Politecnico di Torino

Corso di Laurea
A.a. 2024/2025
Sessione di Laurea Luglio 2025

Surrogate model-based wing sail optimisation

Glaros case study

Supervisor:
Prof. Mauro Bonfanti
Co-supervisor:
Matteo Mastorakis

Candidate:
Dario Vucinic

Abstract

The search for an alternative propulsion system for marine transport has returned back to its roots, to a technology that has been known for thousands of years, wind propulsion. The need to reduce emissions and costs of transporting goods has revived the idea of applying wing sails to ships, starting with the experience of high-performance racing sailboats, that have been experimenting with this concept for 20 years already. The purpose of this work is to provide an innovative framework for wing sail design optimization, known in literature as Surrogate Based Design Optimization. The idea is to reduce the computational burden of high-fidelity fluid dynamic simulations, that would be too impractical to use coupled with an evolutionary algorithm for design space exploration and substituting them with a surrogate model trained on as few samples as possible. To account for the complex dynamics and behaviours that a wing sail would create on the whole boat, a static model is implemented in the workflow to guarantee the analysis of just feasible geometries. Six parameters are used to describe the geometry of the wing sails, and they are initially sampled to create a Design of Experiments (DOE) table through a Latin Hypercube Sampling technique. In order to minimise the necessary DOE individuals, an Adaptive Sampling algorithm is implemented. These initial samples are then simulated with 2D unsteady CFD simulations to extract CL and CD values for each design point; these data are then used as training dataset for a Neural Network. The training of the Neural Network is carried out using the k-fold cross validation technique and the evaluation of different error metrics. A 6 Degrees of freedom static model is used to evaluate the equilibrium conditions, finding the combination of control parameters that maximises forward speed and at the same time guarantees the equilibrium of the system.

The optimal geometry of the wing sail is explored and found by a genetic algorithm that is using the overall VMG (Velocity Made Good) as objective function.

Summary

1. List of Figures .... **VI**

2. List of Tables .... **IX**

3. Introduction **0**

4. Background **5**

4.1 Current technologies 5

4.1.1 Flettner rotors 5

4.1.2 Suction wings 6

4.1.3 Rigid sails 7

4.1.4 Soft sails 7

4.1.5 Kite sails 8

4.2 Key aspects of sails 9

4.3 Fluid dynamic forces 10

4.4 Forces acting on a sailboat 12

4.5 Foiling 15

4.6 Design optimisation 16

4.7 Surrogate-Based Design Optimisation (SBDO) 17

4.8 Design of Experiments (DOE) 18

4.9 Surrogate Model candidates 20

4.9.1 Response Surface Method 20

4.9.2 Artificial Neural Network 21

4.9.3 Kriging 22

4.9.4 Radial Basis Function (RBF) 23

4.10 Surrogate model training 23

4.10.1 Leave-One-Out Cross-Validation 24

4.10.2 $\boldsymbol{k-fold}$ Cross-Validation 24

4.10.3 Bias-Variance trade off 25

5. Glaros' description **27**

6. Optimisation Workflow **34**

7. Geometry creation **36**

8. CFD simulations **41**

8.1 Computational Model 43

8.1.1 Automatization of the simulations 43

8.1.2 Computational domain 43

8.1.3 Mesh 45

8.1.4 Simulation set up 51

8.1.5 Stopping criteria 53

9. Neural Network training **55**

9.1 Error metrics 57

9.2 Adaptive Sampling 57

9.3 Bayesian optimisation 59

9.4 Results 60

9.5 $\boldsymbol{k-fold}$ cross validation 62

10. VPP **63**

10.1 General background knowledge 63

10.2 Glaros' VPP 64

10.2.1 Environmental conditions 65

10.2.2 Force decomposition 65

10.2.3 Aerodynamic forces 68

10.2.4 Hydrodynamic forces 68

10.2.5 Implementation 70

10.2.6 Geometry optimisation 72

11. Results' analysis **73**

12. Conclusions **79**

13. Bibliography 81

1. List of Figures

Figure 1 Concept design of the Oceanbird carrier (2) 1

Figure 2 BMW Oracle on the right, Alinghi Team in the top left image. The two boats side by side in the bottom left image - Americas's Cup 2010 (5) 3

Figure 3 Luna Rossa AC72 mounting a wing sail in San Francisco Bay in 2013 (5) 3

Figure 4 Glaros foiling with two sailors from the Polito Sailing Team 4

Figure 5 Flettner Rotors in action (8) 6

Figure 6 Suction wings application (9) 6

Figure 7 Rigid sail on an AC45 (10) 7

Figure 8 Soft sails on recreational sailboat 8

Figure 9 Kite sail application (13) 8

Figure 10 Aerodynamic forces decomposition (17) 10

Figure 11 CL polar for the AH79-100 B airfoil (14) 11

Figure 12 CD polar for the AH79-100 B airfoil (14) 12

Figure 13 Forces acting on a sailing yacht - top view (15) 13

Figure 14 Apparent Wind Calculations (15) 14

Figure 15 Different foil shapes and relative names (15) 16

Figure 16 Example of distribution of 40 sample points for a two-dimensional problem (left: LHS; right: Uniform Design) (16) 19

Figure 17 Neuron with activation function (19) 21

Figure 18 Multi-Layer perceptron (16) 22

Figure 19 RBF architecture (16) 23

Figure 20 Explanation of Bias-Variance trade-off (21) 25

Figure 21 Glaros flying 27

Figure 22 Glaros rendering 28

Figure 23 Side view of Glaros 29

Figure 24 Front view of Glaros 29

Figure 25 Local reference frame for the boat's dimensions 30

Figure 26 Glaros J foil – front view 31

Figure 27 J foil main dimensions 32

Figure 28 Rudder foil – front view 32

Figure 29 Rudder foil main dimensions 33
Figure 30 Optimisation flowchart 35
Figure 31 Geometry parametrisation adopted in (24) 36
Figure 32 Geometrical parameters adopted in (25) 37
Figure 33 Geometry creation workflow 39
Figure 34 Full geometry-simulation workflow 40
Figure 35 $k - \omega\ SST$ model switches between turbulence model using blending function (30) 42
Figure 36 Computational domain 44
Figure 37 Local refinements between wing edges 45
Figure 38 Offset refinement 45
Figure 39 Domain mesh 46
Figure 40 Local mesh refinements between wings 46
Figure 41 Wake and vertical box refinements 47
Figure 42 Boundary conditions 52
Figure 43 Samples distribution α-δ parameters 55
Figure 44 Samples distribution C1-Gap parameters 56
Figure 45 Samples distribution C1-%C1 56
Figure 46 Upwind/downwind regatta field example 65
Figure 47 Side view forces acting on Glaros 66
Figure 48 Back view forces acting on Glaros 66
Figure 49 Body axes reference frame 67
Figure 50 J foil polar 69
Figure 51 Daggerboard polars 69
Figure 52 Rudder foil polar 70
Figure 53 Outputs of FUNC_mdl6 71
Figure 54 Nonlinear constraint equilibrium function 71
Figure 55 Airfoil sections of optimal sail 73
Figure 56 Isometric view optimal sail 74
Figure 57 Top view at different angles 74
Figure 58 Isometric view at different angles 74
Figure 59 GA convergence 75
Figure 60 Sail's aerodynamic coefficients' polars 77

Figure 61 Comparison between polars of different geometries 77

2. List of Tables

Table 1 Sampling algorithms 18
Table 2 Glaros' measurements 30
Table 3 Main components coordinates 31
Table 4 Sails surface areas 33
Table 5 Final parameters adopted 37
Table 6 Geometrical parameters and relative boundaries 38
Table 7 Mesh parameters 49
Table 8 Mesh default controls 49
Table 9 Prism Layer controls 49
Table 10 Custom controls mesh 50
Table 11 Surface controls 50
Table 12 Simulation time and results for different physical times 53
Table 13 Errors for different physical times 54
Table 14 Some examples of LHS individuals 55
Table 15 Errors after initial training 58
Table 16 Optimal model's hyperparameters 61
Table 17 CL error metrics per iteration 61
Table 18 CD error metrics per iteration 62
Table 19 fmincon boundaries 71
Table 20 Geometrical parameters and relative boundaries 72
Table 21 Optimal design parameters 73
Table 22 Genetic algorithm parameters 75
Table 23 Optimal results 76

3. Introduction

Wind propulsion has been part of human history for thousands of years: the first sailing vessels are traced back to the Egyptians, around 3000/4000 BC. It helped in the development of trade and can be considered a great contributor to the growth of Roman, Greek, Viking and Phoenicians societies, as well as the human species in general, allowing to discover new territories, new cultures and new opportunities. After centuries of sailing boats dominating the seas, with the industrial revolutions of the 18th, 19th and 20th century, this kind of vessels lost to the more powerful and versatile steam or fuel powered ones, setting the decline of this type of propulsion.

Nonetheless, it still kept its presence as a cultural and recreational activity all over the world, in events like the America's Cup, the Vendée Globe, the Ocean Race, SailGP, and also in the Olympic games to mention a few.

The oil crisis that took place in the 1980s, involving the sudden increase of oil price, shook the economy and the industries, leading to an exploration for alternatives. If energy consumption and pollution are taken into considerations too, a need for research for more eco-friendly sources arose quickly, involving all engineering fields, marine transport as well.

Research efforts for more sustainable alternatives to fossil fuel propulsion slowed down as soon as the oil prices dropped down again, but as reported in (1), all experts agree that, since its new rise in 1998, oil price will steadily keep growing, leading to an increase of the shipping cost for approximately 90% of the traded goods. Starting from 2008, many ships reduced their cruising velocity from 25 to 20 knots, or even to 12 knots, but such a solution is in contrast with the need of delivering goods in a timely manner to meet market demands.

As reported in (2), "*In a world with volatile energy prices and limited global availability of several energy carriers, the possibility of directly harvesting wind energy on board decreases operational costs, thereby reducing financial risks as well as emissions*".

Thus, an urge in development of wind propulsion technology is needed, also considering that the worldwide merchant fleet produces more than 3% of the total global carbon emissions (1), and sail technology is likely to be a valid alternative. Electrification, fuel cells, nuclear power, and alternative fuels have the potential to provide carbon-neutral energy for specific trade routes, but each technology still faces unresolved challenges.

Although the exact reduction in fuel consumption is debated, the first applications of wind-assisted propulsion systems made their appearance in 1980, on the tanker "*Shin-Aitoku-Maru*" (3). Since then, there have been various applications and technologies developed. In (1) some examples are reported. In 1986, a cargo wind-ship with rigid wing sails was built and generated an average annual saving of between 15% and 30%. In the year 2000, a product carrier with rigid wing sails was designed by the Danish Ministry of Environment and Energy reported 15% savings of fuel. According to (2), the European Maritime Safety Agency reported energy savings of up to 30% for ships mounting rotor sails, while savings up to 50% for ships designed ad hoc for wind propulsion. An example of such a concept is the *Oceanbird* project, shown in Figure 1.

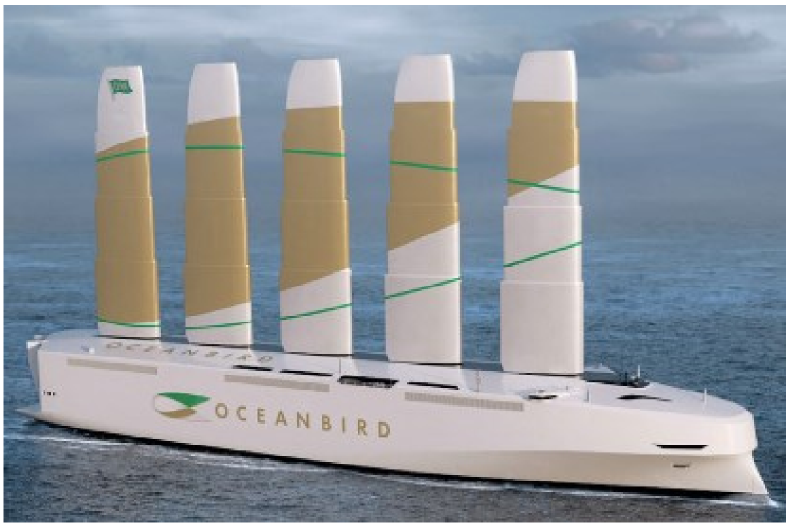


Figure 1 Concept design of the Oceanbird carrier (2)

Due to their conformation and absence of superstructures, low-speed bulk carriers and oil tankers could particularly benefit from wind-assisted propulsion. The long distances covered and mainly not in restricted waters, means that the wing sails could generate useful thrust without compromising significantly manoeuvrability.

Another important application of wing sails that has been researched is autonomous marine systems, that are designed to reach location inaccessible or inhospitable for humans. Regarding marine robots, researchers have been directed to electric motors or combustion engines, but the dependence on board fuel or battery capacity means restrictions in range and endurance. As reported in (3), these limitations can be overcome by wind propelled vessels, due to their ability of harvesting propulsive power from the environment, instead of storing it on board.

This type of sails found its application in high performance sailing as well. The America's Cup represents the pinnacle of high-performance sailboats, and since its first edition, it led to the development of many innovations in sailboat design. The first competition took place in 1851, making it also the oldest trophy in international sport.

As stated in (4), "*high performance sails such as the ones used on the America's' Cup boats, required sails whose aerodynamics approach those of rigid wings*".

The first appearance of this technology in the Cup was in 1988 thanks to the intuition of Dennis Connor. A last minute "Deed of Gift" challenge coming from a New Zealand team meant that the American team, winner of the previous edition, had no time to design and build a new boat, so Dennis Connor partnered up with some airplane designers to design a rigid wing instead of a traditional cloth one.

The difference in performance between the two boats was astonishing, showing that the hard sail was far superior in all points of sail. The wing sail catamaran ended up beating the large monohull with a soft sail by about 20 minutes each race.

Due to the court battles over the validity of the boats, the following 5 editions took place in traditional monohulls, very similar one to another. Rigid wing sails appeared again in the 2010 edition in another Deed of Gift match between Alinghi Team and BMW Oracle Team, in which both teams showed up with two 110 feet long catamarans. Alinghi was mounting a huge soft sail plan capable of doing 2.5 times the wind speed, while BMW Oracle had developed a huge 223 feet wing sail, capable of doing 3 times the wind speed. The American team dominated all the races by more than 5 minutes, leading to the adoption of such wing sails in the rule books of the following editions (5).

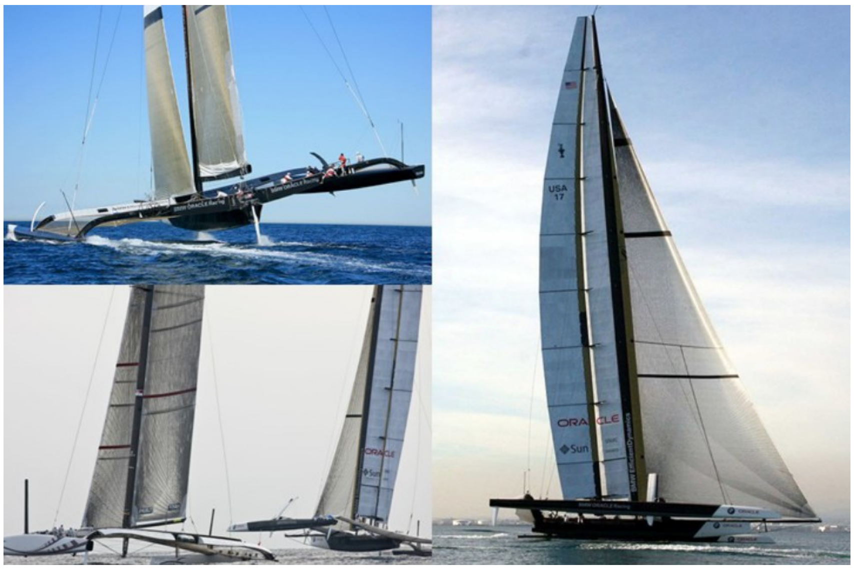


Figure 2 BMW Oracle on the right, Alinghi Team in the top left image. The two boats side by side in the bottom left image - Americas's Cup 2010 (5)

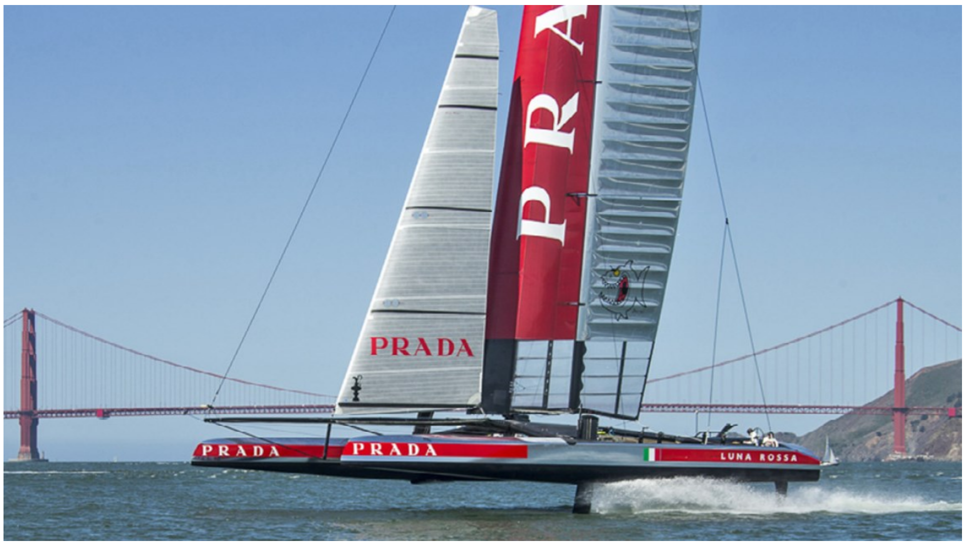


Figure 3 Luna Rossa AC72 mounting a wing sail in San Francisco Bay in 2013 (5)

With this in mind, this work aims at proposing an innovative workflow to optimize a wing sail that could be mounted on a racing boat. The case study has been chosen being Glaros, a R3 class foiling skiff, developed and built by the Polito Sailing Team, in 2020/2021. The Polito Sailing Team is a student team from Politecnico di Torino carrying out the design, construction, and sailing of Skiff and Moth prototypes, focusing on performance and sustainability of the designs. Although wing sails are most often associated with larger vessels, the idea of mounting a wing sail on a boat that is not of the size of an AC72 is not new, in fact, there have been some prototypes of C-class catamarans that implemented one, as reported in (6), thus the inspiration for Glaros.

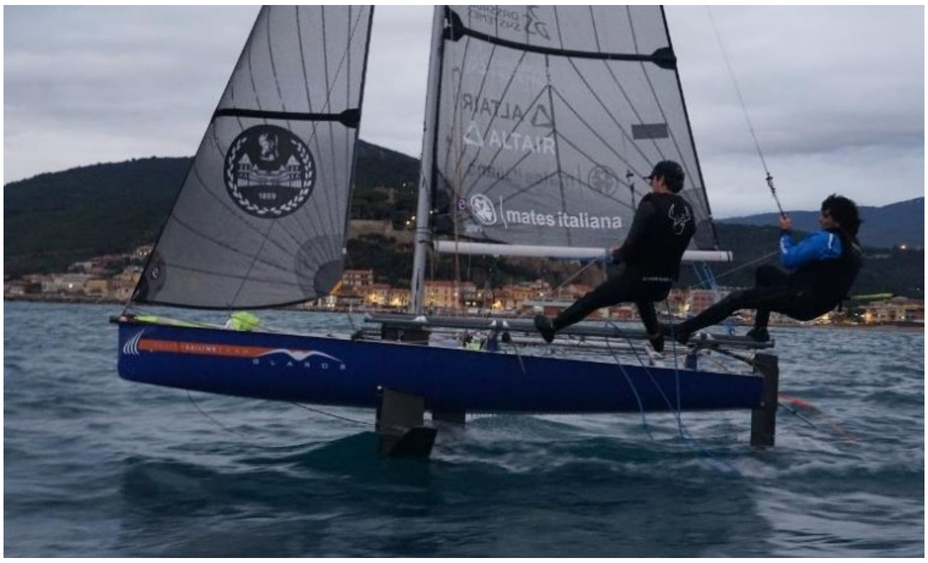


Figure 4 Glaros foiling with two sailors from the Polito Sailing Team

4. Background

4.1 Current technologies

The revived interest in wind assisted propulsion didn't converge on one single type of wing sail for all applications. Alternative types of wing sails emerged and reported below:

- Rotor Sails (Flettner rotor)
- Suction wing
- Hard sail
- Soft sail
- Kite

4.1.1 Flettner rotors

Flettner rotors are vertically rotating cylinders mounted on the deck of ships that exploit the Magnus effect to generate a thrust force. These cylinders can rotate up to 300 rpm, depending on size and application, and are generally made out of composite materials in order to reduce weight.

The Magnus effect is a physical phenomenon where rotation creates a thin boundary layer that creates a pressure differential across the rotor when exposed to wind. This pressure differential creates a force that acts perpendicular to the wind direction. The wind is most effectively harnessed when it blows from the side (beam reach relative to the vessel). In addition, the amount of thrust generated can be increased or decreased by adjusting the speed of the rotor (7). Figure 5 shows a real-world implementation of this technology.

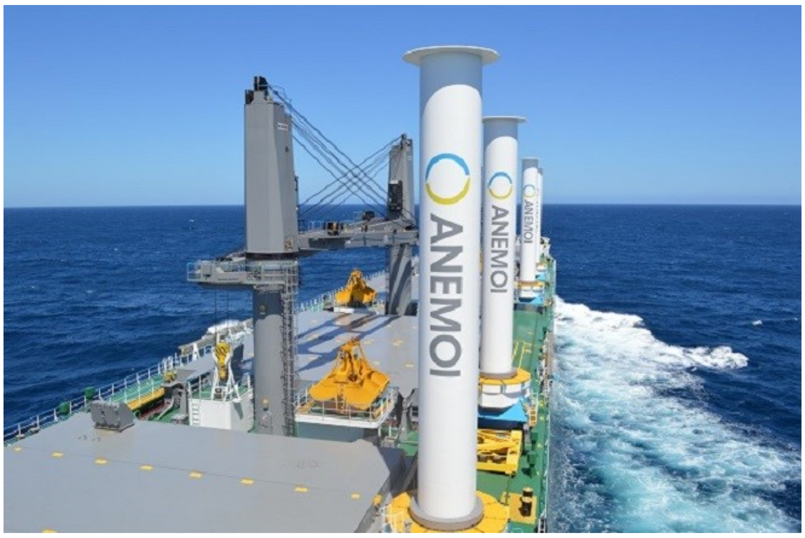


Figure 5 Flettner Rotors in action (8)

4.1.2 Suction wings

Suction wings are advanced propulsion systems that harness wind energy by modifying airflow over a wing using small pumps or fans. In typical wings, air close to the surface can become turbulent and detach, which lowers efficiency. Suction wings overcome this by using pumps to draw in air, eliminating turbulence and maintaining smooth, attached airflow—greatly enhancing lift and overall performance. Figure 6 shows an example of suction wings application.

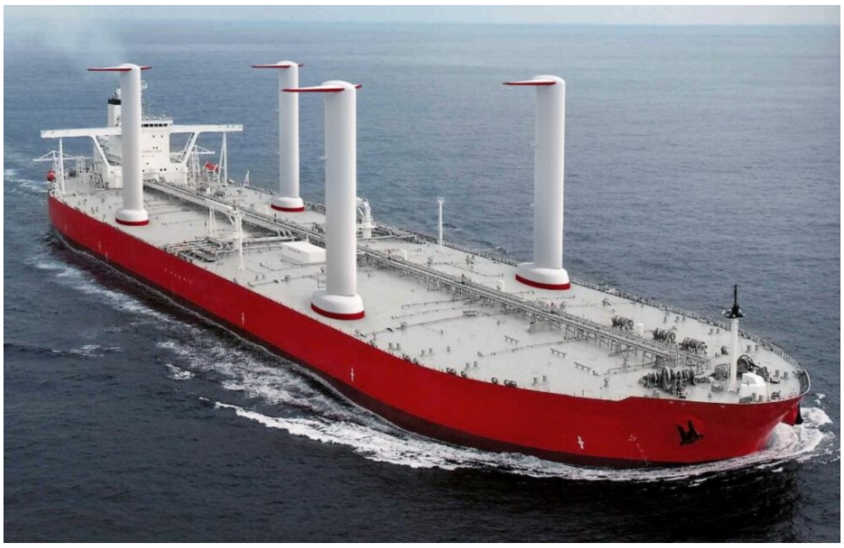


Figure 6 Suction wings application (9)

4.1.3 Rigid sails

Rigid, or hard, sails are wing-like structures installed on ships to harness wind energy for propulsion, with the aim of increasing propulsive efficiency. Unlike traditional cloth sails, hard sails are made of carbon fibre or fibre glass composites, and are designed to function similarly to aircraft wings, generating lift to propel the vessel forward.

