ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

SCUOLA DI INGEGNERIA ED ARCHITETTURA

CORSO DI LAUREA MAGISTRALE IN
INGEGNERIA MECCANICA

FLUID-DYNAMIC AND MECHANICAL DESIGN OF A SCREW PROPELLER FOR AN ELECTRIC CATAMARAN

Tesi di Laurea in Macchine M

Relatore

Prof. Antonio Peretto

Presentata da

Manuel Cotoni

Correlatore

Prof. Andrea De Pascale

Prof. Francesco Melino

Sessione marzo 2024

Anno Accademico 2022/2023

Do not go gentle into that good night...

Acknowledgements

I would like to thank Professor Peretto for his constant interest and support in developing this challenging project, which was a completely new topic not just for me but for him as well.

I would like to thank the amazing *UniBoAT* team, without which this thesis would never been started. It has been a great hands-on practice experience that every engineering student should take part in. I have met wonderful people and applied my engineering knowledge to real problems for the first time, and even if this activity was often overwhelming, in the end, it was worth all the effort.

Last but not least, I want to thank my parents, who have made possible my education and have always motivated me to pursue and achieve better results.

Go UniBoAT!

Abstract

This thesis project presents a methodology to design and numerically validate screw propellers for maritime applications. The method was designed and used during the 2022/2023 season of the *UniBoAT* project.
The theoretical approach applied is *Blade Element Momentum Theory* (*BEMT*). A *MATLAB* code was developed to implement the *BEMT* method. Moreover, an optimization algorithm was implemented in the program to optimize the blade's pitch and chord distributions. Then, CFD and FEA simulations with *Ansys Workbench* were carried out to compare analytical and numerical results. While a good correlation between *BEMT* and CFD has been achieved (particularly for the propeller's thrust and efficiency), analytical methods to compute blade stress have seemed to be not reliable in maximum stress predictions because the blade's root rounding geometry is usually not taken into account in the computation.
Finally, the propellers were manufactured through 3D printing technology and tested on the boat. The propeller's real performance was compared with the previous design and, although the tests weren't comprehensive, the objective of obtaining a bigger thrust from the propeller has been achieved but with bigger power absorption than was expected.
Further investigations and improvements in the simulations set-up will be needed to improve the correlation between *BEMT* and real tests.

Sommario

Questo progetto di tesi presenta un metodo di design e validazione numerica di eliche per applicazioni marittime. Il metodo è stato sviluppato e usato nel corso della stagione 20022/2023 del progetto *UniBoAT*.
L'impostazione teorica utilizzata è la *Blade Element Momentum Theory* (*BEMT*). Un programma *MATLAB* è stato sviluppato per implementare il metodo *BEMT*. Inoltre, un algoritmo di ottimizzazione per ottimizzare le distribuzioni di pitch e corda della pala è stato implementato nel programma. Dopodiché, si sono eseguite simulazioni CFD e FEA tramite *Ansys Workbench* per confrontare risultati analitici e numerici. Benché una buona correlazione sia stata riscontrata tra *BEMT* e CFD (in particulate per la spinta e l'efficienza dell'elica), i metodi analitici per calcolare la distribuzione di stress nella pala sono risultati poco affidabili nella previsione della tensione massima della pala poiché solitamente non tengono conto della geometria raccordata della base della pala.
Infine, le eliche sono state prodotte tramite stampa 3D e testate sulla barca. Le prestazioni finali dellee eliche sono state confrontate con le prestazioni del design precedente e, benché i test non fossero completi, a causa della mancanza di un banco prova per eliche, l'obbiettivo di ottenere una Maggiore spinta dalle eliche è stato raggiunto ma con un assorbimento di potenza maggiore del previsto.
Ulteriori indagini e miglioramenti nel set-up delle simulazioni saranno necessari per migliorare la correlazione tra la *BEMT* e i test reali.

Contents

List of Figures

Chapter 1

Introduction

1.1 UniBoAT Project

The University of Bologna Argonauts Team (UniBoAT) is a fascinating project started by the Department of Industrial Engineering (DIN) of the University of Bologna back in 2019. The project aims to study and develop innovative and green solutions for the maritime sector. In particular, students and doctorates who join the Team design and build a racing catamaran to take part in the Monaco Energy Boat Challenge (MEBC) in the Principality of Monaco, held every year in July.
UniBoAT is a great opportunity for students who want to put their knowledge and theoretical skills into practice, facing real engineering problems and challenges. Moreover, it is a platform for the students to develop and refine new and advanced technologies that could revolutionize the maritime industry in the coming years.
MEBC's goal is to promote a faster technology transition in the maritime industry, from carbon-fuel-based technologies to green and sustainable ones. For this reason, Futura, the UniBoAT's catamaran, is equipped with an advanced propulsion system that is powered by two electric engines. These engines are supplied by lithium-ion battery packs that are recharged by solar panels and a hydrogen fuel cell. The boat is also fitted with two standard Wageningen B-Series propellers, which were designed by the students to achieve maximum efficiency during work. This task is not straightforward since the competition includes four different types of races. These are:

- Endurance race, which tests the boat's reliability and efficiency;
- Speed test, which tests maximum speed;
- Slalom race, which tests manoeuvrability;
- Championship race, which is a 1-vs-1 race that requires both good manoeuvrability and speed.

Fortunately, the competition rules allow every team to change propeller after each race. Therefore, it is convenient to design different propellers depending on the race topology.

Figure 1.1: *Futura*, the UniBoAT's race hydrogen catamaran. Energy Boat Challenge of Monaco, July 2023

1.2 Aim of this work

This thesis work was born from my curiosity to deep dive into the fluid dynamic study of marine propellers and from my pledge to the UniBoAT Team to develop a simple, efficient and rapid propeller design methodology, functional to design quickly new custom propellers based on the boat's thrust, power, and velocity requirements. Since the boat's parameters can change frequently due to changes in race rules and improvements in boat design or components, it is important to have a fast and reliable methodology for designing new propellers when the boat's operating conditions change. Additionally, this methodology should be capable of designing high-efficiency propellers for endurance races and high-speed velocity propellers for championship, slalom, and speed races to achieve the best performance in each race.
For this season, since we already had good propellers that could reach high efficiency, we decided to focus on developing high-velocity propellers. To achieve this goal, we developed a MATLAB program that could design propellers with custom blade geometry and help us reach maximum velocity in a specific work condition. We then compared the results from the MATLAB code with CFD and FEA simulations, and finally with real test data to ensure the reliability of the methodology.

1.3 Structure of the thesis

This thesis is organized into the following chapters and topics:

1. Chapter 2: Blade Element Momentum Theory

1.1 Literature Review of Propeller Design Methodologies: a brief description of the principal design methodologies adopted in propeller design starting from Rankine's Momentum Theory to modern CFD methods, their pros and cons and up-to-date published design results;

1.2 Mathematical Derivation of the *BEMT*: motivations on the chosen model theory, its physical and mathematical derivation and notes about the problem regarding *BEMT* system numerical solution;

1.3 Summary of *BEMT* equations: a summary of the *BEMT* hypothesis and final system of equations.

2. Chapter 2: *BEMT* Implementation in MATLAB

   2.1 MATLAB Code Structure and Functioning: a description of the main ideas the code is based on, the *XFOIL* software usage to get airfoil polar data, the main inputs to start the design, the optimization loop structure and the verification of cavitation;

   2.2 MATLAB Results: discussion of the propeller designed and optimized through the *MATLAB* code and its main performance results;

3. Chapter 3: Propeller CAD model, CFD and FEM analysis

   3.1 PTC Creo CAD Model: description of the propeller's 3D modelization process;

   3.2 CFD Setup and Results: fluid-dynamic simulations to analyze propeller performance and to compare it with the *BEMT* results;

   3.3 FEA Setup and Results: finite-element structural analysis to guarantee mechanical integrity of the designed propeller;

4. Chapter 4: Propeller Manufacturing, Testing and Conclusions

   4.1 Propeller Production and Real Test: 3D printings of the designed propeller, propeller testing in *Marina di Ravenna* e collected data analysis;

   4.2 Final Results: comparison between old and new propellers with the test data;

   4.3 Conclusions and Future Works: final discussion on this thesis results and possible future works to improve the design methodology.

Chapter 2

Blade Element Momentum Theory

2.1 Introduction

Propeller design theories have been evolving since the late 1800s and are currently widely used in the design of marine propellers, wind turbines, and aircraft propellers, especially for drones or small aeroplanes. Carlton [9] provides a comprehensive overview of design methodologies, but we will summarize the most important aspects and the pros and cons of the major theories in this chapter. Moreover, the mathematical derivation of the *BEMT* method adopted for this work is here explained.

2.2 Literature review of propeller design methodologies

Rankine [44] and Froude [21],[20] were the pioneers of providing a method to design and analyze propeller action. The Momentum Theory proposed by Rankine describes the propeller as an actuator disc with an infinite number of blades through which the fluid flows. The fluid is ideal, and no losses due to frictional drag are taken into account. By analyzing the increase in kinetic energy of the slipstream and the flow rate through the disc, it is possible to calculate the thrust generated by the disc and the power absorbed. While Momentum Theory describes the global phenomena of the propeller action, it is useless for designing blade profiles. On the other hand, the Blade Element Theory of Froude examines the local propeller action by studying the blade divided into elementary strips. By knowing the resultant incident velocity, it is possible to calculate the lift and drag experienced by the local strip of the blade and then thrust and momentum through projection. Although the early theories failed in some respects to predict propeller performance accurately, they laid the foundation for all future theories.
In the next decades, thanks to the works of Prandtl, Betz [1], [2], Goldstein [25], and Burrill [8], a comprehensive propeller theory was developed. The advance in the vortex theory by Prandtl, which established that the lift of a swing is generated due to the development of a circulation around the section and that trailing vortices emanated from the blade tip, allowed Betz and Goldstein to characterize the losses which the propeller is subjected due to a finite number of blades and a finite blade. Moreover, Tachmindji [49] expanded this work taking into account the presence of the propeller hub.
Finally, the work of Burrill took into account all the previous works to complete the propeller

design theory which today is widely known as Blade Element Momentum Theory (BEMT). Essentially, Burrill's method is a strip theory which combines the principles of Momentum Theory, Blade Element Theory, and vortex theory. It also introduces induced inflow factors for axial and tangential velocities. Burrill's analysis procedure represents the first coherent step in establishing a propeller calculation procedure. It works quite well for the moderately loaded propeller working at or near its design condition. However, at either low or high loading conditions the theory does not behave as well.
Today, BEMT is largely adopted for marine propellers, wind turbines and aircraft rotors design, due to its simplicity of implementation and low computational effort, which allow designers to define quickly a first-attempt solution that must be validated through CFD simulations later in the design process. Some notable literature examples of propeller design and analysis with BEMT are Benini [19], Phillips [41], Winarto [51], Rwigema [45], Ning [38], Masters [35] and Dehouck [12].
In the following years, several design methods were introduced to cope with the increase in propulsive power transmitted. For instance, Eckhardt and Morgan's [31] design method is an approximated methodology which utilizes empirical data, as the ideal efficiency estimated by Kramer's diagram and minimum cavitation number diagrams, to optimise the propeller. Today this methodology is still widely used thanks to its simple and fast algorithm. However, the accuracy in predicting performance is limited to lightly and moderately loaded propellers. Some up-to-date examples of this design methodology are provided by Parvez [39] and Rahman [43] (who presented a hybrid method with some aspects of more recent theories).
With the advancements in computational capabilities in the second half of the 20th century, several new methods have been introduced and developed. One such method is the Lifting Surface model, which consists of replacing the blade with an infinitely thin surface, which takes the form of the blade camber, upon which a distribution of vorticity is placed in both the spanwise and chordal directions. During the design phase, knowing the approximate distributions of chord, skew, rake and section thickness, source and vortex distributions representing the blades and their wake need to be placed on suitable reference surfaces to enable the induced velocity field to be calculated. Recent applications of this theory were presented for instance by Grassi [26] and Brizzolara [47]. However, since computers with sufficient capabilities were not always available, Lifting Line-lifting Surface hybrid models were introduced. Incorporating surface corrections (Morgan et al. [36]) and cavitation prediction methods faster design computations were admissible.
Thanks mainly to Kerwin's work, during the '60s and '70s Vortex Lattice Method was introduced. This is a subclass of the Lifting Surface method, where the continuous distributions of vortices and sources are replaced by a finite set of straight-line elements of constant strength whose endpoints lie on the blade camber surface. From this system of vortices, the velocities are computed at several suitably located control points between the elements. Some design examples with Vortex Lattice theory were provided by Chung [11], Epps [18] and Stoye [48].
In recent years, there have been significant advancements in the application of Computational Fluid Dynamics. With the use of Reynolds Averaged Navier-Stokes (RANS) method, Large Eddy Simulation (LES) techniques or Direct Numerical Simulations (DNS) is possible to get useful insights into the viscous and cavitating behavior of propellers. However, while these approaches ensure good accuracy and reliability in performance predictions, they are more in-

tended for propeller analysis rather than propeller design and optimization, due to the computational effort required. In literature, considerably large usage of CFD methods has been done. For instance Dubbioso [14], Elghorab [15], Patil [40] and Bulten [7].
It is worth highlighting that several open-source codes, which implement one of the previous theories to design propellers and turbines, are already available online. For instance, pyBEMT, published by Giljarhus [23], is a Python application that implements a BEMT algorithm to design aircraft propellers; OpenProp, instead, is a MATLAB code for designing marine propellers through Lifting-Line Theory and has been developed by Epps [17]; finally, AeroDyn is a tool integrated in OpenFAST, an open source software developed by Jason Jonkman and Mike Sprague [29], to design offshore wind turbine through implementation of BEMT with vortex-wake corrections.
Although open-source codes are often powerful and easy to use with user-friendly interfaces, literature validations are not always available. For this reason, experimental tests are always recommended.

2.3 Mathematical derivation of BEMT

For this thesis development, the BEMT method has been chosen due to its simple software implementation and its fast solution computation, which allows the designer to quickly perform optimization algorithms of the propeller parameters. Validations with both experimental data and CFD simulations of the theory can be easily found in literature, for instance: Bergmann [6], Kamaldeep [46], Abdelkhalig [3] and Bangga [22]. As already has been stated, the BEMT model is not commonly reliable when working conditions are far away from the design working point.
For a comprehensive derivation and explanation of the theory, the book *Wind Turbine Aerodynamics and Vorticity-Based Methods* by Branlard [16] is suggested (note that Branlard derives the method with the turbine sign convention). In this section, the main hypothesis, equations and results of the BEMT method are going to be discussed.

A propeller consists of identical blades attached axisymmetrically to a common hub. Each blade may be thought of as being similar to a high aspect ratio wing, thus the analysis of a propeller may borrow some familiar ideas from the aerodynamic analysis of an aircraft wing. When a wing with an airfoil section crosses a fluid, due to its geometry, fluid particles are accelerated around the curved shape. Since on one side (suction side), the fluid increases its speed, while on the other side (pressure side) the fluid decelerates, due to Bernoulli's equation a static pressure difference is generated between the two sides and a lifting and a drag force is produced on the aerofoil.
In the case of a propeller, the flow analysis is more complex because the propeller ”wing” (the blade) is both moving forward and rotating around the propeller's longitudinal axis. Therefore, the lifting line theory applied to study aerofoil behaviour should be modified to take into account the rotation of the blade.
The blade can be divided into parallel blade elements, each one can be regarded as an infinitesimal aerofoil with a radial length of dr, and chord $c(r)$, with r being the radial distance from the centre of rotation. In this way, the problem of 3-D flow analysis is reduced to a simpler 2-D

analysis, assuming that each blade element does not influence the fluid flow of the other blade elements.
The relative velocity experienced by this element, namely U_{rel}, is decomposed into a longitudinal component normal to the rotor U_n and a tangential component in the rotor plane U_t. Three angles are defined based on the velocity vectors and blade element definition: the inflow angle Φ, the angle of attack α, and the pitch angle θ. The relations between these angles and the velocity components are:

$$\alpha = \Phi - \theta \tag{2.1}$$

$$\Phi = atan\frac{U_n}{U_t} \tag{2.2}$$

$$U_{rel} = \sqrt{U_n^2 + U_t^2}[m/s] \tag{2.3}$$

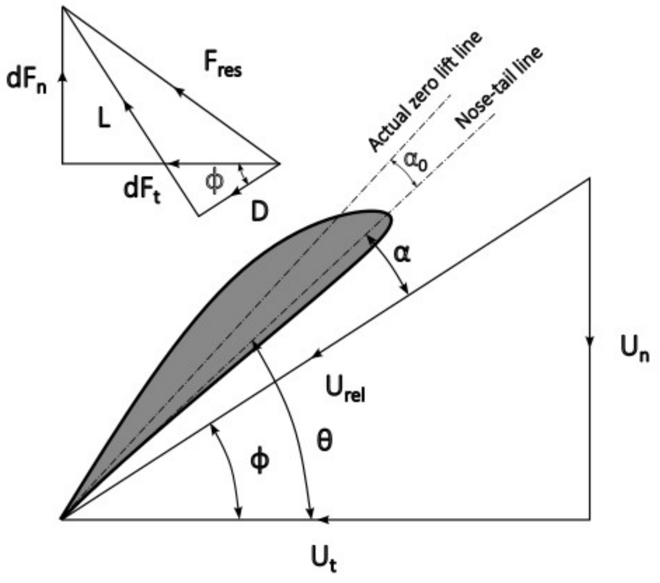


Figure 2.1: Velocity and forces vectors for a blade element

From the previous discussion on the flow around an aircraft wing, we know that aerofoil produces lift and drag forces. Since the lift and drag coefficients of the airfoil are assumed to be known at any given section of the blade, the lift and drag forces applied on the blade element of surface $dS = c(r)dr$ are:

$$dF_L = \frac{1}{2}\rho dS U_{rel}^2 C_l(\alpha) \quad [N] \tag{2.4}$$

$$dF_D = \frac{1}{2}\rho dS U_{rel}^2 C_d(\alpha) \quad [N] \tag{2.5}$$

We can express the two forces per unit of length by dividing by the span dr:

$$L = \frac{1}{2}\rho c(r)U_{rel}^2 C_l(\alpha) \quad [N/m] \tag{2.6}$$

$$D = \frac{1}{2}\rho c(r)U_{rel}^2 C_d(\alpha) \quad [N/m] \tag{2.7}$$

The lift and drag forces can be projected on the longitudinal axis and the rotor plane. In this way, we get an axial component dF_n, the thrust component, and a tangential component dF_t, the torque component:

$$dF_n = dF_L cos(\phi) - dF_D sin(\phi) \quad [N] \tag{2.8}$$

$$dF_t = dF_L sin(\phi) + dF_D cos(\phi) \quad [N] \tag{2.9}$$

By analogy to the lift and drag coefficients, we can express the axial and tangential components as a function of axial and tangential coefficients:

$$dF_n = \frac{1}{2}\rho dS U_{rel}^2 c_n(\alpha) \quad [N] \tag{2.10}$$

$$dF_t = \frac{1}{2}\rho dS U_{rel}^2 c_t(\alpha) \quad [N] \tag{2.11}$$

Where axial and tangential coefficients can be expressed as functions of the lift and drag coefficients:

$$c_n = C_l cos(\phi) - C_d sin(\phi) \tag{2.12}$$

$$c_t = C_l sin(\phi) + C_d cos(\phi) \tag{2.13}$$

It is noted that the radial dependency of the parameters c, C_l, C_d, α, and Φ has been omitted in the above equations.
Since the rotational symmetry hypothesis of the flow holds, the loading on all blades Z is identical, so the local (i.e. elementary) thrust, torque and power are:

$$dT(r) = ZdF_n(r) = \frac{1}{2}\rho U_{rel}^2(r) Z c(r) dr c_n(r) \quad [N] \tag{2.14}$$

$$dQ(r) = ZrdF_t(r) = \frac{1}{2}\rho U_{rel}^2(r) Z c(r) r dr c_t(r) \quad [Nm] \tag{2.15}$$

$$dP(r) = Z\Omega r dF_t(r) = \frac{1}{2}\rho U_{rel}^2(r) Z c(r) r \Omega dr c_t(r) \quad [W] \tag{2.16}$$

We can define dimensionless coefficients as functions of the reference velocity U_{ref}, which is chosen as the free-stream velocity U_0, namely the boat velocity:

$$C_t(r) = \frac{dT}{\frac{1}{2}\rho U_{ref}^2 2\pi r dr} = \frac{U_{rel}^2(r)\sigma(r)c_n(r)}{U_{ref}^2} \tag{2.17}$$

$$C_q(r) = \frac{dQ}{\frac{1}{2}\rho U_{ref}^2 r 2\pi r dr} = \frac{U_{rel}^2(r)\sigma(r)c_t(r)}{U_{ref}^2} \tag{2.18}$$

$$C_p(r) = \frac{dP}{\frac{1}{2}\rho U_{ref}^3 2\pi r dr} = \frac{U_{rel}^2(r)\sigma(r)c_t(r)\lambda_r}{U_{ref}^2} \tag{2.19}$$

Where local solidity $\sigma = Zc/2\pi r$ and local tip-speed ratio $\lambda = \Omega r/U_{ref}$ have been introduced. The local solidity is an important parameter in the study of rotors and it represents the ratio between the surface area of the Z blade elements $dS(r) = Zc(r)dr$ at the radial position r and an annular area $dA(r) = 2\pi r dr$. While the tip-speed ratio λ is similar to the advanced ratio J which is commonly used in propellers analysis:

$$J = \frac{U_{ref}}{n_{rot}D} = \frac{U_{ref}}{\Omega/(2\pi)2R} = \frac{\pi}{\lambda} \tag{2.20}$$

Anyway, in this work λ has been preferred.

Until now, it has been assumed that the presence of the blade element does not affect the inflow. In reality, the inflow velocity is affected by the presence of the blade. Indeed, there is an incremental axial velocity upstream of the blade induced by the blade element, which can be assumed to be a small fraction of the reference forward velocity. Note that the definition of axial induction factor is only appropriate when a free-stream velocity U_0 exists.
Similarly, there is a reduction in the tangential velocity which gives rise to the elemental torque. This decrement is assumed to be a small fraction of the tangential velocity, and it is in the direction opposing the tangential velocity. The distributions of a and b, the axial and tangential induction factors respectively, are unknown and must be computed. This is the essence of the BEMT method.
From the previous discussion, it follows that the axial velocity seen by the blade element is $(1 + a)U_0$, whereas the tangential velocity is $(1 - b)\Omega r$. Then the resultant velocity and the inflow angle are:

$$U_{rel} = \sqrt{(1+a)^2U_0^2 + (1-b)^2\Omega^2 r^2} \tag{2.21}$$

$$tan\phi = \frac{(1+a)U_0}{(1-b)\Omega r} \tag{2.22}$$

The loads from the BET may be rewritten considering the two induction coefficients. Different forms of the equations may be chosen due to different relations between the inflow parameters:

$$\begin{aligned} \frac{dT}{dr} &= \frac{1}{2}\rho Zcc_n(U_n^2 + U_t^2) = \\ &= \frac{\rho Zcc_n U_0(1+a)\Omega r(1-b)}{2sin(\phi)cos(\phi)} \\ &= \pi r\rho\sigma c_n \frac{U_0(1+a)\Omega r(1-b)}{sin(\phi)cos(\phi)} \end{aligned} \tag{2.23}$$

$$\begin{aligned} \frac{dQ}{dr} &= \frac{\rho Zcc_t r U_0(1+a)\Omega r(1-b)}{2sin(\phi)cos(\phi)} \\ &= \pi r^2\rho\sigma c_t \frac{U_0(1+a)\Omega r(1-b)}{sin(\phi)cos(\phi)} \end{aligned} \tag{2.24}$$

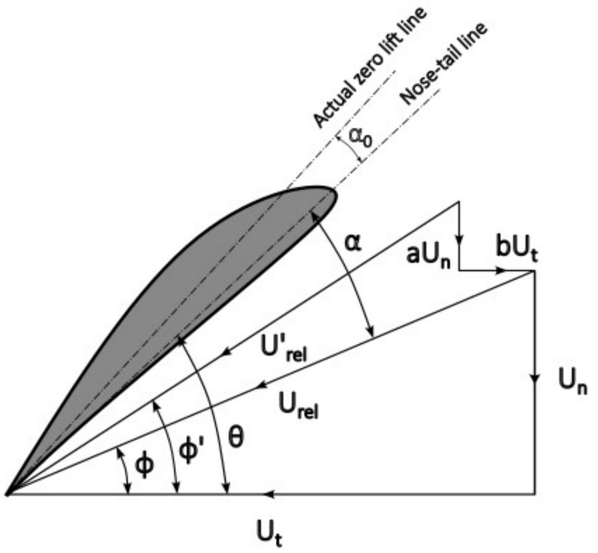


Figure 2.2: Velocity vectors for a blade element considering induction factors. Superscripted quantities refer to the case without velocity induction.

Therefore, the essence of the BET is to reduce the difficulty of modelling a complex 3-dimensional flow by assuming that it can be replaced by a linear summation of a large number of simpler 2-dimensional flows. The blade is divided into several small elements and the flow over each blade element is assumed to be independent of each other. The 3-dimensional effect of the vortex sheet leaving at the trailing edge of the blade is modelled by assuming that as the flow approaches the propeller disc plane, it is affected by an induced velocity in both axial and tangential directions which are expressed through the two coefficients a and b. The momentum theory is then applied to determine the magnitude of these two coefficients.

