UNIVERSITÁ DEGLI STUDI DI GENOVA
SCUOLA POLITECNICA
DIME
Dipartimento di Ingegneria Meccanica, Energetica, Gestionale e dei Trasporti

TESI DI LAUREA MAGISTRALE
IN
INGEGNERIA MECCANICA - ENERGIA E AERONAUTICA

Implementation of the Actuator Disk model in CEASIOMpy for the simulation of propeller aircraft

Relatori:
Chiar.^{mo} Prof. Ing. Alessandro Bottaro
Dott. Ing. Jan B. Vos

Correlatore:
Ing. Aidan Jungo

Allievo:
Giacomo Benedetti

Marzo 2023

Implementazione del modello del disco attuatore in CEASIOMpy per la simulazione di velivoli con elica

Sommario

Nella tesi è presentata l'implementazione di un programma in Python per il calcolo del disco attuatore delle eliche degli aerei. Questo programma è stato sviluppato a partire da uno script Python esistente, modificato in modo da poterlo aggiungere a CEASIOMpy, un ambiente Python per la progettazione di aerei. Grazie a questa nuova funzionalità, CEASIOMpy può ora essere utilizzato per simulare velivoli a elica.

Una volta creato il programma, è stato sottoposto a una serie di test per validarne il corretto funzionamento. I risultati sono stati confrontati con i dati di altre simulazioni numeriche già convalidate ed è stato dimostrato che il software forniva risultati affidabili e abbastanza accurati.

Il software è stato poi testato per diversi casi più complessi. Questi test hanno confermato l'affidabilità e l'efficienza del programma e hanno dimostrato che può essere utilizzato per eseguire calcoli per una prima progettazione.

In conclusione, questo lavoro ha portato alla creazione di un programma che può essere utilizzato per eseguire calcoli di dischi attuatori per eliche di aeromobili, aggiungendo così ulteriori funzionalità e semplificando l'integrazione con gli altri componenti dell'ambiente CEASIOMpy (in particolare l'interazione con lo standard CPACS). CEASIOMpy, e in particolare questo strumento, sarà poi utilizzato per eseguire simulazioni nell'ambito del progetto di ricerca europeo COLOSSUS (Collaborative System of Systems Exploration of Aviation Products, Services and Business Models).

Implementation of the Actuator Disk model in CEASIOMpy for the simulation of propeller aircraft

Abstract

This thesis presents the implementation of a program written in Python for the calculation of the actuator disc of aircraft propellers. This program was developed from an existing python script and was added to CEASIOMpy, a Python environment for aircraft design. With this new functionality, CEASIOMpy can now be used to simulate propeller aircraft.

Once the program had been created, it was subjected to a series of small tests to validate its correct functioning. The results were compared with data from other numerical simulations that had already been validated, and it was shown that the software provided reliable and accurate results.

The software was then tested for several complex cases, covering a wide range of propeller and aircraft configurations. These tests confirmed the reliability and efficiency of the program and showed that it could be used to perform an initial design calculation.

In conclusion, this work has led to the creation of a program that can be used to perform actuator disc calculations for aircraft propellers, thus adding additional functionality and simplifying the integration with the other components of the CEASIOMpy environment (in particular the interaction with the CPACS standard). CEASIOMpy and in particular this tool will then be used to perform simulations within the European research project COLOSSUS (Collaborative System of Systems Exploration of Aviation Products, Services and Business Models).

Ringraziamenti

Si conclude dopo 5 anni il percorso di studi, lo ricorderò sempre come un periodo felice della vita, fatto si di alti e bassi, ma che ha contribuito a rendermi ciò che sono ora. Un ringraziamento speciale va al Prof. Bottaro che mi ha dato l'opportunità di svolgere la tesi all'estero, dove ho potuto fare una bellissima esperienza formativa, la quale mi ha permesso di conoscere diverse persone, tra cui Jan, Alain e Aiden che mi hanno accolto nel loro ufficio e mi hanno fatto sentire a casa.

Un doveroso ringraziamento va sicuramente alla mia famiglia, Alice, Marcella, Alessandro, ma anche i miei nonni Rosa e Guido che mi hanno sempre sostenuto e mi hanno dato la possibilità di studiare e di poter raggiungere questo obiettivo. Obiettivo che non avrei mai raggiunto senza il mio fedele compagno di studi Francesco, era gennaio 2018 quando abbiamo iniziato a studiare insieme per l'esame di termodinamica (isobare e isocore sul piano T-s), da lì in poi è storia. Vorrei poi ringraziare Chiara, la mia forza, che è sempre stata al mio fianco, così come i miei compagni di università Andrea, Matteo; con una menzione particolare per Edoardo e Alberto, con i quali ho condiviso 6 mesi di "vacanza" in Svizzera. Grazie anche a Guido e Maurizio, compagni di università, ma soprattutto grandi compagni di serata.

Un ringraziamento anche al mio zio ingegnere Emilio, che con la sua saggezza e la sua passione mi è sempre stato vicino dispensando consigli utili.

Un ringraziamento a tutti i miei amici, perchè in ogni caso "la felicità è reale solo se è condivisa".

Comincia ora un nuovo capitolo della vita

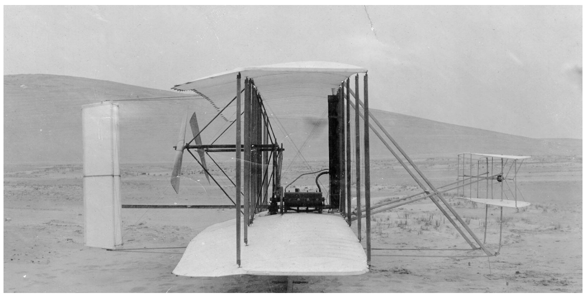
Acknowledgements

The research presented in this paper has been performed in the framework of the Colossus project (Collaborative System of Systems Exploration of Aviation Products, Services & Business models) and has received funding from the European Union Horizon Europe Programme under grant agreement no 101097120. The Swiss participation in the Colossus project was supported by the Swiss State Secretariat for Education, Research and Innovation (SERI) under contract number 22.00609.

Contents

1 Introduction

Over the years, propeller engines have played a central role in aviation. From the first piston engines, the *Wright Flyer*, to today's turboprops, which are widely used in cargo transport, to the futuristic electric propeller engines, the *X-57 Maxwell*, which are being studied to try to decarbonise aviation.


(a) *Wright Flyer*

(b) *NASA X-57 Maxwell*

Figure 1.1: *The first piston prop engine plane and the futuristic electric propeller engine plane*

As a result, each part of the engine has undergone continuous development, with particular attention paid to the propellers and their performance. To do this, it is important to understand and study the different theories: *Momentum Theory* and *Blade Element Theory.*
These models are very useful in describing the characteristics of the propeller and how to get the right amount of thrust. Thrust is an engine parameter of great interest for the CFD simulation of the propeller.

Nowadays CFD simulations are very important to save time (if they are done correctly) and to simplify the design process, so in this work different Eulerian calculations are performed to study the behaviour of the propeller and its integration with the aircraft. The objective is not only to perform numerical simulations, but another important objective is to add a calculation module to the *CEASIOMpy* environment [14]. CEASIOMpy is an *"open source conceptual aircraft design environment"* written in Python, available on *GitHub*, developed at *CFS Engineering* (Computational Fluid and Structure Engineering) [25] in collaboration with *Airinnova* [24]. Throughout the years it has been improved and maintained thanks to these two companies.

2 Theory

This chapter recalled briefly the thermodynamics fundamentals of the turbogas cycle with particular attention to the turboprop engine, furthermore, an analysis of the different methods for studying propellers is carried out.

2.1 Turboprop

Turboprops are engines with a propeller moved by a turbine which is activated by a compressor to self-feeding, so the power provided by the exhausted gases is very low as opposed to a turbofan, the most common engine for commercial flight; hence most of the thrust is provided by the propeller. In the beginning, the propeller was moved by an internal combustion engine, nowadays it is still possible to find some of these engines for regional jets and small tourist aeroplanes.

Turboprops are the most efficient engines at low flight speeds (theoretically below 725 km/h) and usually are used on small subsonic planes and large military aircraft, that's because these engines are cheaper than turbofan and they can provide more power with better efficiency, which means shorter runways and better performance at lower velocities.

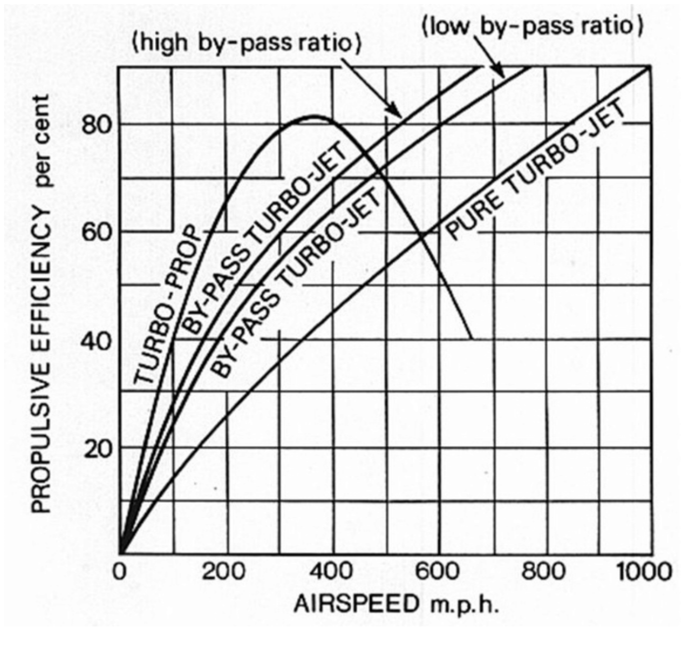


Figure 2.1: *Propulsive efficiency comparison*

A turboprop engine consists of: intake, gearbox, compressor, combustor, turbine and propelling nozzle.