Hard sails are designed to optimise aerodynamic efficiency. Their rigid structure allows precise control of shape and angle, resulting in better performance in different wind conditions. An example of such wings is visible in Figure 7.

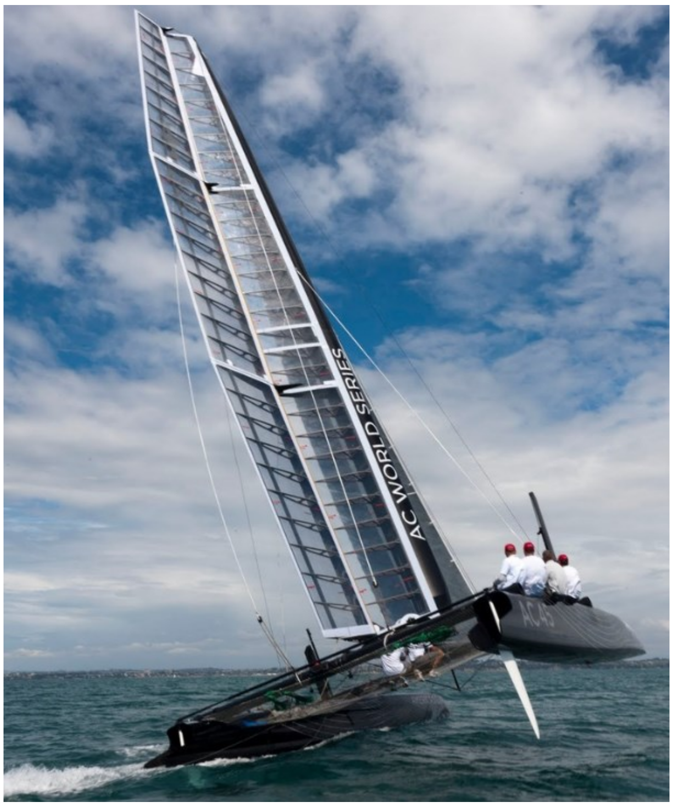


Figure 7 Rigid sail on an AC45 (10)

4.1.4 Soft sails

Soft sails refer to the conventional sails mounted on sailboats, flexible fabric-based structures used to harness wind energy for vessel propulsion. Modern soft sails are constructed from durable, lightweight materials such as polyethylene fibre fabrics, which offer enhanced strength and longevity in marine environments (11). Figure 8 shows an example of a sailboat with common cloth sails.

Figure 8 Soft sails on recreational sailboat

4.1.5 Kite sails

Kite sails are large aerodynamic kites that fly at high altitudes in order to capture stronger and more consistent winds, ranging from 200 to 400 meters. They can be used to assist in towing vessels and can be dynamically controlled to maximise aerodynamic efficiency to generate substantial traction power (often flying in figure eight-shaped patterns) (12). Figure 9 shows a real-world application of a kite sail on a ship.

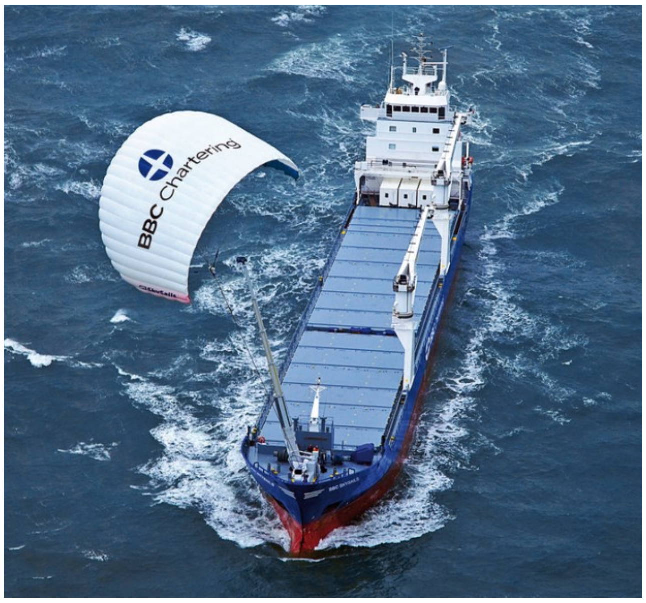


Figure 9 Kite sail application (13)

4.2 Key aspects of sails

It hasn't been easy to find a comprehensive definition of what a wing sail is, but (3) provided one that encompasses the key features: "*a wing sail is a rigid structure presenting an airfoil cross section which can provide a much better lift/drag ratio than conventional sails*" (3).

As it has been done for centuries, sailboats can be powered by traditional cloth sails, and it still remains the most common approach. In the previous section alternative types of propulsive systems have been presented, but the founding principle remains common to all of them: when an airflow encounters a surface, various forces are generated, and if oriented correctly, these forces can lead to a forward thrust that is capable of propelling a boat forward, regardless of whether it is a cloth surface or a wing sail.

Cloth sails are easier to handle and rely on wind pushing against the cloth or flowing around it to generate lift, depending on whether the boat is sailing downwind or upwind. Wing sails are rigid, shaped like an airfoil, thus allowing the airflow to better follow the surface of the wing and generate lift more effectively. The main advantages of flexible sails are:

- They are easier to handle by a human sailor
- More suitable for lowering and storing, especially in adverse weather conditions or mooring
- Allow easy exchange of sails to better adapt to wind conditions
- Easily repairable and modifiable
- Shape and camber of the sail are easily changed by just altering tensions on the lines (ropes)

On the other hand, they also present various drawbacks, like:

- Prone to wearing and tearing
- Being controlled by on-board lines, they're difficult to set perfectly
- Must be furled down or folded to be stored, which degrades the material
- Sensitive to luffing, a fluttering behaviour that occurs then the sail is not correctly trimmed
- Even when perfectly trimmed, the maximum lift that can be extracted is not so high
- Require rigid masts and wiring to be held in place

Meanwhile, the key aspects for wing sails can be summarised as follows:

- Increased Lift/Drag ratio compared to conventional sails, which means that for the same drag they generate more lift
- More efficient for both downwind and upwind navigation
- More reliable and do not suffer from luffing and flapping, even in light wind
- More easily controlled and set at the perfect angle of attack
- Do not require any additional structural elements
- Challenging to design a wing sail that can be reefed
- Difficult to find a compromise between strong, lightweight and cost-effective structure

4.3 Fluid dynamic forces

The aforementioned forces are generated when an air flow, or more generally any fluid, interacts with a surface, and are due to the pressure difference that generates on both sides of such surface. This interaction can be illustrated as shown in Figure 10.

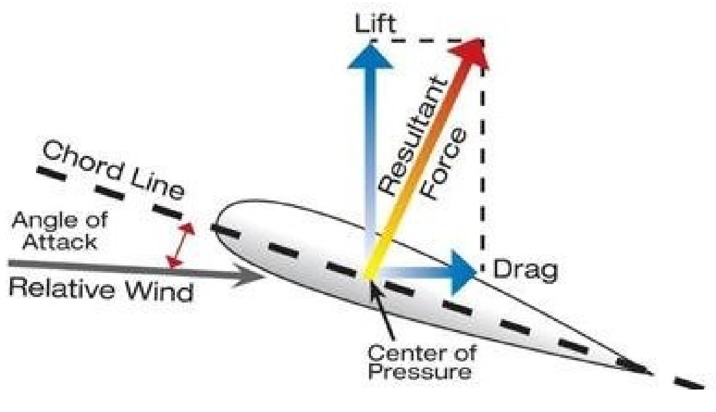


Figure 10 Aerodynamic forces decomposition (17)

The resulting force from the fluid-structure interaction can be decomposed into a component perpendicular to the flow direction named *Lift*, and one parallel to the flow direction called *Drag*. On the upper side there's a higher velocity then the one at the lower side, and according to Bernoulli's principle, this leads to having a higher pressure at the bottom of the airfoil (*pressure side*) and a lower one at the top (*suction side*), which translates into a force.

$$P_{tot} = P_s + \frac{1}{2}\rho V^2 + \rho g z = const$$

This relationship is important because it gives the formulation of the Drag and Lift forces $[N]$, i.e.:

$$L = \frac{1}{2}\rho V^2 S C_L$$

$$D = \frac{1}{2}\rho V^2 S C_D$$

in which, ρ is the fluid density $\left[\frac{kg}{m^3}\right]$, V^2 is the square of the fluid relative velocity $\left[\frac{m}{s}\right]$, S is the planar projection of the surface of the wing $[m^2]$, C_L and C_D are defined as the lift and drag coefficients, respectively. It is important mentioning that C_L and C_D are influenced by the Reynolds number $\left(Re = \frac{\rho L V}{\nu}\right)$ and by the angle of attack α, which is defined as the angle between the chord line of the airfoil and the flow direction. Generally, the C_L and C_D values are found in polars plotted as a function of Re and α, like the ones reported in Figure 11 and Figure 12.

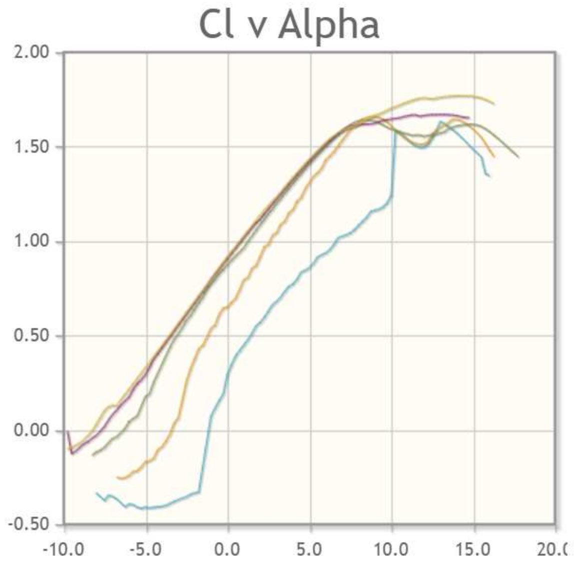


Figure 11 CL polar for the AH79-100 B airfoil (14)

The same plots are found also for C_D. It is worth noticing that C_L and C_D differ by one or two orders of magnitude, depending on the angle of attack.

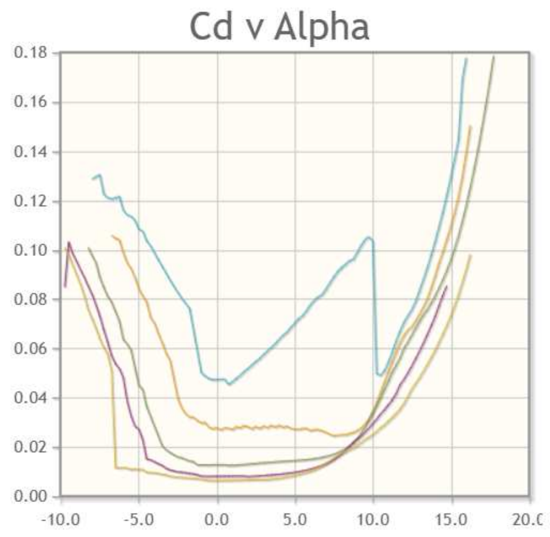


Figure 12 CD polar for the AH79-100 B airfoil (14)

The drag force can be further decomposed into three main components, called *form drag* (or *pressure drag*), *skin friction drag,* and *induced drag*. The first one is generated by the shape and size of the airfoil, the second one is generated by the viscous effects on the interface between the flow and the airfoil surface, and the latter one is linked intrinsically with the lift generation, because at the tip of the wing the low and high pressure zones cause the fluid to move from the higher to the lower pressure zone, creating vortexes, thus dissipation and drag.

These definitions are fundamental to understand the function principle of a sail, in fact both flexible and rigid wing sails produce lift and drag forces. By acting on the angle of attack of the sail, it is possible to modulate the lift force and thus the forward thrust. In fact, the sail's lift is not enough to push the boat forward, but there's another component of a sailboat that is contributing to the generation of the propulsive force, the daggerboard.

4.4 Forces acting on a sailboat

In Figure 13 it is possible to observe the decomposition of the various forces acting on the sail and daggerboard, as well as the resulting driving force.

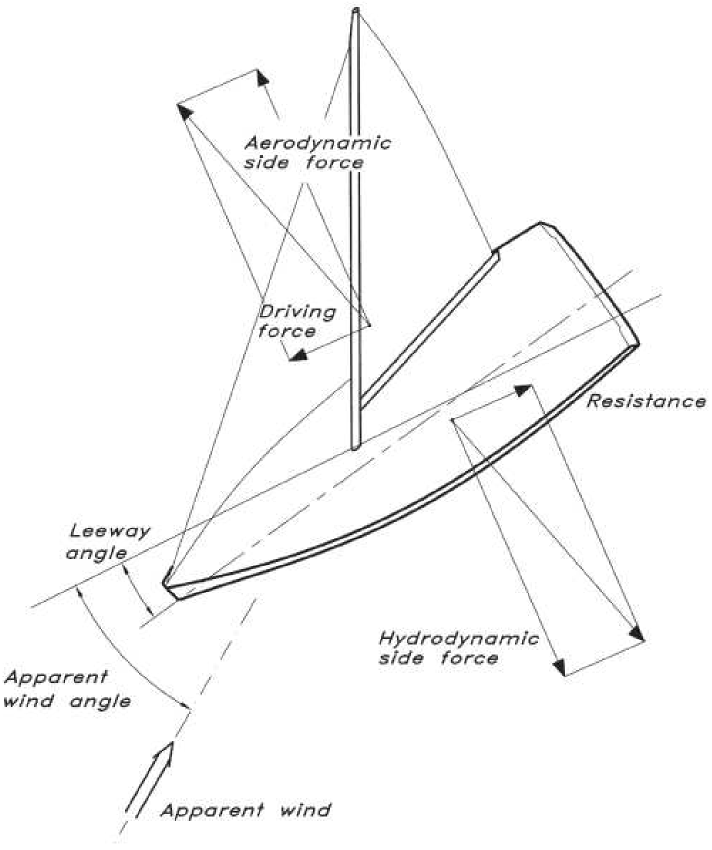


Figure 13 Forces acting on a sailing yacht - top view (15)

To better understand this diagram, another fundamental concept must be introduced, the *Apparent Wind*, which is described by *Apparent Wind Angle (AWA)* and *Apparent Wind Speed (AWS)*. The Apparent wind can be described as the relative wind perceived by the sail when the boat is moving, and it is a combination of the *True Wind,* which is the wind perceived in the surrounding environment standing still and the wind created by the motion of the boat. The True Wind is defined by *True Wind Speed (TWS)* and *True Wind Angle (TWA)*. The formulas expressing these two quantities are reported below:

$$AWA = \operatorname{acos}\frac{TWS * \cos(TWA) + V}{\sqrt{TWS^2 + V^2 + 2 * TWS * V * \cos(TWA)}}$$

$$AWS = \sqrt{TWS^2 + V^2 + 2 * TWS * V * \cos(TWA)}$$

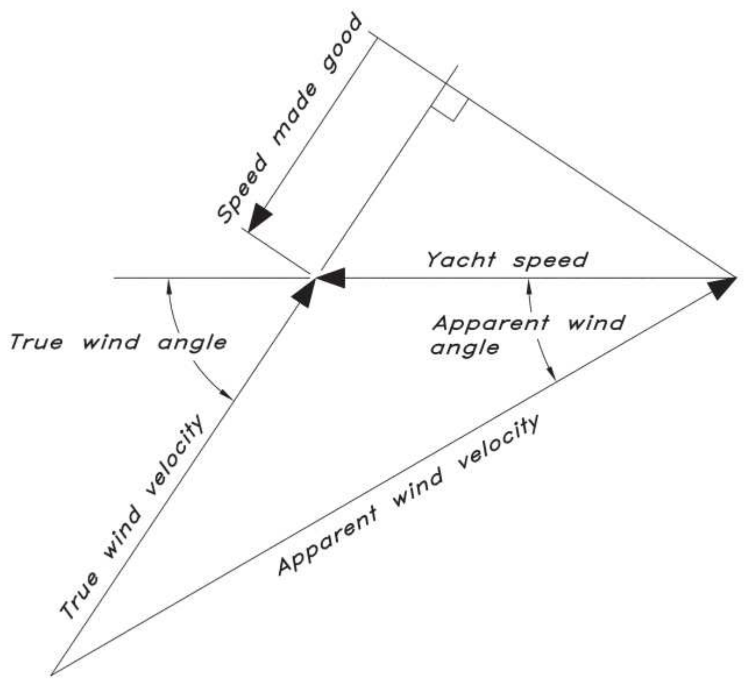


Figure 14 Apparent Wind Calculations (15)

The daggerboard is a component of a sailboat found at the bottom of the hull, with a symmetric airfoil section, whose purpose is to generate a lift force that counteracts the sail force. Since it is operating in water and not in the air, the fluid density is a thousand times higher, meaning that to generate a comparable force to the sail it requires lower angles of attack, lower velocities, and lower surfaces. The lift of the daggerboard is perpendicular to the boat velocity (that corresponds to the velocity of the fluid perceived by the daggerboard) and through a vectorial summation with the sail lift, the thrust force is obtained.

As most of the components installed a sailboat, the daggerboard presents a symmetric airfoil, which means that in order to generate a lift force it requires an angle of attack, corresponding to the so called *leeway angle*, the angle between the actual direction of the boat and the direction the boat is pointing to, and it is caused by the sideways force of the wind of the sails (which pushes the boat slightly off-course).

The thrust of the sail is counteracted by the resistance of the hull and of the various hydrodynamic appendages, which act opposite to the boat's velocity. The main contributor to the drag force is hull resistance, which presents the same components illustrated for airfoil sections but with the addition of *wave resistance*, proportional to the energy lost due to waves generated by the hull moving through water.

4.5 Foiling

The difference between a foiling boat and a traditional one is the fact that the former one is "flying over of the water" thanks to *hydrofoils* (which will hereafter be referred to as *foils*). This technology has been known since the 1920s, but just in the last two decades got more traction thanks mainly to the America's Cup.

Foils can be described as wing-like structures mounted under the hull, to lift the boat out of the water as it gains speed. The idea behind foiling boats is to drastically reduce hull resistance and improve the velocity of the boat (less drag means more forward speed for the same thrust) and allowing also for a smother rider in wavy sea conditions. Naturally, this comes at a price: foils generate not only lift but also resistance, so when the hull is not completely out of the water drag at low velocity is actually higher compared to a conventional boat, but as soon as it gains some speed, the lift produced by the foils lift the hull out of the water and resistance is decreased significantly.

Foils are named and classified after the alphabetic letters that resemble, as displayed in Figure 15, taken from (15). The choice of the airfoil shape and of the foil as a whole, depends on various factors, such as desired speed, manoeuvrability, easiness to control and efficiency.

There are primarily two types of hydrofoils, *surface piercing* and *fully submerged* ones (15).

- Surface piercing foils form a V-shape and during operation break the water surface, thus the name. They are inherently stable because ride height can be controlled passively, so no active control system is required. The height is automatically adjusted so that the submerged part of the foil has just enough lift to carry the weight of the boat, which means that the higher the speed, the smaller the part of the foil in the water.

- T-foils are the simplest example of fully submerged foils, and in this case, the lifting surface is independent of ride height, which means that an active control needs to be implemented. It must be said that there is a weak control mechanism caused by the free-surface effect of the water, but it is not sufficient to be exploited for height regulation purposes. A typical active control systems mounted on T-foils is a flap, that is controlled starting inputs of sensors that measure the ride height.

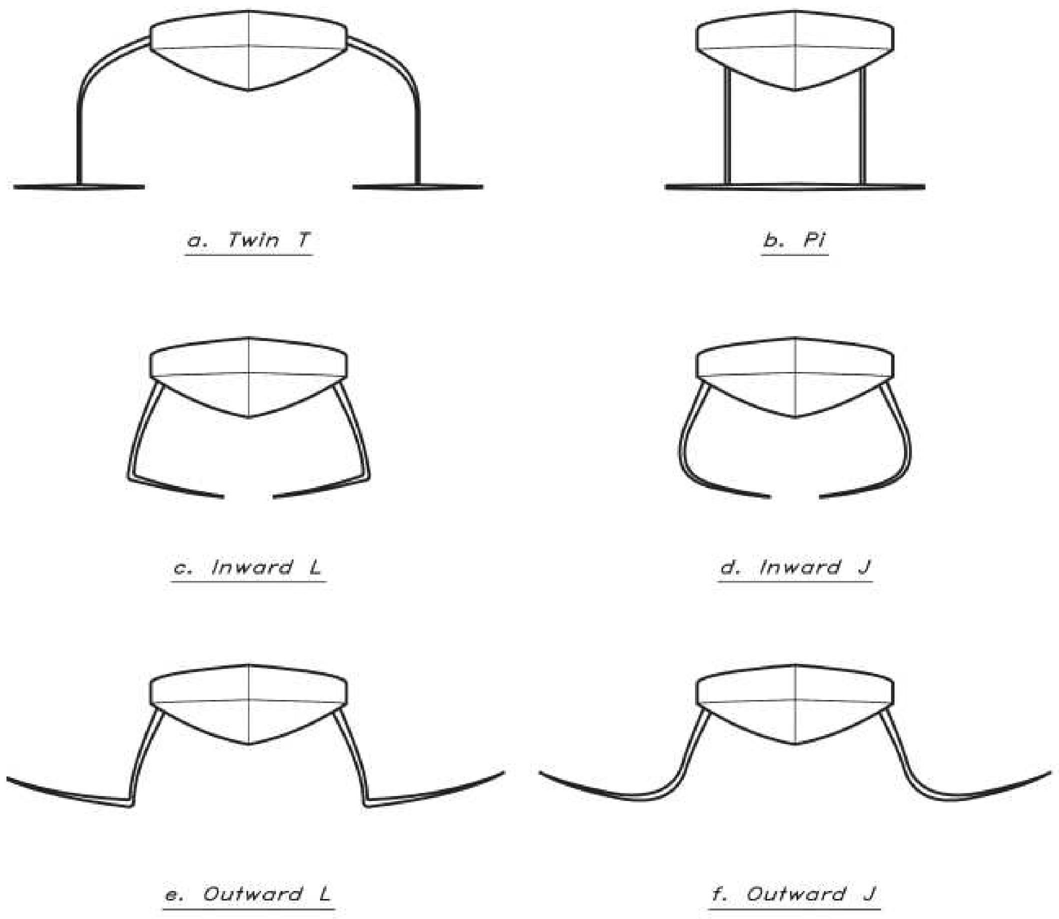


Figure 15 Different foil shapes and relative names (15)

4.6 Design optimisation

Sailboat design is a very specific niche of naval architecture due to the need to account for the presence of sails that apply a driving force and a heeling moment to the yacht. These forces affect the stability, motions and appendage design of the vessels, which translates to a very complex set of constraints and often conflicting objectives that need to be taken into consideration during the optimisation process. An ideal racing yacht design must add further complexity by accounting for different performance characteristics, both upwind and downwind, in a variety of sea and wind conditions.

Despite significant investments in computational tools, design optimization remained largely manual and reliant on designers' expertise, intuition and incremental adjustments up until not so many years ago, due to computational complexity and resources demands. Progress in automating design space exploration – systematically testing parameters to identify optimal configurations was thus needed.

Generally, an optimisation problem can be formulated as:

$$min\ F(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{R}^n$$

$$h(\boldsymbol{x}) < 0$$

$$g(\boldsymbol{x}) = 0$$

where $\boldsymbol{x} = (x_1, x_2, \dots, x_n)^T$ is the vector of independent variables that can be modified and describe the optimisation problem. The *design variables* are subject to *constraints* (equality $h_j(\boldsymbol{x})$ and inequality $g_k(\boldsymbol{x})$ constraints) that define the boundaries of the design space and are evaluated using an *objective function* $F(\boldsymbol{x})$ (16).