The momentum theory (MT) is based on applying the conservation laws of fluid mechanics under the assumption that the flow is steady, incompressible and axisymmetric. Furthermore, MT idealized the propeller as an actuator disk with an infinite number of blades (i.e. the propeller geometry is completely neglected).
The fluid flow pattern is modelled as a stream tube flow that can be divided into three control volumes: the upstream volume, which is somewhat larger than the disk diameter; the downstream volume, which is contracted due to the acceleration imposed by the propeller; and the disk volume, within which the actuator disk is contained. Note that the opposite holds in the case of a turbine: the downstream tube is larger than the upstream one.
In the following section, four planes are defined with the notation 0, w, 1, and 2. The quantities far up-stream and far down-stream are noted with the subscript 0 and w respectively. The quantities just ahead and behind the actuator disc are noted with the subscript 1 and 2 respectively. Across the actuator disc, the longitudinal velocity is continuous and no distinction will be made, thus $U_1 = U_2 = U_z$. The control volumes definition is explained in Figure 2.3.
In the upstream section, the fluid velocity is the speed U_0 of the boat. The flow is accelerated by the propeller action as it moves downstream, reaching a velocity of U_z as it passes through

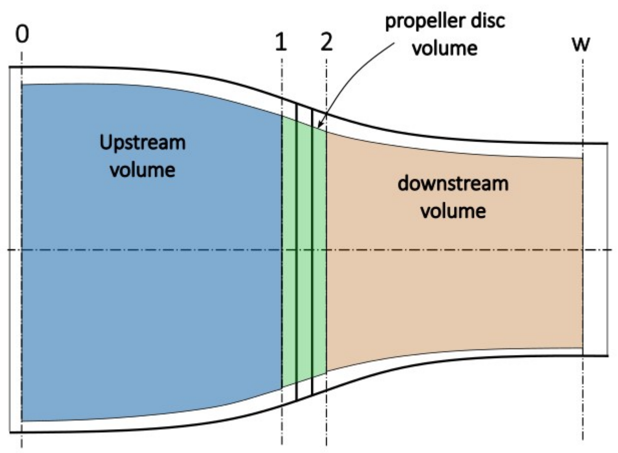


Figure 2.3: Control volume and sections definitions for Momentum Theory

the disc, and settling at a speed of U_w at a sufficiently far downstream location. While the flow velocity is regarded as changing gradually from U_0 to U_w, the static pressure is idealized as being discontinuous as the flow crosses the actuator disc. Just upstream of the disc, the fluid pressure is p_1 and it suddenly jumps to p_2 immediately downstream. The pressure and velocity trends are shown in Figure 2.4.

One of the hypotheses of the MT is the constant enthalpy of the fluid flow. Since enthalpy is constant, therefore Bernoulli's equation can be applied to the three control volumes. For the upstream and downstream regions, respectively, we have:

$$p_0 + \frac{1}{2}\rho U_0^2 = p_1 + \frac{1}{2}\rho U_1^2 \tag{2.25}$$

$$p_2 + \frac{1}{2}\rho U_2^2 = p_w + \frac{1}{2}\rho U_w^2 \tag{2.26}$$

Employing the equilibrium condition (i.e. upstream and downstream sections are sufficiently far away from the rotor disc) and continuity hypothesis, we have that $p_0 = p_w$ and $U_1 = U_2$. Combining the two Bernoulli's equations we get:

$$\Delta p = p_2 - p_1 = \frac{1}{2}\rho(U_w^2 - U_0^2) \tag{2.27}$$

Knowing the pressure difference at the rotor disc, the thrust can be easily determined:

$$T = A(p2 - p1) \tag{2.28}$$

Where $A = \pi R^2$ is the area of the disc.

To determine the relationship between U_z, the axial flow velocity at the disc section, U_w and U_0, we first apply the mass conservation across the three control volumes:

$$\dot{m} = \rho U_0 A_0 = \rho U_z A = \rho U_w A_w \tag{2.29}$$

We can apply the linear momentum conservation to get the thrust expression using the mass conservation just obtained:

$$T = \rho A_w U_w^2 - \rho A_0 U_0^2 = \dot{m}(U_w - U_0) = \rho A U_z(U_w - U_0) \tag{2.30}$$

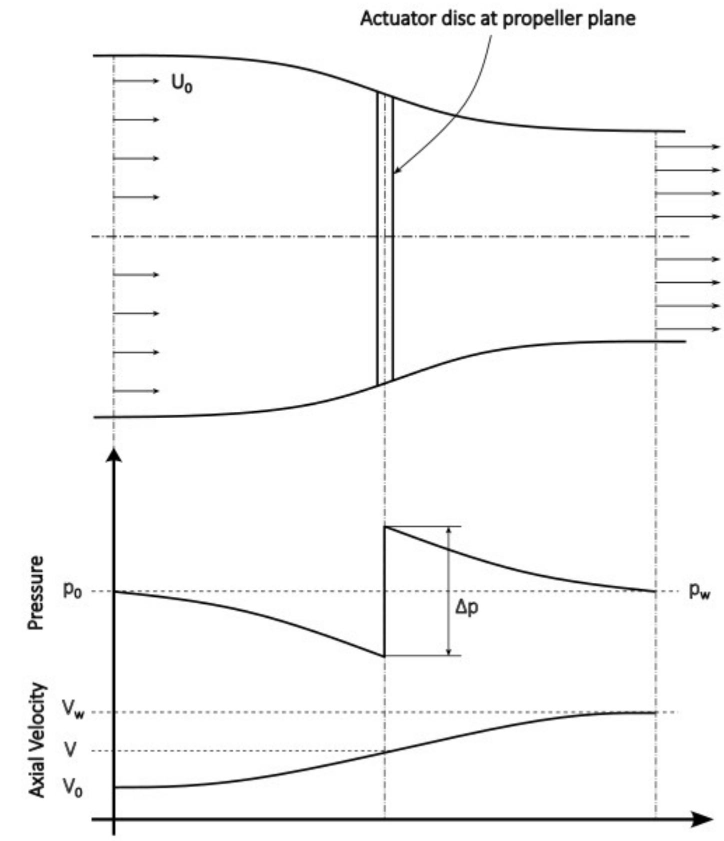


Figure 2.4: Slipstream pressure and velocity of Momentum Theory

To determine the velocity at the rotor disc, we can compare the power applied to the disc expressed through enthalpy conservation and calculated as the product of thrust and velocity, thus:

$$P = \rho(\frac{p_w}{\rho} + \frac{U_w^2}{2})A_w U_w - \rho(\frac{p_0}{\rho} + \frac{U_0^2}{2})A_0 U_0 = \frac{1}{2}\dot{m}(U_w^2 - U_0^2) \tag{2.31}$$

$$P = TU_z = \dot{m}(U_w - U_0)U_z \tag{2.32}$$

Equating the two expressions, we obtain:

$$\frac{1}{2}\dot{m}(U_w^2 - U_0^2) = \dot{m}(U_w - U_0)U_z \quad \Rightarrow \quad U_z = \frac{U_w + U_0}{2} \tag{2.33}$$

Therefore, MT states that axial flow velocity at the rotor disc is the arithmetic average of the far upstream and downstream velocities.
When the boat is cruising, i.e. is not stationary, we can define an axial inflow factor as we did for BET that allows us to express the induced velocity as a function of the reference velocity U_0. Hence, flow velocity can be defined as the sum of the reference velocity U_0 and an induced velocity component aU_0, thus:

$$U_z = U_0(1 + a) \quad ; \quad U_w = U_0(1 + 2a) \tag{2.34}$$

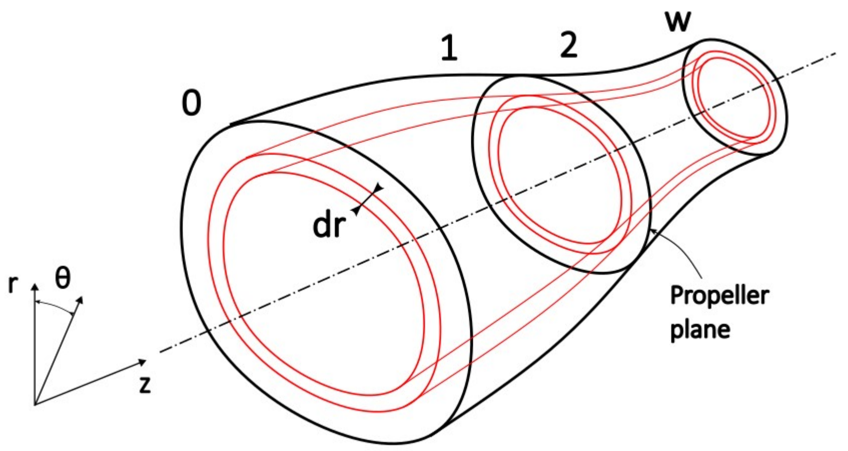


Figure 2.5: Streamtube of Momentum Theory with annuli definition.

Where the above result of U_z being the average of upstream and downstream velocities has been applied.
The MT, originally proposed by Rankine, was modified by Froude to take into account that the inflow factor is not constant across the whole surface of the actuator disc but it is a function of radial distance from the propeller's axis and is therefore determined by the geometry of the propeller.
The actuator disc can be considered as being made up of rings or annuli, the width of which is equal to Δr, and in the limit it can be made to be infinitesimal with a value of dr. Now, the MT is applied for an elementary stream tube or annulus whose cross section is $2\pi r dr$ at the radial station r. The previously obtained results also hold in this case, but now the inflow factor a varies along the propeller radius.

The infinitesimal thrust can be evaluated through Bernoulli's equation previously obtained, considering that the infinitesimal area is $dA = 2\pi r dr$:

$$\begin{aligned} dT(r) = \rho dA U_z (U_w - U_0) &= \rho dA U_0 (1+a)(U_0(1+2a) - U_0) \\ &= \rho U_0^2 \pi r [4a(1+a)] dr \end{aligned} \tag{2.35}$$

As has been done in the case BET, we assume that each annulus does not influence the others.
It is worth pointing out that since the cross sections of the stream tubes vary, a radial component of velocity is inherently present, which contradicts the one-dimensional assumption of the MT.
If we consider axial and radial velocity components, applying Bernoulli's equation to the control volumes we get:

$$p_0 + \frac{1}{2}\rho U_0^2 = p_1 + \frac{1}{2}\rho (U_z^2 + U_r(r)^2) \tag{2.36}$$

$$p_2 + \frac{1}{2}\rho (U_z^2 + U_r(r)^2) = p_w + \frac{1}{2}\rho U_w^2 \tag{2.37}$$

When expressing the pressure drop Δp, the radial velocity terms cancel out and we obtain the same result previously shown. This implies that as long as the velocity at the rotor is constantly radially (i.e. the axial velocity does not depend on the radial coordinate) then the

results previously obtained still apply. For further discussion on the radial velocity implications, the book by Branlard [16] is suggested. Here the radial velocity component is going to be neglected.
To analyze the momentum distribution, it must be considered that a trailing vortex is shed at the tip of each blade. Since the tip traces a helix trajectory as the propeller rotates and advances, the trailing vortex will be of the helical form itself. Without the presence of the trailing vortex, a small elemental streamline would simply move axially. Therefore, the slipstream behind a propeller is found to be rotating, in the same direction as the blades. This rotation is partly due to the circulation around the blades and the remainder is induced by the helical trailing vortices (Houghton and Carpenter [27]. These trailing vortices induce an additional speed upstream of the actuator disc in the tangential direction.
Consider a control volume which encloses the propeller disc. The upstream section is placed immediately ahead of the propeller, whereas the downstream section is placed immediately behind. Ahead of the propeller, at the upstream section, the angular flow velocity is zero. In the propeller plane, the flow angular velocity is $b\Omega$, where Ω is the angular blade velocity and b is the tangential induction factor. If we suppose that upstream and downstream sections are equidistant from the propeller disc and that flow angular velocity increases linearly along the control volume, then at the downstream section the angular velocity of the flow will be equal to $2b\Omega$. Thus the angular velocity of the flow behind the propeller is twice the angular velocity in the propeller plane. Namely:

$$U_{\theta_1} = 0 \quad ; \quad U_\theta = \frac{1}{2} U_{\theta_2} \quad ; \quad U_{\theta_2} = 2br\Omega \tag{2.38}$$

Where subscript θ indicates the tangential velocity, while subscripts 1 and 2 refer to sections just ahead and behind the actuator disc, as done before.
It should be noted that this result is similar to the axial velocity result of the axial momentum theory.
The tangential induction factor b is related to the momentum loss in the tangential direction, which is related to the drag force experienced by the blade. Hence, b is related to the torque required to rotate the blade.
For a further detailed discussion about the tangential speed jump and the induction factor b, see also Houghton and Carpenter [27] and Winarto [51].
As we have already done for the axial case, considering elementary rings of area $dA = 2\pi r dr$ we can express the momentum loss in the tangential direction through the mass flow rate multiplied by the angular induced velocity at the downstream section, thus:

$$\begin{aligned} dQ &= \dot{m} r U_{\theta_2} = \rho r U_z U_{\theta_2} dA = \rho r U_0 (1+a) 2\pi r \Omega b 2\pi r dr \\ &= 4\pi \rho r^3 U_0 \Omega b (1+a) dr \end{aligned} \tag{2.39}$$

Where the definition of axial velocity through the axial induction factor has been applied.
We can also calculate the elementary power as follows:

$$dP(r) = \Omega dQ = 4\pi \rho r^3 U_0 \Omega^2 b (1+a) dr \tag{2.40}$$

As we did for BET, we can calculate the local dimensionless coefficients of thrust, torque and power. Then:

$$C_t(r) = \frac{dT}{\frac{1}{2}\rho U_0^2 dA} = 4a(1+a) \tag{2.41}$$

$$C_q(r) = \frac{dQ}{\frac{1}{2}\rho U_0^2 r dA} = 4b(1+a)\lambda_r \tag{2.42}$$

$$C_p(r) = \frac{dP}{\frac{1}{2}\rho U_0^3 dA} = 4b(1-a)\lambda_r^2 \tag{2.43}$$

Which can be integrated along the radius to get the total dimensionless coefficients of the propeller:

$$C_T = \frac{8}{\lambda^2}\int_{\lambda_h}^{\lambda}(1-a)a\lambda_r d\lambda_r \tag{2.44}$$

$$C_Q = \frac{8}{\lambda^3}\int_{\lambda_h}^{\lambda}(1-a)b\lambda_r^3 d\lambda_r \tag{2.45}$$

$$C_P = \frac{8}{\lambda^2}\int_{\lambda_h}^{\lambda}(1-a)b\lambda_r^3 d\lambda_r \tag{2.46}$$

where λ_h is the tip-speed ratio at the hub, namely $\lambda_h = \Omega r_{hub}/U_0$.

So far we obtained thrust and torque expressions as functions of axial and tangential induction factors. To calculate the radial distribution of these velocity induction coefficients, a and b, the MT and the BET must be combined into the Blade Element Momentum Theory (BEMT), which is an iterative method that combines the results from the two basic theories and imposes that these have to provide the same thrust and torque results. The convergence loop principle is explained by figure 2.5.

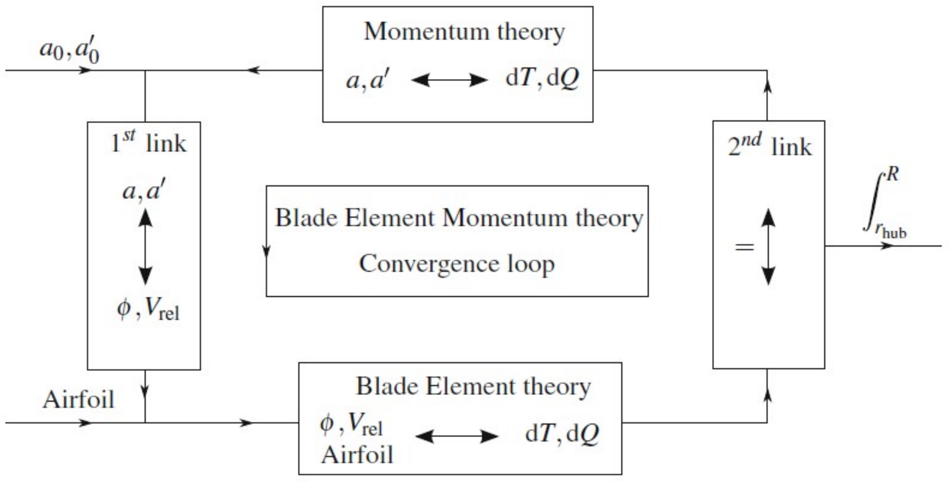


Figure 2.6: Graphic representation of the linking process between BEM and MT, provided by Branlard [16]. In this thesis, a' and V_{rel} have been substituted with b and U_{rel}, respectively.

Basically, it is supposed that the geometry of the airfoil sections along the span of the blade is known, in terms of pitch angle, chord length, rake and skew, which are functions of the radial station. Furthermore, it is also supposed that aerodynamic characteristics of the airfoil profile are known, in terms of lift and drag coefficient as a function of the angle of attack. This can be achieved through CFD simulations, wind tunnel tests or XFOIL utility, which is the method preferred for this work and is going to be explained in the next chapter.
Then, guessing initial values of a and b, it is possible to calculate the inflow angle and the relative velocity, which can be used to get elementary thrust and torque using BET equations. These must be compared with thrust and torque provided by MT equations and the induction factor values must be iteratively adjusted until we get the same results.
We recall now velocity and inflow angle definitions previously seen in BET:

$$U_n = U_0(1+a) \quad ; \quad U_t = \Omega r(1-b) \tag{2.47}$$

$$U_{rel}^2 = U_n^2 + U_t^2 = \frac{U_n U_t}{sin(\phi)cos(\phi)} \tag{2.48}$$

$$tan(\phi) = \frac{U_n}{U_t} = \frac{1+a}{(1-b)\lambda_r} \tag{2.49}$$

To couple the two basic theories, the different approaches each theory is based on must be considered: the BET considers naturally a finite number of blades, whereas the MT assumes a uniform load distribution over the propeller disc with an infinite number of blades. Hence, MT equations must be corrected to take into account a finite number of blades.
The tip-loss factor F, was first developed by Prandtl [42]. Then, Glauert [24] suggested multiplying the momentum equations to account for the flow differences that occur depending on the number of blades. More studies about this effect were conducted by Goldstein [25] and by Tachmindji [49]. Nowadays, several correction factors have been proposed. A more detailed discussion is given by Branlard [16]. Here the formula proposed by Glauert, which is one of the most widely used, is going to be adopted:

$$F = \frac{2}{\pi}cos^{-1}(e^{-f}) \quad where \quad f = \frac{Z(R-r)}{2r sin(\phi)} \tag{2.50}$$

When the number of blades, Z, tends to infinity, F tends to 1.
The MT equations for thrust and torque with tip-loss correction become:

$$dT_{MT} = \frac{1}{2}\rho U_0^2 dA[4aF(1-a)] \tag{2.51}$$

$$dQ_{MT} = \frac{1}{2}\rho U_0^2 r dA[4bF(1-a)\lambda_r] \tag{2.52}$$

While the BET equations are:

$$dT_{BET} = \frac{1}{2}\rho U_{rel}^2 Z c dr c_n \tag{2.53}$$

$$dQ_{BET} = \frac{1}{2}\rho U_{rel}^2 Z c dr r c_t \tag{2.54}$$

Remind that MT loads are defined over an annular cross-section of area $dA = 2\pi r dr$, while BET loads are defined for Z blade elements of area $dS = cdr$.
Equating the expressions for thrust and torque, we get:

$$\frac{U_{rel}^2}{U_0^2}\sigma c_n = 4aF(1+a) \tag{2.55}$$

$$\frac{U_{rel}^2}{U_0^2}\sigma c_t = 4bF(1+a)\lambda_r \tag{2.56}$$

From geometrical considerations on velocity vectors, it can be noted that:

$$U_{rel}^2 sin^2(\phi) = U_0^2(1+a)^2 \tag{2.57}$$

$$U_{rel}^2 sin(\phi)cos(\phi) = U_0(1+a)\Omega r(1-b) \tag{2.58}$$

Using these expressions, the BEMT final equations can be derived:

$$\frac{a}{1+a} = \frac{cZc_n}{4Fsin^2(\phi)2\pi r} = \frac{\sigma c_n}{4Fsin^2(\phi)} \tag{2.59}$$

$$\frac{b}{1-b} = \frac{cZc_t}{4Fsin(\phi)cos(\phi)2\pi r} = \frac{\sigma c_t}{4Fsin(\phi)cos(\phi)} \tag{2.60}$$

The non-linear system of the two equations above must be numerically solved. The dependencies between the variables are highlighted below:

$$iteration \quad n+1 = f(iteration \quad n) \tag{2.61}$$

$$\phi = \phi(a,b) \tag{2.62}$$

$$\alpha = \alpha(\phi) \tag{2.63}$$

$$c_n = c_n(\alpha,\phi) \tag{2.64}$$

$$c_t = c_t(\alpha,\phi) \tag{2.65}$$

$$F = F(\phi) \tag{2.66}$$

$$a = a(F,\phi,c_n) \tag{2.67}$$

$$b = b(F,\phi,c_n) \tag{2.68}$$

All the variables above are also dependent on the radial position.
The BEMT algorithm is by nature an implicit problem, due to the interdependencies between the parameters, and a non-linear problem mainly due to the non-linearity introduced by the profile polars, $C_l(\alpha)$ and $C_d(\alpha)$.
The problem can be solved using non-linear solution techniques. The solution is not always guaranteed and a unique solution is ensured only with some assumptions on the solution domain. The mathematical problem related to the BEMT problem has been analyzed in depth by Ledoux [32].
Several algorithms have been proposed to solve the BEMT system, for instance, Jin [30] and Ning [37]. However, several numerical methods can be adopted to solve this kind of problem,

e.g. the Newton Method or the Method of Secant (for an extensive numerical methods description see Chapra [10]). In this work, for practical purposes, a simple iterative process where an iteration loop is used until convergence is reached has been used.
To solve the BEMT system, all geometric parameters of the propeller must be known a priori. Once the airfoil profile, pitch and chord distribution, diameter, number of blades and so on have been chosen, the numerical algorithm can be performed to calculate the induction factors a and b. Once induction factors are known, it is possible to calculate local values of thrust and torque, and then compute the overall thrust and torque provided by the propeller through integration along the radius. Since the hypothesis of independence of each strip (i.e. radial positions) holds, the convergence loop and the loop over the different stripes may be indifferently swapped.
As we are going to see in detail in the next chapter, the BEMT algorithm can be integrated into an optimization loop to find the best parameters solution.
A generic description of the algorithm that can be implemented to numerically solve the BEMT equations is the following:

```
Start
Input: Diameter, N° Blades, Propeller RPM, $\rho$, $V_a$, N° Blade Sections, Pitch Distribution, Chord Distribution, Airfoil Profile.
Load: Polars $C_l$ and $C_d$.
Compute: $\sigma, \lambda$.
Set: Convergence Tolerance, N° Maximum Iterations, $K_p$,$K_d$, $a$ and $b$ Guess.
For: Every Blade Section:
While: $IterationCounter <$ N° Maximum Iterations AND $Residuals_{1/2} >$ Tolerance:
(1) Compute $U_n, U_t, \phi, \alpha$.
(2) Compute $C_l, C_d$ through interpolation.
(3) Compute $f$ e $F$.
(4) Compute $Residual_1 = \frac{\sigma c_n}{4Fsin^2(\phi)} - \frac{a}{1+a}$ e $Residual_2 = \frac{\sigma c_t}{4Fsin(\phi)cos(\phi)} - \frac{b}{1-b}$.
(5) Compute new values of $a$ e $b$.
(6) If: Residuals $<$ Tolerance Break
End While
Compute: Local values of $dT$ and $dQ$.
End For
Compute: Global values of $T$ and $Q$.
End
```

2.4 Summary of BEMT equations

For clarity reasons, the hypothesis and the main results of the BEMT discussed so far are reported here.
The hypotheses assumed are:

- Propeller geometry in terms of diameter, hub diameter, number of blades, airfoil profile section, chord distribution, and pitch distribution is known;

- Operating conditions in terms of angular velocity and free-stream velocity are known;
- The flow is stationary, incompressible, inviscid, homogeneous and axisymmetrical;
- Each radial station is independent of the others and the radial velocity component is neglected.

The final equations are:

$$\frac{a}{1+a} = \frac{\sigma c_n}{4Fsin^2(\phi)} \quad ; \quad \frac{b}{1-b} = \frac{\sigma c_t}{4Fsin(\phi)cos(\phi)} \tag{2.69}$$

$$tan(\Phi) = \frac{1+a}{(1-b)\lambda_r} \quad ; \quad \alpha = \Phi - \theta \tag{2.70}$$

$$c_n = C_l cos(\phi) - C_d sin(\phi) \quad ; \quad c_t = C_l sin(\phi) + C_d cos(\phi) \tag{2.71}$$

$$F = \frac{2}{\pi} cos^{-1}(e^{-f}) \quad ; \quad f = \frac{Z(R-r)}{2rsin(\phi)} \tag{2.72}$$

$$\lambda_r = \frac{\Omega r}{U_0} \quad ; \quad \sigma = \frac{Zc}{2\pi r} \tag{2.73}$$

$$U_n = U_0(1+a) \quad ; \quad U_t = \Omega r(1-b) \quad ; \quad U_{rel}^2 = U_n^2 + U_t^2 \tag{2.74}$$

Chapter 3

BEMT Implementation in MATLAB

3.1 Introduction

This chapter presents the MATLAB code that has been developed to design a marine propeller through the Blade Element Momentum Theory. First, the key ideas and basic structure of the program will be discussed. Secondly, the software implementation will be briefly shown. The whole code is available in the Appendix A.