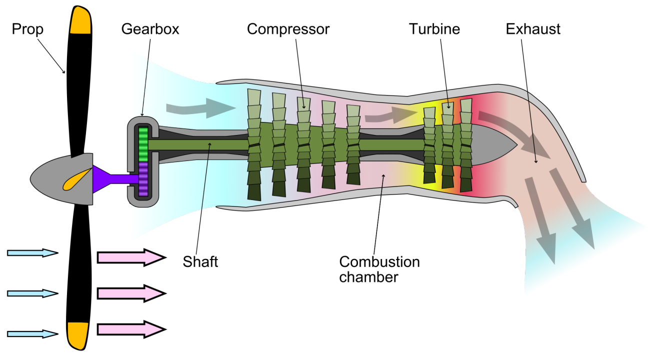


Figure 2.2: *Propeller scheme*

The propeller creates a pressure gradient, which leads to an increase in velocity, thus the propeller generates a thrust. The exhaust gases of the engine create also a thrust, and it is possible to define the total thrust as:

$$F_{total} = F_{propeller} + F_{core} \tag{2.1}$$

deriving the thrust generated by the core, F_{core}, is straightforward:

$$F_{core} = (\dot{m}_0 + \dot{m}_f)V_9 - \dot{m}_0 V_0 + (p_9 - p_0)A_9 \tag{2.2}$$

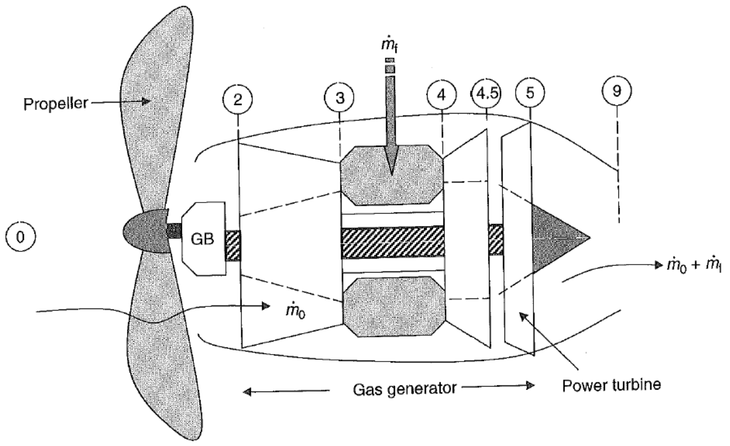
where $\dot{m}_0$ is the air mass flow at the engine inlet and $\dot{m}_f$ is the fuel mass flow, with V_0 being the velocity at the engine inlet and V_9 being the velocity at the nozzle outlet.

For simplicity this equation can be written as:

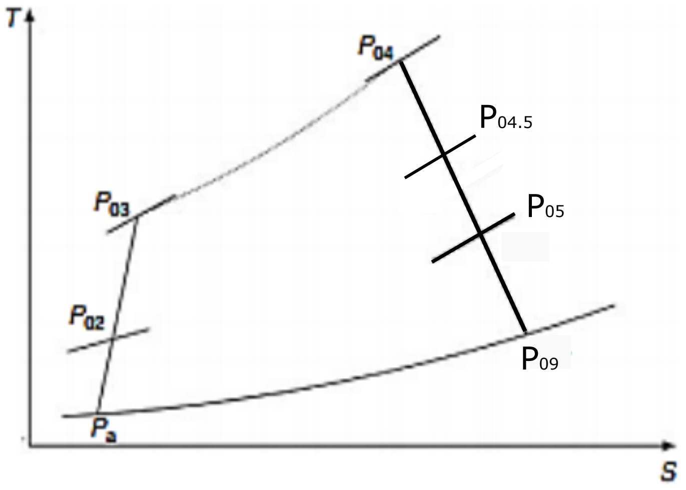
$$F_{core} \simeq (\dot{m}_0 + \dot{m}_f)V_9 - \dot{m}_0 V_0 \tag{2.3}$$

because the thrust coming from the pressure difference between the inlet and the outlet is negligible.

$F_{propeller}$ is not as easy to evaluate as F_{core}. The thrust given by the propeller is linked to the diameter of the blades, so the more power you need, the bigger they have to be, but at the same time they can't be too big, otherwise, the velocity at the tip would reach the sonic regime and so a loss of energy can happen.
In addition, to avoid sonic speeds at the tip of the blades, the rotational velocity of the blades is limited. In fact, usually, the turbine rotates with a velocity higher than the propeller, so it is necessary to use a gearbox to decouple the two parts. Doing this adds another loss because the efficiency of a gearbox must be taken into account, but it permits to have different rotational velocities of the two machines.



(a) *Turboprop engine*



(b) *Thermodynamic cycle of a turboprop*

Figure 2.3: *Turboprop cycle*

As can be seen from the thermodynamic cycle in Fig. 2.3, the propeller has an important role, it accelerates the flow, which also means an increase of enthalpy, so the compressor has to deliver a lower power. The effect of the initial acceleration is at the same time problematic, considering that the flow will have also a swirl that can cause an additional loss.

In describing the thermodynamic cycle, it is also essential to pay attention to the thrust splitting between the turbine (propeller) and the nozzle. As already mentioned, most of the thrust is delivered by the propeller, but it is possible to find an optimal condition. Looking at Fig. 2.4, the optimal condition can be found by introducing α as a ratio between the lower power of the turbine and the overall power (LPT plus nozzle).

$$\alpha = \frac{h_{t4.5} - h_{t5s}}{h_{t4.5} - h_{9s}} = \frac{\frac{P_T/\dot{m}}{\eta_T}}{P_{tot}/\dot{m}} \tag{2.4}$$

The high power turbine is not considered in Eq. (2.4) because the power delivered by it is used to feed the compressor.

The goal is elaborate Eq. (2.4) to obtain an expression which permits the optimal subdivision between the turbine and the nozzle.

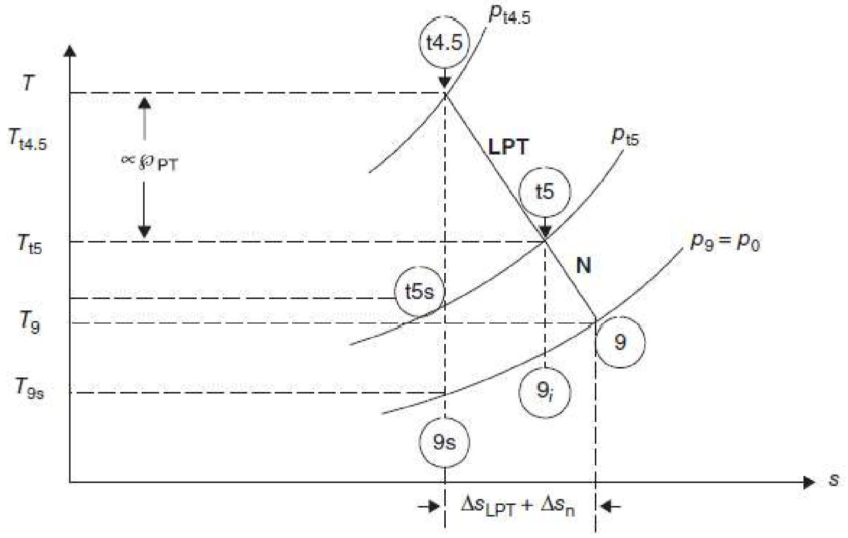


Figure 2.4: *Power division between turbine and nozzle*

The process of expansion happens between two isobars, $p_{t4.5}$ and p_9, which is equal to the static free stream pressure, as it is possible to observe in Fig. 2.4. Considering the turbine is divided into two parts, the high pressure and low pressure part, $p_{t4.5}$ is the pressure outside of the HPT, which the low pressure part can convert, the expansion is until p_{t5} which is the inlet of the nozzle; the optimal position of p_{t5} is unknown. If more work is to be delivered to the turbine, more thrust is given to the propeller instead of the nozzle.

To obtain the optimal condition, it is useful acting on the definition of α, in particular, the aim is to obtain the α_{opt} such that the thrust is maximum. Elaborating Eq. (2.4) it is possible to obtain:

$$\alpha_{opt} = 1 - \frac{\eta_n}{(\eta_{PT}\eta_{LPT}\eta_{gb}\eta_{prop})^2}\frac{\gamma_c - 1}{2}\frac{M_0^2}{\tau_{HTP}\tau_\lambda\left[1-\left(\frac{p_9/p_0}{\pi_r\pi_d\pi_c\pi_b\pi_{HTP}}\right)^{\frac{\gamma_t-1}{\gamma_t}}\right]} \tag{2.5}$$

in which π indicates the pressure ratios, i.e. $\pi_r = \frac{p_{t0}}{p_0}$, $\pi_r = \frac{p_{t2}}{p_{t0}}$ and so on, while τ indicate the temperature ratios, like $\tau_{HTP} = \frac{T_{t4.5}}{T_4}$.