The aim is to push the design of rigid wing sails directly to the optimal zone of the design space, requiring as few iterations and incremental adjustments as possible. In Chapter 6 the flowchart of the whole optimisation process is shown, which is split into two parts, the creation of the surrogate model and the optimization loop itself.

4.7 Surrogate-Based Design Optimisation (SBDO)

In order to perform a useful optimisation within a reasonable time using readily available computer hardware, the number of expensive simulations and test must be reduced to the bare minimum. It is common knowledge that CFD simulations can be extremely computationally expensive, especially in optimisation processes where evolutionary algorithms are adopted and imply testing of thousands of individuals.

Surrogate-based modelling optimisation is a technique used to simplify and speed up the process of optimising complex systems, particularly in cases where the high-fidelity simulations are computationally expensive, like CFD. The idea behind this approach is to build a model that approximates the real model from a small number of sampled data. This approximation model, called *surrogate model* can be calculated in advanced using chosen samples tested on the high-fidelity model, or alternatively, it can be calculated in real time, by progressively being refined using additional samples as the optimisation proceeds.

For a problem in *m-dimensional* space, it is assumed that the focus is on predicting the output of a high-fidelity, costly computer simulation that corresponds to an unknown function $y: \mathbb{R}^m \to \mathbb{R}$. By running these simulations, y is observed at n sites (determined by the DOE, explained in Design of Experiments (DOE)).

$$\boldsymbol{S} = \left[\boldsymbol{x}^{(\boldsymbol{1})}, \dots, \boldsymbol{x}^{(\boldsymbol{n})}\right]^T \in \mathbb{R}^{n \times m}, \quad \boldsymbol{x} = \{x_1, \dots, x_m\} \in \mathbb{R}^m$$

$$\boldsymbol{y_S} = \left[y^{(1)}, \dots, y^{(n)}\right]^T = \left[y\left(\boldsymbol{x}^{(\boldsymbol{1})}\right), \dots, y\left(\boldsymbol{x}^{(\boldsymbol{n})}\right)\right]^T \in \mathbb{R}^n$$

The pairs $(\boldsymbol{S}, \boldsymbol{y_S})$ denotes the sampled data in the vector space (17).

The objective is to build a surrogate model for predicting the output of the "expensive-to-evaluate" simulations for any untried sample $\boldsymbol{x}$ based just on the least possible number of sampled data sets $(\boldsymbol{S}, \boldsymbol{y_S})$, and without compromising the desired accuracy of the results (17).

4.8 Design of Experiments (DOE)

A surrogate model is built on sampled data to obtain an approximation of the cost function of the problem, sufficient to predict the output of an expensive computer code at untried points of the design space. Sample points can be extracted by using a *Design of Experiments* (*DOE*) technique, a procedure that aims at extracting the maximum amount of information from a limited number of sample points. (17) and (16) provide a very detailed description of the most common sampling techniques, which can be classified into two categories, "*classic*" and "*modern*".

CLASSICAL	*MODERN*
Full Factorial	*Latin Hypercube Sampling*
Central Composite Design	*Orthogonal Array Design*
Box – Behnken	*Uniform Design*
D – Optimal Design	*Taguchi*

Table 1 Sampling algorithms

The Latin Hypercube Sampling (LHS) divides the range of each design variable into equal intervals, effectively partitioning the design space into distinct subspaces. By allowing only one sample per subspace, it ensures that the samples do not overlap, leading to a well-distributed and non-redundant set of points.

LHS is particularly useful in applications such as simulation, uncertainty quantification and sensitivity analysis, where it helps to obtain accurate estimates with fewer samples than traditional methods. Its efficiency and effectiveness in representing complex systems make LHS a valuable tool in engineering, risk assessment and other fields where robust statistical analysis is required.

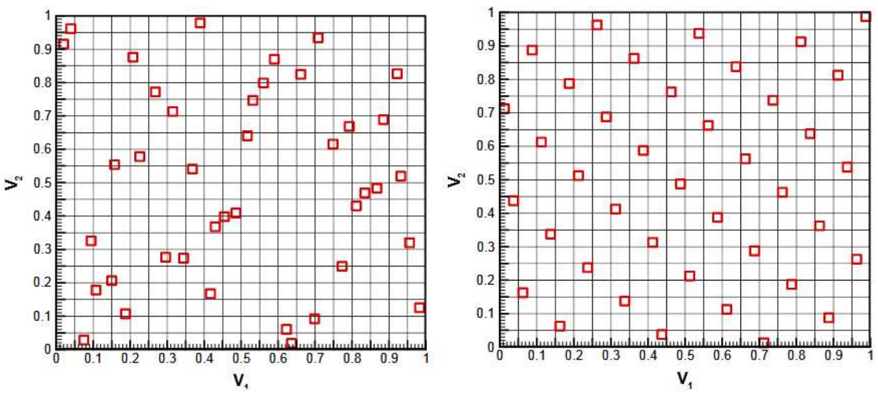


Figure 16 Example of distribution of 40 sample points for a two-dimensional problem (left: LHS; right: Uniform Design) (16)

The number of samples to be added to the DOE is proportional to the complexity of the problem, the number of variables, the number of outputs and the desired accuracy of the surrogate model. There’s no general rule to determine the precise number of samples required, but a rule of thumb commonly adopted is to aim at 10 samples per variable, but it is highly-context dependent. One of the most influential constraints on the number of samples is the available computational resources, since the goal is to obtain enough data to capture all the variables' influences while minimising the cost of simulation.

Another possible approach for determining the number of training data points for surrogate models is known as “*Adaptive Sampling*”. To maximize the efficiency of available computational resources, training points are not preselected using a fixed sampling plan. Instead, they are dynamically added to the dataset based on a specific criterion, so in a way, the surrogate model itself guides the placement of these training points.

Another way to reduce the burden of computational cost is the adoption of multi-fidelity methods, models that are trained on a combination of data from two or more fidelity (accuracy) levels, but these models are beyond the scope of this work (18).

4.9 Surrogate Model candidates

There are a number of surrogate modelling methods in optimisation literature, each one with its advantages and disadvantages. (17) and (16) provide a comprehensive analysis of each one of these methods, being:

- RSM
- Kriging
- Gaussian process
- Artificial Neural Network
- Radial Basis Function
- Support Vector Machine

Different surrogate models have some advantages and disadvantages, and each will show better features for different engineering problems. After a prolonged literature review, no single surrogate model was found to be the most effective for all problems. A deep analysis of each method is beyond the scope of this work, so only the details of the most common ones are reported.

4.9.1 Response Surface Method

The expression *Response Surface Method* (RSM) is usually used to refer to low order polynomial approximation models, especially quadratic polynomial models. Second order polynomials generally offer the best compromise between the modelling accuracy and computational expense, when compared with the linear or higher order polynomial models.

A significant advantage of such models is the capability of smoothing out numerical noise in the data while capturing the global trend of the variation, making it very robust, thus well suited for optimisation problems.

The RSM can be formulated in the following form:

$$y(\boldsymbol{x}) = \hat{y}(\boldsymbol{x}) + \epsilon$$

where $\hat{y}(\boldsymbol{x})$ is the quadratic polynomial approximation and ϵ is the random error (normally distributed with zero mean and variance σ^2.

$$\hat{y}(\boldsymbol{x}) = \beta_0 + \sum_{i=1}^{m} \beta_i x_i + \sum_{i=1}^{m} \beta_{ii} x_i^2 + \sum_{i=1}^{m} \sum_{j\geq 1}^{m} \beta_{ij} x_i x_j$$

In this formulation, $\beta_0, \dots, \beta_{ij}$ are the unknown coefficients to be determined (17).

4.9.2 Artificial Neural Network

An *Artificial Neural Network* (ANN) is a network composed of multiple simple processors, known as units or *neurons*, interconnected by communication channels or connections that facilitate the transmission of encoded numeric data. The units operate exclusively on the local data and input they receive via the connections, applying a multiple linear regression followed by a non-linear transformation on the output value. This transformation, or activation function, is most commonly a sigmoidal function (16).

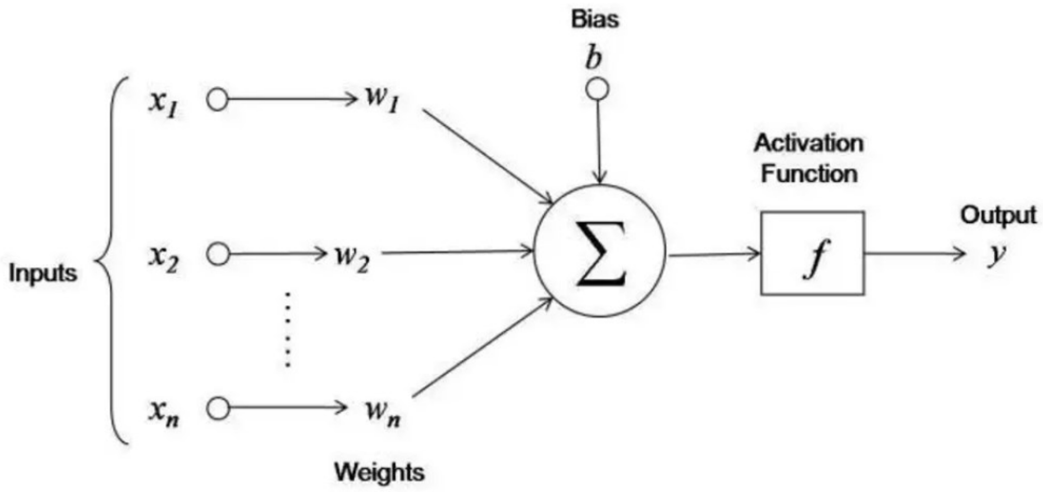


Figure 17 Neuron with activation function (19)

$$\eta = \sum_{i=0}^{n} w_i x_i + \beta$$

$$y = \frac{1}{1 + e^{-\eta}}$$

$\{x_1, x_2, \dots, x_n\}$ are the inputs to each neuron, the regression coefficients are designated by the weights $\{w_1, w_2, \dots, w_n\}$, and β is the "bias value" of the neurons.

Neural network can present in different architectures and topologies, but the most commonly found in engineering applications is the *Multi-Layer Perceptron* (MLP). This type of network shows an input layer, one or more hidden layers and one output layer. Both the number of neurons and the number of layers can vary and need to be optimised.

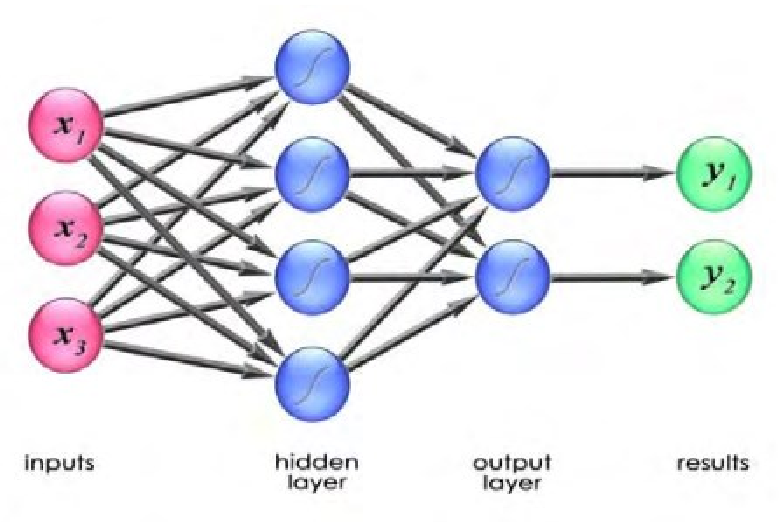


Figure 18 Multi-Layer perceptron (16)

4.9.3 Kriging

Kriging model, also known as *Gaussian process model*, is an interpolation technique based on covariance function. It is based on the assumption that the true response can be modelled as

$$Y = \sum_{i=0}^{m} \beta_i f_i(\boldsymbol{x}) + Z(\boldsymbol{x})$$

where $f_i(\boldsymbol{x})$ is the regression basis function (generally a constant or low-order polynomial), β_i denotes the corresponding coefficient, $Z(\boldsymbol{x})$ is a stochastic process with zero mean, and covariance given by

$$Cov\big(Z(\boldsymbol{x}), Z(\boldsymbol{x}')\big) = \sigma^2 R(\boldsymbol{x}, \boldsymbol{x}')$$

with $R(\boldsymbol{x}, \boldsymbol{x}')$ correlation function dependent only on Euclidean distance between $\boldsymbol{x}$ and $\boldsymbol{x}'$ in the design space. Usually, a Gaussian exponential correlation function is adopted, in the form

$$R(\boldsymbol{x}, \boldsymbol{x}') = \exp\,[-\sum_{i=1}^{m} \boldsymbol{\theta}_k\, |x_k - x_k'|^{p_k} \quad , \qquad 1 < p_k \le 2$$

where θ_k and p_k are the tuneable hyperparameters of the model.

The Kriging model is composed of a parametric regression model and non-parametric random process. Therefore, the Kriging model is more flexible than the general parametric model, and at the same time, it overcomes the limitation of non-parametric model in processing high dimensional data and has stronger prediction ability.

4.9.4 Radial Basis Function (RBF)

Radial Basis Function networks (*RBF*) show a very similar structure to Multi-Layer Perceptron networks, but they use a different activation function. Instead of sigmoidal activation function, RBFs use linear combinations or radially symmetric basis functions that can be modelled as

$$y(\boldsymbol{x}) = \sum_{i=1}^{N} w_i \phi \, || \boldsymbol{x} - \boldsymbol{c_i} ||$$

with $\boldsymbol{x}$ is the vector of inputs, w_i are weighting coefficients, $\boldsymbol{c_i}$ are RBF centres, $|| \cdot ||$ denotes the Euclidean norm, and ϕ is typically a Gaussian function (16)

$$\phi||\boldsymbol{x_i} - \boldsymbol{c_i}|| = \exp\left[-\beta\left||\boldsymbol{x} - \boldsymbol{c_i}|\right|^2\right]$$

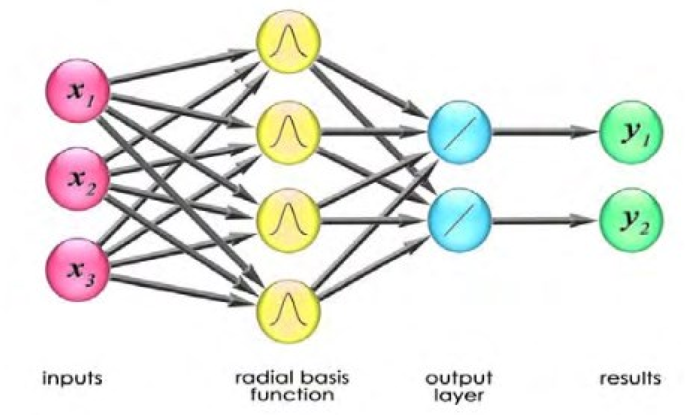


Figure 19 RBF architecture (16)

4.10 Surrogate model training

To estimate the test error associated with fitting a particular statistical learning method on a set of observations, the validation set approach can be employed, a method which involves randomly dividing the available data into two parts: a *training set* and a *validation* (or *hold-out*) *set*. The model is trained on the training set, and its performance is evaluated by predicting the responses for the observations in the validation set. The resulting error rate gives an idea of the accuracy of the model's prediction capability.

The validation set approach is conceptually simple and easy to implement, but it shows a major drawback: the test error can be highly variable depending on which points are included in the training set, and which are included in the validation set (20).

In the following section a refinement of the validation set approach that addresses this issue is presented, *cross-validation*.

4.10.1 Leave-One-Out Cross-Validation

Leave-one-out cross-validation *(LOOCV)* is a specific type of cross-validation where the number of folds equals the total number of data points in the dataset. A single observation (x_1, y_1) is used for the validation set, and the remaining observations $\{(x_2, y_2), \dots, (x_n, y_n)\}$ make up the training dataset. The surrogate model is trained on $n - 1$ observations (where n is the number of total observations) and a prediction $\hat{y}$ is made on the sample x_1.

The $Mean\ Squared\ Error\ MSE_1$ is an unbiased estimate for the test error, since x_1 was not used in the fitting process. This procedure is repeated selecting (x_2, y_2) as new validation set, $\{(x_1, y_1), (x_3, y_3), \dots, (x_n, y_n)\}$ as training set and computing $MSE_2 = (y_2 - \widehat{y_2})^2$, and so on for all n points.

LOOCV estimate for the test MSE is the mean value of all n test error estimates

$$CV_{(n)} = \frac{1}{n}\sum_{i=1}^{n} MSE_i$$

This method maximises the training data used in each iteration, which can lead to an almost unbiased estimate for the model's performance. However, it is computationally intensive, especially for large datasets. Another great advantage of this approach is the fact that there is no randomness associated with the training/validation set splits, in fact, this approach tends not to overestimate the test error rate as much as the validation set approach does (20).

4.10.2 $k - fold$ Cross-Validation

An alternative approach to *LOOCV* that involves randomly dividing the set of observations into k groups, or folds, of approximately equal size. For each iteration, one-fold is reserved as the test set while the remaining $k - 1$ folds are used to train the model. Once the model is trained it is evaluated on the test fold, and MSE_i is computed on the observations in the held-out fold.

This procedure is repeated k times and each time, a different group of observations is treated as a validation set, resulting in k estimates of the test error, $MSE_1, \dots, MSE_k$. The k-fold CV estimate is computed by averaging these values

$$CV_{(k)} = \frac{1}{k}\sum_{i=1}^{k} MSE_i$$

LOOCV can be considered a particular case of k-fold CV, in which $k = n$. The advantage is quite simple: computational effort required. Typical values of k are $k = 5$ and $k = 10$.

Although it may seem a little counterintuitive, apart from the computational cost k-fold presents another advantage, that is related to $bias - variance$ trade off.

4.10.3 Bias-Variance trade off

Bias-variance trade-off is a fundamental concept in machine learning, that explains the balance between two sources of error in predictive models.

Bias is the error arising from oversimplified assumptions in the model. High-bias models underfit, failing to learn meaningful patterns in the data, while variance is an error caused by excessive sensitivity to training data fluctuations. High-variance models overfit, memorising noise instead of generalizable trends and performing poorly on new data.

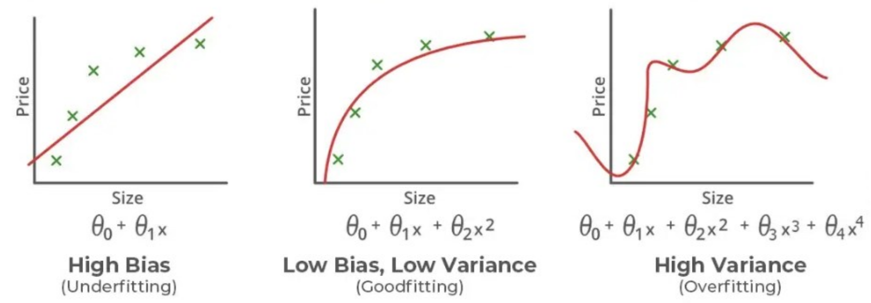


Figure 20 Explanation of Bias-Variance trade-off (21)

The trade-off lies in optimising model complexity to reduce total error, represented mathematically as:

$$Total\ Error = (Bias)^2 + Variance + Irreducible\ Error$$

K-fold cross-validation presents a bias-variance trade-off when estimating test error. Although $LOOCV$ uses nearly the entire dataset for training and thus provides nearly unbiased error estimates, it suffers from high variance because the models, trained on nearly identical data, produce highly correlated predictions. In contrast, k-fold CV (with $k < n$) uses training sets that are slightly smaller, which introduces some bias, but the resulting models are less correlated, leading to lower variance in the error estimates. Empirical evidence suggests that using $k = 5$ or $k = 10$ offers a good balance between bias and variance, resulting in more accurate test error estimates overall (20).

5. Glaros' description

As already mentioned, the case study for this work has been chosen being Glaros, a R3 class foiling skiff, developed and produced by the Polito Sailing Team, competing in the "*1001 Vela Cup*", a student regatta organised in Italy every September.

Glaros is mounting two J-foils (outward) and a T-foil, that serves as the rudder as well. Figure 21 shows Glaros in action flying above the water in its current set up, while Table 2 reports the main dimensions and measurements.

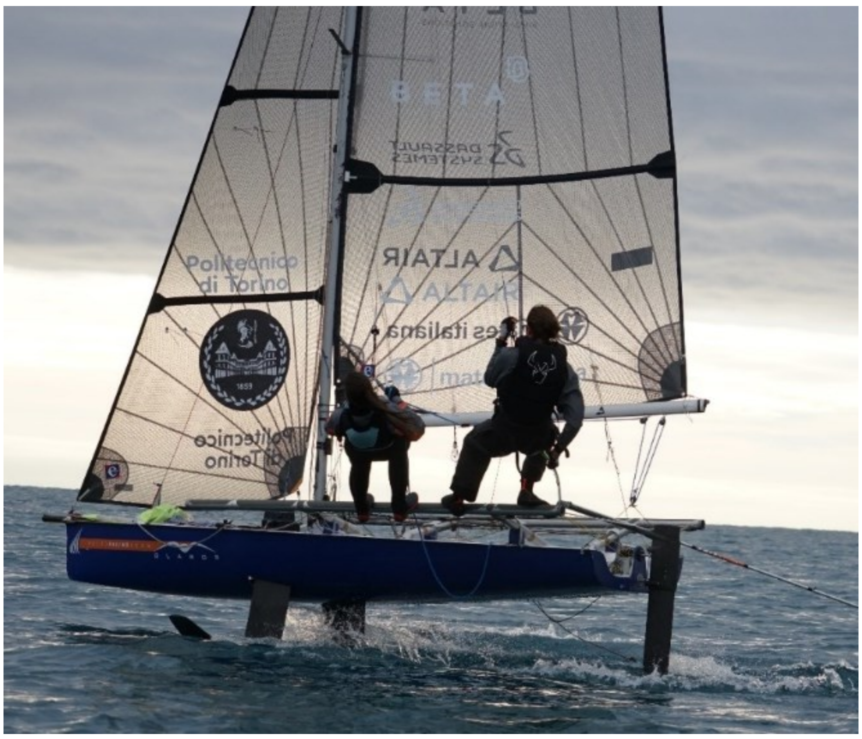


Figure 21 Glaros flying

Figure 24, Figure 23 and Figure 24 show different views of Glaros. They are intended to give the reader a comprehensive idea of the boat.