3.2 MATLAB Code Structure

The basic idea of the MATLAB program is to develop a simple, reliable and fast code that can be used to design new propellers when working conditions change. These propellers should guarantee both big thrust and efficiency at the optimal design point.
The flowchart of the code developed is shown in Figure 3.1.
The first steps take the input data, such as propeller geometric characteristics, working point conditions, pitch-diameter ratio, blade profile geometry, airfoil polar data, etc... As we already said, airfoil polars must be known a priori to start the BEMT computation. To obtain polars extrapolation to 360° angle of attack, which is necessary for the BEMT algorithm, the combination of two open-source software has been used in this project, namely *XFOIL* and *QBlade*. Their usage will be comprehensively explained in the next section.
In the initialization step, the variables used in the following calculations are initialized. Furthermore, the pitch distribution is defined considering the target P/D ratio and an ideal pitch distribution. The choice to impose an ideal distribution will be discussed in the results section.
Then, the optimization loop can start. Firstly, through a gradient algorithm, the optimal chord distribution, initially initialized to a constant distribution, is calculated to maximize thrust and efficiency. At each iteration, the BEMT non-linear system of equations is numerically solved. After achieving convergence, the mechanical validation loop can start and chord distribution is increased to guarantee a sufficient strength of each blade section. The chord distribution is finally interpolated with a parabolic function to avoid any discontinuity.
Since the propeller design is over, the program starts the analysis phase to compute the overall propeller characteristic curves varying the advance ratio. The final results are then plotted and compared with a set of data obtained through CFD simulations of the new propeller and the old

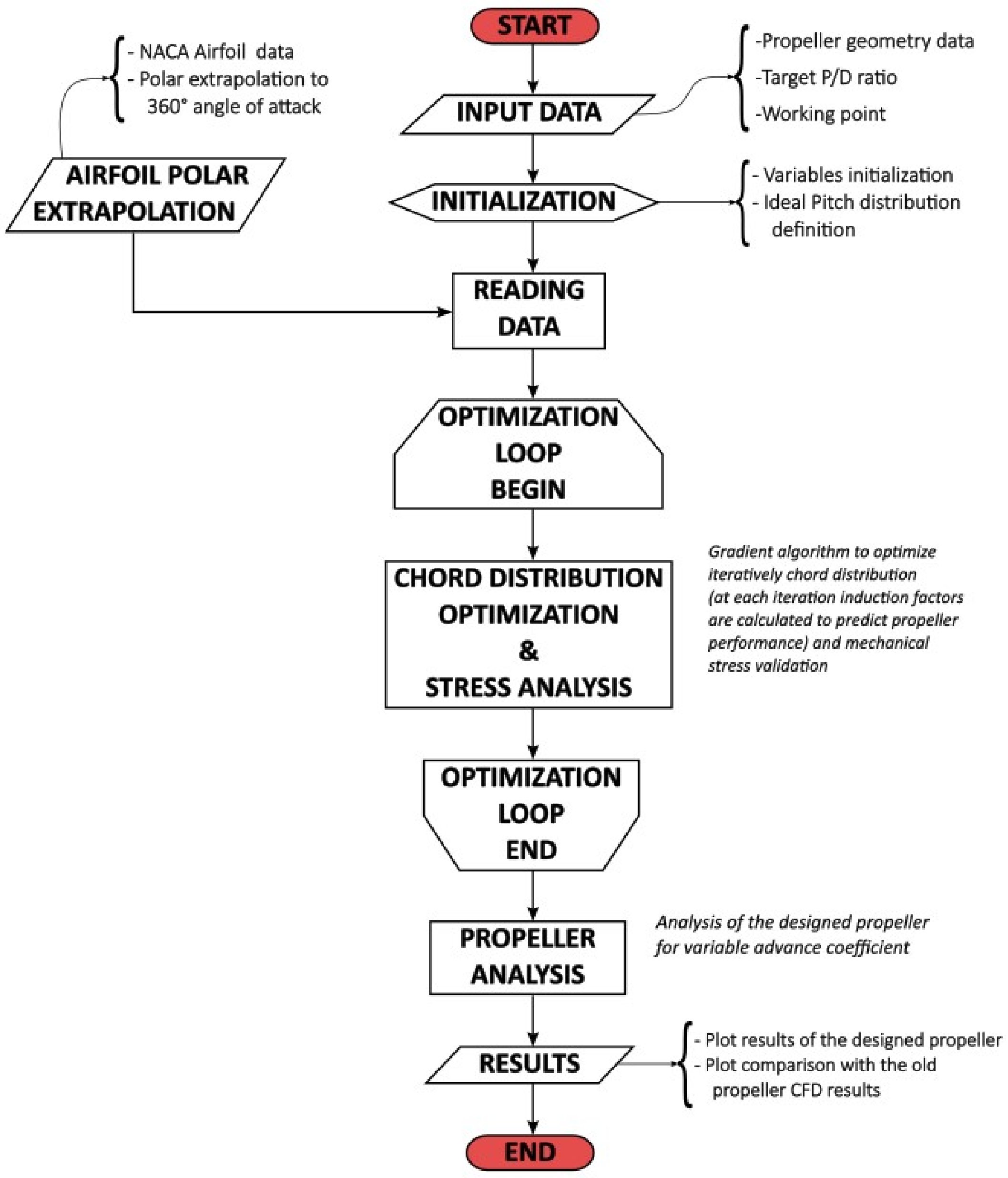

START
INPUT DATA
-Propeller geometry data
-Target P/D ratio
-Working point
- NACA Airfoil data
- Polar extrapolation to 360° angle of attack
AIRFOIL POLAR EXTRAPOLATION
INITIALIZATION
- Variables initialization
- Ideal Pitch distribution definition
READING DATA
OPTIMIZATION LOOP BEGIN
CHORD DISTRIBUTION OPTIMIZATION & STRESS ANALYSIS
Gradient algorithm to optimize iteratively chord distribution (at each iteration induction factors are calculated to predict propeller performance) and mechanical stress validation
OPTIMIZATION LOOP END
PROPELLER ANALYSIS
Analysis of the designed propeller for variable advance coefficient
RESULTS
- Plot results of the designed propeller
- Plot comparison with the old propeller CFD results
END

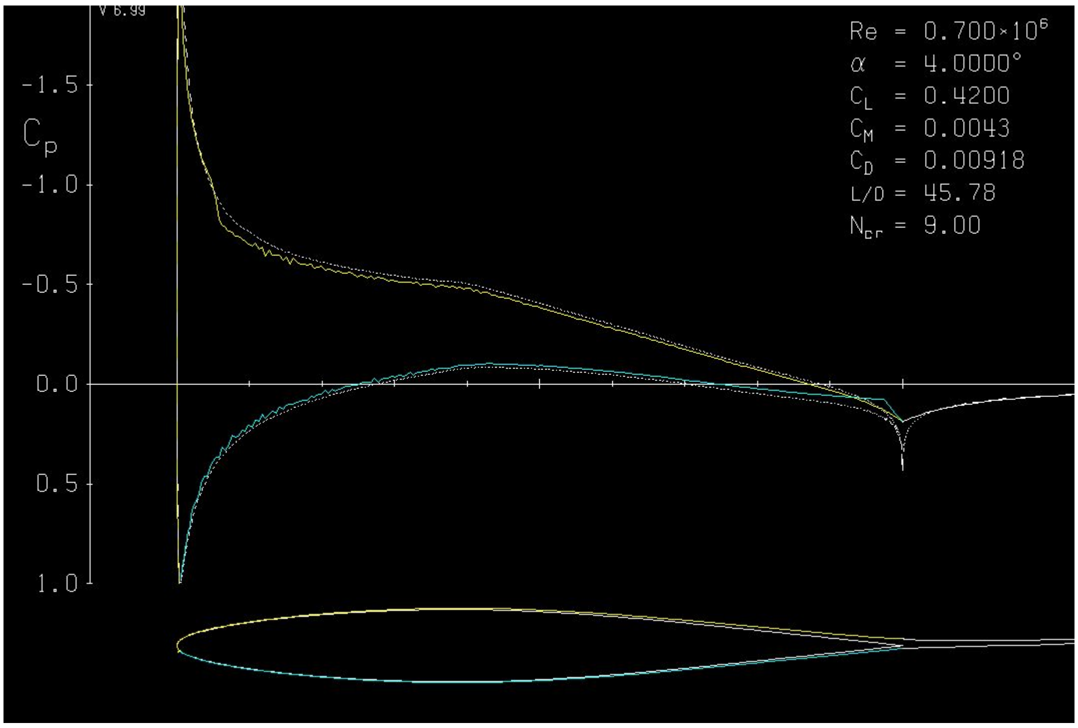
Figure 3.1: MATLAB code Flowchart

one.
Each program step is further explained in the following sections.

3.3 *XFOIL* and Airfoil Polar Extrapolation

XFOIL is an open-source program, written in Fortran, for the design and analysis of subsonic airfoils. Developed by Professor Mark Drela at MIT [13], *XFOIL* has become a valuable tool in the field of aerodynamics. *XFOIL*'s openness and reliability have contributed to its widespread

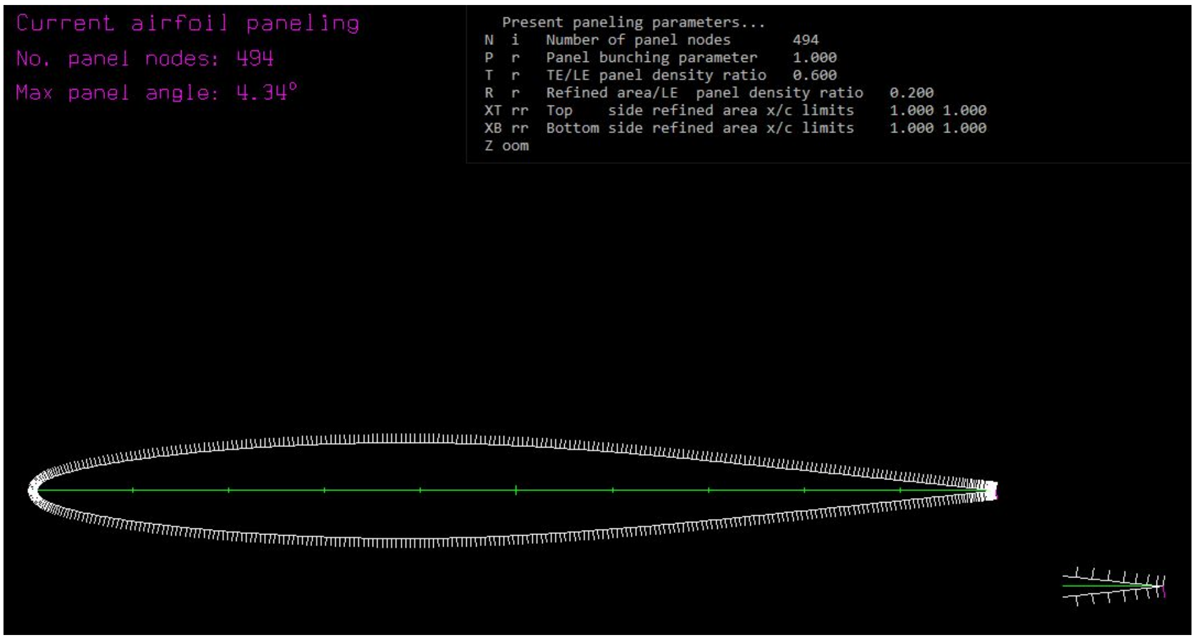
use in the aerodynamics community.
The primary function of *XFOIL* is to calculate the pressure distribution in either a viscid or inviscid scenario on a 2D airfoil given the coordinates specifying the shape of the airfoil, as well as the Reynolds and Mach numbers. To calculate this pressure distribution, *XFOIL* implements a panel-method approach. The airfoil surface is divided into small panels, each represented by a source or a vortex. The strength of these sources and vortices is determined such that the body surface is a streamline of the flow. Finally, from the pressure distribution, *XFOIL* can derive the lift and drag characteristics of the airfoil.
In addition to its analysis capabilities, *XFOIL* also allows for inverse design, meaning that it can vary an airfoil shape to achieve desired parameters.
Figure 3.2 shows an analysis example of the pressure coefficient distribution along the x coordinate of the airfoil for certain values of the Reynolds number and the angle of attack.



$Re = 0.700 \times 10^6$
$\alpha = 4.0000°$
$C_L = 0.4200$
$C_M = 0.0043$
$C_D = 0.00918$
$L/D = 45.78$
$N_{cr} = 9.00$

Figure 3.2: *XFOIL*: airfoil analysis example. The plots show the C_P distribution on the upper and lower surfaces, and the flow separation coordinate as well.

In this project, *XFOIL* has been used to determine the lift and drag characteristics of a given airfoil. The airfoil chosen is the *NACA 64A010* which is a symmetric, thin airfoil. It has been chosen due to its small thickness, allowing the propeller to reach higher efficiency. However, it must be considered that thin airfoils may impact negatively the cavitation characteristic of the propeller.
The airfoil shape and the analysis panel settings are shown in Figure 3.3. To achieve convergent simulations, the number of panels and the number of iterations have been increased as regards the default values, which might not allow to reach convergent results near the stall of the airfoil.



```
Current airfoil paneling
No. panel nodes: 494
Max panel angle: 4.34°
```

```
  Present paneling parameters...
N  i   Number of panel nodes       494
P  r   Panel bunching parameter    1.000
T  r   TE/LE panel density ratio   0.600
R  r   Refined area/LE  panel density ratio   0.200
XT rr  Top    side refined area x/c limits    1.000 1.000
XB rr  Bottom side refined area x/c limits    1.000 1.000
Z oom
```

Figure 3.3: *XFOIL*: airfoil shape and panel settings used in this project

The analysis has been performed in viscous mode to get more realistic results.
The final simulation results with the plot of the polars provided by *XFOIL* are shown in Figure 3.4. The plots include C_L/C_D characteristics, C_L and C_M as functions of α, and transition plots of the upper and lower surfaces for a given C_L.

Since the airfoil is symmetric, the C_L is correctly computed to 0 when α is 0. Furthermore, we can see that the maximum C_L is around 1.16 and the stall of the airfoil begins around $13°$.
There is no point in trying to simulate airfoil characteristics for larger angles of attack because *XFOIL* is not capable of reaching convergence far away from the linear region. Sometimes, even simulations in the stall region are not convergent. However, in performing the BEMT algorithm is essential to provide airfoil polars over the whole range of angles of attack to avoid numerical instability and because we can't know a priori the working conditions (i.e. the angle of attack of a particular blade section) of the propeller. For these reasons, extrapolation procedures are available in the literature and widely used by propeller and turbine designers to estimate C_L and C_D starting from the linear-stall characteristics.
Most used extrapolation techniques are the Viterna [50] and Montgomerie methods [4]. Both methods estimate propeller lift and drag coefficients starting from the linear-stall airfoil behaviour and from geometric considerations. Mahmuddin [34] has presented an assessment of the two methods with experimental data. Moreover, a more recent technique has been proposed by Battisti [33] for further improved extrapolation reliability in the post-stall region.
The Viterna method has been used due to its simple implementation. In particular, it has been used another software to perform polars extrapolation, namely *QBlade*.
QBlade is an advanced multi-physics software designed to facilitate the comprehensive aero-servo-hydro-elastic development, prototyping, simulation, and certification of wind turbines. It is an open-source, cross-platform simulation software for wind turbine blade design and aero-

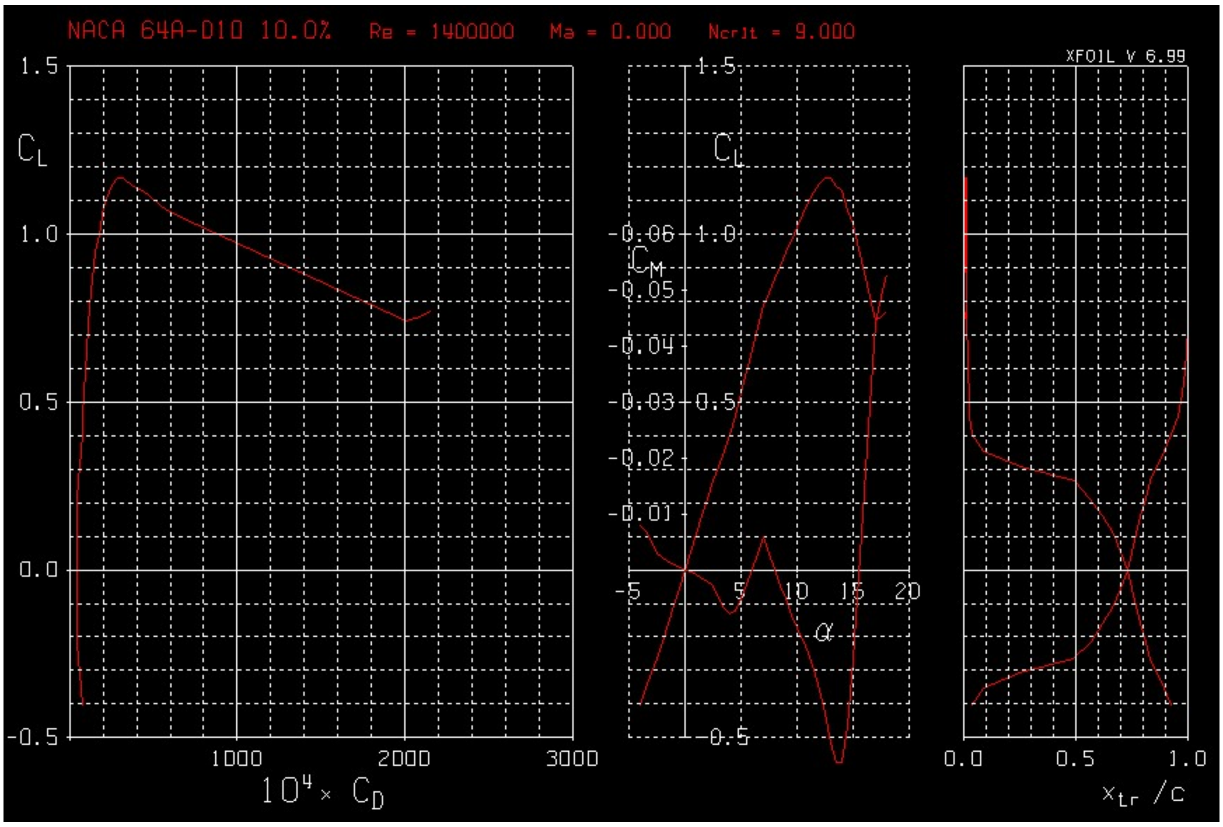
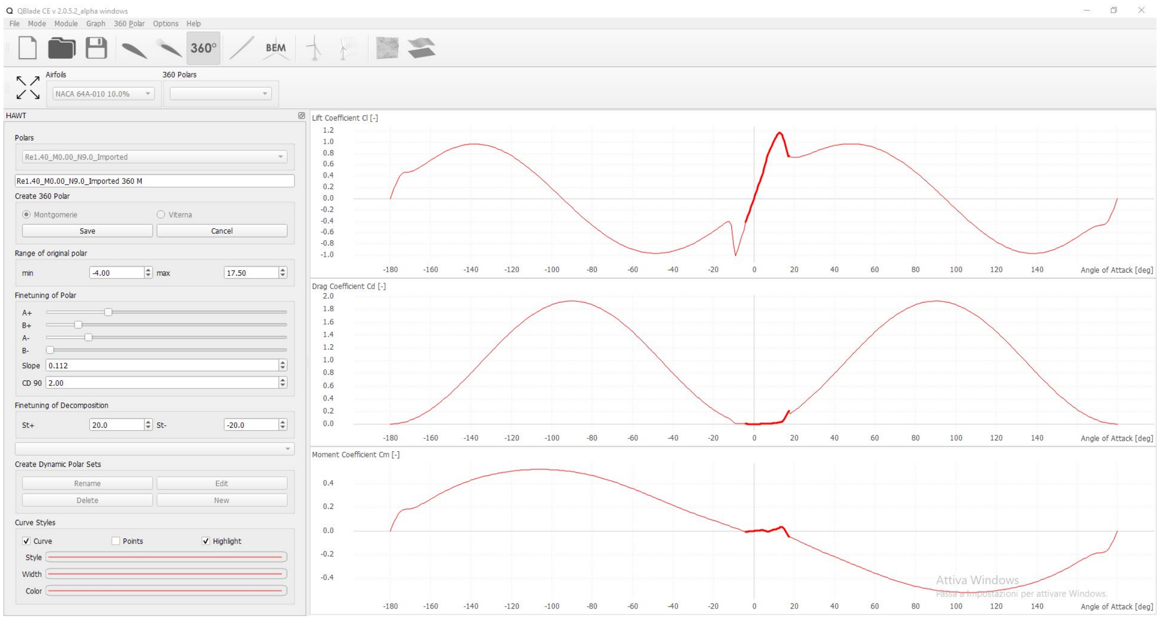


Figure 3.4: *XFOIL*: *NACA64A010* final polars plot

dynamic simulation. The software is integrated into *XFOIL* to allow for the rapid design of custom airfoils and computation of their performance curves, extrapolating the performance data to a range of 360° Angle of Attack, and directly integrating them into a wind turbine rotor simulation. Indeed, *QBlade* can be utilized for various purposes, including Wind Turbine Design (HAWT, VAWT, and Multi-Rotor), Floater Design and Numerical Testing, Controller Testing and Optimization, Wind Farm Simulations and Wake Interaction Studies. One of the significant advantages of *QBlade* is its user-friendly graphical user interface.
Figure 3.5 shows the graphical interface of the extrapolation module which allows for fine-tuning of the extrapolation parameters.
The software returns an Excel file with C_L and C_D defined for the whole range of angles of attack that can be exported to a text format and read by the *Polar Reader* MATLAB function.
At this point, we can proceed with the input data required to perform propeller design and the initialization phase.

3.4 Input Data and Initialization Step

Since now we have the polars data of the selected airfoil, we can now analyze the geometric and working inputs needed for propeller design. Figure 4.1 shows the *MATLAB* section with the input definition.



Airfoils: NACA 64A-010 10.0%

Polars: Re1.40_M0.00_N9.0_Imported

Re1.40_M0.00_N9.0_Imported 360 M

Create 360 Polar: [✓ Montgomerie] Viterna

Save Cancel

Range of original polar: min -4.00 max 17.50

Finetuning of Polar: A+ B+ A- B-

Slope 0.112

CD 90 2.00

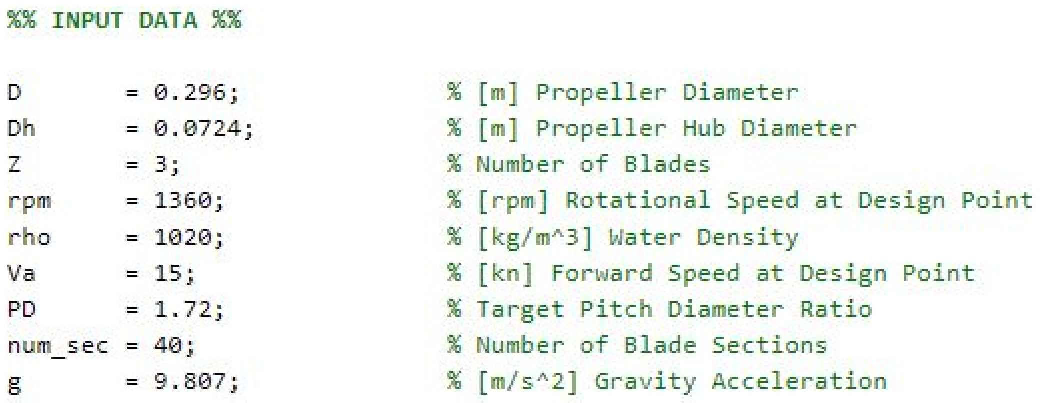
Finetuning of Decomposition: St+ 20.0 St- -20.0

Create Dynamic Polar Sets: Rename Edit Delete New

Curve Styles: [✓ Curve] Points [✓ Highlight]

Style Width Color

Figure 3.5: *QBlade* polars extrapolation module



```
%% INPUT DATA %%

D       = 0.296;            % [m] Propeller Diameter
Dh      = 0.0724;           % [m] Propeller Hub Diameter
Z       = 3;                % Number of Blades
rpm     = 1360;             % [rpm] Rotational Speed at Design Point
rho     = 1020;             % [kg/m^3] Water Density
Va      = 15;               % [kn] Forward Speed at Design Point
PD      = 1.72;             % Target Pitch Diameter Ratio
num_sec = 40;               % Number of Blade Sections
g       = 9.807;            % [m/s^2] Gravity Acceleration
```

Figure 3.6: *MATLAB* code inputs data

Regarding the working conditions, we need to specify the rotational velocity of the propeller at the ideal working point, remembering that our simulation will be just a steady-state computation. Then, we have to define the boat's velocity, i.e. the uniform inflow velocity. Since in our case the boat is a catamaran with the engines sufficiently far away from the hulls, it has been chosen to not consider any wake fraction coefficient, that is the hulls don't affect the uniform inflow velocity at the propellers.
Regarding the geometry, we need to define the propeller diameter, the hub diameter, the number of blades and the P/D ratio. Since we choose to impose an *ideal pitch* distribution, from the P/D ratio we can define the pitch at each section in the following way:

$$\theta_{ref} = atan(\frac{p}{2\pi R}) \tag{3.1}$$

$$k = \theta_{ref} R \tag{3.2}$$

Where k is a constant and θ_{ref} is the pitch at the radius R. Thus, we choose to take the P/D ratio as referred to the radius R. Finally, we have:

$$\theta(r) = \frac{k}{r} \tag{3.3}$$

That defines the pitch value at each blade section during the initialization step. As previously mentioned, the choice of using an *ideal pitch* distribution is explained in the results section.
The last input parameter of interest is the *number of sections* that the blade is divided into. In the previous figure, this parameter is set to the value of 40 because is the best threshold between computational effort and good results convergence. The convergence study on the *number of sections* is shown in the results section.
Lastly, in the initialization phase, the chord distribution is initialized to a constant value that is going to be optimized in the following steps.
Now that the simulation is set and initialized we can proceed with the performance calculation and chord optimization.