Eq. (2.5) optimize the thrust, dividing the enthalpy jump between the nozzle and turbine.
The thrust delivered by the nozzle is only a fraction, usually amounting to between 10% and 20%, most of the power of the turbine is used to move the propeller.

To describe the behavior of the turboprop, and in particular, how the thrust is provided by the propeller, it is useful to explore two different theories:

- Actuator disk also called momentum theory
- Blade element theory

The first one is a mono-dimensional theory which considers only axial velocity, however the blade element theory is a 2D theory, that overlook radial velocity.

2.2 Actuator disk

The study of a propeller is very complicated; to do a complete investigation, it is mandatory to perform a viscous, unsteady, and compressible calculation, but by simplifying the problem it is possible to obtain a reasonable result without much effort.
In the actuator disk model, blades are neglected and the rotor is modeled like a disk of infinitesimal thickness which accelerates the flow in the axial direction. This means that the propeller is seen as a discontinuity and the discontinuity induces a jump in pressure.

2.2.1 Simple momentum theory

To apply this theory some hypotheses have to be formulated, the most important one is that the tangential velocity variation can be neglected, so it exists only a velocity that is normal to the infinitesimal thickness disk of a diameter D.
Moreover, the flow across the disk has to be:

- Steady
- Inviscid
- Incompressible

The flow varies its energy as it passes through the disk, in fact, the disk gives an energy variation, that for the conservation of the energy, has to be equal to the power spent.

To calculate the thrust, a control volume has to be considered.

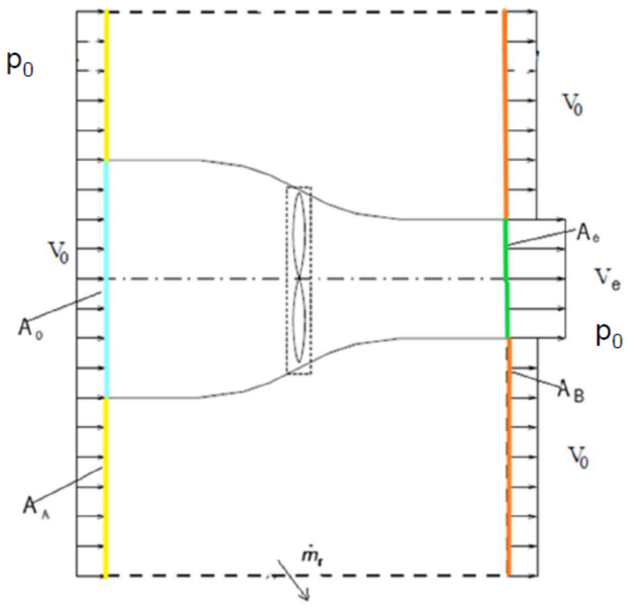


Figure 2.5: *Control volume of turboprop*

Referring to the Fig. 2.5, if we consider only an isolated propeller, the following values can be defined: p_0 is the free stream pressure, V_0 is the free stream velocity and V_e is the velocity downstream of the propeller. Different areas are highlighted in the figure, in particular, there are:

- A_0 that is the inlet area of the capturing streamtube
- A_A is the inlet area outside of the streamtube, that corresponds to $A_{inlet} - A_0$
- A_e is the outlet area of the capturing streamtube
- A_B is the outlet area outside of the capturing streamtube, which corresponds to $A_{outlet} - A_e$

There is also $\dot{m}_r$, which is the mass flow rate which has to be taken into account in order to have the conservation of the total mass flow rate of the control volume.

Doing a balance of forces on the control volume:

$$\sum F_s + \sum F_b = \cancel{\frac{\partial}{\partial t}\int_{CV} V\rho \cdot dV} + \int_{CV} V\rho V \cdot dA \tag{2.6}$$

Looking at each element of Eq. (2.6) it is possible to obtain a simplified equation, in particular, the first integral can be removed as the model assumes that the flow is steady. Concerning the second integral, it is possible to write it as:

$$\begin{aligned} &F_{prop} + p_0 A_0 + p_0 A_A - p_0 A_e - p_0 A_B \\ &= -\rho_0 v_0^2 A_0 - \rho v_0^2 A_A + \rho_0 v_0^2 A_B + \rho_0 v_e^2 A_e + \dot{m}_r v_0 \end{aligned} \tag{2.7}$$

The way the control volume was constructed, it follows that $A_0 + A_A = A_e + A_B$, hence:

$$F_{prop} = -\rho_0 v_0^2 A_0 - \rho v_0^2 A_A + \rho_0 v_0^2 A_B + \rho_0 v_e^2 A_e + \dot{m}_r v_0 \tag{2.8}$$

This equation can be further simplified if two control volumes are considered: VC1, the big control volume and VC2, the control volume of the streamtube; hence putting the two equations together:

$$\begin{cases} VC1 : \dot{m}_0 + \rho_0 v_0 A_A = \rho_0 v_0 A_B + \dot{m}_e + \dot{m}_r \\ VC2 : \dot{m}_0 = \dot{m}_e = \dot{m}_p \end{cases} \tag{2.9}$$

gives as result:

$$\dot{m}_r = \rho_0 v_0 A_A - \rho_0 v_0 A_B \tag{2.10}$$

If Eq. (2.10) is replaced in Eq. (2.8), the following equation is obtained

$$F_{prop} = -\rho_0 v_e^2 A_e - \rho v_0^2 A_0 = \dot{m}_e v_e - \dot{m}_0 v_0 = \dot{m}_p (v_e - v_0) \tag{2.11}$$

The propulsive force is found from Eq. (2.11), and as it is possible to see above, the thrust is a function of the flow rate through the propeller, $\dot{m}_p$, and the difference between inlet and outlet velocities. The problem of this expression is v_e, which is not easy to evaluate.

When only the capturing streamtube is taken as a control volume it is possible to calculate the velocity at the outlet, v_e.

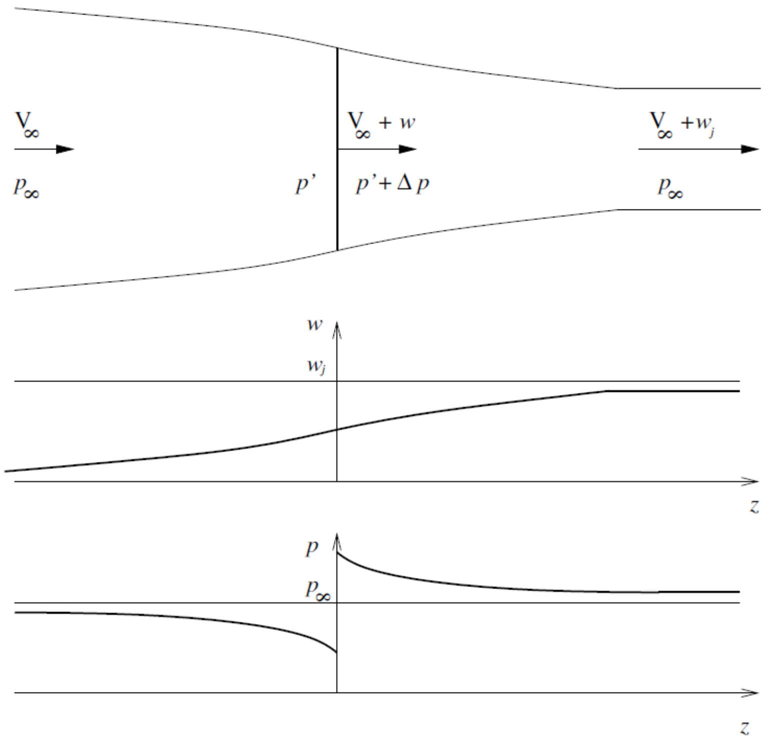


Figure 2.6: *Actuator disk theory*

Inside the control volume different velocities can be defined, in particular: V_0 is meant V_∞, so the free stream velocity, $V_p = V_\infty + w$ which is the velocity in correspondence of the propeller and $V_e = V_\infty + w_j$ the velocity at the exit of the control

volume. w is the local velocity induced by the propeller, in Fig. 2.6 it is possible to see the w trend, the curve reaches the asymptote given by w_j, which is the induced velocity at the end of the control volume.
The control volume has a convergent shape, this because the axial velocity tends to increasing from upstream to downstream of the rotor and at the same time the density is constant, so for the conservation of continuity, streamlines have to be convergent.
The pressure remains constant except at the actuator disk level, where there is a localized pressure drop due to the presence of the rotor.

Considering the portion of the convergent volume between the external streamline, the continuity equation can be defined as:

$$A_0 \rho V_0 = A_e \rho V_e \tag{2.12}$$

To calculate the pressure drop, it is possible to apply two times *Bernoulli equation*, the first one between upstream inlet and the propeller:

$$P_p^- = P_0 + \frac{\rho_0 V_0^2}{2} - \frac{\rho_0 V_p^2}{2} \tag{2.13}$$

The second time the Bernoulli equation is applied between the propeller and downstream outlet:

$$P_p^+ = P_0 + \frac{\rho_0 V_e^2}{2} - \frac{\rho_0 V_p^2}{2} \tag{2.14}$$

Putting together Eq. (2.13) and Eq. (2.14) the result is:

$$P_p^+ - P_p^- = \Delta P_p = \frac{\rho_0}{2} \cdot (V_e^2 - V_0^2) \tag{2.15}$$

with V_e the outlet velocity and V_0 the inlet velocity.