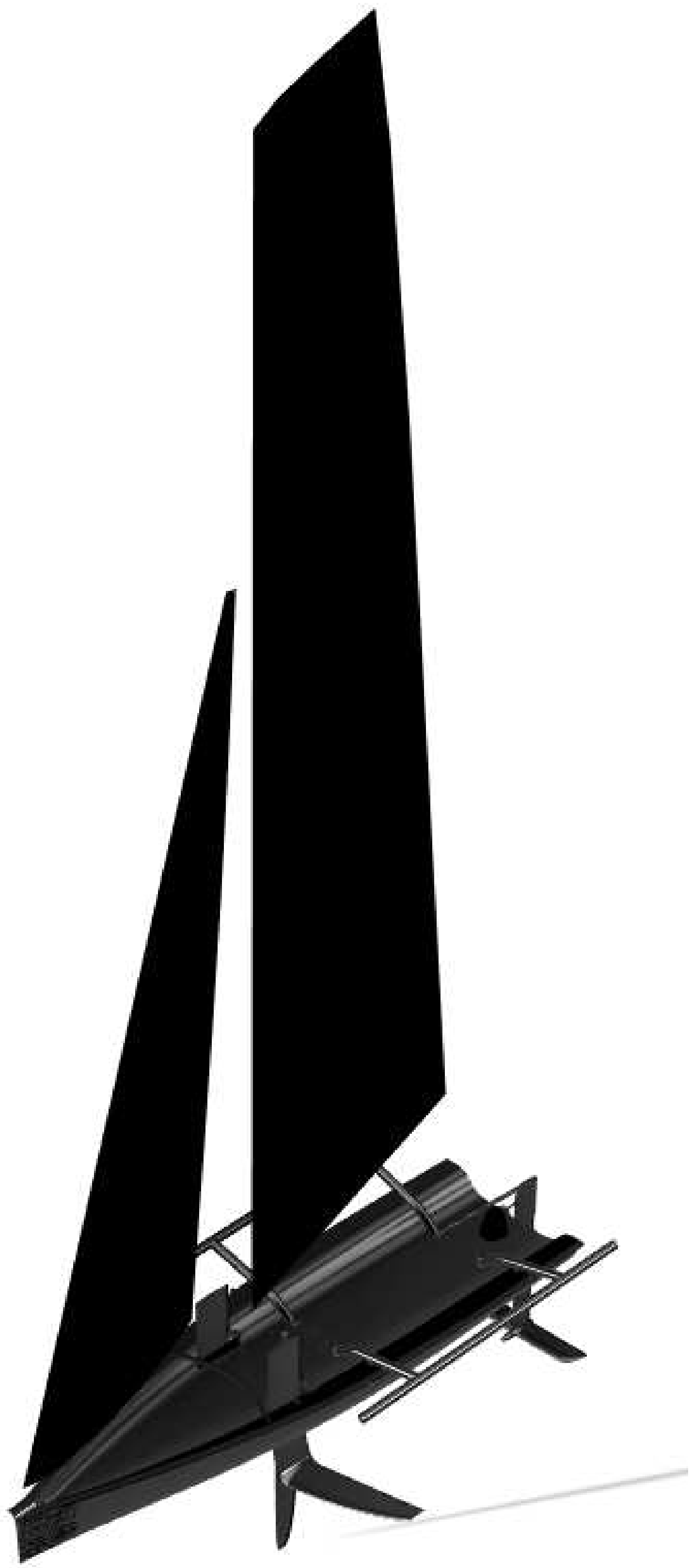


Figure 22 Glaros rendering

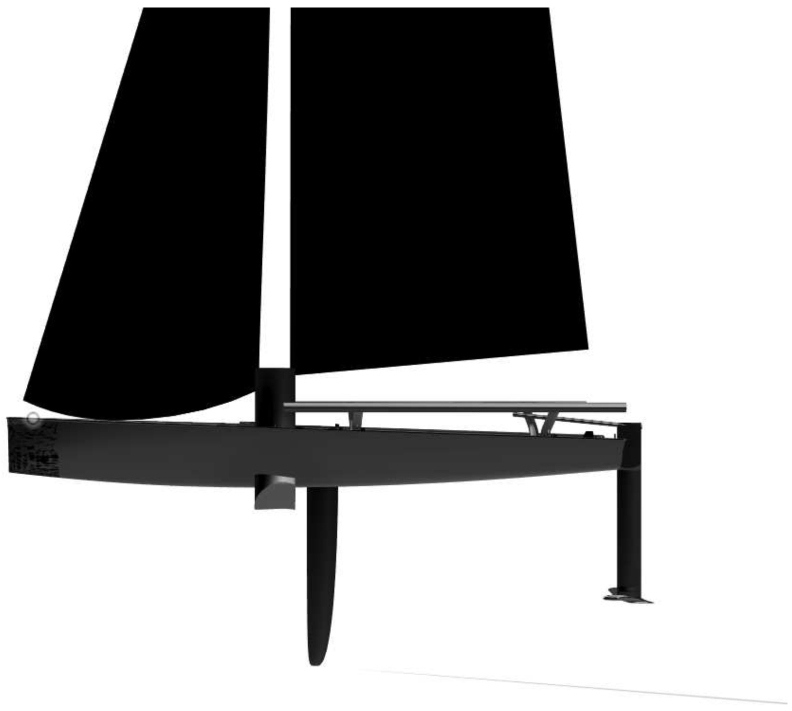


Figure 23 Side view of Glaros

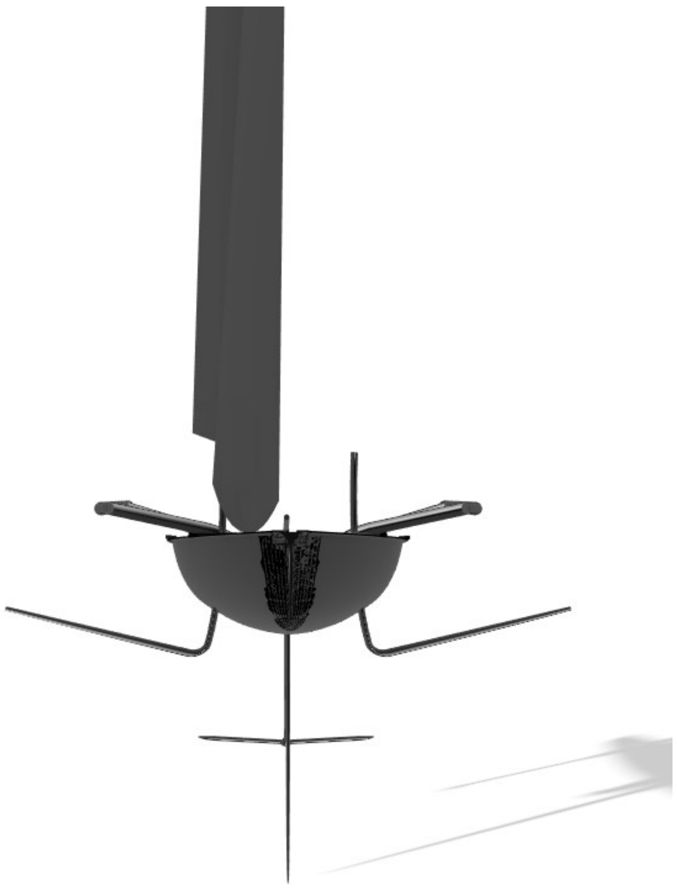


Figure 24 Front view of Glaros

$DIMENSION$	$VALUE$
$Length\ Over\ All\ (LOA)$	$4.62\ m$
$Length\ Water\ Line\ (LWL)$	$4.45\ m$
$Max\ Beam$	$1.34\ m$
$Draft$	$0.12\ m$
$Mass$	$124.5\ kg$
$Sailor's\ mass$	$70\ kg$
$N°\ of\ sailors$	2
$N°\ of\ J-foils$	2
$N°\ of\ T-foils$	1

Table 2 Glaros' measurements

The total lift generated by the foils is equal to the weight of the whole boat (that consists of hull weight and rigging) and the two sailors, split between the two J-foils and the rudder. The distribution is 72% on the J-foils and 28% on the rudder.

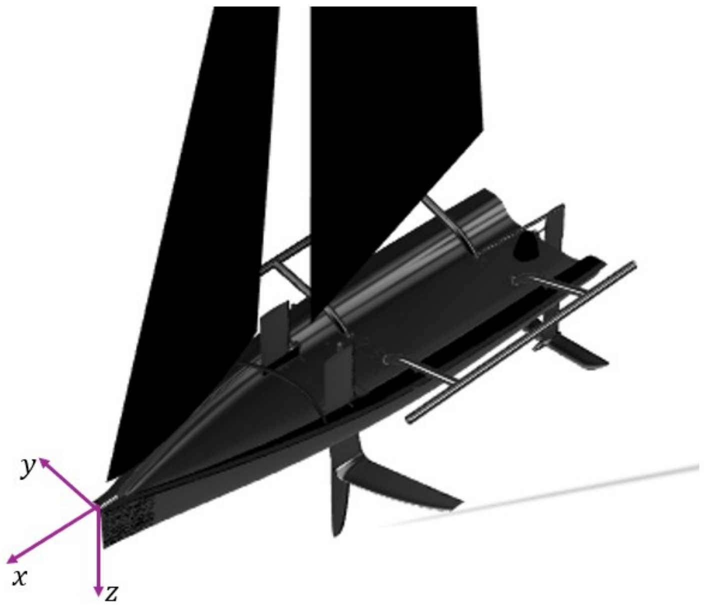


Figure 25 Local reference frame for the boat's dimensions

Other relevant information is contained in Table 3, in which the positions of the CoG of the main components are given with respect to a reference frame whose origin is on the deck at the bow (see Figure 25), x axis aligned with the symmetry axis of the boat pointing forward, z axis on the symmetry plane perpendicular to the x axis pointing downward, and y axis perpendicular to both x and z axes and point to starboard (right-hand side).

$Component$	$X\ coordinate\ [m]$	$Y\ coordinate\ [m]$	$Z\ coordinate\ [m]$
$J\ foil\ DX$	-2.15	0.4	1.1
$J\ foil\ SX$	-2.15	-0.4	1.1
$Rudder$	-4.7	0	1.34
$Daggerboard$	-2.42	0	0.87
$Boat\ CG$	-2.6	0	0.10

Table 3 Main components coordinates

In order to compute the Centre of Gravity of the whole system the sail and the sailors' must be accounted for, but they are not reported in Table 3 because the local CoG of the two elements is a function of design variables used in the optimisation process, thus not reported in this section.

Figure 29 and Figure 29 report the main dimensions of the J-foils and rudder respectively.

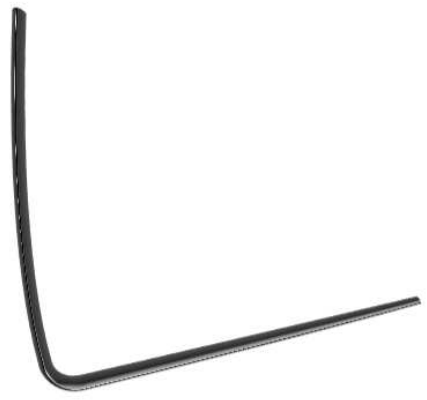


Figure 26 Glaros J foil – front view

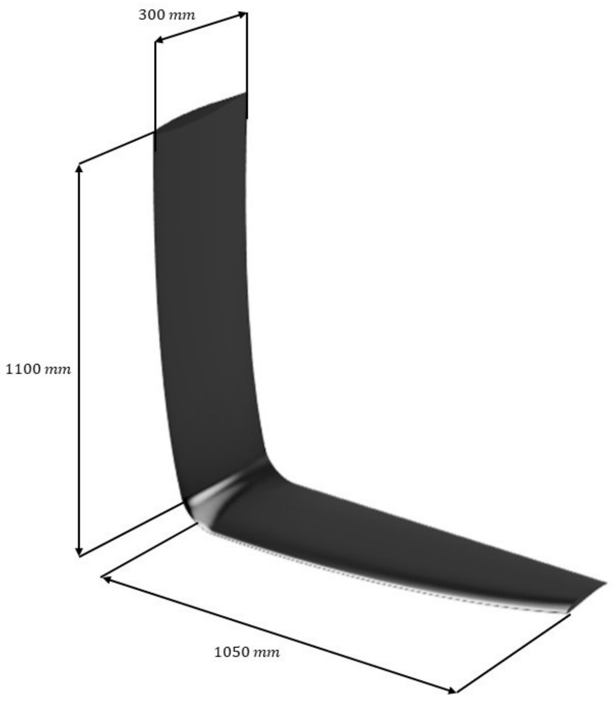


Figure 27 J foil main dimensions

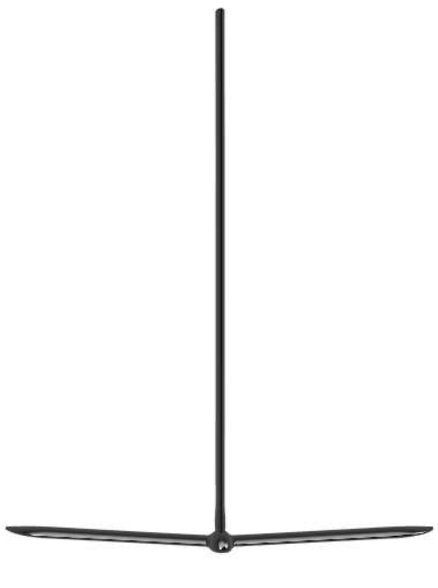


Figure 28 Rudder foil – front view

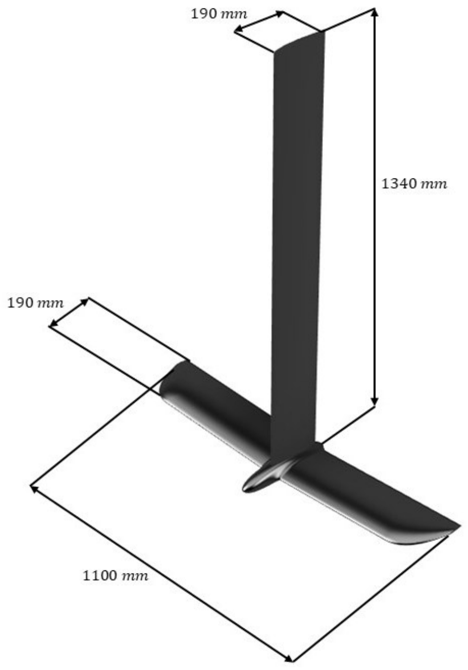


Figure 29 Rudder foil main dimensions

Glaros was conceived with a soft sail plan of 33 m^2, imposed by the 1001 Vela Cup rulebook, distributed on of three sails: a mainsail, a jib and a gennaker. The respective surfaces are reported in Table 4 Sails surface areasTable 4.

$SAIL$	$SURFACE - [m^2]$
$Mainsail$	16
Jib	6
$Gennaker$	11

Table 4 Sails surface areas

This distribution translated into a design constraint for the wing sail as well: the maximum surface of the wing sail has been thus imposed being equal to 16 m^2.

6. Optimisation Workflow

The workflow is split into two branches, one devolved to the creation of the surrogate model inputted to the VPP, and the other carrying out the optimisation itself. Figure 30 reports a flowchart displaying all the steps of the algorithm.

The training begins with the generation of the DOE via a Python script exploiting a specific library. All designs are then simulated with CFD to extract aerodynamic coefficients, which serve as the model's output variables the training is carried out on. In order to minimise the number of individuals and still gather the right amount of data, an adaptive sampling algorithm is implemented and through Bayesian optimisation, the best combination of hyperparameters in pinpointed.

Once the surrogate model has obtained the desired accuracy, it is used as input in Glaros' VPP to predict sail performance. A genetic algorithm is used to explore globally the Design Space of the geometric parameters, and each individual generated is tested with the VPP and highest reachable VMG is evaluated, iterating until the algorithm converges.

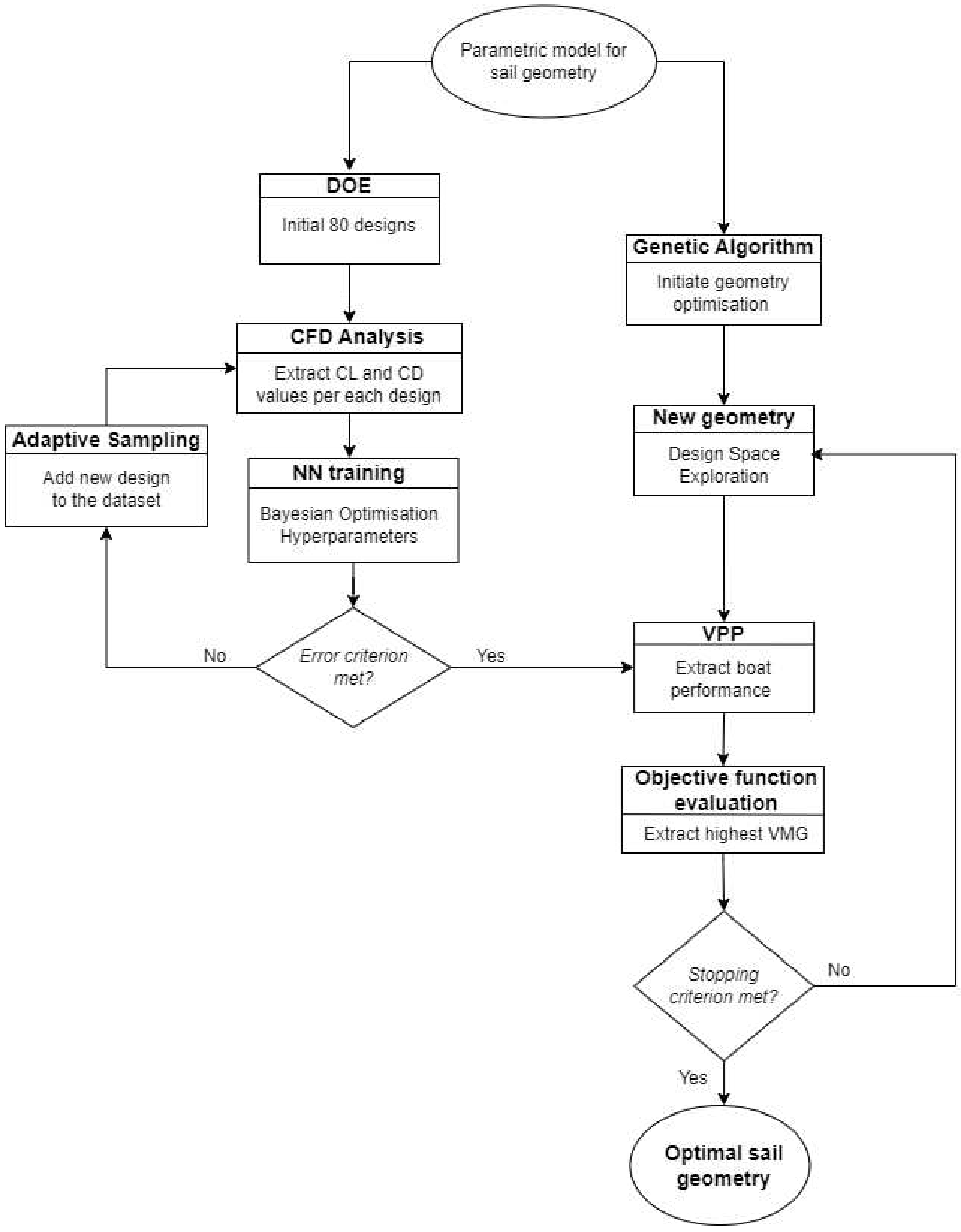


Figure 30 Optimisation flowchart

7. Geometry creation

For the previously explained framework to work, it is necessary to have a method that creates from scratch a complete sail geometry (*parametric modelling*), or alternatively, be able to deform an existing design to match the design parameters required (*parametric transformation*).

In literature several approaches are proposed to parametrise a wing sail geometry. In (16) various technique are reported for hull shape optimisation, but nothing prevents applying the same concepts for sail geometries as well.

When dealing with different geometry testing, a very popular approach is the mesh morphing technique, which has the main advantage of operating directly in the numerical domain in the simulation environment, as shown in (22), but it wasn't quite suitable for this application.

Another interesting approach is proposed in (23), where a simplified CAD-based parametric model developed in Rhino 3D- Grasshopper and based on B-spline curves is presented, though being applied on a soft sail.

Regarding rigid wing sail geometries, two main approaches have been analysed, described in (24) and (25). In the former one, the parameters reported in Figure 31 are used to explore the effect of chord length ratio in a two elements wing sail, meanwhile in the latter an aerodynamic analysis with URANS and LES simulations are carried out.

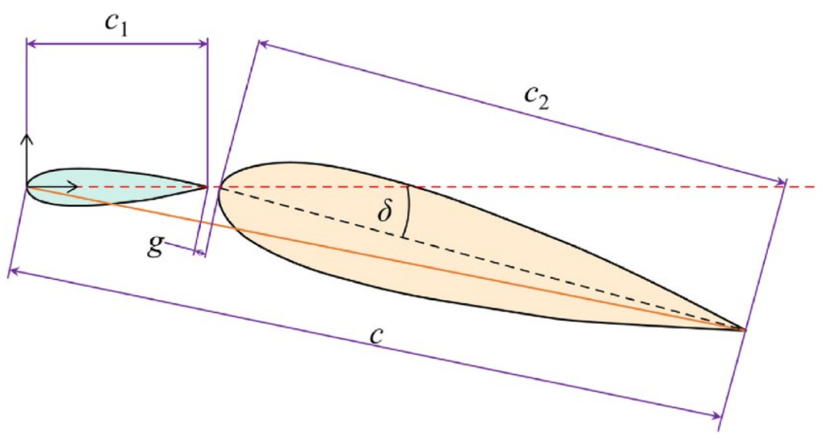


Figure 31 Geometry parametrisation adopted in (24)

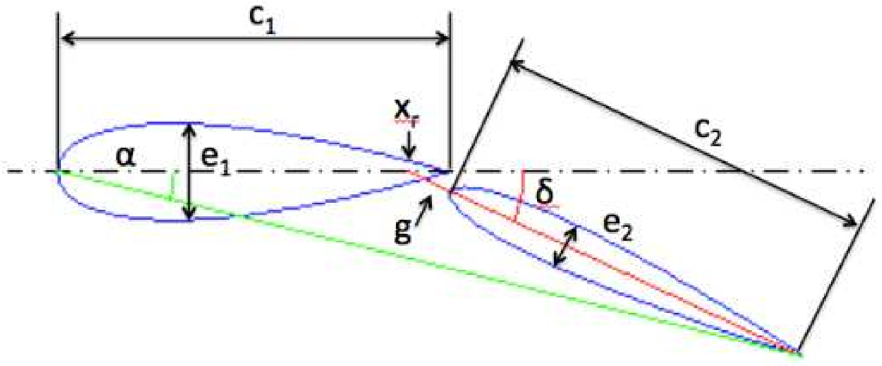


Figure 32 Geometrical parameters adopted in (25)

The final design variables chosen for Glaros' sail are those of Figure 32, schematised in Table 5.

$PARAMETER$	$SYMBOL$
$Main\ element\ chord$	$C1$
$Flap\ element\ chord$	$C2$
Gap	G
$\%\ C1$	PR
$Airfoil\ section\ main$	$E1$
$Airfoil\ section\ flap$	$E2$
$Boom\ angle$	α
$Flap\ deflection$	δ

Table 5 Final parameters adopted

$E1$ and $E2$ are not properly treated as thicknesses of the airfoils, but they are integer values used as indexes to choose different *txt* files from which various airfoils can be imported in the CAD modeller to create different wings. In this particular case, the algorithm can choose between four different symmetric NACA airfoils, precisely $NACA0012$, $NACA0015$, $NACA0018$ and $NACA0020$.

In order to carry out an automatised surrogate model creation process, a Python script has been developed for the open-source *FreeCAD* tool, a general-purpose parametric 3D CAD modeller.

Python's most widespread use is as scripting language embedded in applications, like *FreeCAD*, and it can be run from Python console or from customs scripts, called "*macros*".

For the purpose of this thesis, a custom Python script has been developed, and it is structured in the following way:

- Importing of necessary libraries
- Definition of geometrical parameters and relative boundaries
- DOE creation with sampling algorithm
- Execution of geometry creation function
- Export of DOE parameters, input files for CFD simulations and CAD models' export

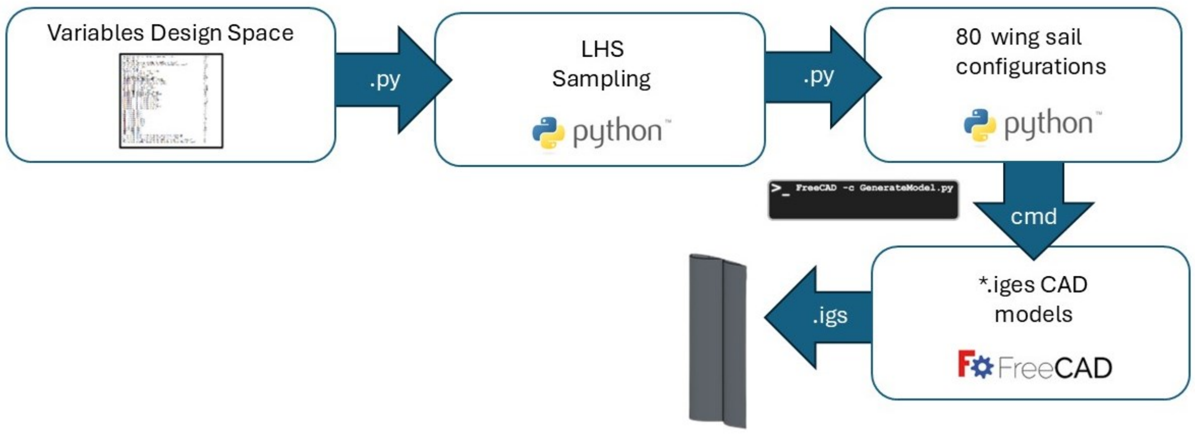
The first step is to add FreeCAD paths to "*sys.path*", a list of directories where Python looks for modules when import statement is used. Then "*numpy*" library is imported together with the "*smt*" library, that stands for "*Surrogate Modelling Toolbox*", an open-source package including libraries for surrogate models fitting, sampling methods and benchmarking problems. This library is necessary to recall the LHS method and generate the initial geometries. Table 6 reports the upper and lower boundaries of each parameter.