3.5 Chord Distribution Optimization and Stress Analysis

In the optimization loop, the chord distribution is modified to improve the propeller's thrust and efficiency. Secondly, the mechanical assessment is performed to guarantee a sufficient thickness of the blade. At each iteration, the function *induction factors calculation*, which implements and solves the *BEMT* equations, is called to allow the calculation of current thrust and efficiency values. Then, a gradient descent algorithm is implemented to optimize the chord distribution.
To implement a gradient descent algorithm, we need to define a *learning rate* and a *cost function* of which we will calculate the gradient at each iteration. To improve algorithm stability, a first-order central difference scheme has been adopted to calculate the gradient. With J representing the cost function, the gradient is:

$$J'_i(r,c) = \frac{J_{i+}(r,c) - J_{i-}(r,c)}{2h} \tag{3.4}$$

where the subscripts i, $+$ and $-$ refer to the $i-th$ iteration, the cost function forward step and the backward step, respectively. While the h parameter is the constant *chord step*, applied to calculate the forward and backward steps, and c refers to dependency by the chord distribution.
The cost function is defined in the following way:

$$J_i(r,c) = \frac{dT(r,c)}{dT_0(r,c)} + \frac{\eta(r,c)}{\eta_0(r,c)} \tag{3.5}$$

where $dT(r)$ and $\eta(r)$ are the infinitesimal thrust and efficiency distribution along the radius respectively, while $dT_0(r)$ and $\eta_0(r)$ are the thrust and efficiency distribution with the current chord distribution. Hence, the thrust and efficiency are normalized with the initial values.
Basically, at each iteration thrust and efficiency distributions are calculated with the current chord distribution. Then, at each blade section, the chord is incremented by h and the new thrust and efficiency distributions are computed. Thus, the forward cost function is calculated

as well. The procedure is repeated applying a decreasing chord step to compute the backward cost function. Finally, the gradient can be computed with (3.1). The gradient descent is then applied through:

$$c_i(r) = c_{i-1}(r) + \alpha J_i'(r, c) \tag{3.6}$$

where c is the chord distribution and α the learning rate. Hence, in our case, the algorithm is more correctly a *gradient ascent* since the main objective is to maximize the cost function.
The learning rate is defined as:

$$\alpha = k_1 + k_2 e^{-k_3 i} \tag{3.7}$$

where k_1, k_2 and k_3 are constants tuned to guarantee as fast and stable as possible algorithm, and i is the current iteration number. Thus, the learning rate decreases exponentially and reaches an almost constant value of k_1 after some iterations.
Once the new chord distribution is computed, the procedure is repeated until the chord difference between two consecutive iterations of each blade section is below a tolerance value. When this is true, the chord optimization is convergent and we can proceed with the stress analysis of the blade.

The mechanical assessment of the blade is performed just after the chord optimization loop. We need to guarantee a suitable thickness (i.e. strength) of the blade that undergoes centrifugal force and bending moment, generated by the hydrodynamic action, during work conditions. Since the dimensionless profile of the blade is fixed, to increment blade thickness we need to vary the chord distribution until the mechanical strength target is achieved.
The methodology implemented here is described by Carlton [9]. The stress at each blade section is generated by the following components: the thrust action, the torque action, the direct centrifugal force and the centrifugal bending (if the blade is swept). At each blade section, the stress is given by:

$$\sigma = \frac{M}{Z} + \frac{F_C}{A} \tag{3.8}$$

where σ is the tensile stress on the section, M is the total bending moment given by the combined effect of hydrodynamic and centrifugal actions, Z is the section tensile modulus, F_C is the centrifugal force and A is the section area.
In our case, the bending moment is determined only by the hydrodynamic action, and thus is:

$$M_H = F_T a cos(\theta) + F_Q b sin(\theta) \tag{3.9}$$

where F_T and F_Q are the integrated means of thrust and torque respectively, θ is the local pitch, and a and b are:

$$a = \int_{r_0}^{R} \frac{F_T' r dr}{F_T' dr} \tag{3.10}$$

$$b = \int_{r_0}^{R} \frac{F_Q' r dr}{F_Q' dr} \tag{3.11}$$

The centrifugal force is:

$$F_C = \rho_m x_c \omega^2 (\int_{x_0}^{1} A dx) \tag{3.12}$$

where:

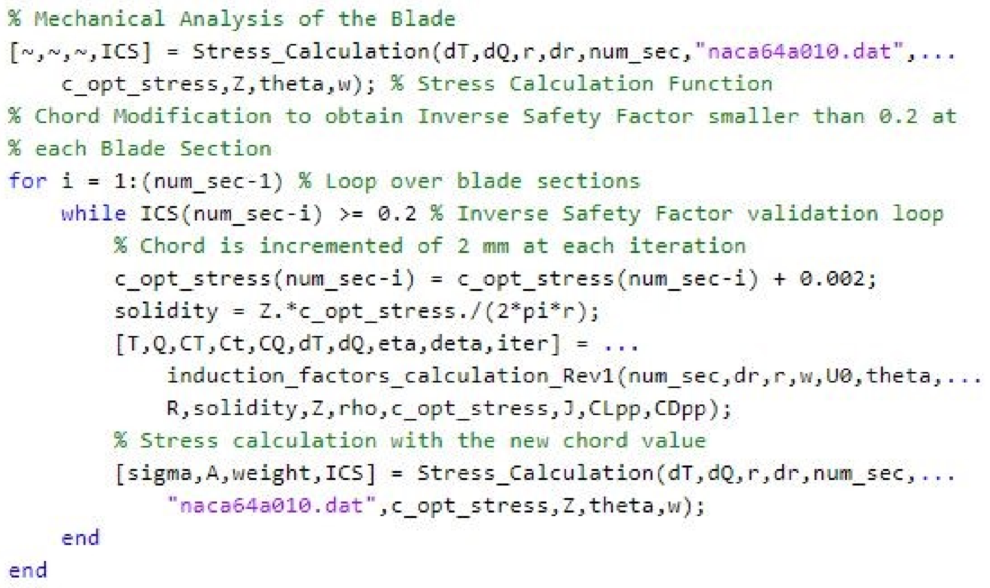
$$x_c = \frac{\int_x^1 Axdx}{\int_x^1 Adx} \tag{3.13}$$

$$A = \int_0^C tdc \tag{3.14}$$

with t and c the local thickness and chord of the blade section respectively, A the local area, x the non-dimensional radius, x_c the position of the blade centroid, rho_m the material density and ω the local angular velocity. Finally, the section tensile modulus is:

$$Z = \frac{2\int_0^C [3y_p(y_p + t) + t^2]tdc \cdot \int_0^C tdc}{3\int_0^C (2y_p + t)tdc} - \frac{1}{2}\int_0^C (2y_p + t)tdc \tag{3.15}$$

where y_p is the pressure face coordinate.
With these formulas, we can estimate the stress at each blade section. Starting from the tip blade, the stress analysis is performed and if the *inverse safety factor* (ICS) is bigger than 0.2, the local chord is iteratively incremented by 2 mm. Then, propeller performance and thus mechanical stress are calculated again until the ICS is satisfied. When the local ICS value is guaranteed, we proceed with the following blade section until a sufficient mechanical strength of the last section, at the root of the blade, is achieved. The reason we chose 0.2 as the ICS target is explained in the results section.
The following Figure shows the piece of code where the mechanical assessment loop is implemented.



```
% Mechanical Analysis of the Blade
[~,~,~,ICS] = Stress_Calculation(dT,dQ,r,dr,num_sec,"naca64a010.dat",...
    c_opt_stress,Z,theta,w); % Stress Calculation Function
% Chord Modification to obtain Inverse Safety Factor smaller than 0.2 at
% each Blade Section
for i = 1:(num_sec-1) % Loop over blade sections
    while ICS(num_sec-i) >= 0.2 % Inverse Safety Factor validation loop
        % Chord is incremented of 2 mm at each iteration
        c_opt_stress(num_sec-i) = c_opt_stress(num_sec-i) + 0.002;
        solidity = Z.*c_opt_stress./(2*pi*r);
        [T,Q,CT,Ct,CQ,dT,dQ,eta,deta,iter] = ...
            induction_factors_calculation_Rev1(num_sec,dr,r,w,U0,theta,...
            R,solidity,Z,rho,c_opt_stress,J,CLpp,CDpp);
        % Stress calculation with the new chord value
        [sigma,A,weight,ICS] = Stress_Calculation(dT,dQ,r,dr,num_sec,...
            "naca64a010.dat",c_opt_stress,Z,theta,w);
    end
end
```

Figure 3.7: *MATLAB* mechanical stress analysis loop

Once the blade is completely assessed, the chord distribution is interpolated with a parabolic

function to avoid discontinuities. With the final distribution, the final performance and mechanical stress are calculated.
Since the propeller design is completed, we can now analyze the propeller characteristics for variable advance ratios and examine the overall results.

3.6 Propeller Analysis and MATLAB Results

The final step is straightforward since the code implements a variable advance velocity and runs the function *induction factors calculation* to evaluate thrust, torque and efficiency at each velocity (with constant angular velocity). After the analysis, the design results can be examined.

Before proceeding with the design results, it is worth testing the convergence behaviour of the *BEMT* algorithm and so of the function *induction factors calculation*. As we saw in the previous section, the code takes as input the number of sections into which the blade is divided. This input has a strong influence on the final result because it allows us to better represent the pitch and chord distributions along the radius, and to decrease the error due to the hypothesis that each section is independent from the others.
For these reasons, the *MATLAB* function was run iteratively with an increasing number of sections with fixed chord and pitch distributions. The thrust was chosen to analyse the convergence behaviour of the results and Figure 3.8 shows the analysis.
As we can see, there is no point in adopting hundreds of sections because the algorithm reaches a good results convergence with just a few dozen sections. In this case, 40 sections were chosen as a good threshold between convergence and computational time.

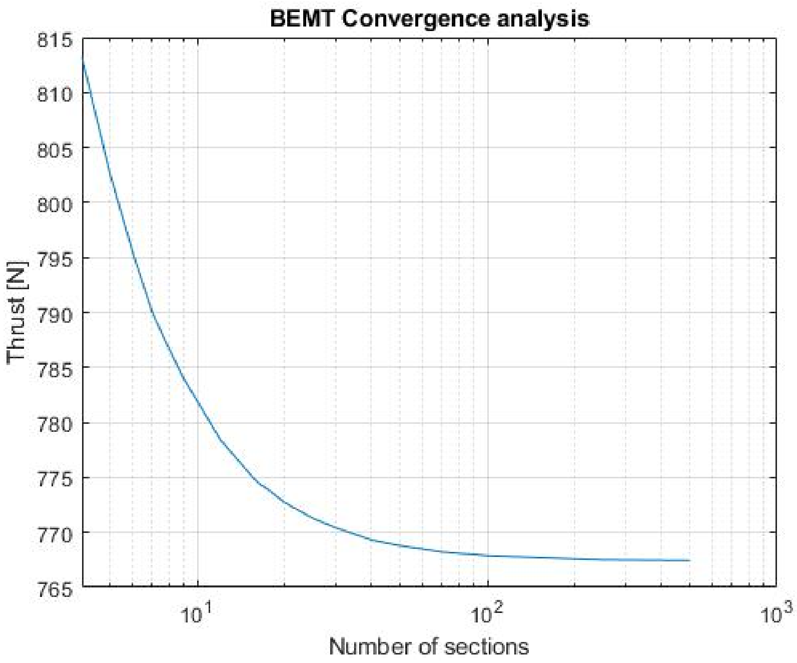


Figure 3.8: *MATLAB* Convergence analysis based on the number of sections

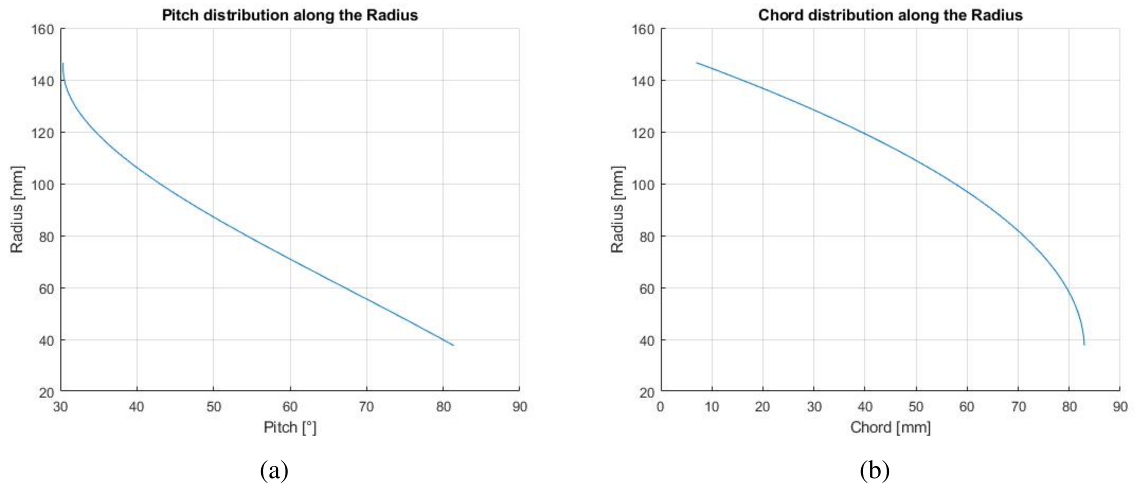


Figure 3.9: (a) Final pitch distribution (b) Final chord distribution

The first interesting design results are the final pitch and chord distributions shown in Figure 3.8.
As we said before, the pitch distribution comes from an interpolation between an ideal distribution and a linear distribution near the propeller hub. The chord distribution instead comes from the performance optimization and mechanical stress assessment. The maximum chord is reached at the root of the blade where the centrifugal force and the bending moment are maximum.

Another interesting result is the thrust distribution along the radius. Figure 3.9 shows the thrust distribution of the final propeller.
As we can see the distribution is quite non-uniform and the maximum thrust is more or less in the middle of the blade span.
In the first version of the code, the thrust distribution was quite different because the pitch distribution was not a quasi-ideal distribution but it was calculated through an optimization loop starting from a constant distribution. The pitch optimization loop was coupled with the chord optimization and the two distributions were iteratively calculated until convergence. In particular, the pitch distribution optimization algorithm followed the same idea and structure as the chord optimization algorithm, but the descent algorithm optimized the pitch instead of the chord. The thrust distribution results were very different with many consequences. Figure 3.10 shows an example of the thrust distribution calculated through the original pitch-chord optimization loop.
As we can see, comparing the two distributions, the optimized-pitch distribution is more non-uniform, has a higher thrust peak and, most of all, the thrust is concentrated nearer the blade tip. These differences produce lots of problems from the mechanical point of view. Indeed, the distribution in Figure 3.9, due to a higher bending moment produced by the thrust distribution, generates much bigger stress at the root of the blade where the mechanical stress reaches its maximum value. This would impose oversizing the chord distribution to guarantee sufficient blade thickness, which is of course a negative drawback.

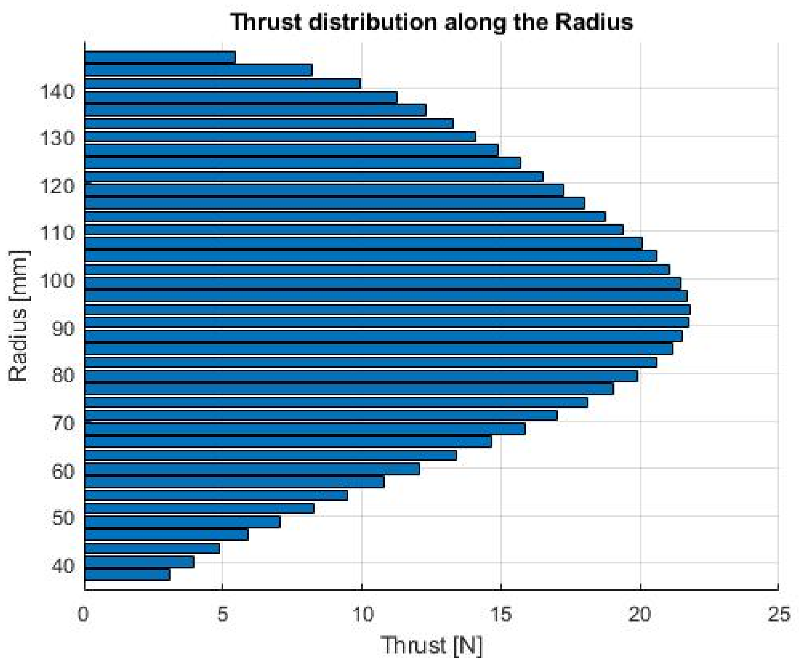


Figure 3.10: *MATLAB* final thrust distribution along the propeller radius

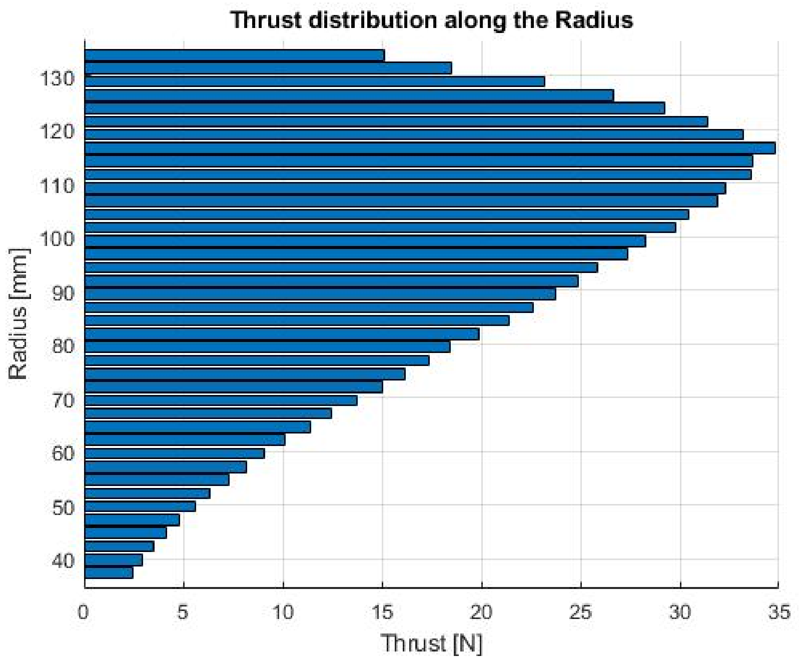


Figure 3.11: *MATLAB* optimized thrust distribution with the old code version

Since the quasi-ideal distribution allows us to produce a more uniform thrust distribution, with a thrust peak nearer the blade root, and thus a better stress distribution, in the final code version the pitch optimization was eliminated and a quasi-ideal pitch distribution was imposed in the initialization step.

Furthermore, about the stress distribution, we saw in the previous section that the analytical stress computation must reach an inverse safety factor of 0.2 (i.e. a safety factor of 5), which is quite an oversized safety factor. But this is necessary since the *finite element analysis* (FEA) shows completely different stress results compared with the analytical procedure, as we will see in the next chapter. For this reason, the value of 0.2 was chosen empirically to guarantee an acceptable maximum stress in the FEA.

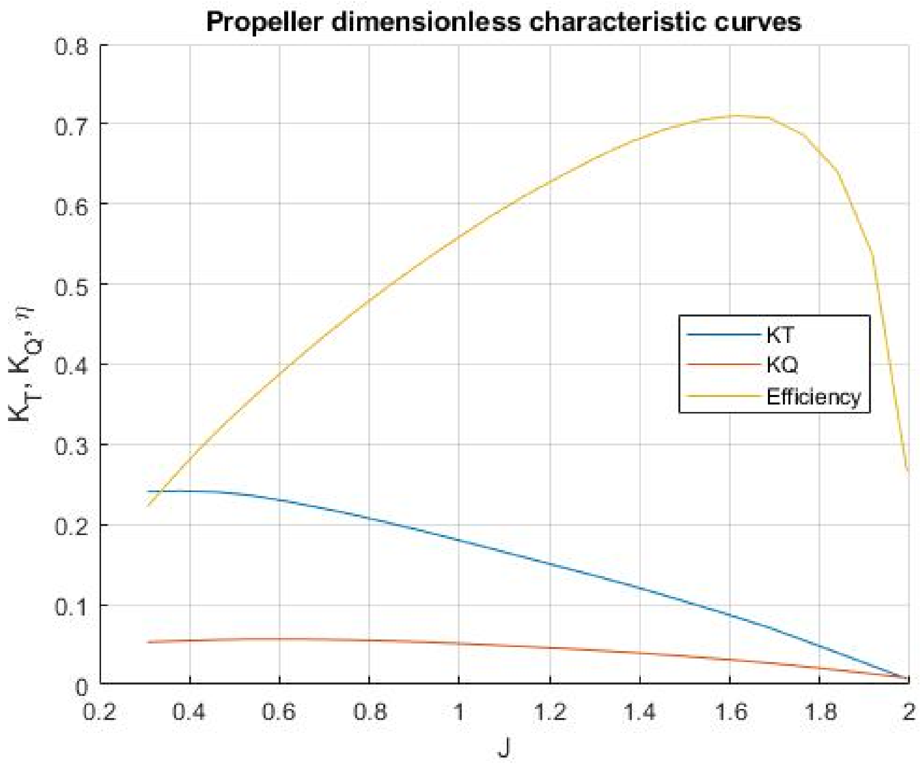


Figure 3.12: *MATLAB* final propeller characteristic curves

Finally, in Figure 3.11, we can see the dimensionless characteristic curves of the designed propeller for variable advance ratio. The maximum efficiency of 0.71 is reached around a value of 1.65 of J, which corresponds to a velocity of around 20 kn with a total thrust of 340 N. Unfortunately, this thrust is not sufficient to reach 20 kn if compared with the theoretical resistance characteristic of the hulls. Anyway, since the objective of this propeller is to provide better acceleration at low velocities and the same maximum velocity as the older propeller, the impossibility of reaching the efficiency peak is not a big issue. Since these performances were considered adequate, we chose to proceed with the CAD design, CFD analysis and FEA validation as we are going to see in the next chapter.

3.7 Cavitation Characteristic

To complete the design process, we need to consider the cavitation tendency of the propeller. As generally accepted, cavitation refers to the phenomenon in which the decreasing static pressure of a liquid, below the liquid's vapour pressure, leads to the formation of small vapour-filled cavities in the liquid. This phenomenon could be dangerous and damage the machinery. A comprehensive explanation of the physics principles behind cavitation, cavitation inception and design considerations are described as usual by Carlton [9] in *Marine Propellers and Propulsion*. Here we are just going to describe the methodology adopted and the consequent results.
As suggested by Batten [5], *XFOIL* can be useful in the preliminary design stage since it allows us to compute the pressure coefficient at each blade section. From the *MATLAB* code analysis we got the effective angle of attack at each blade section with which we can evaluate, using *XFOIL* analysis module, the minimum C_p at each section of the blade. At this point, we can compare the local pressure coefficient with the cavitation number, which is defined as:

$$\sigma = \frac{P_{AT} + \rho g h - P_V}{\frac{1}{2}\rho V^2} \tag{3.16}$$

where P_{AT} is the atmospheric pressure, h is the depth of immersion, ρ is the water density, P_V is the vapour pressure and V is the free stream velocity. Comparing local pressure coefficients and the cavitation number, we can predict if cavitation at a particular blade section will occur or not.
Figure 3.13 shows the comparison between the local C_p and the cavitation number calculated for the whole propeller at the optimal working point.
As we can see, the lower half blade seems to be safe from cavitation, while the upper half may experience cavitation. It is worth highlighting that *XFOIL*, for the used airfoil profile, predicts a huge localized peak of the pressure coefficient (for instance, see Figure 3.2). This peak value was used to evaluate the cavitation characteristic of the propeller but it's not representative of the whole chord of the airfoil. Since the presence of cavitation for a 5÷10$\%$ of the local total chord length is generally considered acceptable, and also the pressure coefficient falls quickly around the peak, the cavitation characteristic was considered acceptable.
We can now proceed with the CAD modelization of the designed propeller.