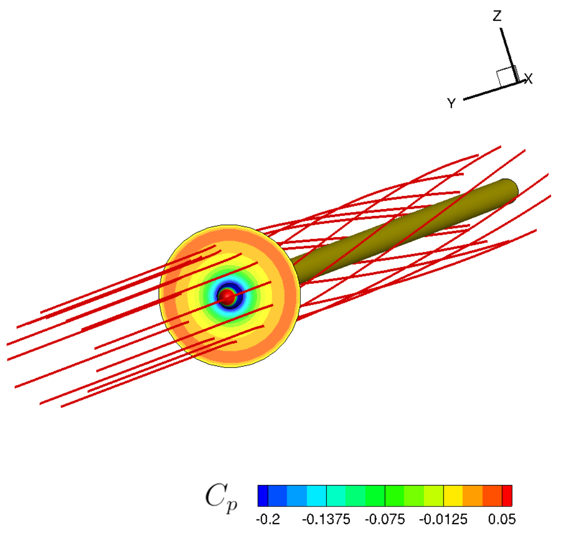


Figure 2.7: *Actuator disk*

Combining two different expressions for the force given by the propeller

$$\begin{cases} F = A_p \Delta P_p = \frac{\rho_0 A_p}{2}(V_e^2 - V_0^2) \\ F = \dot{m}_p (V_e - V_0) = \rho_0 V_p A_p (V_e - V_0) \end{cases} \tag{2.16}$$

Recalling that $\dot{m}_p = \rho_0 V_p A_p$ and performing some algebra, it is possible to obtain an estimation of the velocity at the propeller disk:

$$V_p = (V_e + V_0)/2 \tag{2.17}$$

This is the average between the inlet and outlet velocity, this means that in the first part of the streamtube, the flow undergoes half of the total acceleration.

V_e is not yet known, to obtain it, thrust and power have to be correlated

$$T = \rho_0 V_p A_p (V_e - V_0) = \rho_0 A_p \left(V_0 + \frac{\Delta V_e}{2} \right) \Delta V_e \tag{2.18}$$

$$P = \rho_0 V_p A_p (V_e - V_0) V_p = \rho_0 A_p \left(V_0 + \frac{\Delta V_e}{2} \right)^2 \Delta V_e \tag{2.19}$$

Usually the power is given, thus Eq. (2.19) allows to obtain ΔV_e, hence V_e.

Once the velocity is known is it possible to obtain the thrust that the propeller gives and consequentially propulsive efficiency η_p.

$$\eta_p = \frac{Propulsive\ power}{Rate\ of\ kinetic\ energy} = \frac{2}{1 + V_e/V_0} \tag{2.20}$$

The efficiency is a function of the ratio of volume control exit velocity and free stream velocity, so if $V_e/V_0 \to 1$, then $\eta \to 1$, but at the same time $V_e \simeq V_0$ and so there is no motion.

An useful coefficient, which is important to introduce, is the *advance ratio*, which explains the operation of the propeller:

$$J = \frac{V_\infty}{nD} \tag{2.21}$$

Where n is the propeller rotational speed in revolutions per minute (rpm) and D is the propeller diameter in metres.

Often adimensionalised formulae are used to better visualize and understand the results. The Renard definition is in particular very useful:

$$C_T = \frac{T}{\rho n^2 D^4} \tag{2.22}$$

$$C_P = \frac{P}{\rho n^3 D^5} \tag{2.23}$$

Optimal load distribution

Up to now, it has been assumed that the velocity on the propeller, $V_e = V_\infty + w$, is uniform, but if this assumption is removed, it can be considered that the speed w is a function of the radius, in particular $w = w(r)$, this is assumed to find the *optimal load distribution.*
Thus, the local thrust can be written taking into account the radial distribution of the velocity, by calling $a = V_e/V_0$, it follows that:

$$dT = 4\pi r \rho V_\infty^2 (1+a) a dr \tag{2.24}$$

To obtain the total thrust distribution along the blade Eq. (2.24) has to be integrated and then, the equation became:

$$T = 4\pi \rho V_\infty^2 \int_0^R (1+a) a r dr \tag{2.25}$$

Similarly for power:

$$dP = 4\pi r \rho V_\infty^3 (1+a)^2 a dr \tag{2.26}$$

$$P = 4\pi \rho V_\infty^3 \int_0^R (1+a)^2 a r dr \tag{2.27}$$

This is the differential simple actuator disk theory, which gives a radial distribution of the quantities. To obtain the optimal distribution, a *minimum constrained problem* has to be solved in order to find the maximum efficiency. From Eq. (2.20) it can be seen that the efficiency is a function of the ratio between V_e and V_0, to maximize it is necessary to find an optimal distribution of this ratio, in order to have the minimum power required to obtain a defined thrust. The condition can be solved by imposing $I = P + \Lambda T$ [7], where Λ is a Lagrange multiplier, T and P can be expressed as:

$$T = \int_0^R F[a(r), r]dr \tag{2.28}$$

$$P = \int_0^R G[a(r), r]dr \tag{2.29}$$

The minimum can be obtained by solving the following equation:

$$\frac{\partial G}{\partial a} + \Lambda \frac{\partial F}{\partial a} = 0 \tag{2.30}$$

Eq. (2.30) is satisfied only if $a(r) = constant$, this means that once the thrust is known, the efficiency of the propeller is the maximum if the load is *uniformly distributed*, that it can be observed if the result of Eq. (2.30) is derived:

$$\frac{dT}{dA} = \frac{1}{2\pi r}\frac{dT}{dr} = \rho V_\infty (1 + a)a \tag{2.31}$$

Looking at Eq. (2.31), we can assume that ρ is constant and V_∞ too, hence $\frac{dT}{dA}$ is constant if the axial interference factor is constant.

2.2.2 General momentum theory

When a propeller rotates, it creates a rotational motion of the fluid downstream of the propeller. This rotational motion, which is also known as swirl, was not considered in the simple theory. However, in more advanced theories, the rotational velocity induced by the propeller is taken into account. This is important because the rotational velocity can affect the flow field downstream of the propeller and can have an impact on the propeller's performance.

On the other hand, the radial velocity of the fluid, which is the component of the velocity in the direction perpendicular to the axis of rotation of the propeller, is still not considered. This is because the radial velocity is usually much smaller than the axial and tangential velocities and is often neglected in the analysis of propeller flow.

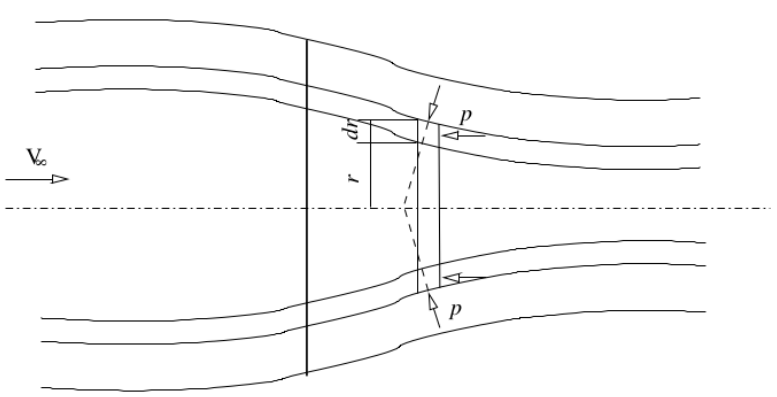


Figure 2.8: *General momentum theory*

Looking at a 3D control volume in Fig. 2.8 and by making the integral momentum balance on $dV = 2\pi r dr dz$, a part between r and $r + dr$, with dz the infinitesimal length, it is possible to obtain:

$$\int_S \rho(\underline{r} \times \underline{V})\underline{V} \cdot \underline{n} dS + \int_S \underline{r} \times p\underline{n} dS = 0 \tag{2.32}$$

where $\underline{r}$ is the radius vector and S is the surface area of the stream tube.
In relation to the propeller axis, the pressure field is symmetric, so the only non-zero elements are given by the momentum convective flux through the two surfaces, perpendicular to the disk axis. Note that $d\dot{m}$ does not vary, the following equation can be derived:

$$-vrd\dot{m} + \left[vr + \frac{\partial}{\partial z}(vr)dz\right] d\dot{m} = 0 \tag{2.33}$$

That means:

$$\frac{\partial}{\partial z}(vr) = 0 \tag{2.34}$$

Integrating this equation and considering that $v = \omega r$, the result is:

$$vr = \omega r^2 = constant \tag{2.35}$$

The angular velocity of a particle, ω, upstream the propeller has to be zero, because there is nothing in the flow which puts it into rotation, while downstream of the disk it has to be different from zero, so through the disk, the angular velocity has to change. This change is due to the action of the torque dQ, given by the propeller.

$$dQ = \omega r^2 d\dot{m} = 4\pi r^3 \rho V_\infty (1 + a)a' dr \tag{2.36}$$

where a' is the rotational interference, that is $a' = \frac{\omega}{2\Omega}$ (where Ω is the angular velocity). This coefficient provides a loss of energy, in fact, there is a kinetic energy loss due to the rotation of the flow around propeller axis.
The rotational effect can't be neglected in the power expression because a variation in kinetic energy has to be considered, and the power expression becomes:

$$P = T(V_\infty + \omega) + \dot{m}\frac{D^2}{4}\Omega^2 a'^2 \tag{2.37}$$

but P can also be obtained by integrating Eq. (2.36):

$$P = \int_0^R \Omega dQ = \dot{m}a'\omega^2\frac{D^2}{4} \tag{2.38}$$

Combining Eq. (2.37) and Eq. (2.38) the ideal efficiency can be derived, and if a simple calculation is performed, the efficiency, as a ratio of interference, can be obtained.

$$\eta = \frac{TV_\infty}{P} = \frac{1 - a'}{1 + a} \tag{2.39}$$

This expression highlights that both axial and rotational interference causes a loss of efficiency.
Also in the general theory, as was done with the simple theory, it is possible to find the optimal condition solving the same minimal constrained problem: $I = P + \Lambda T$. In this case, the solution found is a function of both the axial interference and the rotational interference factor.