$PARAMETER$	$LOWER\ BOUNDARY$	$UPPER\ BOUNDARY$
$C1$	950	1550
$C2$	800	1200
G	10	50
PR	0.5	1
$PROFILE1$	1	4
$PROFILE2$	1	4
α	-15	0
δ	-15	0

Table 6 Geometrical parameters and relative boundaries

The main functions of this Python script are: given a set of geometrical parameters, export the CAD models in *iges* format, generate a *txt* file containing the necessary inputs for CFD simulations in StarCCM+, which are the two profiles' chords and rotation angles, and generate a *csv* file containing the parameters of each individual generated. This last *csv* file is then imported into MATLAB and assembled with CL and CD data to generate the training dataset.

This process is illustrated graphically in Figure 33.



```
>_ FreeCAD -c GenerateModel.py
```

Figure 33 Geometry creation workflow

The Surrogate Modelling Toolbox (SMT) is a Python library designed to facilitate the construction, training and evaluation of surrogate models. It is particularly useful in engineering applications where high-fidelity simulations are computationally expensive, and an approximate model is required to efficiently explore the design space. In addition, it provides several sampling algorithms as well, such as Latin Hypercube Sampling or Full Factorial.

SMT is built on top of several Python libraries and leverages their functionalities to facilitate the user in training a surrogate model. The main libraries that SMT is built upon are:

- *PyDOE3*: sampling for design of experiments
- *SciPy*: numerical computing and optimisation
- *Scikit-learn*: machine learning models and evaluation
- *Matplotlib*: data visualization

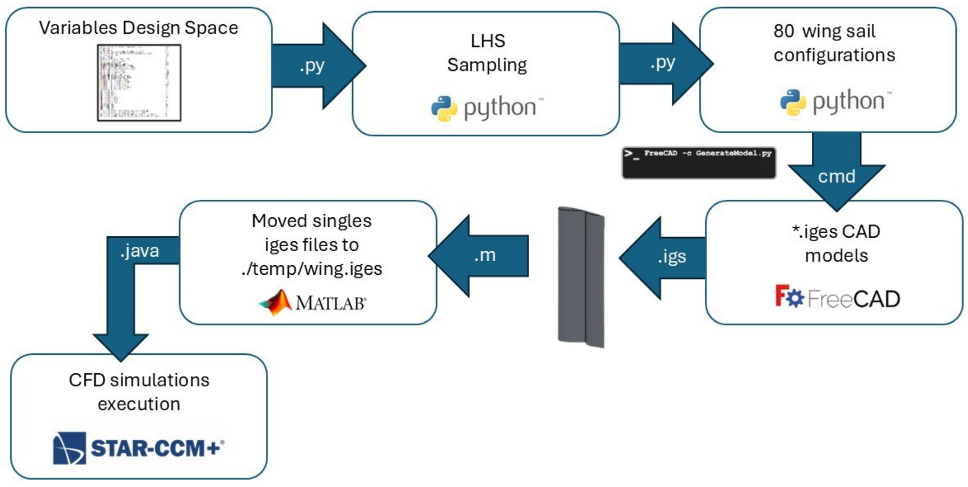
In the framework of this thesis just the sampling algorithm has been exploited, to generate a distribution of individuals with the Latin Hypercube algorithm.

FreeCAD ended up being the most suitable CAD software because of three features it presented:

- Parametric modeller, which means that designs are created using parameters, thus allowing easy editing and modification
- Open source, which means that it is free to use and requires no license to operate
- Scripting and automation capabilities, since it’s highly integrated and scriptable using Python (it is even possible to import FreeCAD libraries into external editors like Visual Studio Code).

Since the CAD models needed to be imported to be analyses with CFD, the *iges* format for the CAD models was adopted because of its seamless integration with Star-CCM+. IGES stands for Initial Graphics Exchange Specification, and it is a vendor-neutral file format, which means that it allows exchange of information among CAD systems and other software, being thus perfect for this application.

A MATLAB script was used to launch the simulations of each wing sail automatically.



```
>_ FreeCAD -c GenerateModel.py
```

Figure 34 Full geometry-simulation workflow

8. CFD simulations

In order to investigate the flow behaviour around the sail Star-CCM+ was adopted, a CFD software developed by Siemens widely spread in aerodynamics and hydrodynamics applications, used to numerically solve Navier-Stokes equations to predict fluid flow, heat transfer, turbulence and other physical phenomena.

Computational Fluid Dynamics (CFD) is a branch of fluid mechanics that uses numerical analysis and algorithms to solve and analyse problems involving fluid flows, under defined boundary conditions (26). It solves the Navier-Stokes equations with techniques like the *Finite Volume Method* (FVM), *Finite Element Method* (FEM) and *Finite Difference Method* (FDM). Star-CCM+ makes use of the FVM method.

Navier-stokes equations describe accurately fluid flow phenomena but are nearly impossible on present-day computers to solve, and a possible solution is to decompose them into what are known as "*Reynolds-Averaged Navier-Stokes Equations*" (*RANS* Equations). In RANS equations the fluid flow is described by separating instantaneous quantities into mean and fluctuating components, so time-dependent turbulent velocity fluctuations are separated from mean flow velocity. This transformation introduces additional unknowns called "*Reynolds stresses*", which depend on the velocity fluctuations, and in order to produce a solvable closed system, require a turbulence model (27). In addition, it is relatively simple to produce an unsteady variant of the RANS equation, known as "*Unsteady Reynolds Averaged Navier-Stokes*" (*URANS* Equations) for transient flows.

Turbulence models are mathematical formulations used to simulate the effects of turbulence without resolving every minor fluctuation, enabling engineers to predict the behaviour of turbulent flows in a computationally feasible way. As most of the aspects related to computational simulations, the choice of the turbulence model is a trade-off between accuracy and computational cost. They are differentiated by the number of equations they are solving, and the most common ones are:

- Zero-Equation Models (algebraic model)
- One-Equation Models
- Two-Equation Models
- Reynolds Stress Models (RSM)

A deeper analysis of each model is out of the scope of this thesis, so only the most adopted models in engineering applications will be examined. Models like the $k - \epsilon$ and $k - \omega$ models have become industry standards for most engineering problems, in which k represents the turbulent kinetic energy and the second transported variable varies between the two. The second variable can be thought of as the variable that determines the scale of the turbulence, while the first variable determines the energy in the turbulence (28).

The $k - \varepsilon$ turbulence model is a commonly employed two-equation closure that computes turbulent kinetic energy k and its dissipation rate ε through distinct transport equations, while the $k - \omega$ turbulence model is a widely used two-equation framework that simulates turbulent flows by solving distinct transport equations for turbulent kinetic energy k and specific dissipation rate ω, which dictates the turbulence length scale.

The Shear-Stress Transport (SST) $k - \omega$ turbulence model is a widely utilized two-equation eddy-viscosity turbulence closure that combines the strengths of the $k - \omega$ and $k - \varepsilon$ frameworks. It employs a $k - \omega$ formulation near solid boundaries, providing precise resolution of flow features down to the viscous sublayer without requiring additional damping functions - a characteristic that makes it robust as a low-Reynolds-number model. In free-stream regions, the model switches to a $k - \epsilon$ type formulation, mitigating its dependence on inlet turbulence specifications. This blended strategy enhances its performance in complex flow scenarios involving adverse pressure gradients and flow separation. However, it may exhibit a tendency to overestimate turbulence levels in zones of high normal strain, although to a smaller degree than conventional $k - \varepsilon$ models (29).

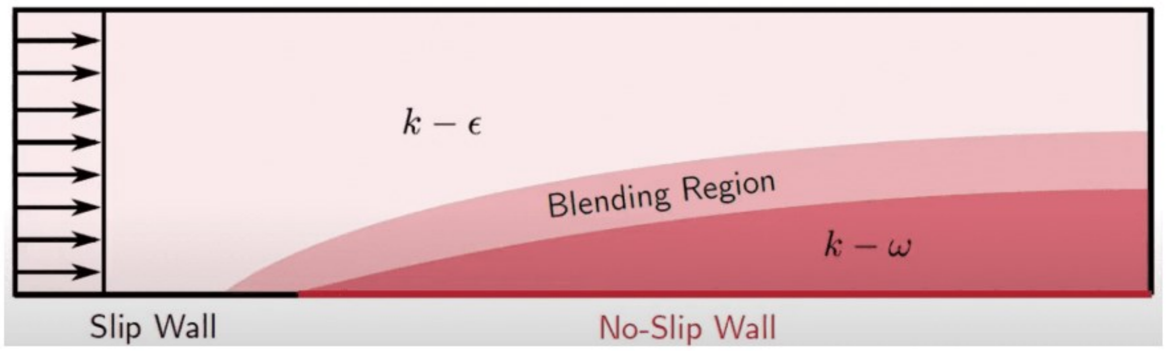


Figure 35 $k - \omega$ SST model switches between turbulence model using blending function *(30)*

In (31) an alternative approach to traditional CFD is proposed, involving 2D-RANS calculations around wing profiles in conjunction with a lifting line method to account for three-dimensional flow phenomena. Nonetheless, in this work 2D-RANS calculations have been adopted because were considered the best trade-off between computational cost and results accuracy, as mentioned in (31).

8.1 Computational Model

8.1.1 Automatization of the simulations

The workflow to set up the CFD simulations in this thesis is the following:

- CAD import as *iges file*
- Meshing
- Physics setup
- Solving
- Extraction of CL and CD values

The automatization of the process was possible thanks to Star-CCM+'s possibility to run Java macros, scripts used to automate tasks like setting up simulations, manipulating geometries, running solvers, and exporting results, allowing thus to create fully automated workflows.

Once the simulation has met the stopping criteria and stopped, the macro exports a *csv* file with the CL and CD values per every time step and saves the results into a dedicated folder, from which, will be uploaded in MATLAB to complete the surrogate model training dataset together with the geometrical parameters.

8.1.2 Computational domain

In this section the CFD model developed in Star-CCM+ is described. For the numerical analysis, we solve two-dimensional unsteady, incompressible RANS equations.

Being two dimensional simulations, the computational domain is treated like a 2D plane, with a length of 280 meters, and a width of 60 meters. The wing sail is positioned at 80 meters from the inlet section, equally spaced between top and bottom edges. All these measures were determined starting from the length of the total chord of the wing sail, which has an average value of 2 meters.

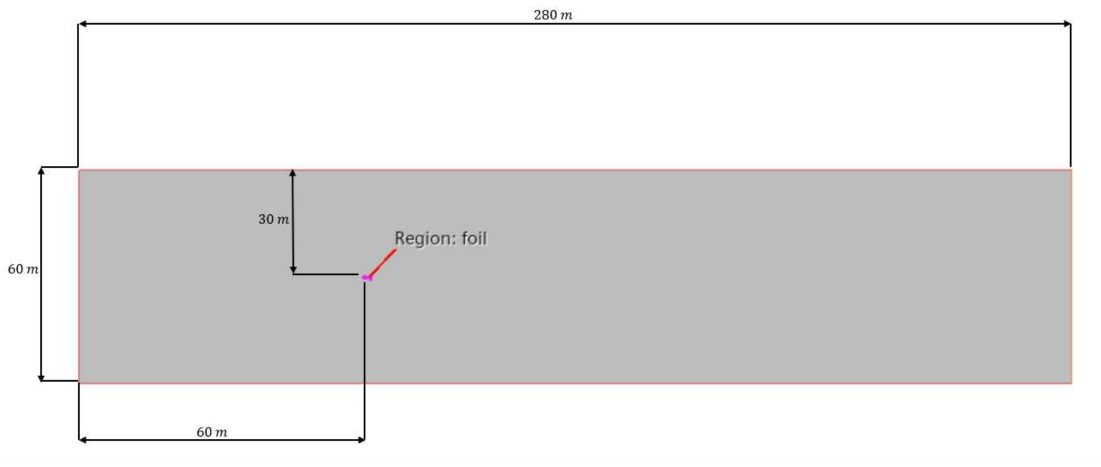


Figure 36 Computational domain

At the leading edge of the first wing, in the space between the trailing edge of the first wing and the leading edge, and at the trailing edge of the second wing a cylinder is positioned and used as local refinement of the mesh afterward. Since the angles of the two elements of the wing sail rotate, the position of the cylinders is parametrised as a function of the angles α and δ. The effect of the cylinders is visible in Figure 40.

The cylinder on the main wing's leading edge is positioned on the reference frame's origin.

$$Start\ coordinate = [0, 0, 0]$$

$$End\ coordinate = [0, 0, 1]$$

The position of the centre of the middle cylinder is expressed by the following relations:

$$Start\ coordinate = [C1, -C1 * \tan(-\alpha), 0]$$

$$End\ coordinate = [C1, -C1 * \tan(-\alpha), 1]$$

Meanwhile, the position of the trailing edge cylinder is expressed as:

$$Start\ coordinate = [C1 + C2 + 0.03, -(C1 + C2) * \tan(-\alpha) - C2 * \tan(-\delta), 0]$$

$$End\ coordinate = [C1 + C2 + 0.03, -(C1 + C2) * \tan(-\alpha) - C2 * \tan(-\delta), 1]$$

All three cylinders present a radius of 0.15 m.

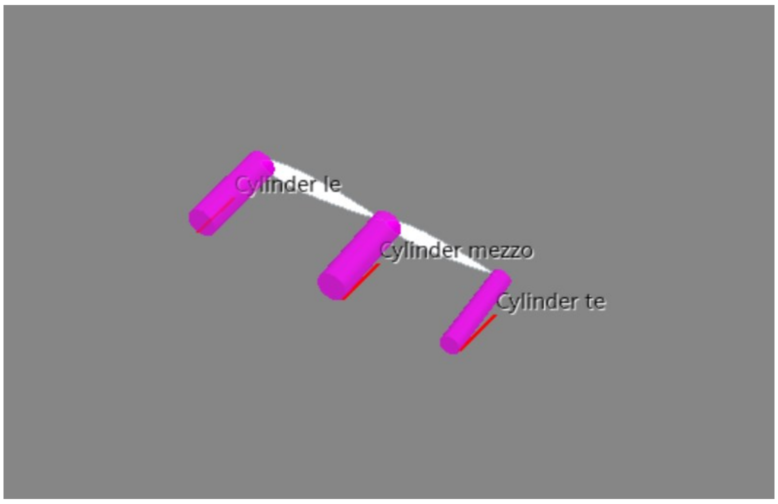


Figure 37 Local refinements between wing edges

A rectangular block was used to set a local mesh refinement above and below the wing sail to better capture the pressures fields above and under the wing. Additionally, an “Offset” part has been added to refine the mesh around the wings, reported in Figure 38.

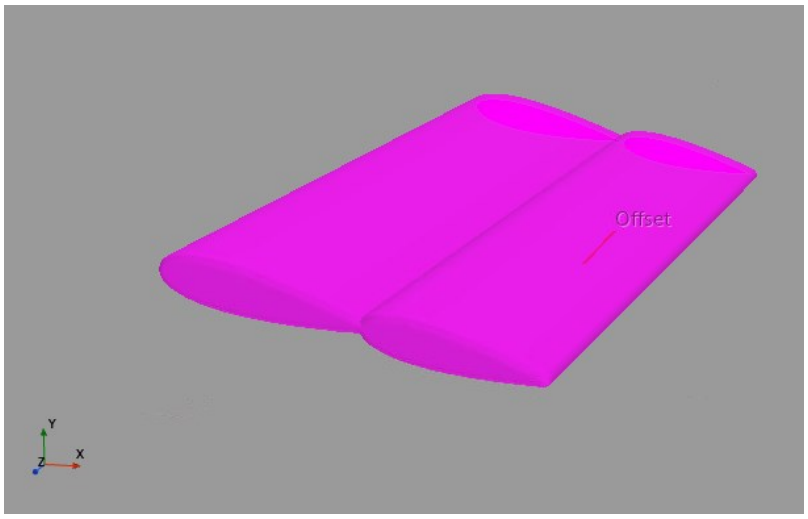


Figure 38 Offset refinement

8.1.3 Mesh

Figure 39 reports the final mesh. Since every tested geometry was different, the total number of cells varied slightly around 110.000 cells. A polyhedral mesh was chosen, and the different refinements were express as a function of the parameters reported in Table 7.

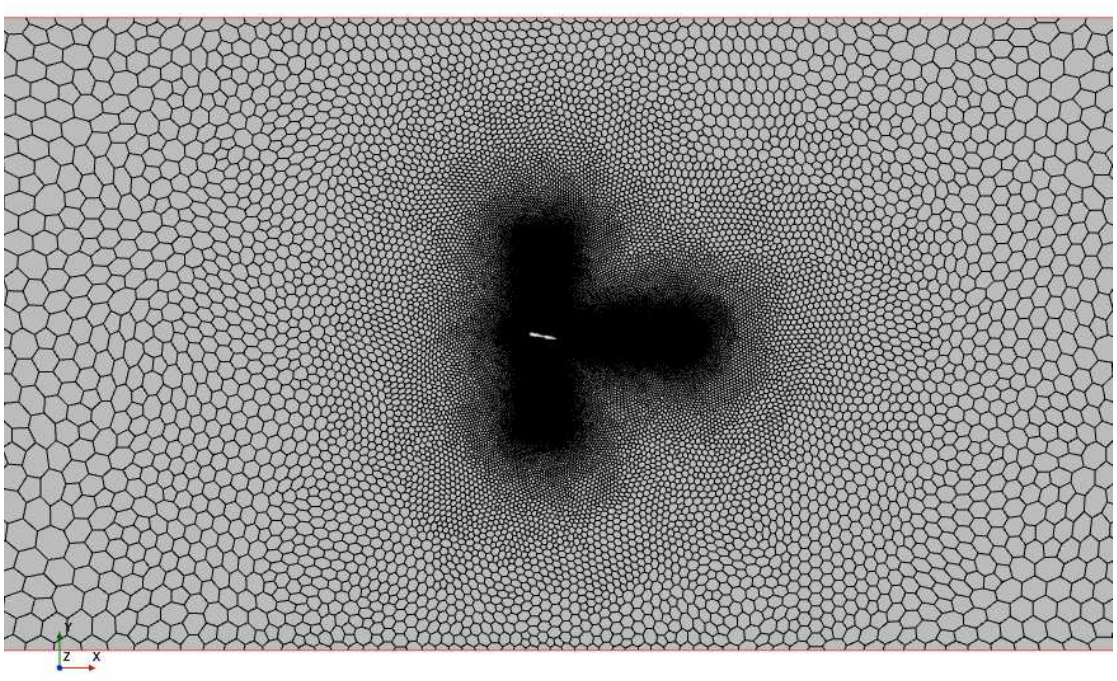


Figure 39 Domain mesh

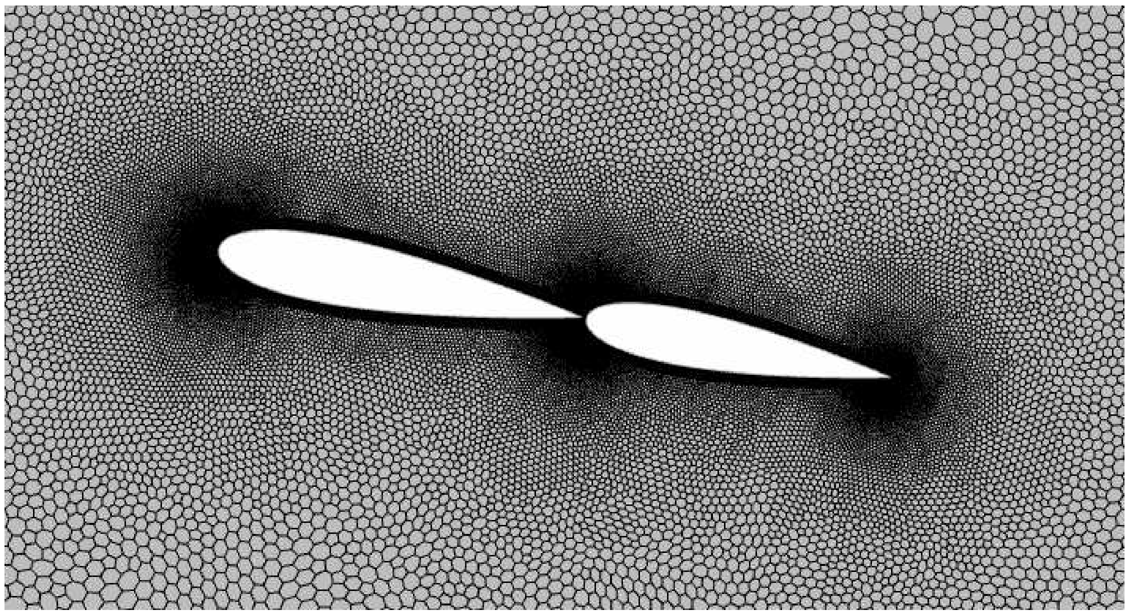


Figure 40 Local mesh refinements between wings

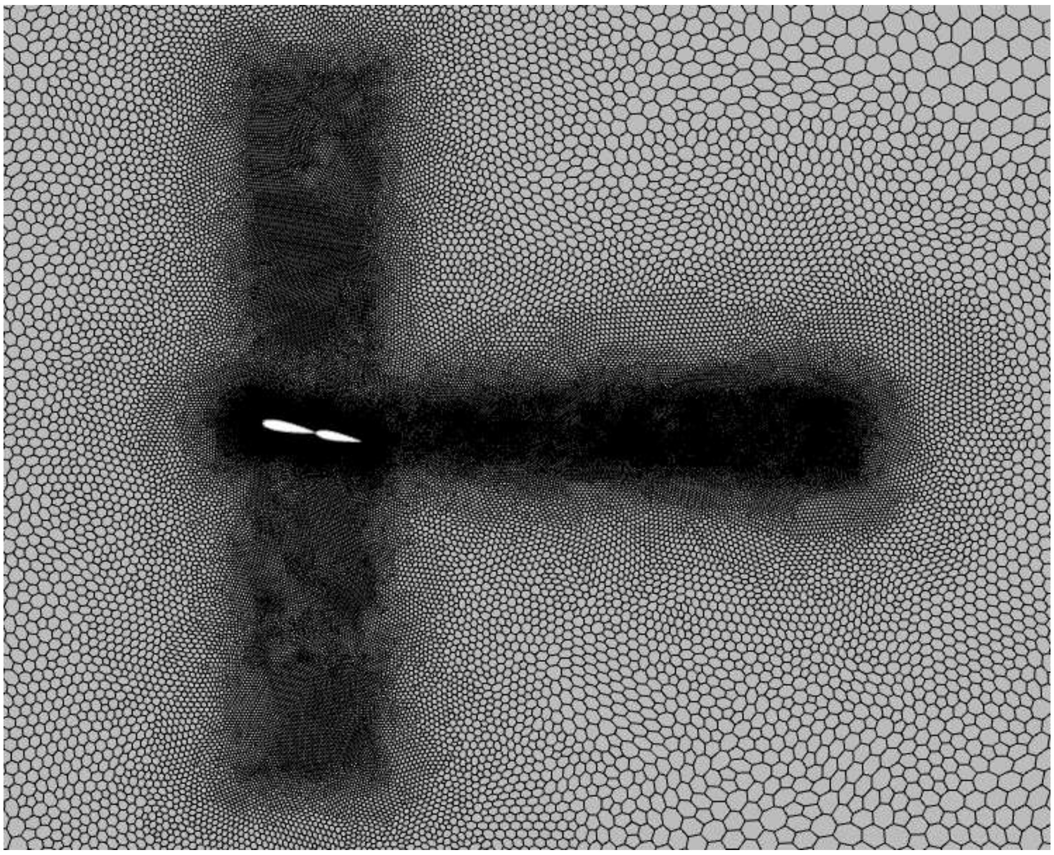


Figure 41 Wake and vertical box refinements

The parameters used to build the mesh are reported in Table 7, Table 8, Table 9, Table 10 and Table 11. $C1$, $C2$, α and β are indicated as "*Variable*" since those values are retrieved from the input file written by the Python script responsible for generating the CAD models. The macro reads the values from that file and updates the parameters in the "*Custom tree*" of the simulations before executing the mesh. This is necessary because the angles and chords change in every geometry and the positions of the cylinders needs to be updated to position them correctly.