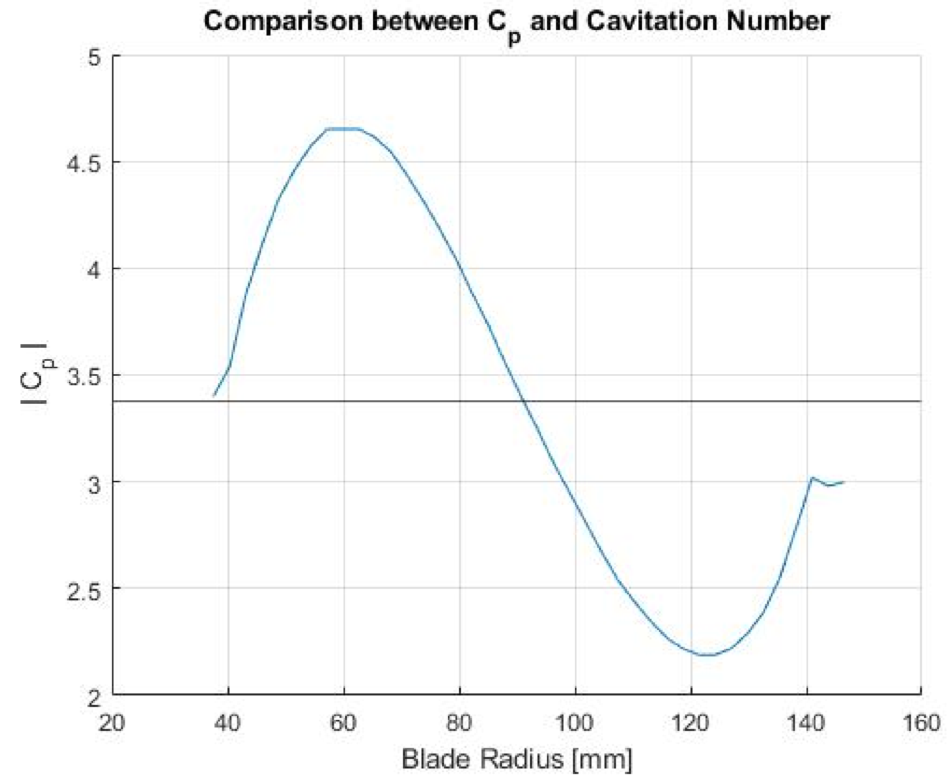


Figure 3.13: *MATLAB* Cavitation Evaluation

Chapter 4

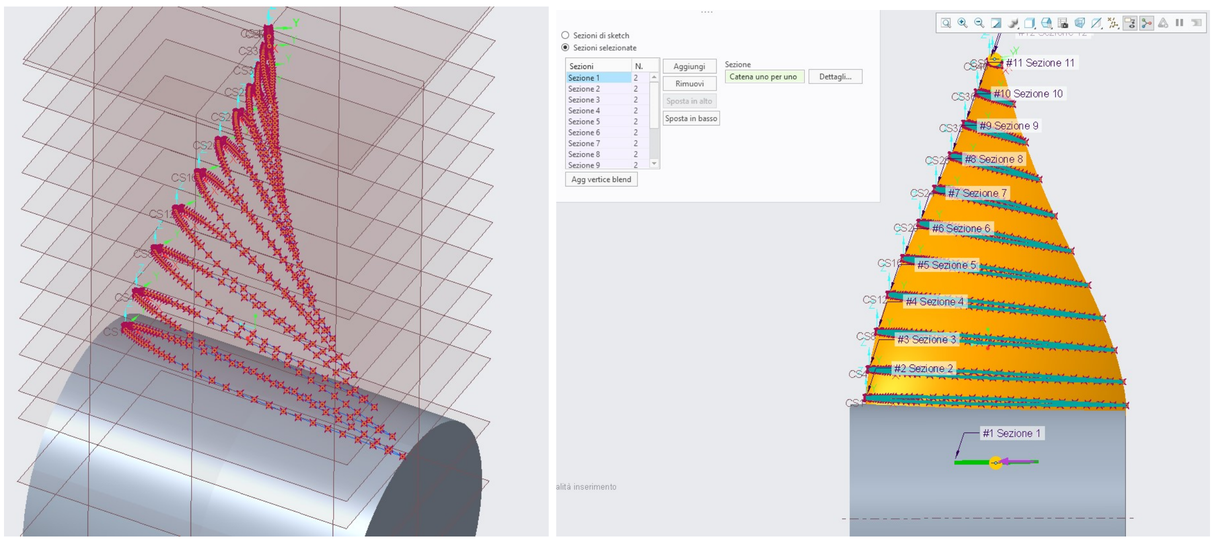
Propeller CAD model, CFD and FEM analysis

4.1 Introduction

This chapter presents the CAD model of the designed propeller, the CFD simulations performed with *Ansys Fluent* module of *Ansys Workbench 2020 R2* to validate the *BEMT* model, and the FEM analysis performed with *Ansys Static Structural* module to guarantee sufficient mechanical strength of the propeller.

4.2 Propeller CAD model

To developed the CAD model of the propeller *PTC Creo 9* was used.
One possible way to create the blade geometry is to import a *.txt* file of airfoil point coordinates for each section of the blade. Airfoil dimensionless point coordinates can be easily found online or in the literature. Once we have the coordinates, we need to calculate the real section coordinates using the chord distribution designed by the *MATLAB* program.
For each section, we must define a local reference system with the axis correctly oriented to represent the blade pitch distribution. In this way, importing point coordinates referred to the local reference system, the airfoil shape is already oriented with the correct local pitch angle. Then, point coordinates can be used to create a line which represents the blade section at each radius using the *Line through points* function. Finally, all the blade sections can be interpolated with the blend function that generates the blade geometry. Figure 4.1 shows the blade sections points and lines of the designed blade on the left, and the blend function definition to create the blade geometry on the right.
Once the blade is generated, we can round the blade's trailing edge, the blade's tip and, especially, the blade's root. As we are going to see in the next sections, the blade root is the most stressed zone of the propeller. Thus, root rounding becomes very important to prevent the yielding or even the break of the blade. Figure 4.2 shows the propeller geometry with the root rounding which was designed to 5 mm.
After the simulations, the hub geometry was completed with the hole and the correct geometry



(a) (b)

Figure 4.1: (a) CAD Blade sections points and lines (b) CAD Blend function definition to create the blade

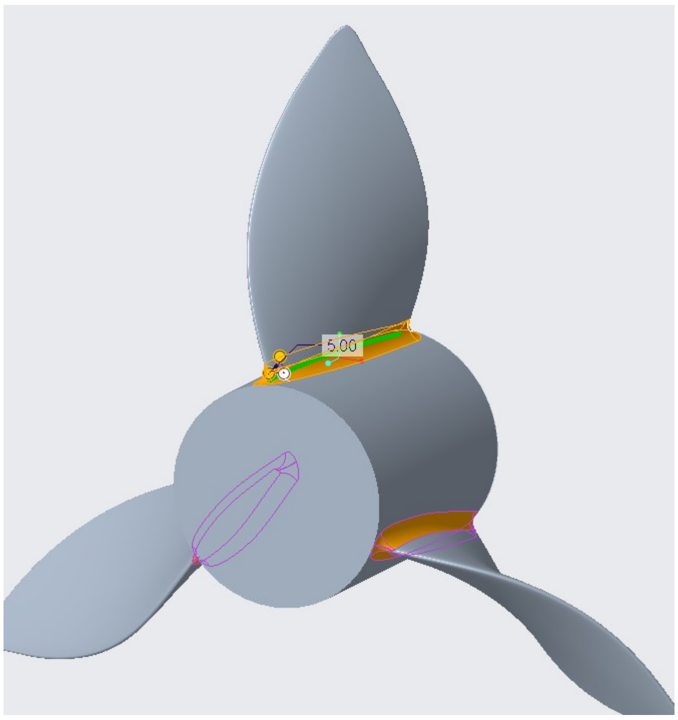
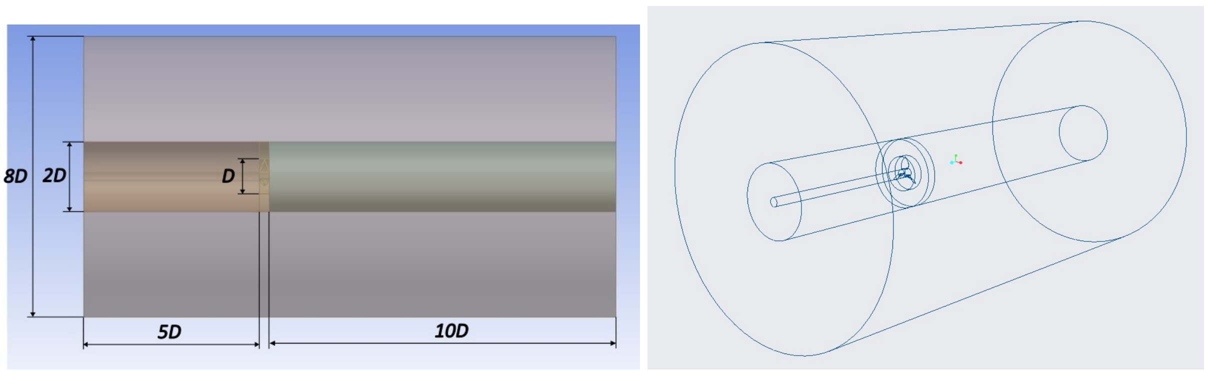


Figure 4.2: CAD Propeller geometry and root rounding

to allow the assembly of the propeller with the engine shaft.
It is worth enlightening that the cylindrical hub geometry was adopted after CFD simulations. Often, propellers for light applications adopt a conic hub geometry, and this was the solution initially adopted in this case as well. Our *BEMT* model doesn't allow us to take into account hub



(a) (b)

Figure 4.3: (a) Computational domain parametric dimensions (b) CAD wireframe of the Computational domain adopted for CFD simulations

different geometries, so CFD was used to test the two different hub configurations. The comparison results will be shown in the next section together with the overall performance analysis. At this point, we want to say that conical geometry seems to produce higher thrust with lower efficiency. Thus, cylindrical geometry was preferred.
With the complete geometry of the propeller, we can now analyze the CFD setup and results.

4.3 CFD Setup and Results

In this section, the CFD setup and the main simulation results will be shown and discussed.

To simulate the propeller open water performance, firstly we need to create the simulation domain. As recommended by the *ITTC Open Water Test Procedures and Guidelines* [28], the test bench must avoid any air drawing from the water surface, and the inflow must be as parallel as possible to the shaft axis. To guarantee these conditions in the simulation environment, the computational domain should be large enough to avoid any disturbances due to the boundary walls (since we performed single-phase simulations, the air drawing from the water surface is automatically guaranteed). Furthermore, a long cylindrical shaft is introduced ahead of the propeller to guarantee as uniform as possible inflow over the propeller. Figure 4.3 shows the cylindrical *Creo* 3D model used to perform CFD simulations and its main dimensions which were parameterized with the propeller diameter.
The model was exported to the *.stp* format and then imported into *Ansys Workbench* environment. The *Ansys Fluent* module was used to perform CFD simulations.
The simulation domain was divided into smaller subdomains to allow better mesh quality control and to facilitate structured mesh creation. Figure 4.4 shows the mesh quality around the rotor subdomain. As you can see, the mesh is structured and has hexahedral elements in each subdomain but in the rotor domain where the mesh is non-structured and has tetrahedral elements.
After the mesh generation, a mesh convergence analysis was carried out to find the best thresh-

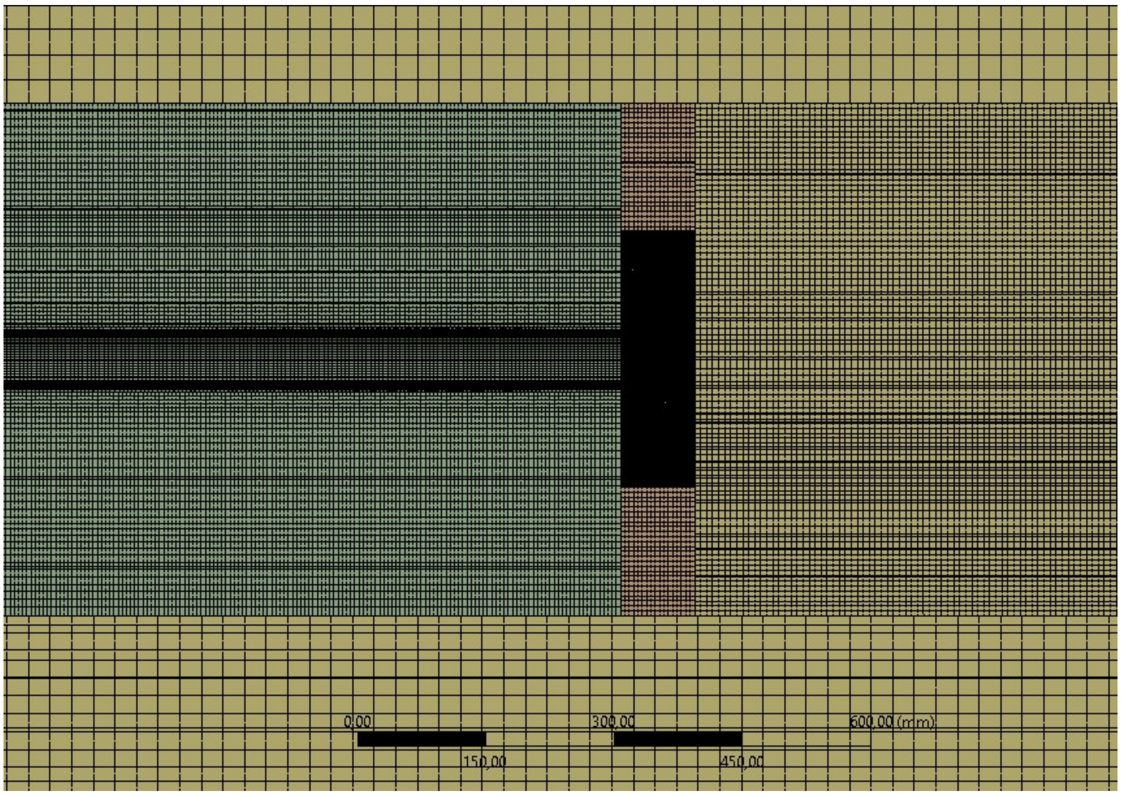


Figure 4.4: CFD mesh around the rotor subdomain

old between results accuracy and computational effort. Figure 4.5 shows the convergence trend for the total number of elements of the mesh. In this case, the output torque applied to the propeller was chosen to analyze the convergence behaviour. Unfortunately, since our computational capability was limited, it was not possible to use a larger number of elements. Thus, the maximum number of elements of the convergence analysis was used to analyse the whole propeller characteristics, namely around 13.4 million elements.
With the optimal mesh number of elements, we carried out simulations with variable inflow velocity and constant propeller rotational velocity to get the complete propeller characteristics. We adopted a steady-state *pressure-based* solver, a viscid *Spalart-Allmaras* model, rotational velocity applied through the *Frame-motion* option, and convergence residual condition set to $1e^{-4}$. The propeller dimensionless characteristics, calculated through CFD, are shown in Figure 4.6.
Figure 4.7 shows the pressure distribution on both propeller sides. As we can see, the pressure calculated is axisymmetric. In reality, since the boat engine is an outboard engine in a pushing configuration, the interaction of the water flux with the immersed part of the engine will exert an influence on the pressure distribution over the propeller. In particular, each blade rotating will see an oscillating pressure distribution due to the periodic presence or absence of the outboard engine. For this reason is reasonable to guarantee a higher structural safety margin to not overcome the material yield strength due to the pressure oscillations.
As stated in the previous section, the hub geometry was chosen after CFD analysis. The first design geometry was a conical geometry with a conicity ratio of $1:4$, similar to the original

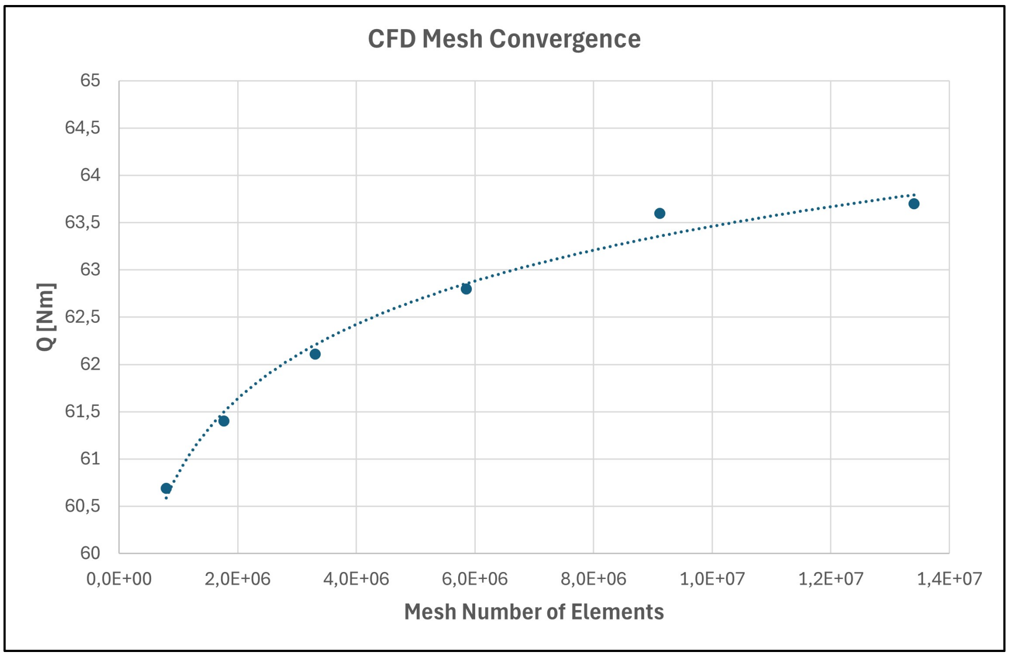


Figure 4.5: CFD mesh convergence analysis

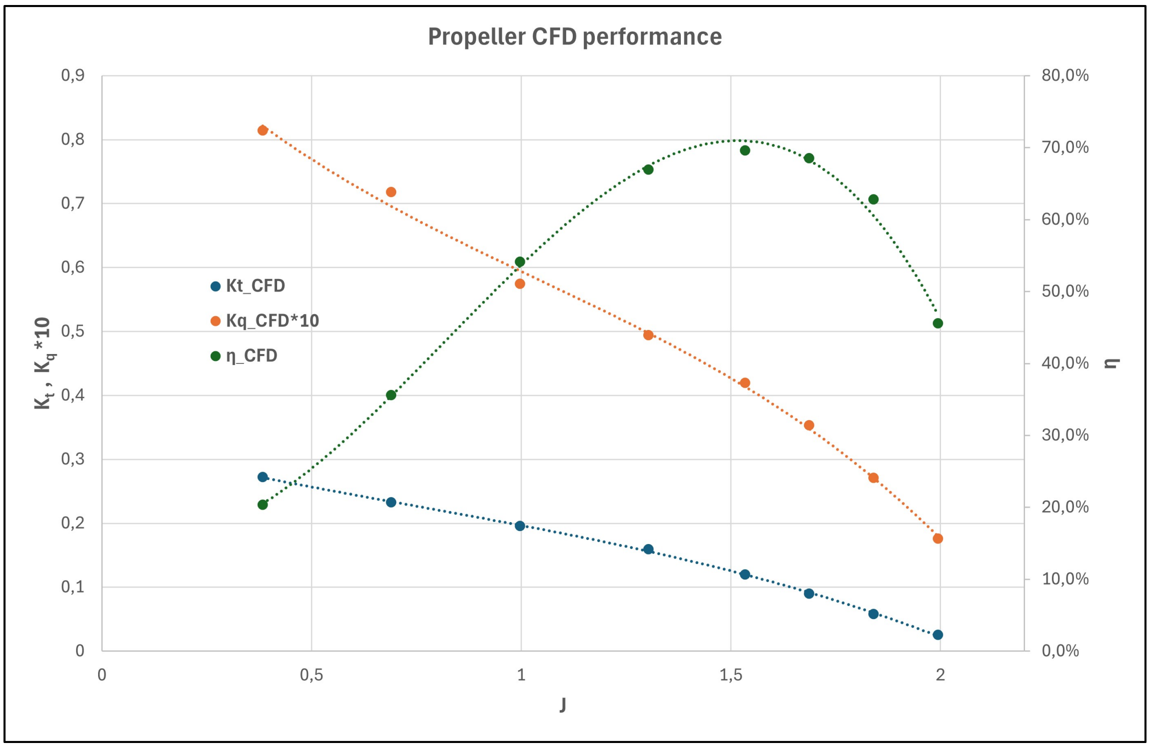
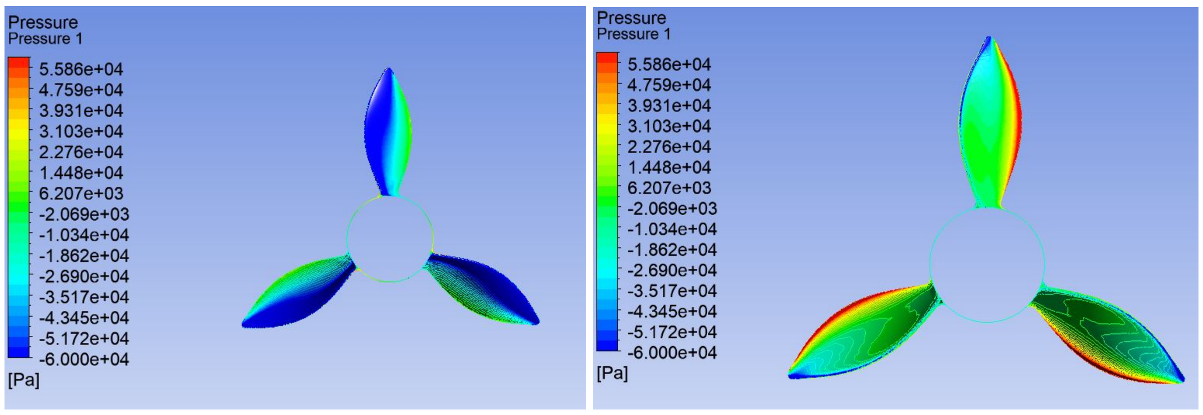


Figure 4.6: CFD Designed Propeller Characteristic Performance

propellers provided by the engine manufacturer, who sponsors the team, and similar to typical hub geometries for light boat propellers. The conical hub geometry was simulated at the optimal working point and compared with a cylindrical hub. Figure 4.8 shows the propeller with conical



(a) (b)

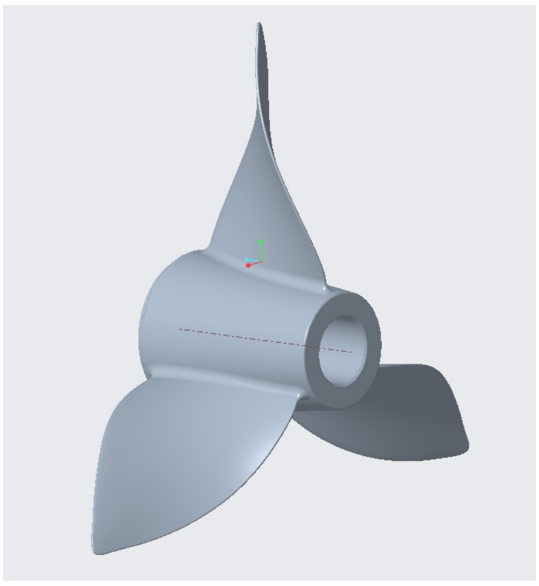
Figure 4.7: (a) Propeller pressure distribution front-side (b) Propeller pressure distribution back-side

geometry and the performance results in comparison between the two different geometries.
As we can see, the conical hub geometry provides better thrust action, due to a larger blade surface. On the other hand, the thrust improvement is obtained despite a 5.4% larger torque requested which reduces the propeller efficiency. Even if the efficiency decrease was not a big issue, the increased total power requested was a problem since the cylindrical geometry already required all the available power from the engine. For this reason, the cylindrical geometry was adopted.
Finally, we can compare CFD results with *BEMT* results. Figure 4.9 shows the comparison.
As we can see, there is a good correspondence between *BEMT* code results and CFD simulations. In particular, thrust and efficiency values are very close around the working point and at the beginning of the curve as well. On the contrary, torque curves, and thus power absorption of the propeller, show little divergence at the optimal working point and a big divergence at the beginning of the curve. Since propeller characteristics show usually a descending behaviour even at low advance-ratio values, we suggest that *BEMT* is not completely reliable in calculating propeller performance far away from the optimal working point, and the bibliography analyzed states the same conclusion. For this reason, CFD results at the beginning of the characteristic curve are considered more accurate.

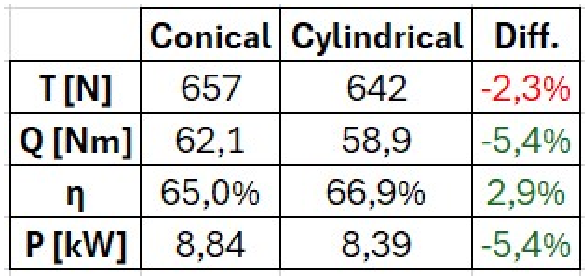
Once we finished analysing propeller fluid-dynamic characteristics, the CFD pressure distribution of the design working point was exported to the *Ansys Static Structural* module to verify the deformation and stress distributions.

4.4 FEA Setup and Results

In this section, the propeller *FEA* setup and results are discussed. Furthermore, as we mentioned before, the decision to implement an ideal pitch distribution due to structural considerations is here explained.



