supporting frames”. In: *Finite Elements in Analysis and Design 40* (2004), pp. 529–540.

[24] M.A. Leschziner. “Computation of aerodynamic flows with turbulence-transport models based on second-moment closures”. In: *Computers and Fluids 24* (1995), pp. 377–392.

[25] M. Lombardi et al. “A strongly coupled Fluid-Structure Interaction model for wind-sail simulation”. In: *4th High Performance Yacht Design Conference*. 2012.

[26] F.R. Menter. “Two-equation eddy-viscosity turbulence models for engineering applications”. In: *AIAA Journal 32* (1995), pp. 1598–1605.

[27] C.E.S. Cesnik M.J. Smith D.H. Hodges. “Evaluation of computational algorithms suitable for fluid-structure interactions,” in: *Journal of Aircraft 37* (2000), pp. 282–294.

[28] D.H. Hodges M.J. Smith C.E.S. Cesnik. “Evaluation of some data transfer algorithms for noncontiguous meshes”. In: *Journal of Aerospace Engineering 12* (2000), pp. 52–58.

[29] *NUMECA Online Documentation Platform*. https://portal.numeca.be/docs/Default.htm. NUMECA International. 2018.

[30] M. Unser P. Thévenza T. Blu. “Interpolation revisited”. In: *IEEE Transactions on Medical Imaging 19* (2000), pp. 739–758.

[31] W.C. Lasher P.J. Richards. “Wind Tunnel and CFD Modelling of Pressures on Downwind sail”. In: *BBAA VI International Colloquium on: Bluff Bodies Aerodynamics Applications*. 2008.

[32] Y. Roux et al. “Strongly coupled VPP and CFD RANSE code for sailing yacht performance prediction”. In: *3rd High Performance Yacht Design Conference, Auckland*. 2008.

[33] D. Choudhury S.-E. Kim. “A near-wall treatment using wall functions sensitized to pressure gradient”. In: *ASME FED 217* (1995).

[34] D. Trimarchi. “Analysis of Downwind Sail Structures Using Non-linear Shell Finite Elements”. PhD thesis. University of Southampton, 2012.

[35] D. Trimarchi. “Implementazione in Ambiente Matlab di un Codice FEM per la Deformazione delle Vele”. MA thesis. Università degli Studi di Genova, 2008.

[36] P. Heppel V.G. Chapin N. de Carlan. “A Multidisciplinary Computational Framework for Sailing Yacht Rig Design Optimization through Viscous FSI”. In: *The 20th Chesapeake Sailing Yacht Symposium*. 2011.

[37] P. Heppel V.G. Chapin N. de Carlan. *Performance optimization of interacting sails through Fluid Structure coupling*. 2011.

[38] I. M. Viola et al. “Detached Eddy Simulation of a sailing yacht”. In: *Ocean Engineering 90* (2014), pp. 93–103.

[39] I.M. Viola. “Downwind sail aerodynamics: A CFD investigation with high grid resolution”. In: *Ocean Engineering 36* (2009), pp. 974–984.

[40] J. Sonnenmeier W. Lasher. “An Analysis of Practical RANS Simulations for Spinnaker Aerodynamics”. In: *Journal of Wind Engineering and Industrial Aerodynamics 96* (2008), pp. 143–165.

[41] P.J. Richards W.C. Lasher. “Validation of RANS simulations for spinnaker force coefficients in an atmospheric boundary layer”. In: *Journal of Ship Research 51* (2007), pp. 22–38.

[42] D.C. Wilcox. “Reassessment of the scale-determining equation for advanced turbulence models”. In: *AIAA Journal 26* (1988), pp. 1299–1310.

[43] A.M. Wright et al. *Off-wind Sail Performance Prediction and Optimization*. 2010.