$PARAMETER$	$VALUE/FORMULA$
$C1$	$Variable$
$C2$	$Variable$
α	$Variable$
δ	$Variable$

$External\ Reynolds\{Re\_from\_out\}$	$2 * 10^5$
$Density\ \{\rho\}$	$1.18415\ Kg/m^3$
$Dynamic\ viscosity\ \{\mu\}$	$1.85508 * 10^{-5}\ Pa$
$Velocity\ from\ Reynolds\ \{V_{from_{Re}}\}$	$\frac{Re * C1}{\rho * \mu}$
$Velocity\ \{V\}$	$[V_{from_{Re}}, 0, 0]$
$Base\ Size\ \{BS\}$	1.4
$Number\ of\ layers\ main\ wing\ \{NLayers\}$	35
$Number\ of\ layers\ flap\ wing\ \{NLayers\ Back\}$	32
$Stretch\ Factor\ \{SF\}$	1.13
$Y +$	0.5
$Near\ Wall\ Thickness\ \{NearWallT\}$	$\frac{Y + *\ \mu}{\rho * U^*}$
$Total\ Thickness\ main\ wing\ \{TotalT\}$	$NearWallT * \frac{1 - SF^{NLayers}}{1 - SF}$
$Total\ Thickness\ flap\ wing\ \{TotalT\ Back\}$	$NearWallT * \frac{1 - SF^{NLayers\ Back}}{1 - SF}$
$Reynolds\ on\ main\ wing$	$C1 * \lVert V \rVert * \frac{\rho}{\mu}$
$Reynolds\ on\ flap\ wing$	$C2 * \lVert V \rVert * \frac{\rho}{\mu}$
$Skin\ friction\ coefficient\ \{C_f\}$	$C_f = [2 * \log_{10}(Re) - 0.65]^{-2.3}$
$Wall\ Shear\ Stress\ \{\tau_w\}$	$\tau_w = \frac{1}{2}\rho U^2 C_f$
$Friction\ velocity\ \{U^*\}$	$\sqrt{\frac{\tau_w}{\rho}}$
$Model\ coefficient\ \{C_\mu\}$	0.09
$Turbulence\ Length\ scale\ \{l\}$	$0.4 * TotalT$
$Turbulent\ kinetic\ energy\ \{k\}$	$\frac{3}{2}{V_{from}}_{Re}^{2} Turb_{intensity}^2$

$Turbulence\ intensity\ \{Turb_{intensity}\}$	0.03
$Turbulence\ dissipation\ rate\ \{\epsilon\}$	$C_\mu * \frac{k^{\frac{3}{2}}}{l}$
$Real\ total\ thickness\ main\ wing$	$0.37 * \frac{C1}{Re^{0.2}}$
$Real\ total\ thickness\ flap\ wing$	$0.37 * \frac{C2}{Re^{0.2}}$

Table 7 Mesh parameters

$DEFAULT\ CONTROLS$	$VALUE$
$Mesher$	$Polygonal\ mesher$
$Prism\ Layer\ mesher$	$Enabled$
$Base\ Size$	$\{BS\}$
$Target\ Surface\ Size\ (Relative\ to\ base)$	200%
$Minimum\ Surface\ Size\ (Relative\ to\ base)$	0.05%
$Surface\ Growth\ rate$	1.05

Table 8 Mesh default controls

$PRISM\ LAYER\ CONTROLS$	$VALUE$
$Stretching\ Function$	$Geometric\ Progression$
$Distribution\ Mode$	$Stretch\ Factor$
$Number\ of\ Prism\ Layers$	$35\ \{Nlayers\}$
$Prism\ Layer\ Stretching$	$1.13\ \{SF\}$
$Prism\ Layer\ Total\ Thickness\ (Absolute\ value)$	$0.0405\ \{TotalT\}$

Table 9 Prism Layer controls

$CUSTOM\ CONTROLS$	$VALUE$
$Dominio\ (Relative\ to\ base)$	200%

$Cylinders' size\ (Absolute\ value)$	$\left\{\frac{C1}{250}\right\}$
$Offset\ size\ (Absolute\ value)$	$\left\{\frac{C1}{100}\right\}$
$Horizontal\ Wake\ (Absolute\ value)$	$\left\{\frac{C1}{20}\right\}$
$Vertical\ Box\ refinement\ (Absolute\ value)$	$\left\{\frac{C1}{20}\right\}$

Table 10 Custom controls mesh

$Surface\ contro$	$Target\ Surface\ Siz$	$Minimum\ Size$	$Isotropic\ size\ wake\ refinemen$
$Main\ wing$	$\left\{\frac{C1}{100}\right\}$	$0.05\%\ Relative\ to\ Bas$	$\left\{\frac{C1}{30}\right\}$
$Flap\ wing$	$\left\{\frac{C2}{100}\right\}$	$0.05\%\ Relative\ to\ Bas$	$\left\{\frac{C2}{30}\right\}$

Table 11 Surface controls

All the formulas in the tables are taken from (32), a comprehensive guide for correct modelling of the boundary layer. Firstly, the height of the first cell adjacent the surface of the body needs to be calculated, starting from the Reynolds number.

$$Re = \frac{\rho U L}{\mu}$$

$$C_f = [2 \log_{10} Re - 0.65]^{-2.3}$$

The skin friction coefficient is determined by an empirical correlation for fully developed turbulent flow over a flat plate (which is valid for turbulent flows with $Re < 10^9$).

Once the skin friction coefficient is determined, the wall shear stress τ_w is calculated as

$$\tau_w = \frac{1}{2} \rho U^2 C_f$$

Friction velocity is then computed from wall shear stress

$$u_\tau = \sqrt{\frac{\tau_w}{\rho}}$$

Having chosen a $y+= 0.5$, it is possible to calculate the height of the wall adjacent cell, also referred to as "*Near Wall Thickness*"

$$Near\ Wall\ Thickness = \frac{y + * \mu}{u_\tau \rho}$$

Having defined the total number of layers in the boundary layer and the growth ratio, also referred to as "stretch factor", it is possible to compute the total thickness of the boundary layer

$$Total\ Thickness = NearWallThickness \cdot \frac{1 - r^N}{1 - r}$$

When implementing inflation layers, it is preferable that the transition from the final inflation layer to the freestream mesh maintains a consistent cell volume. Abrupt variations in cell volume can induce sudden fluctuations in sub-grid viscosity and potentially lead to inaccuracies and instability.

To provide the necessary background information, an inflation layer is a region in a computational mesh where cells are refined near a surface – typically a wall – to capture more accurately the steep gradients in physical quantities, such as velocity and temperature, that occur in the boundary layer. These layers comprise multiple rows of cells, with a gradual increase in thickness as they extend away from the wall. This ensures that the simulation accurately resolves the near-wall flow dynamics without excessively increasing the overall mesh size. This approach is critical in Reynolds-Averaged Navier-Stokes (RANS) modelling, where the accurate resolution of the boundary layer is paramount for predicting aerodynamic or hydrodynamic behaviour.

8.1.4 Simulation set up

Firstly, the boundary conditions were assigned to the respective regions, as reported in Figure 422. Velocity inlet is assigned to the top, bottom and front edges, while pressure outlet is assigned to the back one. The wing sail is modelled as a no-slip wall condition.

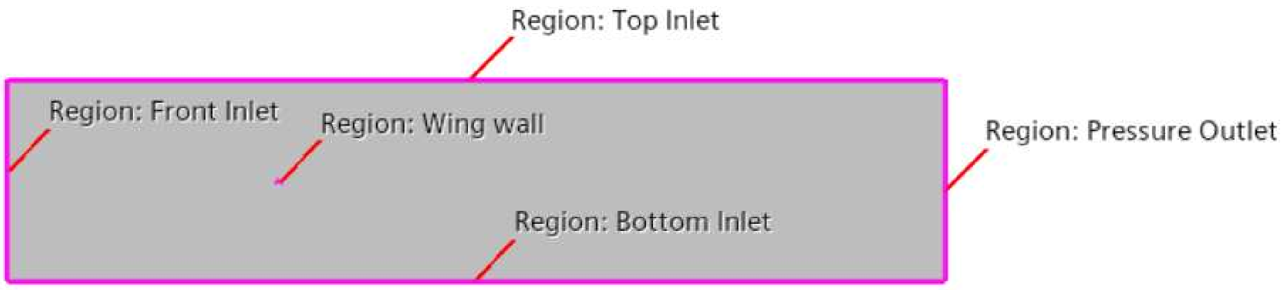


Figure 42 Boundary conditions

Following, the solver's settings are selected:

- Implicit unsteady
- Segregated flow
- $SST\ k-\omega$ turbulence model
- $\gamma - Re\theta$ transition model

A segregated solver has been chosen in order to reduce memory usage and reduce computational cost since it solves momentum, pressure, and turbulence equations in sequence, reusing smaller matrices, resulting lighter for moderate-sized clusters.

An unsteady solver is selected to solve for time-dependent phenomena that can occur at high angles of attack, like vortex shedding and stall, while an implicit scheme is chosen to have an unconditionally stable problem.

$$CFL = \frac{|u|\Delta t}{\Delta x} = 1$$

The time step has been set equal to $\Delta t = 0.01\ s$ and considering an average mesh element size of $\Delta x = 0.02$ m around the sail, the CFL is kept around 1. $|u|$ is calculated starting from a Reynold number of 200000, which equates to a velocity of 2.3 m/s.

The $SST\ k-\omega$ turbulence model is widely used in CFD due to its effectiveness in capturing both near-wall and free-flow turbulence (33). However, it assumes that the flow is fully turbulent, thereby ignoring the critical laminar-to-turbulent transition essential for accurate simulations in aerodynamics, turbomachinery and marine applications. To overcome this limitation, the $\gamma - Re\theta$ transition model is incorporated to explicitly predict the onset and evolution of turbulence. Unlike conventional models that assume fully turbulent flow, the $\gamma -$

$Re\theta$ model considers laminar, transition and turbulent regions, allowing for a more accurate characterisation of the boundary layer. In practice, the inclusion of a transition model is critical - especially when analysing airfoil performance - as relying solely on the $SST\ k - \omega$ model can lead to overestimation of turbulence, resulting in inaccurate drag predictions and suboptimal boundary layer control.

The coefficients of the turbulence model are computed starting from the turbulent kinetic energy k, from the turbulence intensity I and the inlet velocity U (32):

$$k = \frac{3}{2} U^2 I^2$$

The specific dissipation rate ϵ is calculated next, and from that, the specific dissipation rate ω:

$$\epsilon = \frac{C_\mu k^{\frac{3}{2}}}{l}$$

$$\omega = \frac{\epsilon}{C_\mu k}$$

8.1.5 Stopping criteria

Physical time has been chosen as the stopping criteria of the simulation. A trade-off between accuracy and computational cost was necessary in order to allow the physical phenomena to develop fully but avoid unnecessary iterations. Thus, to determine the optimal physical time, a convergence study was carried out on the aerodynamic coefficients' values, reported in Table 132.

$PHYSICAL\ TIME$	$EXECUTION\ TIME$	CL	CD
$10s$	$8\ min$	2.1171	0.1026
$20\ s$	$14\ min$	2.1263	0.0864
$30\ s$	$22\ min$	2.1087	0.0846
$45\ s$	$38\ min$	2.0944	0.0852
$60\ s$	$41\ min$	2.0955	0.0855

Table 12 Simulation time and results for different physical times

This convergence study is performed by calculating the percentage error, using the result from the longest computational time as reference. Results are reported in Table 13.

$PHYSICAL\ TIME$	$ERROR\ ON\ CL$	$ERROR\ ON\ CD$
$10\ s$	-1.03%	-20%
$20\ s$	-1.47%	1.05%
$30\ s$	-0.63%	1.05%
$45\ s$	0.05%	0.35%
$60\ s$	0%	0%

Table 13 Errors for different physical times

Considering the computational time necessary to carry out each simulation, a physical time of 28 seconds has been chosen as most appropriate compromise. Given that the initial pool individuals to be tested was of 80 designs, the total simulation time amounted to roughly 28 hours, without considering the time consumed by the adaptive sampling algorithm. The simulations were run on a desktop computer mounting two Intel Xeon Silver 4114 CPUs and 128GB of RAM and required on average 20 minutes each.

The variation of values per iteration was used as the convergence metric. To make this criterion more conservative, only the second half of the simulation was considered, specifically, the deviation during the last 14 seconds of physical time. First, the mean values of CL and CD were calculated. Then, the maximum recorded values were identified. If the difference between the mean and the maximum was less than 0.1, the case was considered converged; otherwise, it was discarded.

$$CL_{mean} = \frac{1}{n} \sum_{i=14s}^{m=28s} CL_i$$

$$err_{CL} = \max(CL) - CL_{mean} < 0.1$$

In order to mitigate influence of possible small oscillations of the coefficients, the mean value of CL was added to the dataset, not the value of the last iteration. The same concept and procedure were applied to the drag coefficient.

9. Neural Network training

Since the objective was to reduce the computational burden of the CFD simulations during the optimisation process, the adoption of a surrogate model was imperative. As reported in Design of Experiments (DOE), a Latin Hypercube Sampling algorithm was adopted, and following the general rule of thumb of the 10 individuals per design variable, 80 initial designs were generated. Table 14 reports a few examples of individuals extracted from the DOE.

$C1$	$C2$	$\%C1$	G	$PROFILE\ 1$	$PROFILE\ 2$	$ALPHA$	$DELTA$
1028.8	1147.5	0.92	27.25	1	2	−7.78	−0.84
1351.3	917.5	0.91	35.75	3	1	−8.91	−11.72
1253.8	877.5	0.71	38.25	1	1	−9.47	−14.34
1441.3	1007.5	0.67	49.75	2	2	−5.72	−7.78

Table 14 Some examples of LHS individuals

Since the Design Space is eight dimensional it would it be impossible to represent the distribution of samples generated, so Figure 433, Figure 44, Figure 45 report just 2D projections of such space.

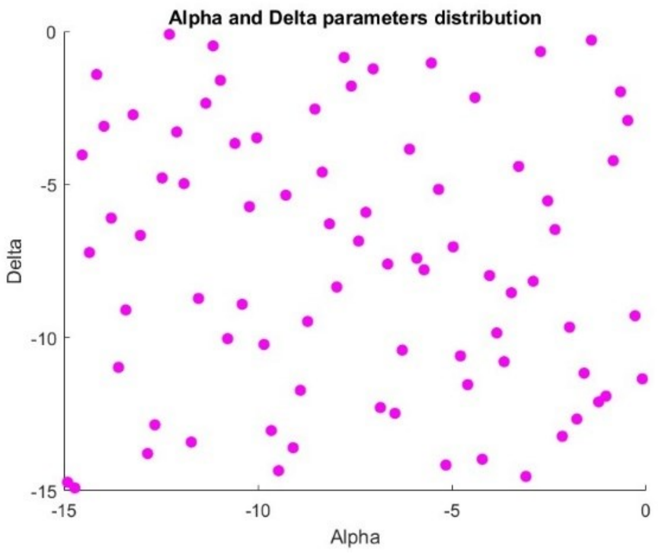


Figure 43 Samples distribution α-δ parameters

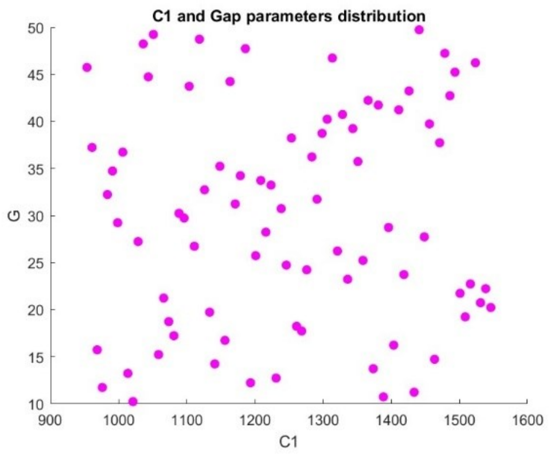


Figure 44 Samples distribution C1-Gap parameters

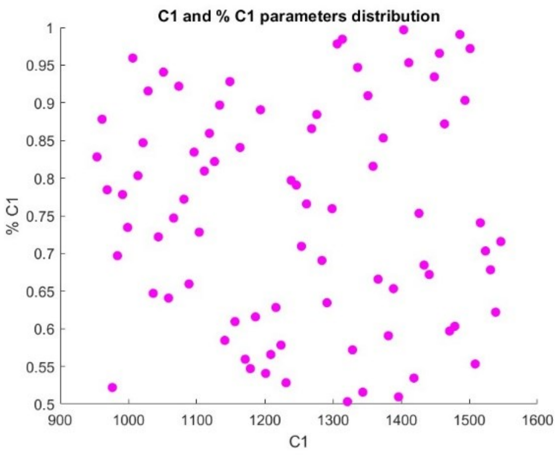


Figure 45 Samples distribution C1-%C1

Given the limited computational power available, in order to reduce to the bare minimum the number of simulations necessary, an Adaptive Sampling technique was implemented in MATLAB. Since the number of individuals generated by LHS could have been insufficient, the idea behind this algorithm was to add individuals to the initial DOE until the *Maximum Absolute Error* (MAE) reported by the model was lower than 0.1 for both CL and CD predictions. Other indicators adopted to evaluate the accuracy of the models where the *Coefficient of Determination* R^2 and *Root Mean Squared Error* $RMSE$. R^2 and $RMSE$ were not used as criterions for convergence but still observed during the iterations and taken into consideration when evaluating the correctness of the optimal solution.

9.1 Error metrics

In the surrogate modelling context, *Absolute Error (AE)* is defined as the difference between the predicted value and the true value of a measurement (candidate). The Maximum Absolute Error is just the highest *AE* reported while evaluating the test dataset with the model.

$$AE = |y_{pred} - y_{true}|$$

$$MAE = \max(AE) = \max(|y_{pred} - y_{true}|)$$

The *Coefficient of Determination*, most commonly referred to as R^2, expresses a measure of how well the model's predictions match the actual data. It ranges from 0 to 1, where 1 means perfect matching and 0 means the model is not capable of catching the variability of the data. Its expression is the following (34):

$$R^2 = 1 - \frac{\sum(y_{true} - y_{pred})^2}{\sum(y_{true} - \bar{y})^2}$$

where $\bar{y}$ is the *mean value* of actual measurements.

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$$

The *Root Mean Square Error (RMSE)* is a measure of the average error in predictions. RMSE help assess the error in a regression or statistical model, with a value of 0 indicating a perfect match between predicted and actual values. Lower RMSE values suggest more accurate predictions and a better model fit, while higher RMSE values indicate greater errors and less reliable forecasts (35).

9.2 Adaptive Sampling

The Adaptive Sampling routine was dealt with a MATLAB code, in which the initial dataset was built, coupling design variables and CL and CD values extracted from CFD simulations, creating a 80×10 table. This dataset was used to build the *training dataset* with 80% of the samples and the remaining 20% formed the *testing dataset*.

For this particular application, a *Regressional Neural Network* was chosen as most suitable, due to its improved performance in dealing with complex non-linear problems. The functioning of this type of model is equal to that explained in Artificial Neural Network, and *Regressional* refers to the fact that the neural network is being used to predict continuous numerical values rather than discrete categories.

A first surrogate model was trained on this data and showed poor performance, thus the necessity of adding extra points to the dataset. The errors obtained after the initial training are reported below in Table 15.

$Coefficient$	$Error$
C_L	0.2219
C_D	0.0115

Table 15 Errors after initial training

As infilling criterion, the Maximum Absolute Error has been chosen and 0.1 has been set as threshold for the stopping of the sampling.

At every iteration of the algorithm, three new candidates were generated via LHS, with the aim of maintaining uniformity of points distribution. Each one of these candidates was tested on CFD and if its convergence criterion (on CL and CD oscillations) was met, it was added to the dataset, otherwise discarded.

Every iteration of the algorithm lead to an increase of the dataset size and an optimisation of the surrogate model parameters was carried out. As before, individuals were split into *training dataset* and *testing dataset* and provided as input to an optimisation function. The purpose of this function was to find the best combination of parameters affecting the predictive performance of the surrogate model.

The optimisation function made use of a Bayesian optimisation algorithm, since in literature it is considered one the most suitable for neural network hyperparameter tuning due to its efficiency in handling "expensive-to-evaluate" functions.

9.3 Bayesian optimisation

Bayesian optimisation is a commonly used strategy for fine-tuning neural network hyperparameters, which have a fundamental impact on model's performance and are unrelated to the dataset. A hyperparameter is a parameter that is used to control the learning process of the model, and an optimisation process searches for the best combination of such parameters with the objective of minimising a predefined loss function.

This optimisation algorithm is significantly more efficient compared to random search or evolutionary algorithms, because instead of testing random values, it builds a probabilistic model of the objective function and uses it to select the most promising hyperparameter settings to evaluate next (36).

A probabilistic model is a mathematical framework that uses probability theory to represent and analyse uncertain events or phenomena, by defying a set of possible outcomes and assigning probabilities to each one of these, allowing for predictions and inferences even when there is randomness or uncertainty involved (37).

Bayesian optimization follows a structured process involving the following key steps:

- Surrogate model: A probabilistic approximation of the objective function—in this case, the Mean Absolute Error (MAE) of the predictions. This model is typically built using Gaussian Processes and serves to estimate the performance of different hyperparameter configurations without evaluating them all directly.
- Acquisition function: A strategy used to decide which set of hyperparameters to evaluate next. It balances exploration (trying new areas) and exploitation (focusing on promising regions) based on the surrogate model's predictions.
- Evaluation and update: The selected hyperparameters are tested on the actual objective function, here represented by the MAE of CL predictions on validation data. Based on these results, the surrogate model is updated to reflect the new information.
- Iteration: These steps are repeated iteratively until a predefined stopping criterion is met (e.g., a maximum number of iterations or convergence of the objective function).

Instead of attempting to optimize the true objective function, $f(x)$, directly, which may prove prohibitively expensive to evaluate, a surrogate model is constructed to approximate that function, employing a set of sampled data points. The purpose of this model is twofold: firstly, to provide a prediction (mean) of the function value at any given point, and secondly, to provide an uncertainty (variance) estimate. The acquisition function is defined on this surrogate model and is used to decide which point in the input space to evaluate next. It quantifies the utility of sampling each point by balancing exploration (searching where the uncertainty is high) and exploitation (searching where the predicted value is promising). The subsequent selection and evaluation of the next sample point is undertaken using the true objective function, resulting in the updating of the surrogate model with the new data. This iterative process continues until a predetermined termination criterion is fulfilled.

Two commonly adopted acquisition functions are the following:

- *Expected improvement* (EI), which chooses the combination with the highest expected improvement
  The Expected Improvement at a point (x) is given by
  $$EI(x) = \mathrm{E}\left[\max\left(0, f(x) - f(x^{+})\right)\right]$$
  where $f(x^{+})$ is the current best observed value.

- *Upper Confidence Bound* (UCB), which chooses points that have a good balance of high guess and high uncertainty, effectively balancing exploration and exploitation
  The Upper Confidence Bound at a point (x) is given by
  $$UCB(x) = \mu(x) + k\sigma(x)$$
  where $\mu(x)$ and $\sigma(x)$ are the mean and standard deviation predicted by the Gaussian Process, and k is a parameter balancing exploration and exploitation (38).

After selecting a new point x_{new} by optimising the acquisition function, the true objective function is evaluated at this point. Then the Gaussian Process model is updated with this new data, enhancing its accuracy in approximating the objective function. In this work, the algorithm was run with default settings, thus using the Expected Improvement function.

9.4 Results

The hyperparameters that led to the best accuracy of the model are reported in Table 16.

$Number\ of\ hidden\ layers$	2
$Neurons\ per\ layer$	5
$Lambda$	0.001
$Activation\ function$	$tanh$

Table 16 Optimal model's hyperparameters

Table 17 reports the accuracy metrics per every iteration of the sampling algorithm. Convergence was met after 8 iterations, with a total of 101 individuals in the dataset. The same procedure was followed for the CD model as well and hereafter the metrics per iteration are reported in Table 18.