(a)



	Conical	Cylindrical	Diff.
T [N]	657	642	-2,3%
Q [Nm]	62,1	58,9	-5,4%
η	65,0%	66,9%	2,9%
P [kW]	8,84	8,39	-5,4%

(b)

Figure 4.8: (a) Propeller CAD with conical hub geometry (b) CFD comparison between conical and cylindrical hub geometries

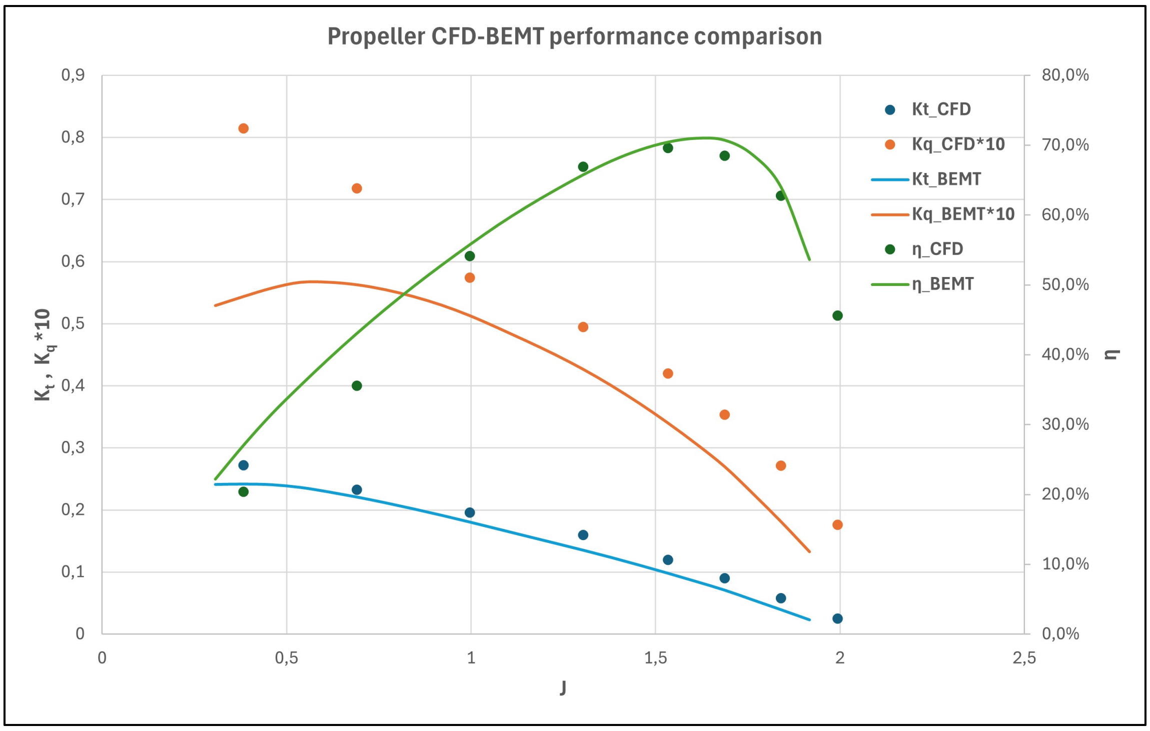


Figure 4.9: CFD-BEMT Performance comparison

Ansys Fluent allows easily to export a pressure distribution from a *Named Selection* object of the *Fluent* module into the *Static Structural* module. Thus, it allows us to export the surface

pressure distribution over the propeller geometry into another module. Once we have generated a mesh of the geometry, the pressure load is applied through a pressure interpolation between the *Fluent* and *Mechanical* meshes.
The pressure load is the most important load component, but in addition to the flow dynamic condition we must consider the centrifugal force which was applied through a centrifugal load. Furthermore, a fixed boundary condition, applied to the hub hole, was applied.
Regarding the propeller material, the propellers have been manufactured through 3D printing by *Continental Engineering Services* thanks to a sponsorship agreement. The technology used was *MJF* (Multy-Jet Fusion) and the filament has been *Nylon PA12*. Since there are many filaments of this material with different characteristics on the market, and since we didn't have any kind of control of the printing parameters, a conservative tensile strength value of *41.8 MPa* was considered.
As we did for the CFD analysis, first a convergence analysis is carried out to guarantee that the dimensions of the elements are suitable to correctly represent the stress distribution. Figure 4.10 shows the results of the analysis. Here, the maximum stress was chosen to probe the dependency on the mesh.

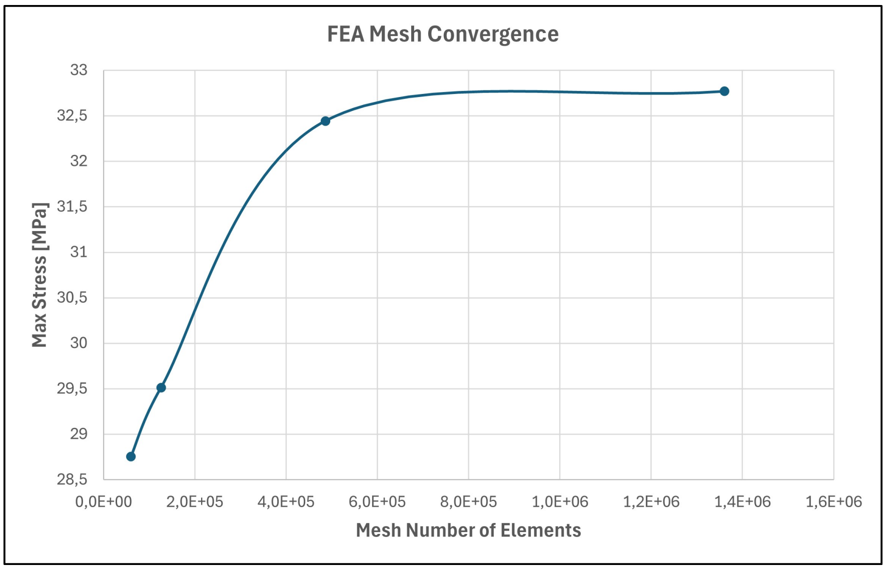
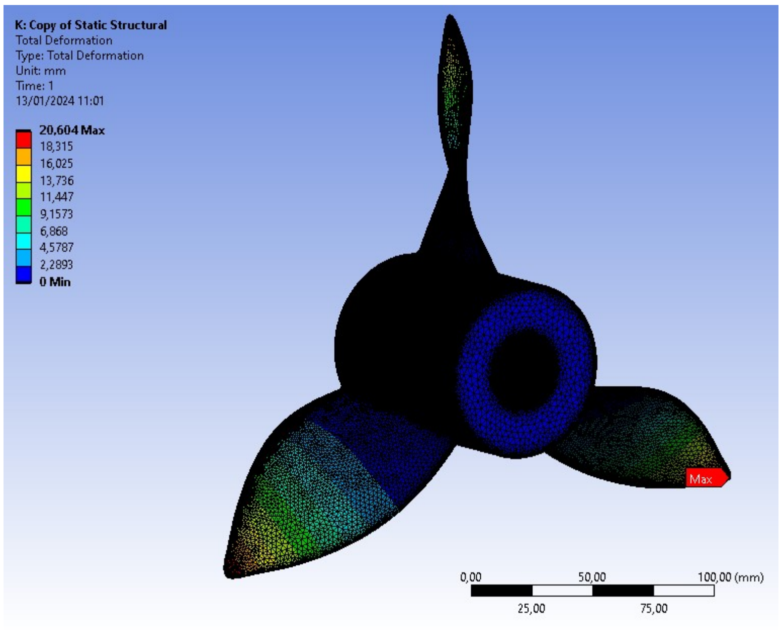
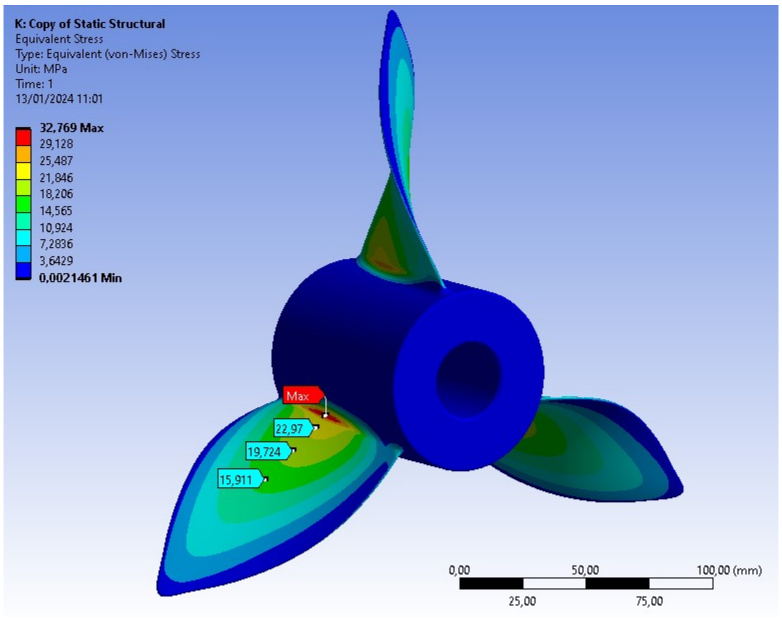


Figure 4.10: FEA mesh convergence analysis

As we can see, a number of elements around $6 \cdot 10^5$ is sufficient to guarantee mesh convergence. Then, with a suitable element dimension, the working point stress distribution was simulated. Figure 4.11 shows deformation and stress distributions. As was predictable, the maximum deformation is at the blade tip while the maximum stress is on the rounding at the blade root. The maximum stress of around $32MPa$ was calculated, thus a *safety factor* of around 1.3 is guaranteed. This result led us to conclude that the analytical method, proposed by *Carlton* and implemented in our code, is not reliable in predicting propeller stress since it doesn't take into account the presence of a radius at the blade root. We recall that a *safety factor* of 5 had been considered in the *MATLAB* stress analysis, which seems to be very different from the *finite-element analysis* result, which was considered more accurate.
The deformation distribution shows a maximum of around $20mm$ at the blade tip. Since we



(a)



(b)

Figure 4.11: (a) Propeller FEA mesh and deformation distribution (b) Propeller FEA stress distribution

considered a conservative *Young modulus* of $1250MPa$ and since the propellers, after printing, would have been covered with a layer of epoxy resin, which has higher stiffness after cure than *Nylon PA12*, we considered the deformation acceptable and the *safety factor* sufficient.

As we stated before, the first version of the *MATLAB* code performed a pitch distribution optimization with a similar algorithm used for the chord distribution. The optimized propeller, for the same diameter, number of blades, and optimal working point of the final propeller, showed higher thrust force despite a lower efficiency. However, the main issue was the thrust distribution along the radius (remember the comparison of Figures 3.10 and 3.11) which led to a too-large bending momentum on the blade with large stress at the blade root rounding. Figure 4.12 shows the stress calculated through the FEA of the blade rounding with the optimization of the pitch distribution.

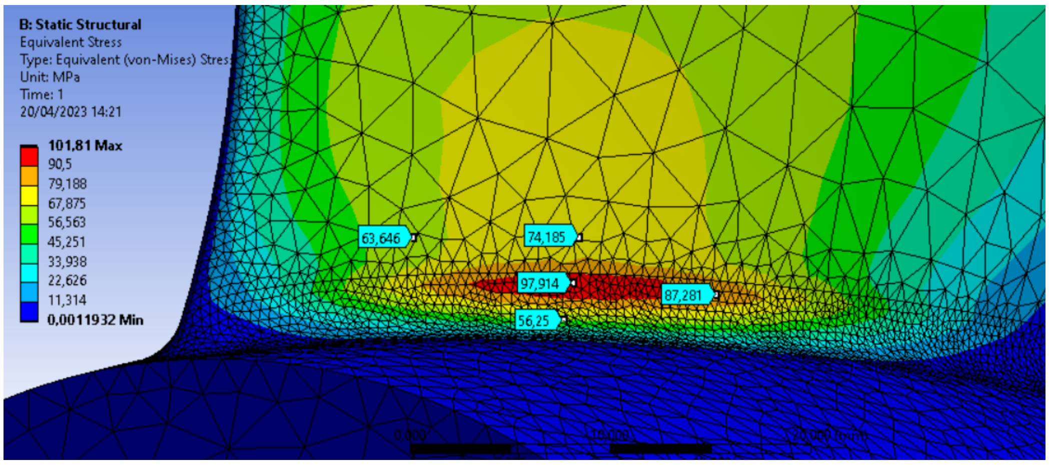


Figure 4.12: Propeller with optimized pitch stress at the blade's root

As we can see, the mechanical stress reaches a very large value of around $101 MPa$, which overcomes the material yield strength and is more than three times the stress calculated for the final propeller, which was designed with an ideal pitch distribution. The difficulty of reaching an acceptable stress distribution led us to the final version of the program, in which the pitch optimization has been substituted with an ideal pitch distribution known *a priori*.

At this point, since the results were considered acceptable, the design process was completed. Then, the propellers were printed, covered with epoxy resin and finally tested on the boat, as we are going to see in the next chapter.

Chapter 5

Propeller Manufacturing, Testing and Conclusions

5.1 Introduction

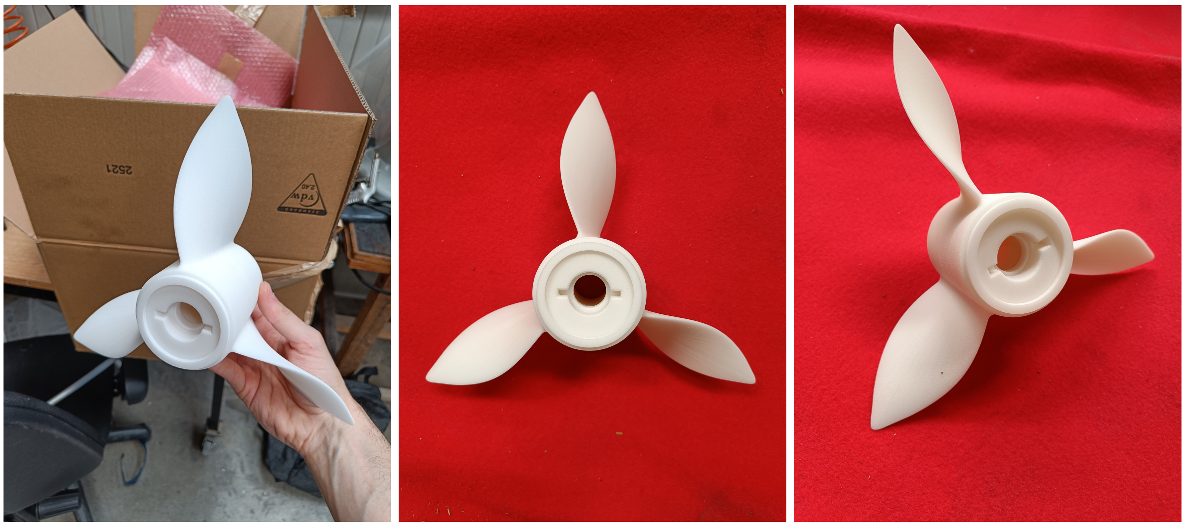
In this final chapter, we are going to see the results of real tests we carried out with the designed propellers. Then, the results from CFD simulations and experimental tests will be compared and discussed. Finally, we are going to provide our conclusions and future works that will be carried out.

5.2 Propellers Manufacturing and Treatments

Since the catamaran is powered by two engines with opposite directions of rotation, to improve thrust and manoeuvrability, we designed and printed two propellers, a left-oriented one and a right-oriented one. As we said before, the propellers were printed by *Continental Engineering Services*, through a sponsorship agreement, in *Nylon PA12* filament with *MJF* technology. Figure 5.1 shows the propellers as just arrived at the UniboAT lab. To improve the assembly, the hole tolerance was corrected at the milling machine since the printing process was not capable of reaching the correct tolerance. Then the propellers were smoothed with glass paper and covered with epoxy resin with the addition of a yellow pigment. Figure 5.2 shows the propellers mounted on the engines.

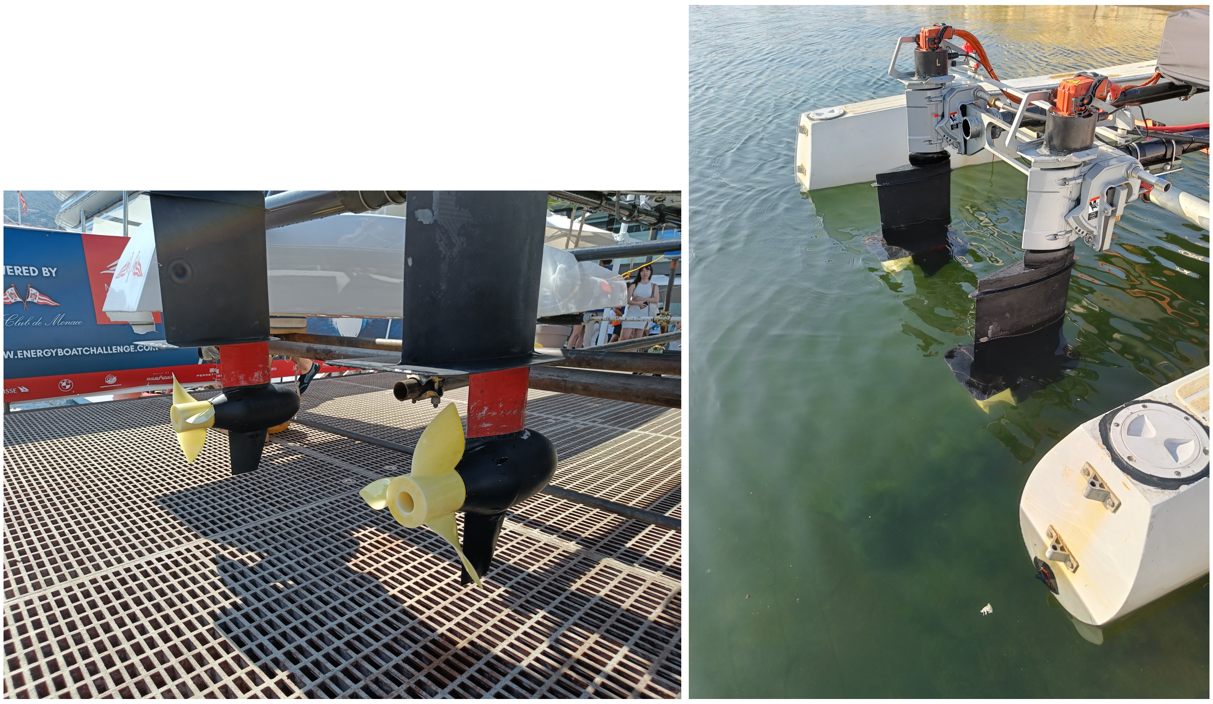
5.3 Propellers Testing and Data Analysis

Unfortunately, we don't have a testing bench for propellers, therefore we are not able to test the complete characteristics of the propellers.
Our testing session was conducted in open water in Ravenna at the *Circolo Velico Ravennate* during the *Navigare per Ravenna* event in September. The next pictures show the boat during the event in *Ravenna* when we tested the new propellers.
In these conditions, the data we can extract from the tests are limited. Indeed, we are not able



(a) (b) (c)

Figure 5.1: Final printed propeller



(a) (b)

Figure 5.2: (a) Propellers mounted on the engines (b) *Futura* with the new propellers

to directly measure the thrust of the propeller and the torque provided by the engines. Furthermore, since we don't have the complete characteristic curves of the electric engines used during the test, which had been supplied by a sponsor when the project started, it is not possible to indirectly calculate thrust and torque with a power balance. Anyway, we can compare the

(a) (b)

Figure 5.3: *Futura* attending the *Ravenna*'s event with the new propellers

power absorbed and the engine's angular velocities, as functions of the boat velocity, with our CFD results and with old propeller test data as well.
After we collected the test data, we could proceed to the analysis phase. The *MATLAB* code used to analyse the data is included in *Appendix B*. The variables of interest are *Motor Speed* of the two engines, *Inverter Current*, *Inverter Power* supplied to the two inverters, and the *GPS Speed* which is the boat's velocity. The sample frequency for each of these variables is 10ms.
Since the event was divided into three main parts (outward from the sea to the dock, a short exhibition of the boat, and finally the return to the arbour), the data sampled are divided into three groups as well. In particular, in the first part boat's velocity was quite low and constant while in the second and third groups, the velocity was higher and quite variable. Since we are interested in *steady-state* conditions, data of each group were filtered to consider only steady functioning conditions for at least 2-3 seconds. This assumption reduces the acceptable samplings from the second and third groups due to high data variability in time. Figure 5.4 shows the valid velocity data sampled during the overall event.
As we can see, we were able to collect a lot of data at low velocities and very few at high velocities. Moreover, since we were inside the dock area, we weren't allowed to test the maximum boat velocity. Anyway, these data allow us to make some considerations about the low-middle velocity range.
Figure 5.5 shows a comparison between the newly tested propellers and the old ones.
The chart on the left shows the power absorption as a function of the boat speed. The new propellers absorb more power at high velocities and less power at low velocities. In the middle range, the two absorptions are very similar. The chart on the right shows the engine rotational velocity as a function of the boat speed. Since data were collected in steady-state conditions, namely when thrust and boat resistance are equal, we can conclude that new propellers provide bigger thrust than the old ones, that was one of the main objectives of this project. However, this is obtained with higher power absorption, which might be an important limitation. Indeed, since our maximum deliverable power is limited by the temperature constraint of the inverters, which is reached in the range of 8÷9 kW, with the new propellers the maximum achievable

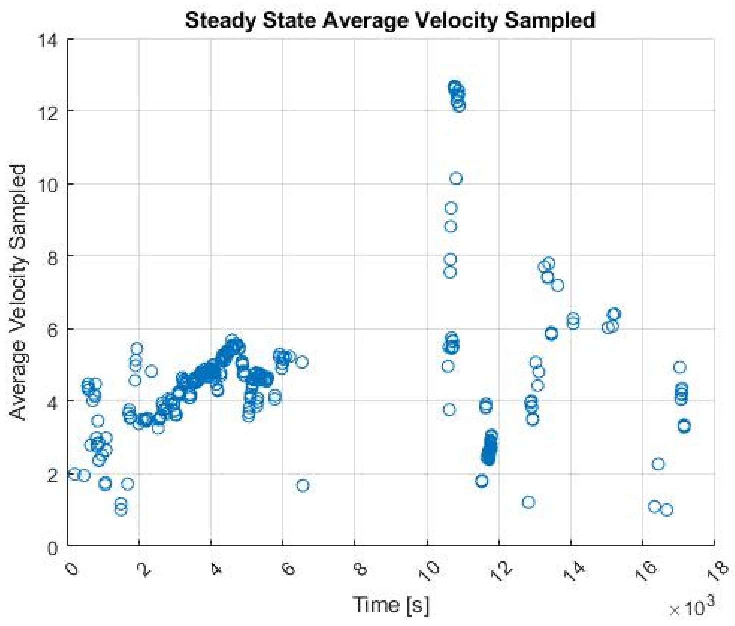


Figure 5.4: Velocity test sampled during *Navigare per Ravenna* event

velocity is around 14÷15 kn instead of $16kn$ with the old ones. Actually, during the test, the maximum velocity sampled was $16kn$, but it was sampled just for a few instants and not in steady-state conditions, therefore it was disregarded. Probably, higher velocities can be reached in favourable sea and wind conditions, but usually, our performance is limited by the power delivered from the inverters.
Finally, Figure 5.6 shows the power curves compared with the new propeller's CFD results.
We must highlight that the test data and CFD results comparison is not straightforward. Indeed, our data on the power are provided by sensors placed between the batteries and the inverters. Thus, the power we calculate is the total power absorbed by the powertrain system, and not just the power absorbed by the propellers. On the contrary, CFD results regard only the propeller, and this partially justifies the huge difference between real power absorbed and CFD prediction. Anyway, since our inverters and electric engines should have an efficiency higher than 90% for a wide part of the operating range, which partially covers the difference in the power curves obtained, we can say it is very likely that CFD simulations predict a lower power absorption compared with real test data. This is probably due to the ”bench test” configuration adopted in CFD simulations. Indeed, we considered a uniform and steady inlet flow and neglected the interaction of the propeller with the engine pod. It seems that these conditions are not very reliable in power absorption predictions and further investigations should be carried out in future analysis.
In future tests, we hope to be capable of a more comprehensive analysis of propeller performance thanks to either a test bench or more sensors installed on the boat. At least, with complete characteristic curves of inverters and engines we should be able to indirectly estimate the thrust and efficiency of the propellers.

So far, we explained our design method implemented in *MATLAB* and we commented on

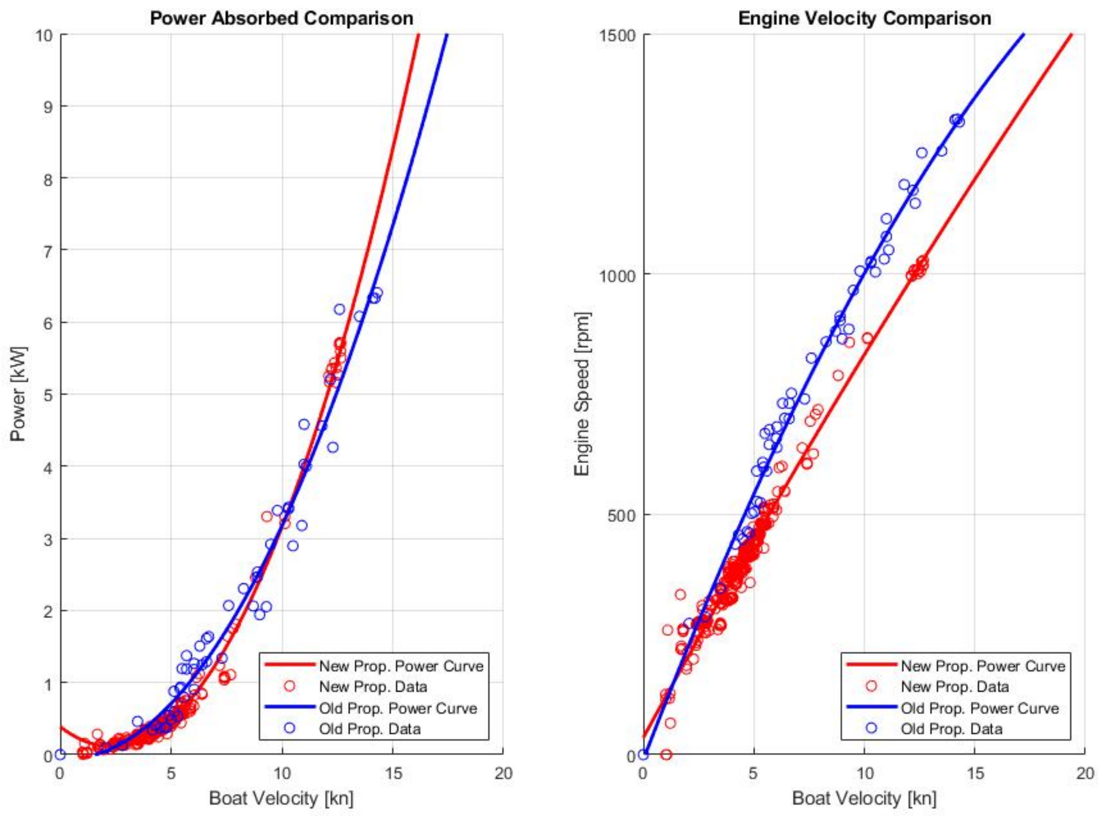


Figure 5.5: Absorbed power and velocity comparison between the new propellers and the old ones

the analysis carried out to validate the method. Furthermore, the tests completed, even if not comprehensive, allowed us to highlight some interesting results that will help the team to revise and improve the future design methodology. We can now look at the conclusions of this project and analyze possible future works.