2.3 Blade element theory

Blade element theory or BET is a method to calculate the behavior of the propeller. It is a 2D theory, so in addition to V_∞, the tangential velocity Ωr has to be considered.

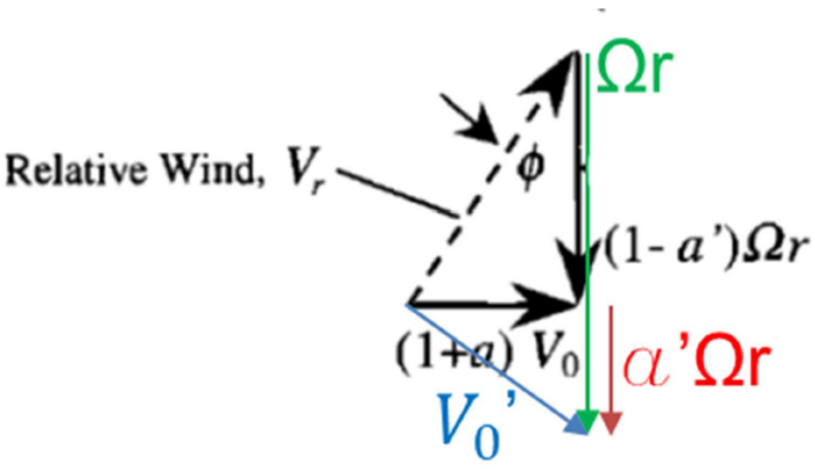


Figure 2.9: *Velocity triangle for blade element theory*

As for the general momentum theory it is necessary to take into account a and a', the *axial interference factor* and the *rotational interference factor* respectively. The first factor is used because the flow arrives with a pre-compression, that is actuated in the capturing streamtube, so the velocity viewed from the blade will be greater than the free stream velocity. Instead a', that is smaller than a (in particular it is about 1/100 of a), considers the fact that there is a pre-rotation due to the drag effect, so that the flow does not arrive axially.

In this model, the blade is discretized into several small elements, and for simplicity the interaction between different elements is not considered.

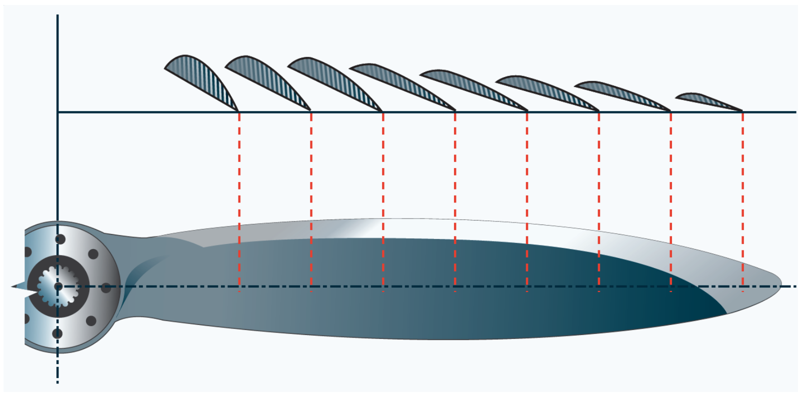


Figure 2.10: *Blade discretization*

It should be noted that the chord and twist variation of the blades is critical to optimising the performance of the propeller. This is because the blade design directly affects the aerodynamic efficiency of the propeller and therefore its ability to convert power into thrust. This theory incorporates these geometric parameters to provide a more accurate prediction of the propeller performance, resulting in a more efficient and effective design compared to the *Actuator Disk Theory*, which provides a simplified approach to analysis that may not always capture the subtleties of propeller behaviour. It is therefore important to consider all geometrical parameters to gain a more accurate and comprehensive understanding of the propeller's operation.

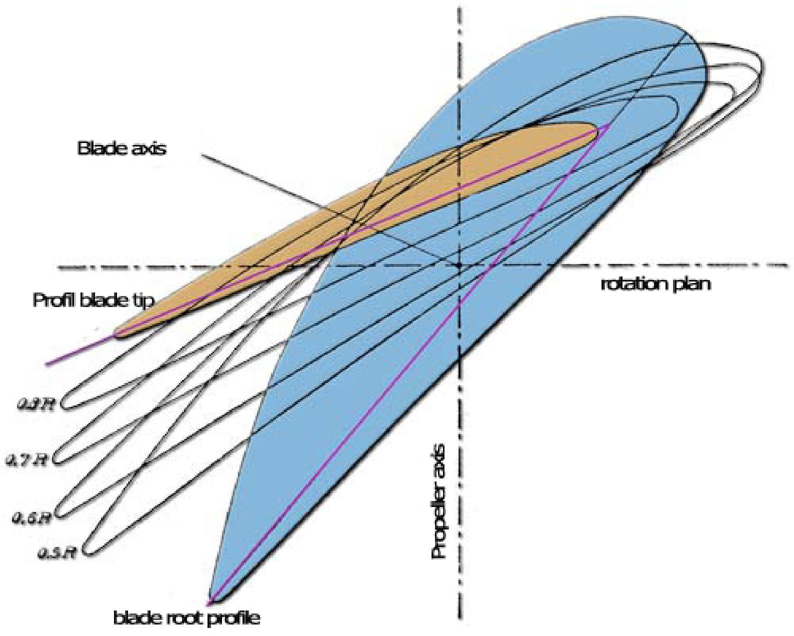


Figure 2.11: *Twist variation*

Looking at the force acting on each element of the blade, that means on every airfoil, it can be separated into:

- Torque, that is a tangential force
- Thrust
- Lift
- Drag

These forces, when paired, torque with thrust and lift with drag, give a resultant which is the *total airfoil force*, dR.

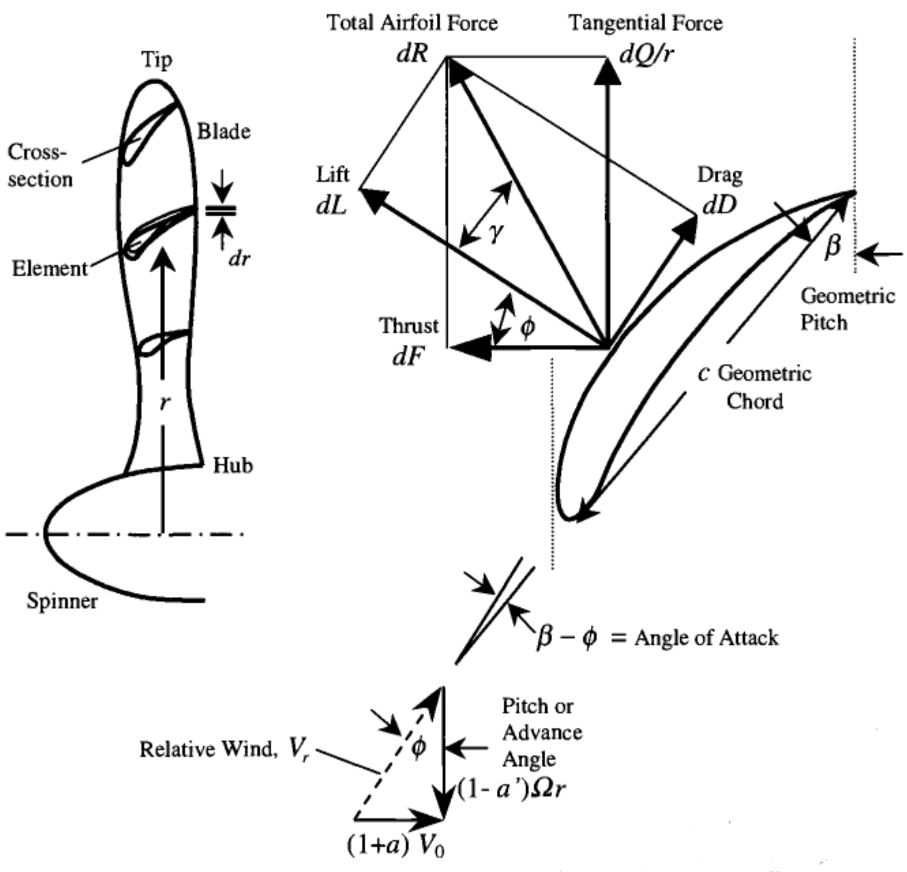


Figure 2.12: *Blade Forces*

If all these forces are known, the propeller performance can be calculated.
As can be seen from Fig. 2.12, a velocity triangle can be constructed, the meaningful velocities are: V_0 which is the *free stream velocity*, V_{tan} which is the *tangential velocity*, equal to Ωr and V_r that is the *relative velocity* also called *local velocity*. V_0 and V_{tan} are corrected considering the inferential factors.
The angles that appear in the Fig. 2.12 are geometric and flow angles, in particular: β is the twist angle, so is imposed by the blade designer, ϕ is the flow angle, and $\alpha = \beta - \phi$ is the *angle of attack*.