$ITERATION$	AE	R^2	$RSME$
1	0.1336	0.9785	0.0628
2	0.1832	0.9687	0.0707
3	0.1478	0.9742	0.0626
4	0.1079	0.9759	0.0589
5	0.2934	0.9394	0.1045
6	0.1045	0.9856	0.0520
7	0.1165	0.9831	0.0518
8	0.0954	0.9820	0.0491

Table 17 CL error metrics per iteration

$ITERATION$	AE	R^2	$RSME$
1	0.0153	0.8423	0.0053
2	0.0081	0.9227	0.0035
3	0.0078	0.8511	0.0041
4	0.0108	0.7912	0.0045

5	0.0106	0.8823	0.0044
6	0.0083	0.8375	0.0042
7	0.0079	0.8991	0.0032
8	0.0032	0.9771	0.0015

Table 18 CD error metrics per iteration

The training of the surrogate model ended when the desired accuracy was met, and as it can be seen from Table 17 and Table 18, the maximum error on the CL was 0.0954, while the maximum error on the CD was 0.0032. The maximum error on the validation data was chosen as convergence criterion because it was more conservative than considering the mean absolute error, which would've in fact, been even lower, for both CL and CD models.

Nonetheless, the mean absolute error was used as objective function for neural network hyperparameters optimisation, which was carried out with the $k-fold$ cross validation.

9.5 $\boldsymbol{k-fold}$ cross validation

k-fold cross validation was carried out during the optimisation of the surrogate model hyperparameters, because it represents a more robust and reliable estimate of model performance compared to single validation set testing, which might be biased or unrepresentative of the whole dataset.

For this application, $k=5$ has been chosen and mean absolute error as evaluation metric, and consequently cost function of the optimisation, because it quantifies the average magnitude of the errors in predictions, independent of their direction. The interpretation of MAE is straightforward since it is expressed in the same units as the target variable. Evaluating MAE over k-fold cross-validation provides a consistent measure of the model's prediction accuracy across different data splits, thus helping to identify a hyperparameter configuration that minimises error reliably.

10.VPP

In order to test the suitability of each sail geometry candidate, the behaviour of the boat with such sail needs to be evaluated, so a VPP is required. VPP stands for “Velocity Prediction Program”, a widely used tool in yacht design for speed assessment, which depends on the boat characteristics and sail performance, in a way that requires several aspects of physics involved to be opportunely modelled (22).

10.1 General background knowledge

VPPs are procedures that evaluate the global equilibrium of the system, balancing aerodynamic and hydrodynamic forces and thus calculating actual sailing performance data for any sailboat design generated during the exploration of the design space (22).

In the field of sailing, various methodologies for VPPs have been proposed, with each method tailored to the specific type of boat. Nonetheless, the fundamental principle of a VPP is the adoption of an iterative procedure at each wind speed and angle to find an “equilibrium sailing condition”, defined by a unique combination of parameters that satisfy equilibrium equations (39).

$$\sum F = \sum M = 0$$

In this work, a six degrees of freedom VPP is integrated in the optimisation environment, which permits to find the sail shape and trim that maximises the boat’s VMG. VMG stands for “Velocity Made Good” and corresponds to the effective speed at which a boat is progressing in a desired direction. The component of the boat's speed in the direction of the target is calculated by multiplying the boat's speed by the cosine of the angle between its heading and the direction of the wind. This measure is of crucial importance in the field of sailing, particularly in the context of racing, whether upwind or downwind.

$$VMG = V_{boat} \cdot \cos(TWA)$$

where V_{boat} is the velocity of the boat with the selected sail geometry and appendages.

A similarly important concept is the VMC, “Velocity Made on Course”, which is the component of a boat’s velocity in the direction of a specified waypoint or destination. It represents the effective speed at which the boat is approaching the target, regardless of wind direction.

$$VMC = V_{boat} \cdot \cos(\theta)$$

where θ is the angle between the boat’s heading and the direct course to the waypoint. In contrast to VMG (Velocity Made Good), which measures speed relative to the wind direction, VMC focuses on progress towards a specific location, thus making it an important metric for route optimisation in offshore and long-distance racing.

The motions of a marine craft exposed to wind, waves and ocean currents takes place in 6 DOF, thus the need to use a complex six-dofs model.

10.2 Glaros’ VPP

A VPP states the boat’s optimal speed through the water as a function of the sailing conditions at best possible trimming at each point of sailing, so the resultant of all forces and moments must equal to zero.

Nonetheless, not every equilibrium condition is acceptable, in fact, if for instance the computed equilibrium condition resulted with a pitch angle of 40° it wouldn’t be a physical sailing condition. For this reason, constraints on the optimisation parameters need to be set.

The accuracy of a VPP is strictly linked to the accuracy with which the individual components of the sailboat are modelled. Since Glaros is a foiling boat, no hull resistance modelling was required since it was assumed to be capable of foiling in the wind conditions analysed. It is evident that J and T foils necessitate distinct modelling strategies, a consequence of their inherent behavioural properties.

In order to model adequately the behaviour of Glaros, a six-dofs static model was implemented in MATLAB. The roll degree of freedom was deliberately neglected based on the observations of the boat made during tests on the field, that showed consistently that the best condition to facilitate take-off and keep a steady flight condition was keeping the boat flat, which would have translated in a very limiting constraint on the roll angle. For this reason, it has been considered acceptable setting that angle to zero but still retaining the degree of freedom in the problem formulation.

10.2.1 Environmental conditions

The sail optimisation was carried out on the most critical section of a regatta course, the upwind leg. Regatta courses can have different shapes depending on the competition but in the two most common scenarios, Olympic ring course and upwind/downwind course, the first two buoys lie directly upwind, perpendicular to the start line, as shown in Figure 46.

(40) offers an interesting example of optimisation for this type of regatta course. Additionally, this assumption led to a simplification of the problem, since the heading of the boat was tightly linked to the TWA and the leeway angle.

A trade-off for wind speed was required, since it had to be high enough to guarantee full foiling conditions for Glaros but not too extreme, so a $TWS = 6.3\ m/s$ was considered.

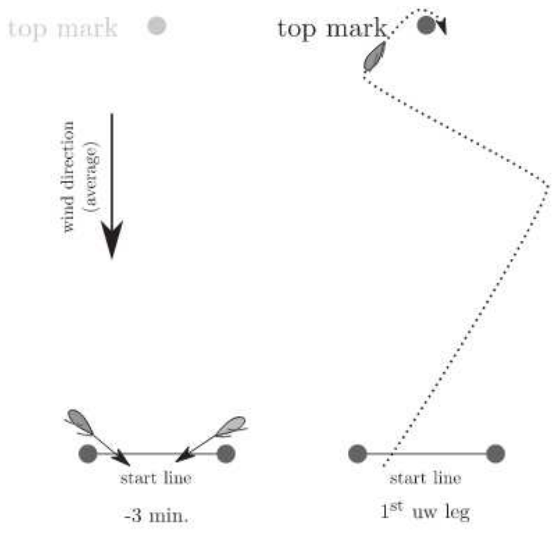


Figure 46 Upwind/downwind regatta field example

10.2.2 Force decomposition

Given that no hull modelling is required, all the other forces are aerodynamic forces and can be modelled as such, with the relation

$$F = \frac{1}{2}\rho S V^2 CF$$

where CF is a six-element vector, one row per each degree of freedom.

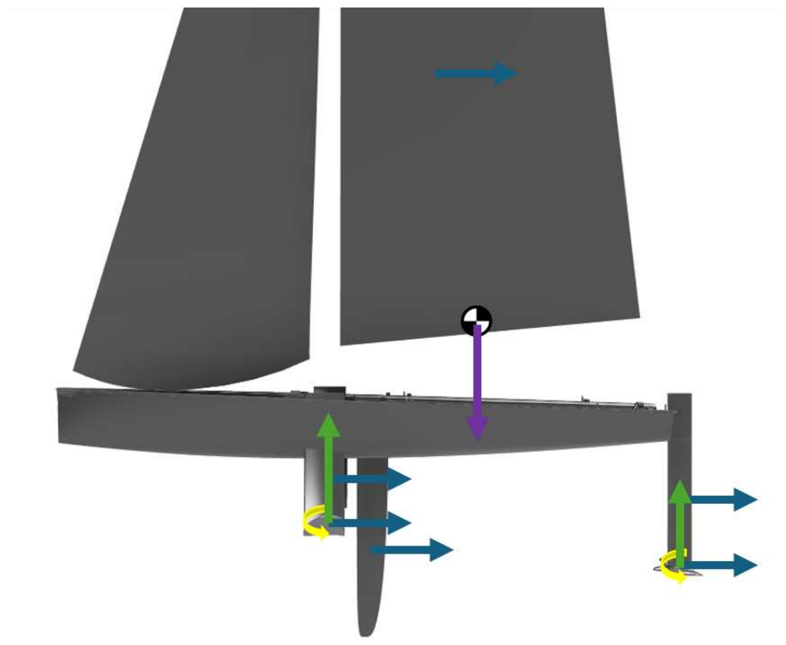


Figure 47 Side view forces acting on Glaros

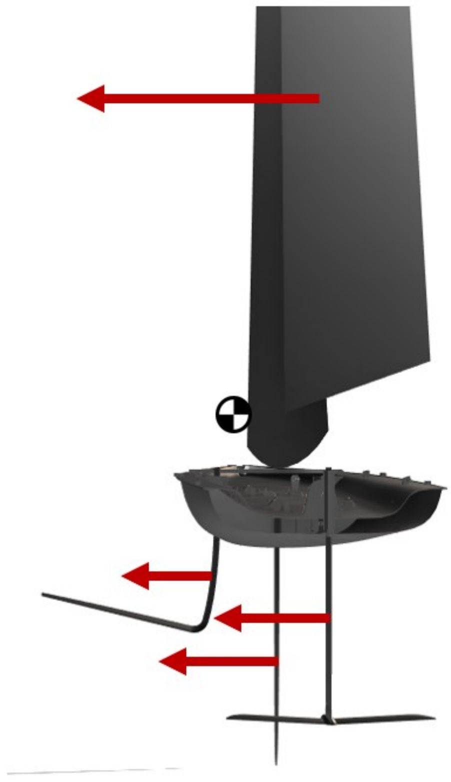


Figure 48 Back view forces acting on Glaros

Figure 47 and Figure 48 display all the forces acting on the system. For clarity reasons, only the left J foil is presented in the pictures, and the relative forces are sketched. Naturally, the same forces are present on the right foil as well. Forces generated by the daggerboard and by the rudder are presented too.

The weight of the sailors was not considered as an external force but as part of the system's mass, and by varying the position of the sailor, the position of the global CoG was recomputed as well.

$$tot.CG = boat.CG * boat.mass + sailor.CG * sailor.m + sail.CG * sail.m$$

tot, $boat$, $sailor$ and $sail$ are structures created in the MATLAB workspace to store in an orderly manner parameters and input data relative to that specific part of the boat. In addition to these, each foil has its own structure, namely, JDX, JSX, $dagg$ and rud, respectively for right J foil, left J foil, daggerboard and rudder.

Appendages were considered in the boat CoG term since their position was fixed, while the sail was considered separately since its height was a function of the design parameters.

Appendages' forces were expressed initially in their local $FLOW$ reference frame defined by three axes and two angles, α and β, and then roto-translated into body axes together with all other forces. α indicates the incidence angle on the *xz* plane, while β is the "side" incidence angle for vertical appendages, in plane *xy*.

The total CoG represented the origin of the body axes reference frame and reference point for moment calculations, as displayed in Figure 49.

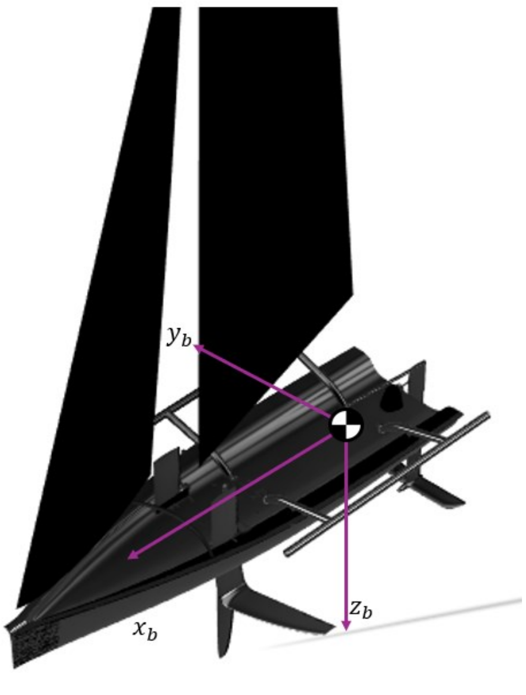


Figure 49 Body axes reference frame

10.2.3 Aerodynamic forces

The sail force is assumed being applied in the centre of gravity of the sail due to its regular shape. Figure 48 and Figure 48 report the original sail of Glaros for sake of simplicity but the same concept is valid for wing sails. The expression of the wing sail centre of gravity with respect to the mast position is the following:

$$sail.CG = \left[C1, 0, \frac{sail.height}{2}\right]$$

The aerodynamic centre was supposed to be coincident with the sail's CoG given the rectangular shape of the sails.

For sake of simplicity, sail forces were directly expressed in body axes, by inserting the aerodynamic coefficients coming from the surrogate model in the following equations:

$$F_x = CL \cdot \sin(AWA) - CD \cdot \cos(AWA)$$

$$F_y = CL \cdot \cos(AWA) + CD \cdot \sin(AWA)$$

10.2.4 Hydrodynamic forces

Hydrodynamic forces are modelled identically to aerodynamic forces with the only difference being the density of water entering the formula instead of air density.

The J foils generate a lift force, a drag force and a moment around the y-axis due to the focal moment of the airfoil section. Their vertical sections also generate a lateral (side) force along the y-axis and additional drag. The rudder similarly is generating lift, drag and a moment about y-axis. The daggerboard is generating just drag and a side force, the horizontal- foil equivalent of lift.

The hydrodynamic coefficients' polar for the J foils have been computed with CFD simulations, using a template developed by Polito Sailing Team's fluid dynamics department. The polar plot is displayed below in Figure 50.

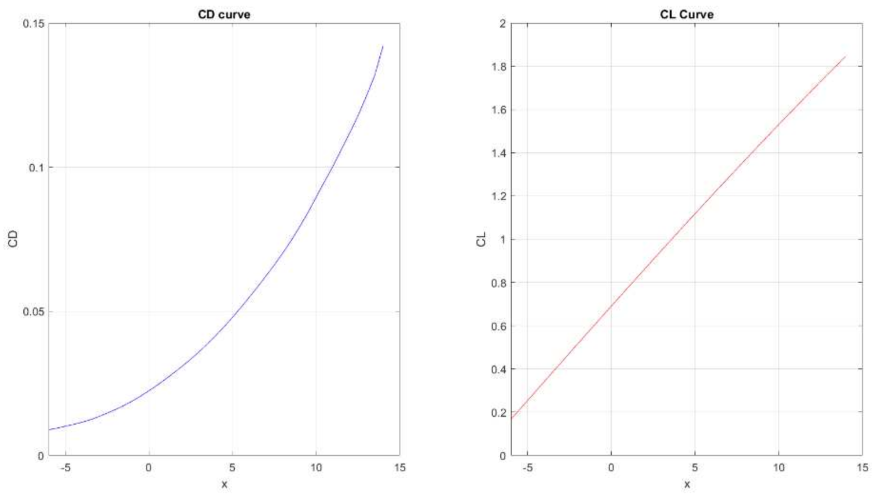


Figure 50 J foil polar

On the other hand, given the simplified shape of the rudder and daggerboard, simulations exploiting the Vortex Lattice Method were considered accurate enough for this purpose and significantly less computationally onerous. Polar graphs are reported in Figure 51 and Figure 52.

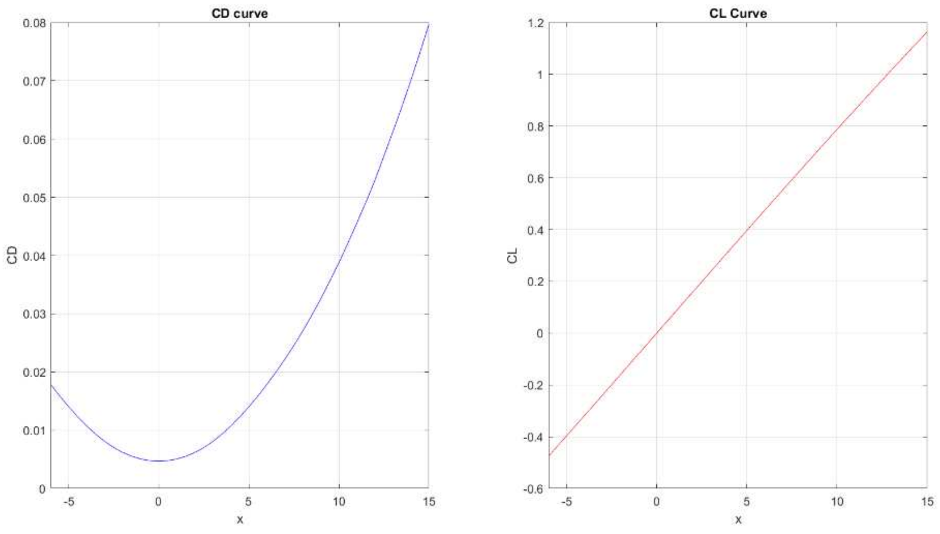


Figure 51 Daggerboard polars

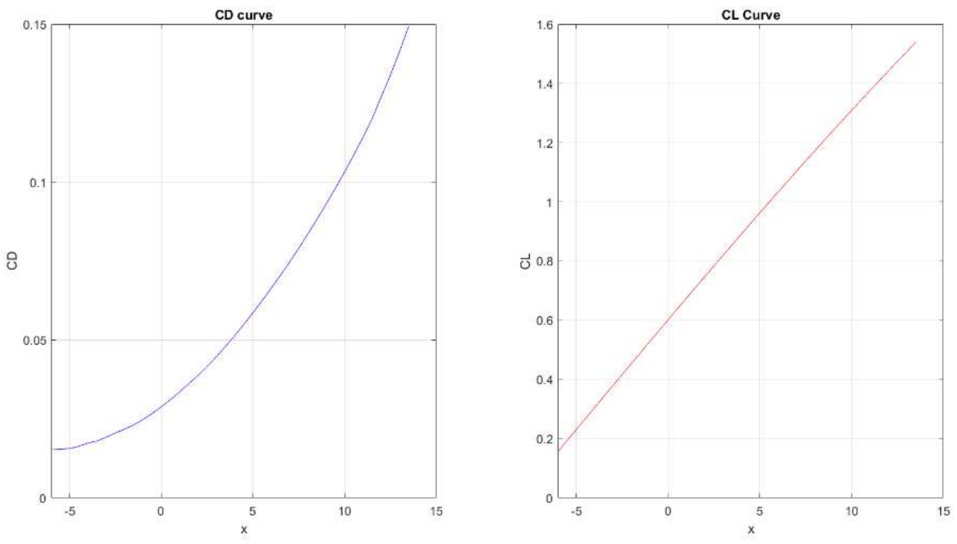


Figure 52 Rudder foil polar

10.2.5 Implementation

A MATLAB routine was developed to determine the boat's equilibrium state given the environmental conditions and the sail geometry as inputs. The optimization algorithm chosen to calculate the equilibrium points was *fmincon*, a gradient-based method widely adopted for this purpose. It has the advantage of being significantly quicker than other global optimisation algorithms while still maintaining its capability to find the global equilibrium of the problem, especially when combined with MATLAB's MultiStart algorithm to explore multiple local minima.

The decision variables manipulated by fmincon are reported below and addressed to as the vector xu (a blend of x state vector and u control vector), representing the boat-trim condition. Their corresponding boundaries are listed in Table 19.

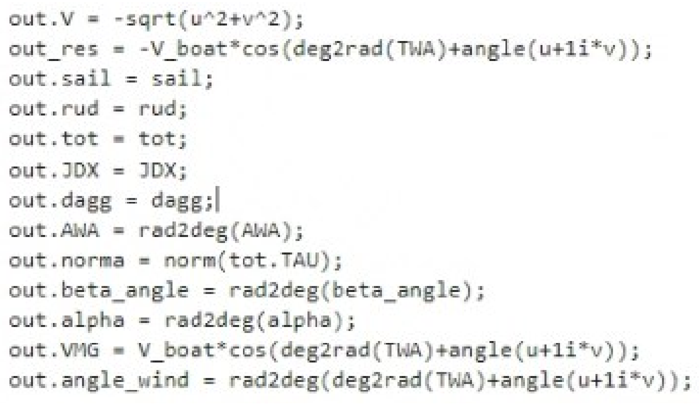
$$xu = [z\ u\ v\ \theta\ \psi\ \alpha 1\ \alpha 2\ x_s\ y_s\ rud.delta]$$

$Parameter$	$Lower\ Boundary$	$Upper\ Boundary$	$Unit\ of\ measure$
Z	$-1.1 + tot.cg(3)$	$-0.3 + tot.cg(3)$	m
U	1	15	m/s
V	0	2	m/s

$Theta$	-5	-5	°
Psi	30	140	°
$Alpha1$	-15	-2	°
$Alpha2$	-15	-2	°
X_s	-4.5	-1.5	m
Y_s	-2	0	m
Rud_delta	-10	10	°

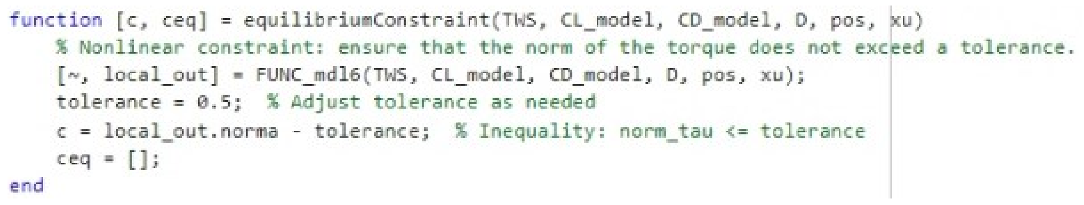
Table 19 fmincon boundaries

The optimisation itself was carried out in a script called "FUNC_mdl6" which returned the boat speed in the xy plane, the VMG and a structure in which all the coefficients and forces' values were saved. This script was used as cost function for the *fmincon* optimisation, and in addition, a nonlinear constraint function was implemented in order to guide *fmincon* towards solutions that maximised VMG while still guaranteeing equilibrium. The system was considered in equilibrium when the norm of the forces vector was close to zero, within a 0.5 tolerance.



```
out.V = -sqrt(u^2+v^2);
out_res = -V_boat*cos(deg2rad(TWA)+angle(u+1i*v));
out.sail = sail;
out.rud = rud;
out.tot = tot;
out.JDX = JDX;
out.dagg = dagg;
out.AWA = rad2deg(AWA);
out.norma = norm(tot.TAU);
out.beta_angle = rad2deg(beta_angle);
out.alpha = rad2deg(alpha);
out.VMG = V_boat*cos(deg2rad(TWA)+angle(u+1i*v));
out.angle_wind = rad2deg(deg2rad(TWA)+angle(u+1i*v));
```

Figure 53 Outputs of FUNC_mdl6



```
function [c, ceq] = equilibriumConstraint(TWS, CL_model, CD_model, D, pos, xu)
    % Nonlinear constraint: ensure that the norm of the torque does not exceed a tolerance.
    [~, local_out] = FUNC_mdl6(TWS, CL_model, CD_model, D, pos, xu);
    tolerance = 0.5;  % Adjust tolerance as needed
    c = local_out.norma - tolerance;  % Inequality: norm_tau <= tolerance
    ceq = [];
end
```

Figure 54 Nonlinear constraint equilibrium function

FUNC_mdl6 begins by collecting the decision variables supplied of *fmincon*. Then it calculates the boat's incidence angles and uses these to determine hydrodynamic forces in the flow reference frame. These forces are then rotated and translated into the body-fixed axes, where the resulting moments are computed and summed into a single vector, TAU. The Euclidean norm of TAU serves as the equilibrium indicator.