5.4 Conclusions

- A *BEMT* algorithm with chord distribution optimization has been designed and implemented in *MATLAB* to design new propellers. The *BEMT* algorithm is quite easy to implement and allows fast computations with good reliability of the results, especially around the optimal working point. The main idea of the newly designed propellers was to improve acceleration and thrust without losing too much efficiency.

- CFD and FEA simulations were carried out to validate the code. Compared with CFD simulations, the *BEMT* algorithm seems reliable in thrust and efficiency predictions. Smaller reliability is highlighted in the torque prediction. Regarding FEA results, it is worth highlighting that analytical methods to predict blade maximum stress are unreliable

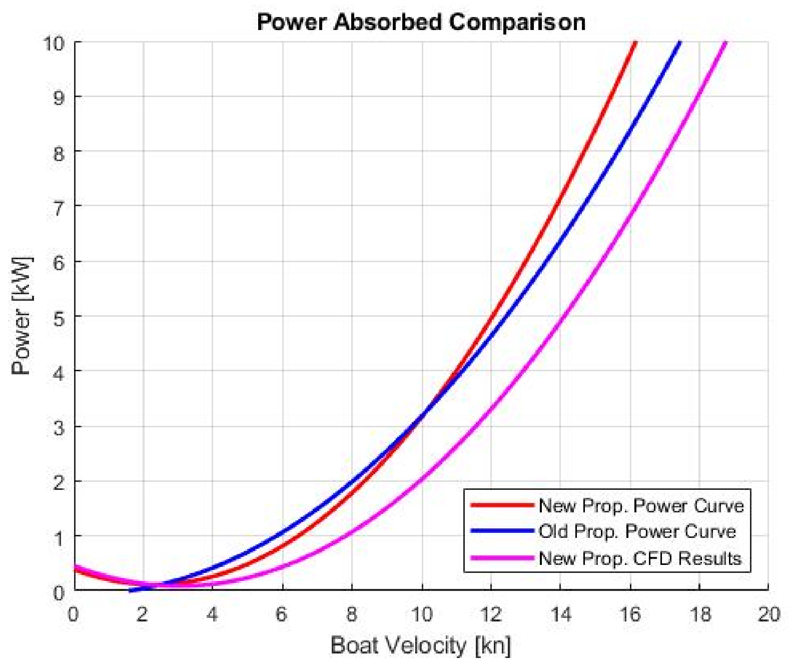


Figure 5.6: Absorbed power comparison between real test data of the new propellers, CFD results and test data of the old propellers.

unless they consider the rounding geometry at the blade's root. Otherwise, finite-elements analysis should be always carried out to validate the mechanical integrity of the propeller.

- Real tests were carried out to compare the designed propellers with the old ones. Even if we are not capable of analyzing the complete propeller characteristics, we can conclude that the objective of reaching a bigger thrust has been achieved. Anyway, the bigger power absorption than the predicted one by *BEMT* and CFD limits the maximum velocity we can reach with these new propellers. It seems that *BEMT* code and CFD methodology should be revised taking into account a non-uniform inlet flow due to the presence of the engine pod to predict a more realistic power absorption.

5.5 Future Works

- The *MATLAB* code should be improved in the coming years to achieve better correlation with CFD results and real test data, particularly regarding the power absorbed by the propeller. Furthermore, a pitch algorithm may be implemented to reach the optimal design for a particular working condition, taking into account the thrust requested as well.

- The *MATLAB* code should be improved by introducing other airfoil sections. An investigation of the effects on the propeller's performance of different airfoil sections might be interesting since there isn't comprehensive literature about that.

- The CFD methodology should be revised to take into account the engine pod interaction with the propeller since this seems to affect the power absorption prediction. Furthermore, the team will try to test the propeller's performance completely through a test bench, provided by a sponsor company, to improve the correlation between analytical methods, CFD and real test results.

Appendices

Appendix A

This appendix contains the main file of the MATLAB code and all the function files.

```
PropBEMT_v4.m Main File

%*************************************************************************
% Main File
% PropBEMT_v4 - Thesis Project
% Mechanical Engineering Master's Degree - University of Bologna
%
% Blade Element Momentum Method - Marine Propeller Design and Optimization
% Author: Manuel Cotoni
% 27/12/2023
%*************************************************************************

clc; clear;

% This program implements a Blade Element Momentum method to design a
% marine propeller to reach optimum thrust and efficiency at a particular
% working point.

%% INPUT DATA %%

D       = 0.296;              % [m] Propeller Diameter
Dh      = 0.0724;             % [m] Propeller Hub Diameter
Z       = 3;                  % Number of Blades
rpm     = 1360;               % [rpm] Rotational Speed at Design Point
rho     = 1020;               % [kg/m^3] Water Density
Va      = 15;                 % [kn] Forward Speed at Design Point
PD      = 1.72;               % Target Pitch Diameter Ratio
num_sec = 40;                 % Number of Blade Sections
g       = 9.807;              % [m/s^2] Gravity Acceleration

%% INITIALIZATION %%

% Constants
R       = D/2;                % [m] Propeller Radius
rh      = Dh/2;               % [m] Propeller Hub Radius
rps     = rpm/60;             % [rev/s]
w       = pi/30*rpm;          % [rad/s] Angular Velocity
U0      = Va/1.944;           % [m/s] Forward Speed
J       = U0/rps/D;           % Advance Ratio

% Blade Elements Position and Thickness
r = zeros(1,num_sec);
dr = (R-rh)/num_sec;          % Thickness of Blade Elements
for i = 1:num_sec
    r(i) = rh+(i-1)*dr+dr/2;  % Radius of Sections Position
end

tipspeed = w.*r/U0;           % Tip-Speed Ratio

% Chord Distribution Initialization
```

```
c = zeros(1,num_sec)+0.05;    % Constant Chord Distribution

% Ideal Pitch Distribution
% Pitch distribution is calculated through PD ratio and an interpolation
% with a linear distribution near the hub to avoid an infinite pitch angle
p = PD*D;                      % Target Pitch
theta_deg_ref = rad2deg(atan(p/(2*pi*R))); % Reference Pitch Angle
k_pitch = R*theta_deg_ref;
theta_deg_ideal = k_pitch./r;
% Linear distribution is imposed in the last 10 blade sections
theta_deg = [linspace(80,theta_deg_ideal(10),10) theta_deg_ideal(11:end)];
% Pitch distribution polynomial interpolation to avoid discontinuities
h_t = polyfit(r,theta_deg,3);

% Final Pitch Distribution in degrees
theta_deg = h_t(1)*r.^3+h_t(2)*r.^2+h_t(3)*r+h_t(4);
% Final Pitch Distribution in radians
theta = deg2rad(theta_deg);

%% POLARS INPUT DATA %%
% Polars must be provided through a .txt file properly formatted
[alpha,CLpp,CDpp] = polar_reader("NACA64A010.txt"); % NACA64A010 Airfoil

%% OPTIMIZATION LOOP
% Chord is optimized to reach high thrust and efficiency and to guarantee
% the mechanical strength of the blade
[c_opt,complete_chord] = chord_optimization_Rev1(Z,rho,num_sec,R,w,U0,J,...
    r,dr,theta,c,CLpp,CDpp); % Chord Optimization Function
solidity = Z.*c_opt./(2*pi*r); % Blade Solidity

% Convergence check
if complete_chord == 1
    fprintf("Chord convergence achieved\n");
else
    fprintf("Chord convergence not achieved\n");
end

[T,Q,CT,Ct,CQ,dT,dQ,eta,deta,iter] = induction_factors_calculation_Rev1(...
    num_sec,dr,r,w,U0,theta,R,solidity,Z,rho,c_opt,J,CLpp,CDpp);
c_opt_stress = c_opt;

% Mechanical Analysis of the Blade
[~,~,~,ICS] = Stress_Calculation(dT,dQ,r,dr,num_sec,"naca64a010.dat",...
    c_opt_stress,Z,theta,w); % Stress Calculation Function
% Chord Modification to obtain an Inverse Safety Factor smaller than 0.2 at
% each Blade Section
for i = 1:(num_sec-1) % Loop over blade sections
    while ICS(num_sec-i) >= 0.2 % Inverse Safety Factor validation loop
        % Chord is incremented of 2 mm at each iteration
        c_opt_stress(num_sec-i) = c_opt_stress(num_sec-i) + 0.002;
        solidity = Z.*c_opt_stress./(2*pi*r);
        [T,Q,CT,Ct,CQ,dT,dQ,eta,deta,iter] = ...
            induction_factors_calculation_Rev1(num_sec,dr,r,w,U0,theta,...
            R,solidity,Z,rho,c_opt_stress,J,CLpp,CDpp);
        % Stress calculation with the new chord value
        [sigma,A,weight,ICS] = Stress_Calculation(dT,dQ,r,dr,num_sec,...
            "naca64a010.dat",c_opt_stress,Z,theta,w);
    end
end

% Chord interpolation with a parabolic function
C = [r.^2; r; ones(size(r))]';
d = c_opt_stress;
a = [];
b = [];
Aeq = [R^2 R 1;r(1)^2 r(1) 1;2*rh 1 0];
beq = [0.005; c_opt_stress(1);0];
h_c = lsqlin(C,c_opt_stress,a,b,Aeq,beq);
% Final chord distribution and solidity
c_final = h_c(1)*r.^2 + h_c(2)*r + h_c(3);
solidity = Z.*c_final./(2*pi*r);

%% ANALYSIS OF THE DESIGNED PROPELLER

fprintf("Starting analysis of the propeller...\n")
% Propeller analyzed between 4 and 25 kn
Vaa = [4:1:(Va-1) Va (Va+1):1:25];
U0 = Vaa/1.944;

% Variable rotational speed with respect to the design point
%rpm = 1350;
%rps = rpm/60;
%w = pi/30*rpm;
J = U0/rps/D;

% Performance and Stress Calculation with variable advance velocity
for i = 1:length(Vaa)
    [T(i),Q(i),CT(i),Ct(i,:),CQ(i),dT(i,:),dQ(i,:),eta(i),deta(i,:),...
        iter] = induction_factors_calculation_Rev1(num_sec,dr,r,w,U0(i),...
        theta,R,solidity,Z,rho,c_final,J(i),CLpp,CDpp);
    [sigma(i,:),A,weight,ICS(i,:)] = Stress_Calculation(dT(i,:),dQ(i,:),...
        r,dr,num_sec,"naca64a010.dat",c_final,Z,theta,w);
end

% Final Coefficients Calculation
KT = T./(rho*rps.^2*D^4);
KQ = Q./(rho*rps.^2*D^5);
Pin = w*Q/1000;
eta = T.*U0./(w*Q)*100;
fprintf("Analysis completed.\n")

%% GRAPHIC RESULTS

% Thrust Distribution
figure(1)
hold on
```

```
barh(r*1000,dT(13,:))
title("Thrust distribution along the Radius")
grid on

% Final Propeller Characteristic Curves
figure(2)
hold on
grid on
yyaxis left
ylim([0,2000])
plot(Vaa,T,Color="r")
plot(Vaa,Q*10,Color="b",LineStyle="-")
yline(620)                  % Target Thrust 1
yline(720,Color="magenta")  % Target Thrust 2
yyaxis right
ylim([0 80])
plot(Vaa,eta,Color="g")
title("Propeller characteristic curves")
legend("Thrust","Torque*10","Thrust Target 17kn","Thrust target 20kn",...
    "Efficiency","Location","SouthWest")

% Final Propeller Dimensionless Characteristic Curves
figure(3)
hold on
grid on
plot(J,KT,J,KQ,J,eta/100)
title("Propeller dimensionless characteristic curves")
legend("KT","KQ","Efficiency","Location","best")

%% RESULTS COMPARISON
% Comparison of the results obtained through MATLAB and CFD analysis
% Actual Propeller CFD Results
Jst = [0.3927 0.7853 1.1780 1.5707];
Ktst = [0.3267 0.2384 0.1350 0.0079];
Kqst = [0.06129 0.05069 0.03619 0.01332];
etast = [0.333 0.588 0.7 0.149];
% Designed Propeller CFD Results
nnew = 1360;
Vanew = [13 17 20];
Tnew = [789.468 642.318 482.4204];
Qnew = [68.3738 58.9488 50.0208];
U0new = Vanew/1.944;
rpsnew = nnew/60;
wnew = pi*nnew/30;
Jnew = U0new/(D*rpsnew);
Ktnew = Tnew./(rho*rpsnew^2*D^4);
Kqnew = Qnew./(rho*rpsnew^2*D^5);
etanew = Tnew.*U0new./(wnew*Qnew)*100;
% Comparison Plot
figure(4)
hold on
grid on
plot(Jnew,Ktnew,Jnew,Kqnew,Jnew,etanew/100)
plot(Jst,Ktst,Jst,Kqst,Jst,etast)
legend("KT new","KQ new","Efficiency new","KT old","KQ old",...
    "Efficiency old","Location","northwest")
title("CFD Results Comparison");

% End of the file
```

chord_optimization_Rev1.m Function File

```
%**************************************************************************
% Function File
% Chord Optimization Loop
% Mechanical Engineering Master’s Degree - University of Bologna
%
% Blade Element Momentum Method - Marine Propeller Design and Optimization
% Author: Manuel Cotoni
% 27/12/2023
%**************************************************************************

% This function optimizes the chord of the propeller through a gradient
% descent algorithm. The cost function is defined as the sum of thrust and
% efficiency normalized with their initial values, respectively. The
% algorithm optimizes the chord maximizing the cost function value. The
% learning rate is optimized to guarantee convergence and good
% computational time in the case of 40-blade sections.

function [c_opt,complete_chord] = chord_optimization_Rev1(Z,rho,num_sec...
    ,R,w,U0,J,r,dr,theta,c,CLpp,CDpp)

% Input Variables
k_min      = 0.00003;     % Learning Rate Minimum Value
k_per      = 0.00015;     % Learning Rate Multiplication Constant
k_exp      = -0.02;       % Learning Rate Exponent Constant
chord_step = 0.004;       % Chord Optimization Step
toll       = 1.5e-4;      % Convergence Residual
iter_max   = 300;         % Maximum number of iterations

% Initialization
convergence = 0; % Convergence Boolean
sec_conv = zeros(1,num_sec);
i = 1; % Counter

% Gradient Descent Algorithm
while i <= iter_max && convergence ~= 1

    iter1(i) = i; % Iteration counter
    % Exponential Learning Rate
    learning_rate = k_min + k_per*exp(k_exp*iter1(i));

    solidity = Z.*c(i,:)./(2*pi*r); % Solidity of the Blade
    % Initial Values
    [~,~,~,~,~,dT(i,:),~,~,deta(i,:),~] = ...
```

```
        induction_factors_calculation_Rev1(num_sec,dr,r,w,U0,theta,R,...
        solidity,Z,rho,c(i,:),J,CLpp,CDpp);

    % Forward Step
    c_temp = c(i,:)+chord_step;
    solidity = Z.*c_temp./(2*pi*r);
    [~,~,~,~,~,dT1,~,~,deta1,~] = induction_factors_calculation_Rev1(...
        num_sec,dr,r,w,U0,theta,R,solidity,Z,rho,c_temp,J,CLpp,CDpp);
    % Cost Function Definition Normalized with the Initial Values
    cost_function1(i,:) = dT1./dT(i,:)+deta1./deta(i,:);

    % Backward Step
    c_temp = c(i,:)-chord_step;
    solidity = Z.*c_temp./(2*pi*r);
    [~,~,~,~,~,dT2,~,~,deta2,~] = induction_factors_calculation_Rev1(...
        num_sec,dr,r,w,U0,theta,R,solidity,Z,rho,c_temp,J,CLpp,CDpp);
    % Cost Function Definition Normalized with the Initial Values
    cost_function2(i,:) = dT2./dT(i,:)+deta2./deta(i,:);

    % Two-Points Gradient Calculation
    grad_cost_function(i,:) = ...
        (cost_function1(i,:)-cost_function2(i,:))/(2*chord_step);

    % New Chord Calculation
    for j = 1:num_sec
        if i == 1 % First Optimization Iteration
            c(i+1,j) = c(i,j) + learning_rate*grad_cost_function(i,j);
        else
            % Thrust Imposed Limitation to Blade Radius
            if dT(i,j) <= (R*100+1/3*R*100)
                if (j <= num_sec/8 || j >= num_sec*9/10)
                    % Slower Computation
                    c(i+1,j) = c(i,j) + ...
                        1*learning_rate*grad_cost_function(i,j);
                else
                    % Faster Computation
                    c(i+1,j) = c(i,j) + ...
                        5*learning_rate*grad_cost_function(i,j);
                end
            else
                c(i+1,j) = c(i,j);
            end
            % Section Convergence Check
            if abs(c(i+1,j)-c(i,j)) <= toll && i > 20
                sec_conv(j) = 1;
            else
                sec_conv(j) = 0;
            end
        end
    end
    % Blade Convergence Check
    if sum(sec_conv) == num_sec
        convergence = 1;
        c_opt = c(end,:);
        complete_chord = 1;
    end

    i = i+1;
    % Maximum Number of Iterations Reached
    if i == iter_max
        c_opt = c(end,:);
        complete_chord = 0;
    end
end
end

% End of the file
```

polar_reader.m Function File

```
%***************************************************************************
% Function File
% Polar Reader
% Mechanical Engineering Master's Degree - University of Bologna
%
% Blade Element Momentum Method - Marine Propeller Design and Optimization
% Author: Manuel Cotoni
% 27/12/2023
%***************************************************************************

% This function reads a .txt file containing the airfoil's lift and drag
% coefficients as functions of the angle of attack alpha. Then, the
% function returns the lift and drag interpolated through a spline function.

function [alpha,CLpp,CDpp] = polar_reader(fileID)

filename = fileID;

% Table Format
data = readtable(filename,"format","%f%f%f","Delimiter",["","\t"]);
rows = height(data);
cols = width(data);

% Initialization
alpha = zeros(rows,1);       % Attack Angle
CL = zeros(rows,1);          % Lift Coefficient
CD = zeros(rows,1);          % Drag Coefficient
column = zeros(rows,cols);

% Data Reading
for i = 1:cols
    column_index = data.Properties.VariableNames{i};
    column_data = data.(column_index);
    column(:,i) = column_data;
end
```

```
for i = 1:rows
    alpha(i) = column(i,1);
    CL(i) = column(i,2);
    CD(i) = column(i,3);
end

% Cubic Spline Interpolation of the coefficients
CLpp = spline(alpha,CL);
CDpp = spline(alpha,CD);

end

% End of the file
```

induction_factors_calculation_Rev1.m Function File

```
%***************************************************************************
% Function File
% Induction Factors Calculation Loop
% Mechanical Engineering Master’s Degree - University of Bologna
%
% Blade Element Momentum Method - Marine Propeller Design and Optimization
% Author: Manuel Cotoni
% 27/12/2023
%***************************************************************************

% This function calculates the velocity induction factors of the BEMT
% method and return the thrust and torque characteristics of the input
% propeller.

function [T,Q,CT,Ct,CQ,dT,dQ,eta,deta,iter] = ...
    induction_factors_calculation_Rev1(num_sec,dr,r,w,U0,theta,R,...
    solidity,Z,rho,c,J,CLpp,CDpp)

% Initialization
% Proportional constants to calculate the numerical correction
Kp1 = 0.006; Kp2 = 0.006;
% Derivative constants to calculate the numerical correction
Kd1 = 0.001; Kd2 = 0.0001;

iter_max = 10000;               % Maximum number of iterations
toll = 0.005;                   % Convergence tollerance
err = 0;                        % Error Boolean

a = zeros(num_sec,iter_max);
b = zeros(num_sec,iter_max);
convergence = zeros(1,num_sec);
phi_deg = zeros(num_sec,iter_max);
alpha_deg = zeros(num_sec,iter_max);
res1 = zeros(num_sec,iter_max);
res2 = zeros(num_sec,iter_max);
```

```
eps1 = zeros(num_sec,iter_max);
eps2 = zeros(num_sec,iter_max);

% Calculation Loop
for z = 1:num_sec

    % Error Check
    if err == 1
        break;
    end

    i = 1;    % Counter
    conv1 = 0;
    conv2 = 0;
    iter(z) = 0;

    while (i <= iter_max && convergence(z) ~= 1)

        if i == 1 % First Iteration

            % Induction factors initial guess values
            a(z,i) = 0;
            b(z,i) = 0;

            Un = U0*(1+a(z,i));                % Axial velocity
            Ut = w*r(z)*(1-b(z,i));            % Tangential velocity
            phi = atan(Un/Ut);                 % Intake flux angle
            phi_deg(z,i) = rad2deg(phi);
            alpha = theta(z)-phi;              % Effective angle of attack
            alpha_deg(z,i) = rad2deg(alpha);

            % Lift and drag coefficients interpolation (ppval is
            % preferable since it is faster than interp1)
            Cl = ppval(CLpp,alpha_deg(z,i));
            Cd = ppval(CDpp,alpha_deg(z,i));
            %Cl = interp1(alpha_polar,CL,alpha_deg(z,i),"linear");
            %Cd = interp1(alpha_polar,CD,alpha_deg(z,i),"linear");

            % Dimensionless coefficients
            cn = Cl*cos(phi)-Cd*sin(phi);
            ct = Cl*sin(phi)+Cd*cos(phi);

            f = Z*(R-r(z))/2/sin(phi);
            F = 2/pi*acos(exp(-f));

            % Residuals calculation
            res1(z,i) = solidity(z)*cn/(4*F*sin(phi)^2)-a(z,i)/(1+a(z,i));
            res2(z,i) = solidity(z)*ct/(4*F*sin(phi)*cos(phi))-...
                b(z,i)/(1-b(z,i));

            % Numerical correction calculation
            eps1(z,i) = Kp1*res1(z,i)+Kd1*res1(z,i);
            eps2(z,i) = Kp2*res2(z,i)+Kd2*res2(z,i);
```

```
else

    Un = U0*(1+a(z,i));                 % Axial velocity
    Ut = w*r(z)*(1-b(z,i));             % Tangential velocity
    phi = atan(Un/Ut);                  % Intake flux angle
    phi_deg(z,i) = rad2deg(phi);
    alpha = theta(z)-phi;               % Effective angle of attack
    alpha_deg(z,i) = rad2deg(alpha);

    % Lift and drag coefficients interpolation (ppval is
    % preferable since it is faster than interp1)
    Cl = ppval(CLpp,alpha_deg(z,i));
    Cd = ppval(CDpp,alpha_deg(z,i));
    %Cl = interp1(alpha_polar,CL,alpha_deg(z,i),"linear");
    %Cd = interp1(alpha_polar,CD,alpha_deg(z,i),"linear");

    % Dimensionless coefficients
    cn = Cl*cos(phi)-Cd*sin(phi);
    ct = Cl*sin(phi)+Cd*cos(phi);

    f = Z*(R-r(z))/2/sin(phi);
    F = 2/pi*acos(exp(-f));

    % Residuals calculation
    res1(z,i) = solidity(z)*cn/(4*F*sin(phi)^2)-a(z,i)/(1+a(z,i));
    res2(z,i) = solidity(z)*ct/(4*F*sin(phi)*cos(phi))-...
        b(z,i)/(1-b(z,i));

    % Numerical correction calculation
    eps1(z,i) = Kp1*res1(z,i)+Kd1*(res1(z,i)-res1(z,i-1));
    eps2(z,i) = Kp2*res2(z,i)+Kd2*(res2(z,i)-res2(z,i-1));

end

% Application of the numerical corrections to the induction factors
% and convergence check
% Axial Induction Factor
if (res1(z,i) >= toll || res1(z,i) <= -toll)
    conv1 = 0;
    a(z,i+1) = a(z,i) + eps1(z,i);
else
    conv1 = 1;
    a(z,i+1) = a(z,i) + eps1(z,i);
end
% Tangetial Induction Factor
if (res2(z,i) >= toll || res2(z,i) <= -toll)
    conv2 = 0;
    b(z,i+1) = b(z,i) + eps2(z,i);
else
    conv2 = 1;
    b(z,i+1) = b(z,i) + eps2(z,i);
end
```

```
        iter(z,i) = i;

        % Infinitesiaml Thrust, Torque and Efficiency calculation if
        % convergence has already been reached
        if (conv1 == 1 && conv2 == 1)
            convergence(z) = 1;

            dT(z) = 0.5*rho*(Un^2+Ut^2)*Z*c(z)*cn*dr;
            dQ(z) = 0.5*rho*(Un^2+Ut^2)*Z*c(z)*r(z)*ct*dr;
            deta(z) = dT(z)*U0/(w*dQ(z))*100;

            Ct(z) = (Un^2+Ut^2)/U0^2*solidity(z)*cn;
            Cq(z) = (Un^2+Ut^2)/U0^2*solidity(z)*ct;
        else
            i = i + 1;

            % Maximum number of iterations allowed
            if i == iter_max
                fprintf("Induction factors calculation: Max..." + ...
                    " number of iteration is reached. Convergence..." + ...
                    " not achieved.\n");
                err = 1;
                break;
            end
        end
    end
end

% Calculation of the total Thrust, Torque and Efficiency of the Propeller
if err == 0
    T = sum(dT);
    Q = sum(dQ);
    CT = 2/R^2*sum(r.*Ct*dr);
    %CT = T/(0.5*rho*pi*R^2*U0^2);
    CQ = 2/R^3*sum(r.^2.*Cq*dr);
    %CQ = Q/(0.5*rho*pi*R^3*U0^2);
    %eta = CT*J/CQ/2/pi*100;
    eta = mean(deta);
else
    fprintf("Calculation stopped (section %i)\n",z);
    T = 0; CT = 0; Ct = zeros(1,num_sec); CQ = 0; dT = zeros(1,num_sec);
    dQ = zeros(1,num_sec); eta = 0; Q = 0; deta = zeros(1,num_sec);
end
end

% End of the file
```