ϕ be defined as:

$$\phi = \arctan\left(\frac{(1+a)V_0}{(1-a')\Omega r}\right) \tag{2.40}$$

Another important angle is γ, which is the angle between the resultant force on the airfoil and the lift force.

$$\gamma = \arctan\left(\frac{D}{L}\right) = \arctan\left(\frac{C_D}{C_L}\right) \tag{2.41}$$

Typically, propeller blades are designed with a decreasing geometrical angle β, as a result of the increase in local rotational velocity Ωdr which is lower at the hub and higher at the tip. This decrease in β is required to prevent the flow from becoming supersonic at the tip. Consequently, the flow angle ϕ also needs to decrease. It is important to note that a high angle of attack α can lead to poor blade performance, hence good efficiency requires that the blade is designed with an appropriate angle of attack.

To calculate the thrust generated by each blade of the propeller, the blade needs to be divided into multiple elements, or airfoils, as mentioned previously. The thrust generated by each blade can then be approximated by summing the forces generated by each element.

This can be expressed using the equation:

$$dF = dR\cos(\phi+\gamma) \tag{2.42}$$

However, since the total airfoil force, dR, is not known in advance, it can be rewritten as the lift force, dL, divided by the cosine of the angle γ, according to the force decomposition of Fig. 2.12. This leads to the following equation:

$$dF = \frac{dL}{\cos\gamma}\cos(\phi+\gamma) \tag{2.43}$$

The lift force is defined in the following way:

$$dL = \frac{1}{2}\rho_0 C_L V_r^2 cdr = \frac{1}{2}\left(\frac{(1+a)V_0}{\sin\phi}\right)^2 C_L cdr \tag{2.44}$$

Substitution of Eq. (2.44) in Eq. (2.43) and performing some algebra, the final expression of dF is obtained:

$$dF = dL\frac{\cos(\phi+\gamma)}{\cos\gamma} = \frac{\rho_0\{(1+a)V_0\}^2}{2\sin^2\phi}\frac{cos(\phi+\gamma)}{\cos\gamma}C_L cdr \tag{2.45}$$

Integrating dF along the radius, the total thrust acting on the blade is obtained.

$$F = \int_{r_{hub}}^{r_{tip}} dF \tag{2.46}$$

A similar procedure is followed to calculate the torque force on the airfoil:

$$dQ = rdF_\theta = rdR\sin(\phi+\gamma) = r\frac{dL}{\cos\gamma}\sin(\phi+\gamma) \tag{2.47}$$

Also in this case the lift definition of Eq. (2.44) is replaced inside the equation in order to have:

$$dQ = rdL\frac{\sin(\phi+\gamma)}{\cos\gamma} = \frac{\rho_0\{(1+a)V_0)^2\}}{2\sin^2\phi}\frac{\sin(\phi+\gamma)}{\cos\gamma}C_L cdr \tag{2.48}$$

To get the distribution of torque along the blade Eq. (2.48) has to be integrated from the hub radius to the tip radius:

$$Q = \int_{r_{hub}}^{r_{tip}} dQ \tag{2.49}$$

Thrust and torque distribution have similar equations, because they are linked by the angles ϕ and γ to the total force that acts on the blade.

As was done for the *actuator disk theory*, for them*blade element theory* it is also possible to find the optimal condition. In this case, the distributions of the interference factors are only an approximation.

2.4 Computational fluid dynamics

Computational Fluid Dynamics or CFD is a discipline in which the physics of fluid motion is modeled and a numerical solution is obtained thanks to CFD software. The aim is to describe the motion of the fluid in different situations, to do this it is mandatory to use the Navier-Stokes equations, a system of partial differential equations (PDE), these equations are:

$$\frac{\partial \rho}{\partial t} + \frac{\partial(\rho u_i)}{\partial x_i} = 0 \tag{2.50}$$

$$\frac{\partial(\rho u_i)}{\partial t} + \frac{\partial(\rho u_i u_j)}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \frac{\partial \tau_{ij}}{\partial x_j} + \rho f_i \tag{2.51}$$

$$\frac{\partial(\rho e)}{\partial t} + (\rho e + p)\frac{\partial u_i}{\partial x_i} = \frac{\partial(\tau_{ij} u_j)}{\partial x_i} + \rho f_i u_i + \frac{\partial(\dot{q}_i)}{\partial x_i} + r \tag{2.52}$$

The first one is the *continuity* Eq. (2.50), then there is the *conservation of momentum* Eq. (2.51) and in the end the *conservation of energy* Eq. (2.52).
The analytic solution of these equations is possible only when making a large number of simplifications, and usually numerical techniques are used to solve the Navier Stokes equations. However, this requires in general a large computational effort.

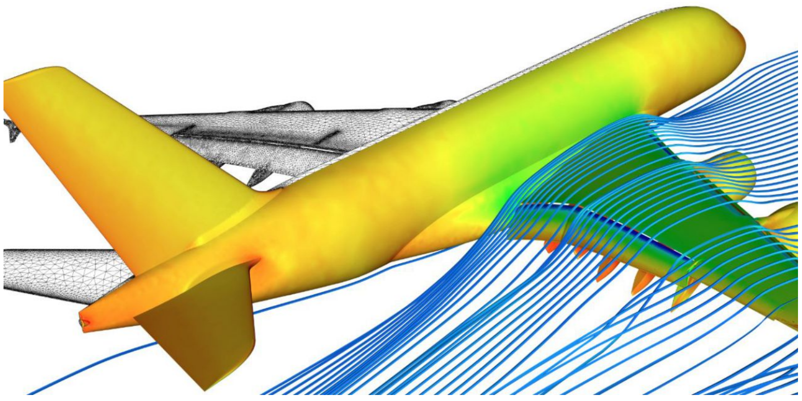


Figure 2.13: *plane CFD simulation*

When using CFD the equations summarized above are solved iteratively, so the result will be approximate, but usually accurate. When starting a CFD calculation it is needed to discretize the domain, and a mesh has to be created. The mesh is an important factor, in fact by changing the density of the mesh the calculation can be more or less accurate; with a coarse mesh the degree of accuracy will be less, with a fine mesh it is the opposite. At the other hand, by increasing the size of the mesh, the computational costs increase too, and the calculation can become too time consuming. So a compromise needs to be found between the accuracy of the calculation and the computational costs.

CFD has some limitations, in fact, the models used are not always accurate and above all the model has to be be calibrated to the problem and once the numerical solution is obtained, it's better to validate the data with experiments. Various approaches have been developed over the years, in particular, three numerical methods are used: the RANS method, which is less accurate but computationally lighter, the LES method, with intermediate characteristics, and the DNS method, which is the most accurate but requires a large computing power.

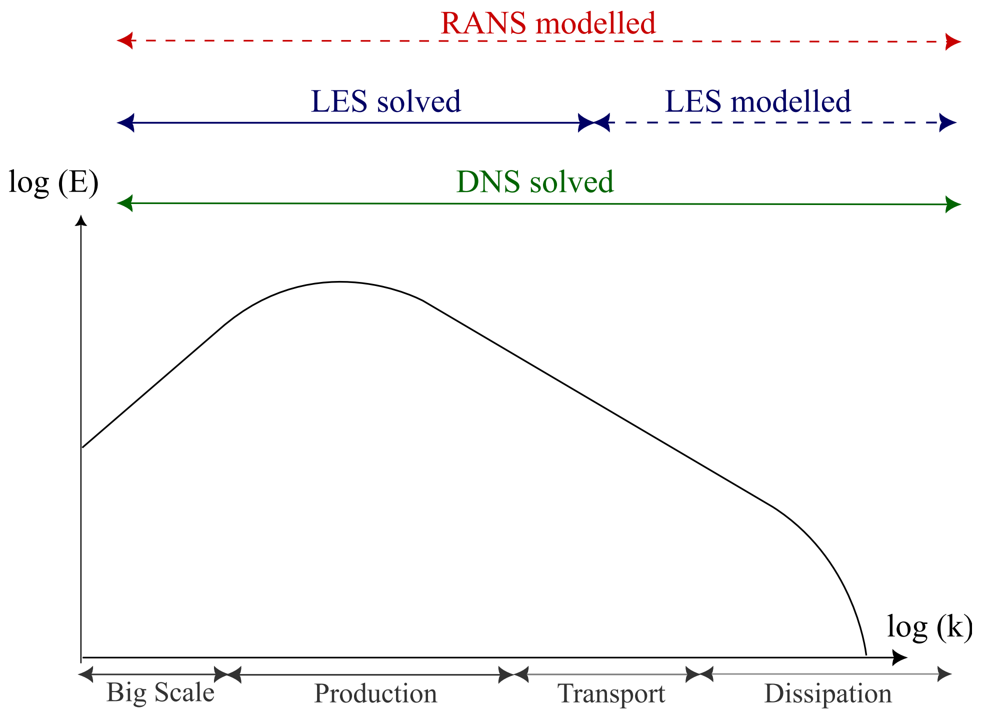


Figure 2.14: *Qualitative representation of the scales of turbulence in the wave number domain that can be solved or modeled by different numerical approaches*

DNS which means *Direct Numerical Simulation*, is the best simulation approach that can be used, in fact, *Navier-Stokes* equations are solved with numeric methods but without applying models, which means more accuracy but more computational time.