10.2.6 Geometry optimisation

The geometry optimisation is carried out in two optimisation loops, an external one where a genetic algorithm is exploring the geometric parameters design space, and an inner one represented by the fmincon routine with the aim of finding the optimal boat-sail trim to maximise velocity for that particular sail geometry.

Two additional scripts are required to execute this routine:

- OPT_SAIL is devolved to setting up the genetic algorithm (GA) parameters and the post processing of the results
- OBJECTIVE_FUNCTION_VMG is used to launch the FUNC_mdl6 for each design, verify equilibrium for the analysed sail and extract the maximum VMG, which serves as objective function for the GA.

The boundaries of the geometric parameters are initially presented in Table 6 and reported here below in Table 20.

$PARAMETER$	$LOWER\ BOUNDARY$	$UPPER\ BOUNDARY$
$C1$	950	1550
$C2$	800	1200
G	10	50
PR	0.5	1
$PROFILE1$	1	4
$PROFILE2$	1	4
α	-15	0
δ	-15	0

Table 20 Geometrical parameters and relative boundaries

11. Results' analysis

The optimisation routine has been run several times changing for example initial conditions and sail surfaces, to test its robustness, and it always converged on the final result reported in Table 21.

$C1$	$C2$	GAP	$\%C1$	$PROFILE\ 1$	$PROFILE\ 2$
1550	1189.5	10.09	0.58	$NACA0012$	$NACA0012$

Table 21 Optimal design parameters

The analysis of the optimisation results shows that the chord of the wing sail was maximised on the first element and flap moved towards the upper boundary as well, which means that the height of the sail was being reduced. This suggests that the algorithm was finding shorter sails to be more robust and better performing than taller configurations. The rotation point of the flap lies almost at mid-length of the first wing segment, and the gap between the two is minimised as much as possible. Among all the tested profiles, he NACA 0012 was perceived as the best performing airfoil, likely because of its slenderer shape.

Figure 55, Figure 56, Figure 57 and Figure 58 present the final CAD model of the optimised sail. The mast and the flap rotation axis positions are displayed as well.

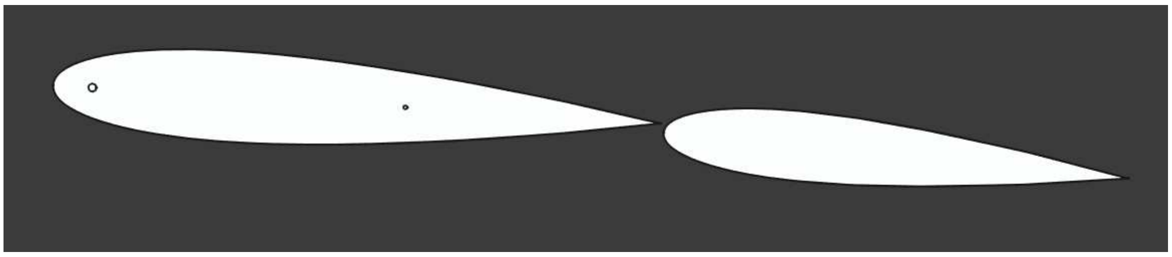


Figure 55 Airfoil sections of optimal sail

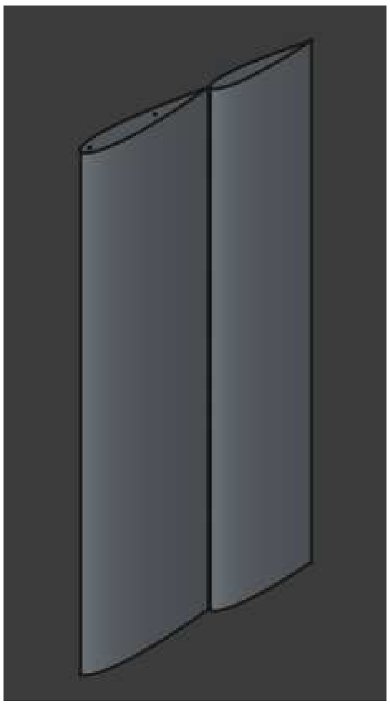


Figure 56 Isometric view optimal sail

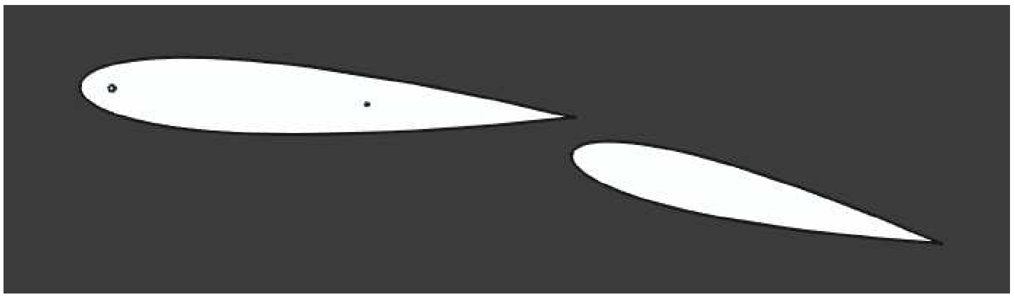


Figure 57 Top view at different angles

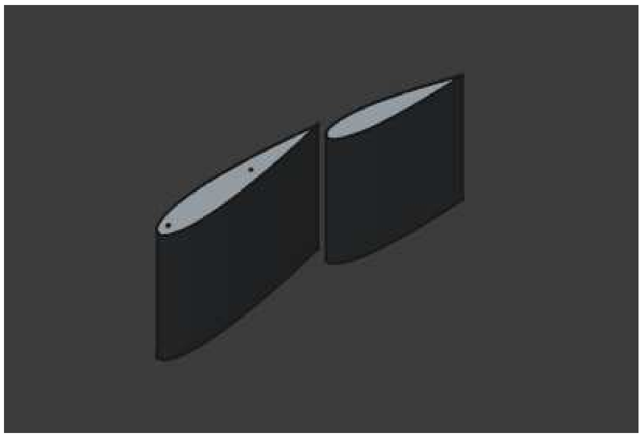


Figure 58 Isometric view at different angles

The GA had following parameters reported in Table 22, following the values recommended by the MATLAB documentation. Figure 59 plots the iterations of the algorithm over successive generations and shows that the GA terminated after just 108 generations, even though the stopping criteria was set to 200. This means that the algorithm couldn't find any better sail configurations as the iterations progressed, strongly suggesting convergence to a global optimum. Additionally, the points' distribution exhibits a clear downward trend, which indicates good convergence of the GA.

$N°\ of\ individuals\ per\ generation$	100
$N°\ of\ generations$	200

Table 22 Genetic algorithm parameters

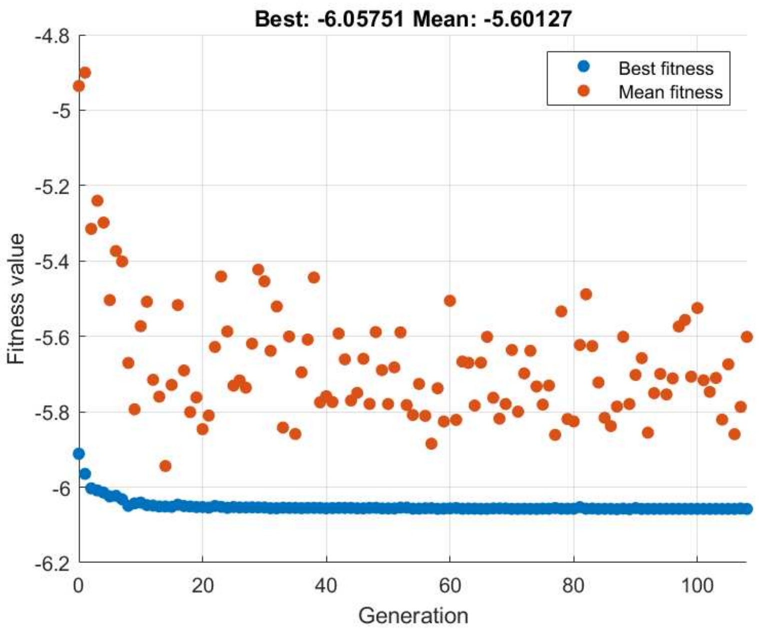


Figure 59 GA convergence

Once the GA was completed, Glaros's performance was reevaluated, and the final results are reported in Table 23 below. Equilibrium was reached since the norm of the force vector *tau* was equal to 0.50, and the largest residual was of 0.4 N, meaning that on the other dofs was even lower, around 0.01 N.

$Quantity$	$Value$	$Unit$
VMG	6.05	m/s
$Optimal\ heading$	42,38	°
$Boat\ velocity$	8.8	m/s
$U\ component$	8.78	m/s
$V\ component$	0.64	m/s
$Leeway\ angle$	4.17	°
$Effective\ optimal\ angle$	46,58	°
$Highest\ equilibrium\ residual$	0.4	N

Table 23 Optimal results

The optimal heading to be followed by Glaros in order to obtain the best performance for the first leg of the regatta is 42.38°, with a speed of 8.8 m/s, corresponding to roughly 17 knots. For this condition the leeway ends up being around 4.1° degrees, in line with theoretical expectations.

The polar plot of the optimised sail, displayed in Figure 60, exhibit the expected trend for CL and CD curves: the CL is increasing linearly with the angle of attack, while CD is increasing quadratically, as a demonstration of the model's predictive accuracy.

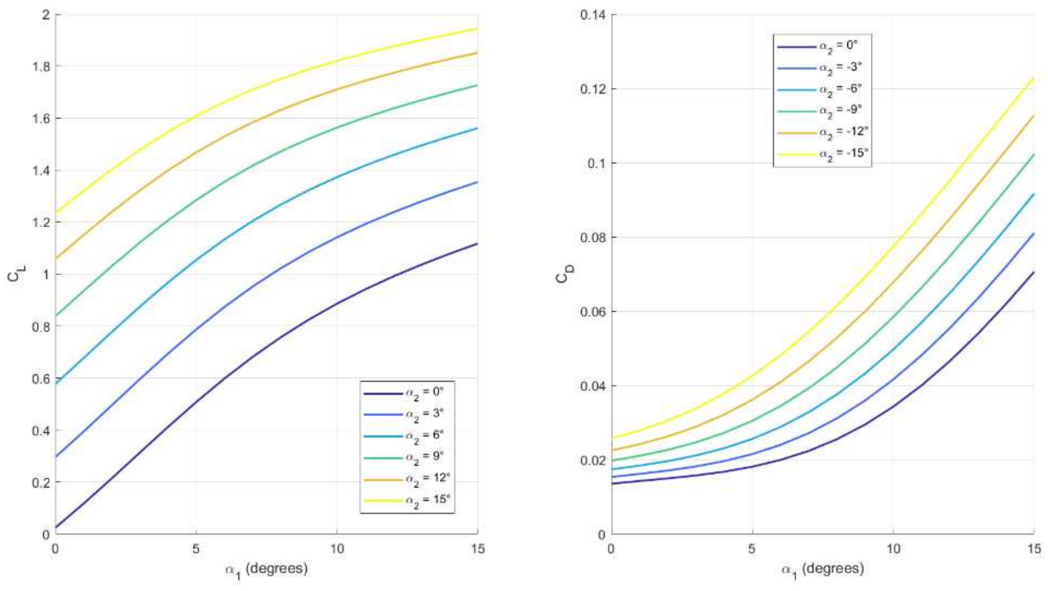


Figure 60 Sail's aerodynamic coefficients' polars

Figure 61 reports polars graphs for different sail geometries, clearly illustrating that the geometric parameters' variation was significantly affecting the aerodynamic performance of the configurations evaluated.

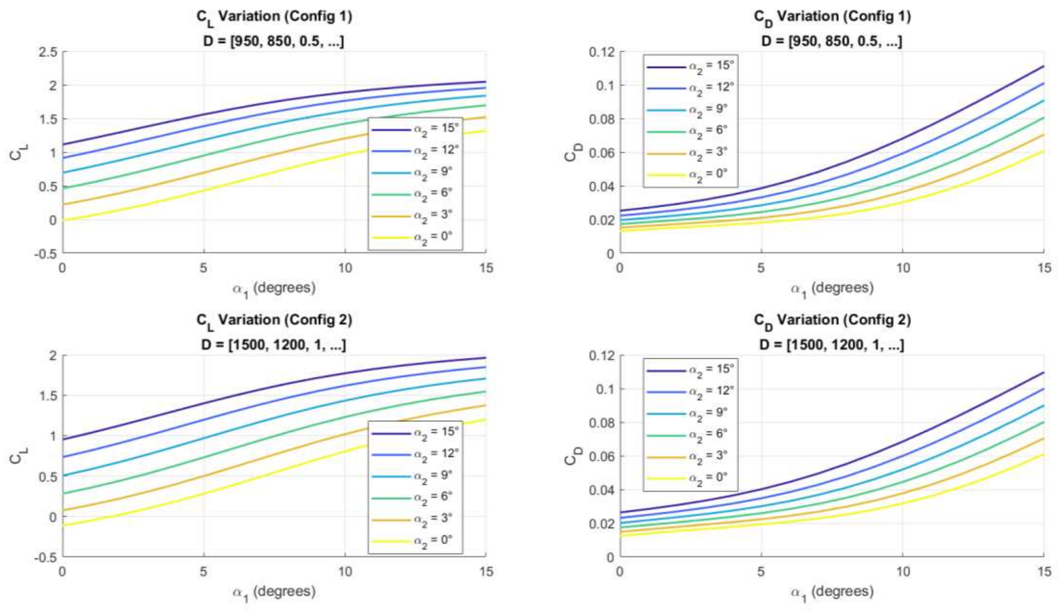


Figure 61 Comparison between polars of different geometries

The obtained results are consistent with the physical principles governing them, demonstrating the predictive capabilities and reliability of the developed system, despite the simplifications that were assumed during the development of this work. By integrating more accurate physics and models if more computational power was available, the quality of the results would increase even more, but the purpose of this work was to set the basis for a geometry optimisation. Four core components can be identified as the building block of this workflow: geometry parametrisation, CFD model, neural network training and VPP. By refining each of these “blocks” even more accurate and robust optimisations can be achieved, underscoring the significant potential of this approach.

12.Conclusions

In this work, a comprehensive modelling and simulation framework was developed to predict and optimize the performance of Glaros under upwind sailing conditions. A static six-degree-of-freedom MATLAB model was implemented to compute equilibrium trim states and predict boat behaviours, into which environmental inputs (wind speed, course geometry) and sail-geometry parameters were fed. The sail aerodynamic coefficients were modelled with an accurate neural network, and by replacing costly, time-intensive CFD runs with near-instantaneous model evaluations, this approach eliminates the simulation stage as a bottleneck in fluid-dynamics optimizations. This means that it is possible to iterate through design variations far more efficiently and still achieve CFD-level accuracy and reliability.

The several limitations that have been presented do not reduce the value of the workflow, rather they pave the way for future research to improve its optimisation capabilities. Future extensions are limitless and concern all fields of this thesis.

The sail's geometry can be captured more precisely by adding extra shape parameters, like a twist angle, and by expanding the airfoil library with additional sections. Instead of describing the leech as a simple straight line and getting thus just rectangular shapes, it can be described with a flexible B-spline curve. This combination would give us far greater design fidelity and control.

CFD simulations could be improved by fine-tuning the turbulence model's parameters, by carrying out a more thorough mesh and CFL convergence study. 3D simulations would further improve the reliability of results significantly but would also require more computational power than the one that was available for this work.

By training the neural network on data from simulations of varying fidelity, computational efficiency could improve even further. A multi-fidelity surrogate model would be built by generating a high number of low fidelity samples through simple but fast simulations, like a panel method algorithm or CFD models with a coarser mesh. This dataset would then be enriched by few selected high-fidelity simulations, yielding the same level of accuracy as a single-fidelity model but at a fraction of the cost.

Adaptive sampling can be made more efficient by choosing which new points to evaluate based on a variance analysis, rather than indiscriminately adding samples each iteration. In practice, this means that instead of simulating three new candidates every time, just the single most informative candidate would be identified, simulated, and added to the dataset, thereby cutting down on computation time even further.

Lastly, the static model for boat-trim evaluation can be substituted by a transient dynamic model, and by adopting more precise foil polars, more accurate results could be predicted. A more robust controller for the sailor behaviour and adding the roll degree of freedom would be beneficial as well. Additionally, instead of considering just one wind condition, the optimisation could result more robust if a more realistic wind distribution was considered, implemented as a Markov chain model or by consider real wind fields taken from Windy's API. An example of Markov chain implementation is presented in (40).

13.Bibliography

1. Viola IM. A numerical method for the design of ships with wind-assisted propulsion.

2. Malmek K. Rapid aerodynamic method for predicting the performance of interacting wing sails. Ocean Engineering. 2024;

3. Silva MF. Rigid wing sailboats: A state of the art survey. Ocean Engineering. 2019;

4. Caraher SP, Hobson GV, Platzer MF. Aerodynamic Analysis and Design of High-Performance Sails.

5. Harrison.Turner S15.

6. Turnock SR, Cambell IM, Magherini M. PARAMETERS AFFECTING THE PERFORMANCE OF THE C-CLASS WINGSAIL.

7. What are Flettner rotors and how do they work? – IMO [Internet]. 2024 [citato 17 marzo 2025]. Disponibile su: https://futurefuels.imo.org/faq/what-are-flettner-rotors-and-how-do-they-work/

8. Diederichsen K. Back To The Future: How Rotor Sails are now a credible option to tackle climate challenge [Internet]. SAFETY4SEA. 2023 [citato 17 marzo 2025]. Disponibile su: https://safety4sea.com/back-to-the-future-how-rotor-sails-are-now-a-credible-option-to-tackle-climate-challenge/

9. Habibic A. CRAIN Technologies suction wing gets green light from Bureau Veritas [Internet]. Offshore Energy. 2022 [citato 17 marzo 2025]. Disponibile su: https://www.offshore-energy.biz/crain-technologies-suction-wing-gets-green-light-from-bureau-veritas/

10. Pinterest [Internet]. [citato 25 giugno 2025]. SailRaceWin: America's Cup: AC45 Wing-Sailed Catamaran Under Sail in Auckland | Sailing, Americas cup sailing, Catamaran. Disponibile su: https://www.pinterest.com/pin/sailracewin-americas-cup-ac45-wingsailed-catamaran-under-sail-in-auckland-in-2023--394627986095621363/

11. What are soft wing sails and how do they work? – IMO [Internet]. 2024 [citato 17 marzo 2025]. Disponibile su: https://futurefuels.imo.org/faq/what-are-soft-wing-sails-and-how-do-they-work/

12. SkySails Marine [Internet]. [citato 17 marzo 2025]. Home. Disponibile su: https://skysails-marine.com/

13. ResearchGate [Internet]. [citato 17 marzo 2025]. Figure 2.2: A ship sailing with the assistance of a kite sail. (Photo:... Disponibile su: https://www.researchgate.net/figure/A-ship-sailing-with-the-assistance-of-a-kite-sail-Photo-Skysails_fig5_338901158

14. Airfoil data information [Internet]. [citato 15 marzo 2025]. Disponibile su: http://airfoiltools.com/airfoil/

15. Larsson L, Eliasson RE, Orych M. Principles of yacht design. Fifth edition. London$aOxford$aNew York$aNew Delhi$aSydney: Adlard Coles; 2022. 400 p.

16. Mason AP. STOCHASTIC OPTIMISATION OF AMERICA'S CUP CLASS YACHTS.

17. Han ZH, Zhang KS. Surrogate-Based Optimization. In: Roeva O, curatore. Real-World Applications of Genetic Algorithms [Internet]. InTech; 2012 [citato 10 marzo 2025]. Disponibile su: http://www.intechopen.com/books/real-world-applications-of-genetic-algorithms/surrogate-based-optimization

18. Peart T, Aubin N, Nava S, Cater J, Norris S. Multi-Fidelity Surrogate Models for VPP Aerodynamic Input Data. Journal of Sailing Technology. 9 febbraio 2021;6(01):21–43.

19. Nandi S. 12 Types of Activation Functions in Neural Networks: A Comprehensive Guide [Internet]. Medium. 2025 [citato 24 giugno 2025]. Disponibile su: https://medium.com/@sushmita2310/12-types-of-activation-functions-in-neural-networks-a-comprehensive-guide-a441ecefb439

20. James G, Witten D, Hastie T, Tibshirani, Taylor J. An Introduction to Statistical learning - Python. An Introduction to Statistical learning - Python.

21. GeeksforGeeks [Internet]. 21:10:47+00:00 [citato 16 marzo 2025]. ML | Underfitting and Overfitting. Disponibile su: https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/

22. Cella U, Salvadore F, Ponzini R, Biancolini ME. VPP Coupling High-Fidelity Analyses and Analytical Formulations for Multihulls Sails and Appendages Optimization.

23. Melis MF, Tannenberg R, Boyd S, Abdel-Maksoud M. AC75 Aerodynamic Performance Prediction via BEM. Journal of Sailing Technology. 19 dicembre 2024;9(01):143–74.

24. Kuang L. Effect of chord length ratio on aerodynamic performance of two-element wing sail. Ocean Engineering. 2023;

25. Chapin V, Gourdain N, Verdin N, Fiumara A, Senter J. AERODYNAMIC STUDY OF A TWO-ELEMENTS WINGSAIL FOR HIGH PERFORMANCE MULTIHULL YACHTS. 2015 [citato 21 agosto 2024]; Disponibile su: http://rgdoi.net/10.13140/RG.2.1.2891.8883

26. Computational fluid dynamics. In: Wikipedia [Internet]. 2025 [citato 11 marzo 2025]. Disponibile su: https://en.wikipedia.org/w/index.php?title=Computational_fluid_dynamics&oldid=1279487154

27. Reynolds-Averaged Navier-Stokes Equations | Symscape [Internet]. [citato 11 marzo 2025]. Disponibile su: https://www.symscape.com/reynolds-averaged-navier-stokes-equations

28. Two equation turbulence models -- CFD-Wiki, the free CFD reference [Internet]. [citato 11 marzo 2025]. Disponibile su: https://www.cfd-online.com/Wiki/Two_equation_models

29. SST k-omega model -- CFD-Wiki, the free CFD reference [Internet]. [citato 11 marzo 2025]. Disponibile su: https://www.cfd-online.com/Wiki/SST_k-omega_model

30. SimScale [Internet]. [citato 16 marzo 2025]. K-Omega Turbulence Models | Global Settings. Disponibile su: https://www.simscale.com/docs/simulation-setup/global-settings/k-omega-sst/

31. Graf K. Comparison of full 3D-RANS simulations with 2D-RANS/lifting line method calculations for the flow analysis of rigid wings for high performance multihulls.

32. Fluid Mechanics 101 [Internet]. [citato 15 marzo 2025]. Disponibile su: https://www.fluidmechanics101.com/pages/tools.html

33. Fiumara A. Numerical and experimental analysis of the flow around a two-element wingsail at Reynolds number 0.53×106.

34. Coefficient of determination. In: Wikipedia [Internet]. 2025 [citato 13 marzo 2025]. Disponibile su: https://en.wikipedia.org/w/index.php?title=Coefficient_of_determination&oldid=1277871013

35. Arize AI [Internet]. [citato 13 marzo 2025]. Root Mean Square Error (RMSE) In AI: What You Need To Know. Disponibile su: https://arize.com/blog-course/root-mean-square-error-rmse-what-you-need-to-know/

36. Hyperparameter optimization. In: Wikipedia [Internet]. 2025 [citato 14 marzo 2025]. Disponibile su: https://en.wikipedia.org/w/index.php?title=Hyperparameter_optimization&oldid=1278767439

37. Probabilistic Model - an overview | ScienceDirect Topics [Internet]. [citato 14 marzo 2025]. Disponibile su: https://www.sciencedirect.com/topics/materials-science/probabilistic-model

38. Prabhakaran S. Bayesian Optimization for Hyperparameter Tuning - Clearly explained. [Internet]. Machine Learning Plus. 2024 [citato 14 marzo 2025]. Disponibile su: https://www.machinelearningplus.com/machine-learning/bayesian-optimization-for-hyperparameter-tuning/

39. Hagemeister N, Flay RGJ. Velocity Prediction of Wing-Sailed Hydrofoiling Catamarans.

40. Dalang RC. Stochastic optimization of sailing trajectories in an upwind regatta.