Stress_Calculation.m Function File

```
%***************************************************************************
% Function File
% Stress Calculation
% Mechanical Engineering Master’s Degree - University of Bologna
%
% Blade Element Momentum Method - Marine Propeller Design and Optimization
% Author: Manuel Cotoni
% 27/12/2023
%**************************************************************************

% This function determines the stress generated at each blade section due
% to the centrifugal force and the bending torque

function [sigma,A,weight,ICS] = Stress_Calculation(dT,dQ,r,dr,num_sec,...
    fileID,c,Z,theta,w)

A = zeros(1,num_sec); % Blade section area

% Material properties
rho_mat = 1010;     % [kg/m^3] Nylon PA12
sigma_max = 41.8;   % [MPa]

% Coefficients calculation. A and b depend on thrust and torque
% distributions on the blade respectively. Mh instead is a bending module.
for i = 1:num_sec
    a(i) = sum(dT(i:end).*(r(i:end)-r(i))*dr)/sum(dT(i:end)*dr);
    b(i) = sum(dQ(i:end).*(r(i:end)-r(i))*dr)/sum(dQ(i:end)*dr);
    Mh(i) = sum(dT(i:end))*a(i)*cos(theta(i))+...
        sum(dQ(i:end))*b(i)*sin(theta(i));
end

% Reading data of the airfoil. Yup, ylow, cxup and cxlow are the
% dimensionless airfoil points coordinate (the origin (0,0) is the
% leading edge)
filename = fileID;
data = readtable(filename,"format","%f%f","Delimiter",[""," \t"]);
cx = data.(1);
y = data.(2);
[k,~] = size(cx);
yup = flip(y(1:round(k/2),1));
ylow = y(round(k/2):end,1);
cxup = flip(cx(1:round(k/2),1));
cxlow = cx(round(k/2):end,1);
t = abs(yup)+abs(ylow); % Thickness

% Actual coordinates of the section taking into account the real chord of
% the airfoil
yup = yup.*c;
ylow = ylow.*c;
t = t.*c;
cxup = cxup.*c;
cxlow = cxlow.*c;

% Section area calculation
for i = 1:num_sec
```

```
    for j = 1:height(cxup(:,1))-1
        A(i) = A(i) + t(j,i)*abs((cxup(j+1,i)-cxup(j,i)));
    end
end

weight = rho_mat*Z*sum(A*dr); % Weight of the blades

% Centrifugal force calculation
for i = 1:num_sec
    Fc(i) = rho_mat*w^2*sum(A(i:end).*(r(i:end)-r(i))*dr);
end

% Tensile module calculation
for i = 1:num_sec
    for j = 1:round(k/2)
        if j == 1
            tensile_mod(i) = (2*sum((3*yup(:,i).*(yup(:,i)+t(:,i))+...
                t(:,i).^2).*t(:,i)*abs(cxup(j,i)))*sum(t(:,i)*cxup(j,i)))...
            /(3*sum((2*yup(:,i)+t(:,i)).*t(:,i)*(cxup(j,i))))-0.5*...
            sum((2*yup(:,i)+t(:,i)).*t(:,i)*cxup(j,i));
        else
            tensile_mod(i) = (2*sum((3*yup(:,i).*(yup(:,i)+t(:,i))+...
                t(:,i).^2).*t(:,i)*abs(cxup(j,i)-cxup(j-1,i)))...
                *sum(t(:,i)*(cxup(j,i)-cxup(j-1,i))))/(3*sum((2*yup(:,i)...
                +t(:,i)).*t(:,i).*(cxup(j,i)-cxup(j-1,i))))-0.5*sum((2*...
                yup(:,i)+t(:,i)).*t(:,i).*(cxup(j,i)-cxup(j-1,i)));
        end
    end
end

% Stress calculation
sigma = (Mh./tensile_mod+Fc./A)/10^6;

% Inverse safety factori calculation
ICS = abs(sigma)/sigma_max;

end

% End of the file
```

Appendix B

This appendix contains the scripts used to analyze test data and to compare them with CFD simulation and PropBEMT code results.

```
DataAnalysis.m

clc;clear;

%*********************************************************************
% Test Data Analysis
% Data Analysis
% Mechanical Engineering Master's Degree - University of Bologna
%
% Blade Element Momentum Method - Marine Propeller Design and Optimization
% Author: Manuel Cotoni
% 27/12/2023
%***************************************************************************

% This script analysis the data acquisited during propellers test in open
% water. The data acquisitions are 3 and are analyzed one at a time and
% stored in a data struct.
% The sample time is 10 ms.

counter = 0;
sampledData = zeros(1,3);
nPrevious = 0;
relaxation = 0;

% Since in the second and third acquisitions the velocity shows
% considerable variations, we apply a coefficient to the interval range
% and standard deviation to loosen up the limits and evaluate if in each
% interval variables are steady state or not
coeff = 1.4; % Multiplication Coefficient for second and third acquisitions

speedZeroLimit = 1; % Minimum valid Speed  [kn]
stdZeroLimit = 0.002; % Minimum valid Deviation

% Loading one of the data acquisitions at a time
for iter = 1:3
    load(['acquisition',num2str(iter),'.mat'])
    load(['acqPower',num2str(iter),'.mat'])
    n = size(GPS_Speed,1); % Array dimension of the data acquisited

    if iter == 1
        step = 250; % 100 steps = 1 s interval
        speedRangeLimit = 0.012; % Maximum valid Speed Variation ...
        % Acceptable Range is 2*speedRangeLimit
        stdSpeedVarLimit = 0.01; % Maximum valid Speed std Deviation
    else
        step = round(step/coeff); % 100 steps = 1 s interval
        speedRangeLimit = speedRangeLimit*coeff; % Maximum Speed ...
        % Variation % Acceptable Range is 2*speedRangeLimit
        stdSpeedVarLimit = stdSpeedVarLimit*coeff; % Maximum Speed ...
        % std Deviation
    end

    relaxLimit = step/2; % Relaxation between a valid interval and the
    % following one

    % For each acquisition we analyze a data interval calculating average
    % values. Then, if the data acquisited in that interval are of
    % interest, we stored them in a struct variable.
    for i = 1:(n-step)
```

```
        % Average values computation
        interval = GPS_Speed(i:(i+step-1));
        intervalCurrent1 = InverterCalc_TargetCurrent(i:(i+step-1));
        averageSpeed = sum(interval)/step;
        averageCurrent = sum(intervalCurrent1)/step;
        averagePower = sum(PackCalc_Power)/step;
        maxSpeed = max(interval);
        minSpeed = min(interval);
        stdDeviation = std(interval);
        topLimit = maxSpeed-averageSpeed;
        bottomLimit = averageSpeed-minSpeed;

        % Evaluation of the interval. We don't want to keep an interval in
        % which either velocity or current are zero.
        if relaxation >= relaxLimit % Relaxation Control
            if averageSpeed >= speedZeroLimit && stdDeviation ...
                    >= stdZeroLimit && averagePower > 0 % Minimum ...
                % Speed and Power Control
                if topLimit <= speedRangeLimit && bottomLimit ...
                        <= speedRangeLimit && stdDeviation <= ...
                        stdSpeedVarLimit % Maximum Variation Control
                    counter = counter+1;
                    validData.index(counter) = i + nPrevious;
                    validData.averageSpeed(counter) = averageSpeed;
                    validData.maxSpeed(counter) = maxSpeed;
                    validData.minSpeed(counter) = minSpeed;
                    validData.stdDeviation(counter) = stdDeviation;
                    validData.motor1Speed(counter) = ...
                        InverterCalc_MotorSpeed(i); % Inverter 1 current
                    validData.motor2Speed(counter) = ...
                        InverterCalc2_MotorSpeed(i); % Inverter 2 current
                    validData.inverter1Current(counter) = ...
                        InverterCalc_TargetCurrent(i);
                    validData.inverter2Current(counter) = ...
                        InverterCalc2_TargetCurrent(i);
                    validData.Power1(counter) = PackCalc_Power(i);
                    validData.Power2(counter) = PackCalc2_Power(i);
                    relaxation = 0;
                end
            end
        else
            relaxation = relaxation + 1;
        end
    end
    sampledData(1,iter) = counter;
    nPrevious = nPrevious + n;
end

% Valid Data saved in a single array
sampledData(1,3) = sampledData(3)-sampledData(2);
sampledData(1,2) = sampledData(2)-sampledData(1);

% Valid Data Plot
```

```
hold on
fig = plot(validData.index/100,validData.averageSpeed,'o');
grid on
xtickangle(45)
ax = ancestor(fig,'axes');
ax.XAxis.Exponent = 3;
xlabel("Time [s]")
ylabel("Average Velocity Sampled")
title("Steady State Average Velocity Sampled")

save("TestDataRev1.mat",'validData')

% End of the file
```

DataComparison.m

```
clc;clear;

%**************************************************************************
% Test Data Comparison
% Data Comparison
% Mechanical Engineering Master's Degree - University of Bologna
%
% Blade Element Momentum Method - Marine Propeller Design and Optimization
% Author: Manuel Cotoni
% 27/12/2023
%**************************************************************************

% Load Test Data already analyzed
load("TestDataRev1.mat")

power_new = max(validData.Power1,validData.Power2);
rpm_new = max(validData.motor1Speed,validData.motor2Speed);

% Polynomial interpolation of the test data
t = linspace(0,20,200);
fitPower_New = polyfit(validData.averageSpeed,power_new/1000,2);
Power_New = polyval(fitPower_New,t);
fitRpm_New = polyfit(validData.averageSpeed,rpm_new,2);
RPM_New = polyval(fitRpm_New,t);

%%% -------------------------------------------------------------------- %%%
% Old propeller data from previous test sessions

power_old = [0;96; 126; 339; 367;373;375;409;460;489;530;546;594;796;879;...
906;915;939;1181;1188;1194;1246;1265;1288;1338;1374;1506;1611;1637;1941;...
2052;2062;2067;2303;2466;2534;2896;2918;3177;3387;3408;3432;4000;...
4027;4262;4559;4580;5209;6079;6176;6330;6333;6406]; % [W]

Boat_speed_kn_old = [0;2.07;2.8;4.16;4.5;4.8;4.7;4.3;3.49;5.02;5.3;4.9;5.14;...
    5.58;5.13;6.04;5.4;5.44;6.02;5.7;5.5;6.4;6.04;6.6;7.3;5.7;6.3;6.6;...
    6.7;9;9.3;8.7;7.6;8.27;8.9;8.9;10.5;9.5;10.9;9.8;10.3;10.3;11.1;11;...
    12.3;11.8;11;12.2;13.5;12.6;14.2;14.1;14.3];

Motor_speed_rpm_old =[0;273;287;438;449;461;464;457;346;505;524;502;527;590;...
    590;639;608;598;658;645;668;700;682;699;740;676;731;731;752;865;885;...
    881;825;859;912;903;1004;966;1031;1006;1023;1026;1050;1078;1147;...
    1186;1115;1174;1256;1252;1322;1321;1316];

% Polynomial interpolation of the old test data
fitPower_Old = polyfit(Boat_speed_kn_old,power_old/1000,2);
Power_Old = polyval(fitPower_Old,t);
fitRpm_Old = polyfit(Boat_speed_kn_old,Motor_speed_rpm_old,2);
RPM_Old = polyval(fitRpm_Old,t);

%%% ------------------------------------------------------------------- %%%

%%% ------------------------------------------------------------------- %%%
% CFD new propellers results

CFD_Speed = [2.6;5;8;11;13.5];

CFD_Motor_Speed = [250;440;675;910;1100];

CFD_Torque = [2.24;6.54;15.05;26.98;39.06];

CFD_Power = CFD_Torque.*CFD_Motor_Speed*pi/30/1000;

% Polynomial interpolation of the new propellers CFD results
fitPowerCFD = polyfit(CFD_Speed,CFD_Power,2);
Power_CFD = polyval(fitPowerCFD,t);
fitRPMCFD = polyfit(CFD_Speed,CFD_Motor_Speed,2);
RPM_CFD = polyval(fitRPMCFD,t);

%%% ------------------------------------------------------------------- %%%

% Figures
% Data Comparison

figure(1)
subplot(1,2,1)
title("Power Absorbed Comparison")
hold on
plot(t,Power_New,'LineWidth',2,'Color','r')
plot(validData.averageSpeed,power_new/1000,'o','Color','red')
plot(t,Power_Old,'LineWidth',2,'Color','b')
plot(Boat_speed_kn_old,power_old/1000,'o','Color','b')
ylim([0,10])
ylabel("Power [kW]")
xlabel("Boat Velocity [kn]")
legend("New Prop. Power Curve","New Prop. Data", ...
    "Old Prop. Power Curve", "Old Prop. Data","Location","southeast")
grid on
subplot(1,2,2)
title("Engine Velocity Comparison")
```

```
hold on
plot(t,RPM_New,’LineWidth’,2,’Color’,’r’)
plot(validData.averageSpeed,rpm_new,’o’,’Color’,’r’)
plot(t,RPM_Old,’LineWidth’,2,’Color’,’b’)
plot(Boat_speed_kn_old,Motor_speed_rpm_old,’o’,’Color’,’b’)
ylim([0,1500])
ylabel("Engine Speed [rpm]")
xlabel("Boat Velocity [kn]")
legend("New Prop. Power Curve","New Prop. Data",...
    "Old Prop. Power Curve", "Old Prop. Data",’Location’,’southeast’)
grid on

figure(2)
title("Power Absorbed Comparison")
hold on
plot(t,Power_New,’LineWidth’,2,’Color’,’r’)
plot(t,Power_Old,’LineWidth’,2,’Color’,’b’)
plot(t,Power_CFD,’LineWidth’,2,’Color’,’m’)
ylim([0,10])
ylabel("Power [kW]")
xlabel("Boat Velocity [kn]")
legend("New Prop. Power Curve", "Old Prop. Power Curve", "New Prop. CFD" + ...
    " Results","Location","southeast")
grid on

% End of File
```

Bibliography

[1] Betz A. Ergebn aerodyn vers anst. gottingen. 1919.

[2] Betz A. Eine Erweiterung der Schrauben Theorie. Zreits. Flugtech. 1920.

[3] Abdelkhalig Ashraf, Mahmoud Elgendi, and Mohamed Y. E. Selim. Review on validation techniques of blade element momentum method implemented in wind turbines. *IOP Conference Series: Earth and Environmental Science*, 08 2022. doi: 10.1088/1755-1315/1074/1/012008.

[4] Montgomerie B. and Totalförsvarets forskningsinstitut. *Methods for Root Effects, Tip Effects and Extending the Angle of Attack Range to +- 180°, with Application to Aerodynamics for Blades on Wind Turbines and Propellers*. FOI-R. Swedish Defence Research Agency, 2004. URL https://books.google.it/books?id=uJqltwAACAAJ.

[5] WMJ Batten, AS Bahaj, AF Molland, and JR Chaplin. Hydrodynamics of marine current turbines. *Renewable energy*, 31(2):249–256, 2006.

[6] Ole Bergmann, F. Götten, C. Braun, and F. Janser. Comparison and evaluation of blade element methods against RANS simulations and test data. *CEAS Aeronautical Journal*, 13(2):535–557, 2022.

[7] Norbert Bulten, Petra Stoltenkamp, and Judith van Hooijdonk. Efficient propeller designs based on full-scale cfd simulations. 2014. URL https://api.semanticscholar.org/CorpusID:133599858.

[8] Burrill L. C. Calculation of marine propeller performance characteristics. The North East Coast Institution of Engineers and Shipbuilders, Boldec Hall, 1944. URL http://resolver.tudelft.nl/uuid:e6fa1a58-9e71-4a59-a3bc-4ce5c02daef3.

[9] J. S. Carlton. *Marine Propellers and Propulsion*. Elsevier Ltd., 2012. URL https://api.semanticscholar.org/CorpusID:149740974.

[10] Steven C. Chapra and Raymond P. Canale. *Numerical Methods for Engineers*. McGraw-Hill Education, 2015.

[11] H. L. Chung. An enhanced propeller design program based on propeller vortex lattice lifting line theory. 2007. URL https://api.semanticscholar.org/CorpusID:109748688.

[12] V. Dehouck, Mohamed Lateb, Jonathan Sacheau, and Hachimi Fellouah. Application of the blade element momentum theory to design horizontal axis wind turbine blades. *Journal of Solar Energy Engineering-transactions of The Asme*, 140, 2018. URL `https://api.semanticscholar.org/CorpusID:117164661`.

[13] Mark Drela. Xfoil: An analysis and design system for low reynolds number airfoils. In *Low Reynolds Number Aerodynamics: Proceedings of the Conference Notre Dame, Indiana, USA, 5–7 June 1989*, pages 1–12. Springer, 1989.

[14] Giulio Dubbioso, Roberto Muscari, and Andrea Di Mascio. Cfd analysis of propeller performance in oblique flow. 2013. URL `https://api.semanticscholar.org/CorpusID:128355812`.

[15] M. A. Elghorab, A. Abou El-Azm Aly, A. S. Elwetedy, and M. A. Kotb. Open water performance of a marine propeller model using cfd. 2013. URL `https://www.researchgate.net/publication/262004663`.

[16] Branlard Emmanuel. *Wind Turbine Aerodynamics and Vorticity-Based Methods Fundamentals and Recent Applications*. Springer International Publishing, 2018.

[17] Brenden P. Epps, M. Jordan Stanway, and Richard Warren Kimball. Openprop: An open-source design tool for propellers and turbines. 2009. URL `https://api.semanticscholar.org/CorpusID:12070519`.

[18] Brenden P. Epps, Bernard T. Roesler, Richard B. Medvitz, Yeunun Choo, and Jarlath Mcentee. A viscous vortex lattice method for analysis of cross-flow propellers and turbines. *Renewable Energy*, 143:1035–1052, 2019. URL `https://api.semanticscholar.org/CorpusID:191177231`.

[19] Benini Ernesto. Significance of blade element theory in performance prediction of marine propellers. *Significance of blade element theory in performance prediction of marine propellers*, 31(8-9):957–974, 2004.

[20] R. E. Froude. On the part played in propulsion by difference in pressure. pages 390–423, 1889.

[21] W. Froude. On the elementary relation between pitch, slip, and propulsive efficiency. 1878. URL `https://ntrs.nasa.gov/api/citations/19930080719/downloads/19930080719.pdf`.

[22] Bangga Galih. Comparison of Blade Element Method and CFD Simulations of a 10 MW Wind Turbine. *Fluids*, Vol. 3(n.4), 2018.

[23] Knut Erik Teigen Giljarhus. pybemt: An implementation of the blade element momentum theory in python. *J. Open Source Softw.*, 5, 2020. URL `https://api.semanticscholar.org/CorpusID:225318660`.

[24] H. Glauert. *The Elements of aerofoil and airscrew theory*. Cambridge University Press, 1983.

[25] Sydney Goldstein. *On the Vortex theory of screw propellers*. Royal Society, 1929.

[26] Davide Grassi and Stefano Brizzolara. Numerical analysis of propeller performance by lifting surface theory. 2007. URL `https://api.semanticscholar.org/CorpusID:157064070`.

[27] E. L. Houghton and P. W. Carpenter. *Aerodynamics for engineering students*. Butterworth-Heinemann, 2003.

[28] ITTC. Ittc quality system manual, recommended procedures and guidelines - open water test. 2014. URL `https://www.ittc.info/media/8025/75-02-03-021.pdf`.

[29] Jonkman Jason, Emmanuel Branlard, Matthew Hall, Greg Hayman, Andy Platt, and Amy Robertson. Implementation of Substructure Flexibility and Member-Level Load Capabilities for Floating Offshore Wind Turbines in OpenFAST. Golden, CO: National Renewable Energy Laboratory, 2020. URL `https://api.semanticscholar.org/CorpusID:225515389`.

[30] Meng Jin and Xiaogang Yang. A new fixed-point algorithm to solve the blade element momentum equations with high robustness. *Energy Science & Engineering*, 9:1734–1746, 2021. URL `https://api.semanticscholar.org/CorpusID:237645883`.

[31] Eckhardt M. K. and Morgan W. B. A propeller design method. *Trans. Soc. Naval Architects and Marine Engrs.*, Vol. 63:325–374, 1955.

[32] Jeremy Ledoux, Sebastián Riffo, and Julien Salomon. Analysis of the blade element momentum theory. *SIAM J. Appl. Math.*, 81:2596–2621, 2020. URL `https://api.semanticscholar.org/CorpusID:216080655`.

[33] Battisti Lorenzo, Zanne Luca, Castelli Marco Raciti, Bianchini Alessandro, and Brighenti Alessandra. A generalized method to extend airfoil polars over the full range of angles of attack. *Renewable Energy*, 155:862–875, 2020.

[34] Faisal Mahmuddin, Syerly Klara, Husni Sitepu, and Surya Hariyanto. Airfoil lift and drag extrapolation with viterna and montgomerie methods. *Energy Procedia*, 105:811–816, 2017. URL `https://api.semanticscholar.org/CorpusID:114642632`.

[35] Ian Masters, J.C. Chapman, M. R. Willis, and J. A. C. Orme. A robust blade element momentum theory model for tidal stream turbines including tip and hub loss corrections. *Journal of Marine Engineering & Technology*, 10:25–35, 2011. URL `https://api.semanticscholar.org/CorpusID:107245242`.

[36] W. B. Morgan, Vladimir Silović, and Stephen B. Denny. Propeller lifting-surface corrections. 1968. URL `https://api.semanticscholar.org/CorpusID:107599437`.

[37] Andrew Ning. A simple solution method for the blade element momentum equations with guaranteed convergence. *Wind Energy*, 17:1327–1345, 2013. URL https://api.semanticscholar.org/CorpusID:55523098.

[38] Andrew Ning, Greg Hayman, Rick R. Damiani, and Jason M. Jonkman. Development and validation of a new blade element momentum skewed-wake model within aerodyn. 2015. URL https://api.semanticscholar.org/CorpusID:54689246.

[39] Mohamad Ziauddin Parvez, R V Shashank Shankar, and MP Mathew. Methodology to develop propeller using circulation theory: Review and application. *OCEANS 2022 - Chennai*, pages 1–11, 2022. URL https://api.semanticscholar.org/CorpusID:248922887.

[40] Veeranagouda B. Patil, H. R. Purushothama, Ashwathnarayana Manjunatha, and Vijay Kumar Mishra. Performance evaluation of marine propeller using numerical simulation. *Indian Journal of Science and Technology*, 9, 2016. URL https://api.semanticscholar.org/CorpusID:125540606.

[41] Alexander B. Phillips, Stephen R. Turnock, and Maaten E. Furlong. Evaluation of manoeuvring coefficients of a self-propelled ship using a blade element momentum propeller model coupled to a reynolds averaged navier stokes flow solver. *Ocean Engineering*, 36:1217–1225, 2009. URL https://api.semanticscholar.org/CorpusID:4995230.

[42] Ludwig Prandtl and Albert Betz. Vier abhandlungen zur hydrodynamik und aerodynamik. 1927. URL https://api.semanticscholar.org/CorpusID:161095014.

[43] Ataur Rahman. Development of a marine propeller design method based on lifting line theory and lifting surface correction factors. 2015. URL https://api.semanticscholar.org/CorpusID:114951462.

[44] W. J. M. Rankine. On the mechanical principles of the action of propellers. *Transaction of the Institute of Naval Architects*, Volume 6:13–39, 1865.

[45] Manelisi Kagame Rwigema. Propeller blade element momentum theory with vortex wake deflection. 2010. URL https://api.semanticscholar.org/CorpusID:26565325.

[46] Kamaldeep Singh. Blade element analysis and experimental investigation of high solidity wind turbines. 2014. URL https://api.semanticscholar.org/CorpusID:218728759.

[47] Brizzolara Stefano, Grassi Davide, and Tincani Emilio P. Design Method for Contra-Rotating Propellers for High-Speed Crafts: Revising the Original Lerbs Theory in a Modern Perspective. *The International Journal of Rotating Machinery*, 2012:1–18, 2012. doi: 10.1155/2012/408135.

[48] Thomas Stoye. Propeller design and propulsion concepts for ship operation in off-design conditions. 2011. URL https://api.semanticscholar.org/CorpusID:38267496.

[49] A. J. Tachmindji and A. B. Milam. The calculation of the circulation distribution for propellers with finite hub having three, four, five and six blades. *International shipbuilding progress*, Volume 4:467–477, 1957. URL https://api.semanticscholar.org/CorpusID:117381805.

[50] Larry A. Viterna and Robert D. Corrigan. Fixed pitch rotor performance of large horizontal axis wind turbines. 1982. URL https://api.semanticscholar.org/CorpusID:109053586.

[51] Hadi Winarto. Bemt algorithm for the prediction of the performance of arbitrary propellers. 2004. URL https://mirror.unpad.ac.id/orari/library/library-non-ict/aero/docs/CoEAL%20report%20BEMT.pdf.