RANS, *Reynolds Averaged Navier-Stokes*, is one of the most used approaches, this method is based on the Reynolds decomposition, and the flux is divided into the average and oscillating components. Doing that reduces the calculation time but the approach involves the use of turbulence models and small turbulence scales cannot be captured. Through RANS stationary problem can be solved, in case the flux is unsteady URANS are used, in this approach temporal derivative in the Navier-Stokes equation is not neglected.

LES, *Large Eddy Simulation* is in the middle between DNS and RANS, with this approach, large eddies are solved directly, in fact, these eddies are the most energetic and are the easiest to solve. But the small scales are modelled mathematically.

2.4.1 SU2 CFD code

SU2 is the acronym of *Stanford University Unstructured* and is an open source CFD code written in C++ developed by the Aerospace Design laboratory at Stanford University, available on GitHub [22]. SU2 is a powerful software package that offers a wide range of tools for computational fluid dynamics and optimization problems. The software is based on a finite-volume approach, and it is particularly suited for simulating aerodynamic problems. SU2 can handle both steady and unsteady flows, and can be used to solve problems in both two and three dimensions.

The SU2 software package includes many tools:

- SU2_MSH is the tool to do the grid adaptation based on different techniques
- SU2_CFD is the module to solve direct, adjoint, and linearized problems for the Euler, Navier-Stokes, and Reynolds-Averaged Navier-Stokes (RANS) equation sets.
- SU2_DOT this module calculates the partial derivative of a functional, taking into account the variation of the aerodynamic surface
- SU2_DEF calculates the geometrical deformation of surfaces
- SU2_GEO is the tool to preprocess geometrical information, it is used to compute the geometric constraints
- SU2_SOL is the tool thanks to which is possible to have output files with volume and surface solutions

Inside SU2 is possible to perform both calculations for in-compressible and compressible flow, in this project, inviscous compressible CFD calculations are performed, and to do that, the Euler equations are solved.

One of the main benefits of SU2 is that it is an open-source software package, which means that it is freely available to anyone who wants to use it. The software is also well-documented and well-maintained, with a strong community of users and developers who are constantly working to improve the software and add new features.

2.4.2 Euler equations

The Euler equations are a simplification of the Navier-Stokes equations obtained by neglecting the viscous terms. The equations are therefore *Conservation of Mass, Momentum Equation and Conservation of Energy*:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \underline{u}) = 0 \tag{2.53}$$

$$\frac{\partial \rho \underline{u}}{\partial t} + \nabla \cdot (\rho \underline{u} \otimes \underline{u} + pI) = 0 \tag{2.54}$$

$$\frac{\partial E^*_{tot}}{\partial t} + \nabla \cdot [\underline{u}(E^*_{tot} + P)] = 0 \tag{2.55}$$

The Euler equations are a first-order system of non-linear coupled partial differential equations. The solution of these equations requires a spatial discretization method, as well as an integration method in time. The applicability of the Euler equations is to high Reynolds numbers flow for which viscous effects can be neglected.

The Euler equations will be solved using SU2 to do simulations in the CEASIOMpy environment.

3 CEASIOMpy environment

CEASIOMpy is a powerful tool for aircraft design and optimization, which is freely available to the public as an open-source software downloadable from GitHub. It has been developed by CFS Engineering in collaboration with Airinnova, with the aim of providing a comprehensive and user-friendly platform for the design of new aircraft.
The development of CEASIOMpy was part of the AGILE project, which was initiated by the European Commission in 2015 to improve the efficiency and collaboration among teams of experts in the aircraft design process. The project aimed to reduce the time-to-market and development costs of new aircraft, which are two critical objectives for the aeronautical industry.
One of the key features of CEASIOMpy is its use of Python as a programming language. Python is an increasingly popular language in the scientific and engineering communities due to its simplicity and versatility. With its modular design, CEASIOMpy allows users to easily customize and extend the functionality of the software to meet their specific needs.

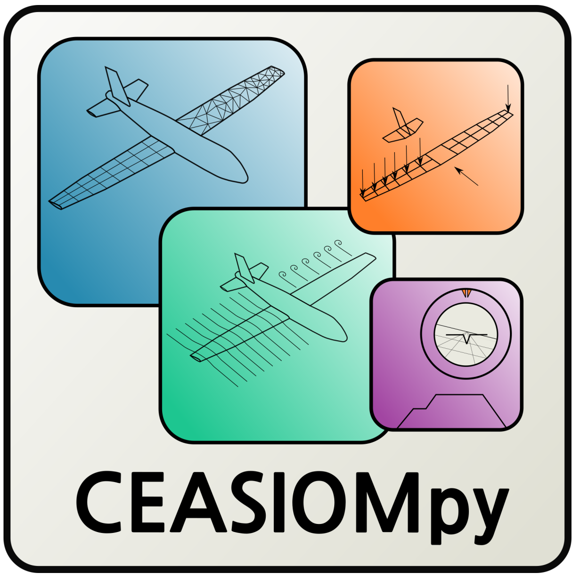


Figure 3.1: *CEASIOMpy logo*

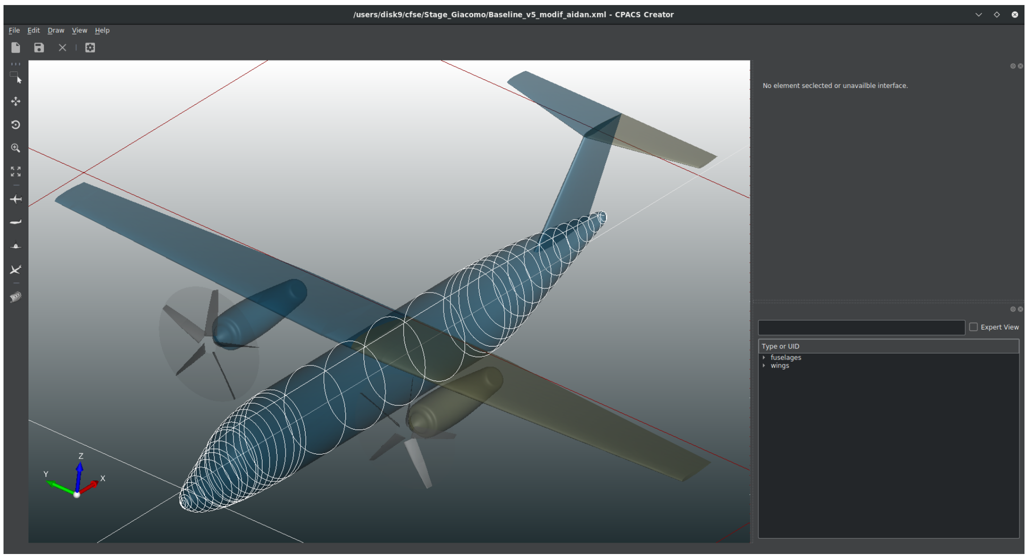
CEASIOMpy provides a wide range of tools for the design and optimization of various aircraft components such as wings, fuselage, and propulsion systems. It includes various methods for aerodynamic and structural analysis, optimization, and sizing. The software also includes a graphical user interface (GUI) that allows users to interact with the software easily and efficiently.
Overall, CEASIOMpy represents a significant step forward in the field of aircraft design and optimization. Its open-source and collaborative nature, combined with its powerful features and user-friendly interface, make it a valuable resource for aeronautical engineers and researchers alike.

3.1 Modules

Over the years, a variety of tools to design and optimise workflows were implemented in CEASIOMpy. These tools include general modules, as well as more specific modules for different components of the aircraft design process. Some of the main modules in CEASIOMpy are:

- *General modules*: these modules provide the core functionality of CEASIOMpy, including the ability to import and export data, manage projects and configure settings.
- *Geometry and Mesh module*: this module contains the "CPACSCreator" tool, which is a CAD tool used to create and modify CPACS files. CPACS [16], or Common Parametric Aircraft Configuration Scheme, is a data definition model for air transport systems that is based on the TiGL language, developed by the German Aerospace Center (DLR) and the European aircraft manufacturer Airbus. The Geometry and Mesh module allows users to generate a 3D model of an aircraft from a CPACS file, which can be used for further analysis and optimization.
- *Aerodynamics module*: this module provides a set of tools for aerodynamic analysis, including computational fluid dynamics (CFD) simulations. The CFD tool can be used to simulate the airflow around an aircraft, providing valuable data for the design and optimization of the aircraft's shape and components. A vortex lattice calculation tool is also available.
- *Weight and Balance*: this module provides tools to estimate the weight and balance of an aircraft. This is a critical step in the design process, as it affects the performance and safety of the aircraft.
- *Mission Analysis*: this module is under development, it allows users to simulate the flight of an aircraft, taking into account factors such as fuel consumption, altitude, and weather conditions. This information can be used to optimize the aircraft's design for a specific mission or route.
- *Structure*: this module is not yet developed yet, but in the future, it will provide structural analysis tools that will allow users to evaluate the strength and durability of the aircraft's components. This is essential to ensure that the aircraft is safe and reliable.

Together, these modules provide a comprehensive and integrated environment for aircraft design and optimisation. By using CEASIOMpy, engineers and researchers can streamline their workflow and optimise the design process, resulting in more efficient and cost-effective aircraft development. It is possible to exchange information between different tools.


/users/disk9/cfse/Stage_Giacomo/Baseline_v5_modif_aidan.xml - CPACS Creator

No element seclected or unavailble interface.

Figure 3.2: *CPACS CAD*

As mentioned above, CEASIOMpy uses CPACS geometry, which is an XML-based data exchange format used in the aerospace industry to share and store information about aircraft design and configuration. It is used by various organisations in the aerospace industry, including aircraft manufacturers, suppliers, research institutes and regulatory bodies. The benefit of CPACS is that it provides a standardised way of representing the geometry, structure, aerodynamics and other important characteristics of an aircraft design. This enables different teams and organisations to collaborate more effectively and efficiently during the design and development process.
The CPACS interface allows to modify every part of the plane, in particular, it is possible to add and remove sections from *fuselage*, *wing*, *nacelle* and the *propeller disk*.

Two different modules are available to generate a mesh, these are *CPACS2GMSH* which uses GMSH [15] and *SUMOAutoMesh* which converts CPACS geometry to SUMO [26] geometry and then generates the mesh. Both modules allow the user to obtain a mesh automatically. However, SUMO currently has one limitation: it does not allow the modelling of an actuator disk.

The *Aerodynamic module* is the most important of the entire python environment, there are two programs to perform calculations: *pyTornado* [27] a vortex latex method-based program and *SU2Run*. The one that will be used is *SU2Run*, which uses the SU2 CFD software.

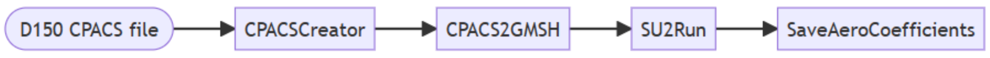
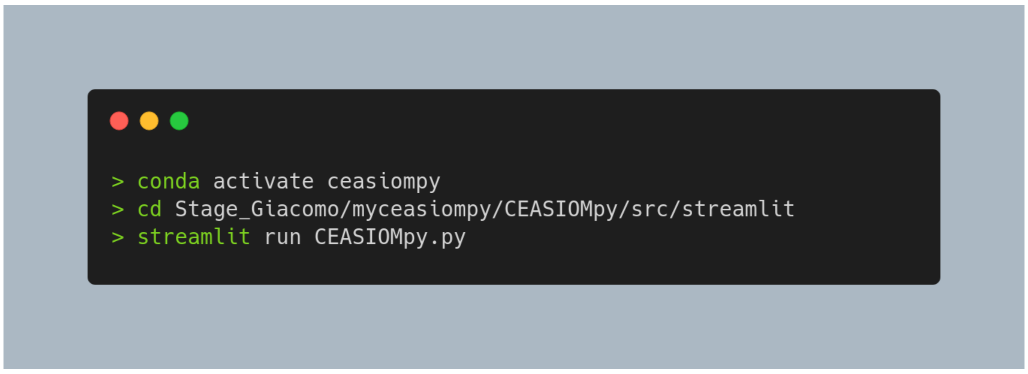


Figure 3.3: *Typical workflow*

In Fig. 3.3 is possible seeing how a workflow can be, each module exchanges information with the following one in order to obtain the solution.

3.2 How to use it

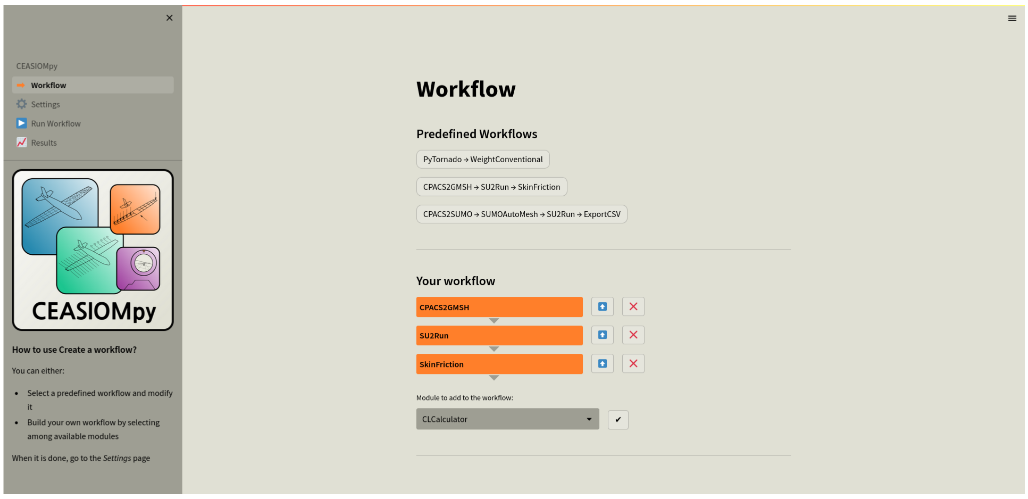
To use CEASIOMpy, you will first need to clone it from its GitHub repository and follow the instructions provided on the website [14].
Once the installation is complete, you can begin using it by activating the *ceasiom environment* from the command prompt. To open the CEASIOMpy Graphical User Interface (GUI), which is created using *streamlit* [23], you can use the following commands:



```
> conda activate ceasiompy
> cd Stage_Giacomo/myceasiompy/CEASIOMpy/src/streamlit
> streamlit run CEASIOMpy.py
```

Figure 3.4: *Command line to run CEASIOMpy from the command prompt*

When the program is launched, the first step is to choose the CPACS file to be used for the study.
After that, modules have to be selected; in order to do a CFD calculation, the workflow at least has to consist of a mesh module and the SU2Run module to perform the Euler calculation. To take into account the effect of viscosity, *SkinFriction* module has to be added.



How to use Create a workflow?

You can either:

- Select a predefined workflow and modify it
- Build your own workflow by selecting among available modules

When it is done, go to the *Settings* page

Workflow

Predefined Workflows

PyTornado → WeightConventional

CPACS2GMSH → SU2Run → SkinFriction

CPACS2SUMO → SUMOAutoMesh → SU2Run → ExportCSV

Your workflow

CPACS2GMSH

SU2Run

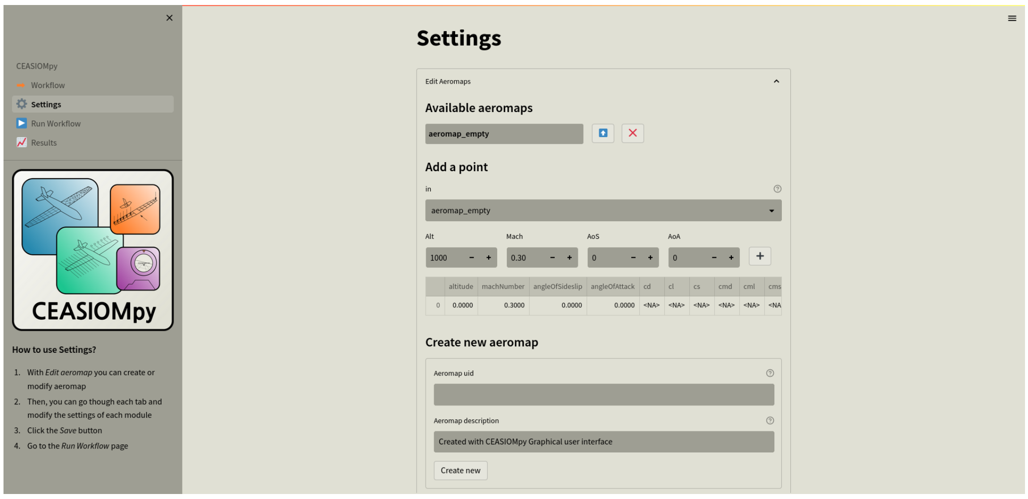
SkinFriction

Module to add to the workflow:

CLCalculator

Figure 3.5: *Workflow*

Once the workflow is chosen, input variables for each module need to be set. First of all, the flight conditions have to be imposed, which means choosing an *altitude* and a cruise velocity, hence a *Mach number*, this option in the GUI is called *aeromap*. Aeromap settings are very important because from the altitude it is possible to get the air density and from the Mach number, it is possible to obtain the speed of sound and so the velocity in m/s. Furthermore, a different angle of attack can be chosen to simulate in the best way different mission phases.



How to use Settings?

1. With *Edit aeromap* you can create or modify aeromap
2. Then, you can go though each tab and modify the settings of each module
3. Click the *Save* button
4. Go to the *Run Workflow* page

Settings

Edit Aeromaps

Available aeromaps

aeromap_empty

Add a point

in: aeromap_empty

Alt: 1000 | Mach: 0.30 | AoS: 0 | AoA: 0

	altitude	machNumber	angleOfSideslip	angleOfAttack	cd	cl	cs	cmd	cml	cms
0	0.0000	0.3000	0.0000	0.0000	<NA>	<NA>	<NA>	<NA>	<NA>	<NA

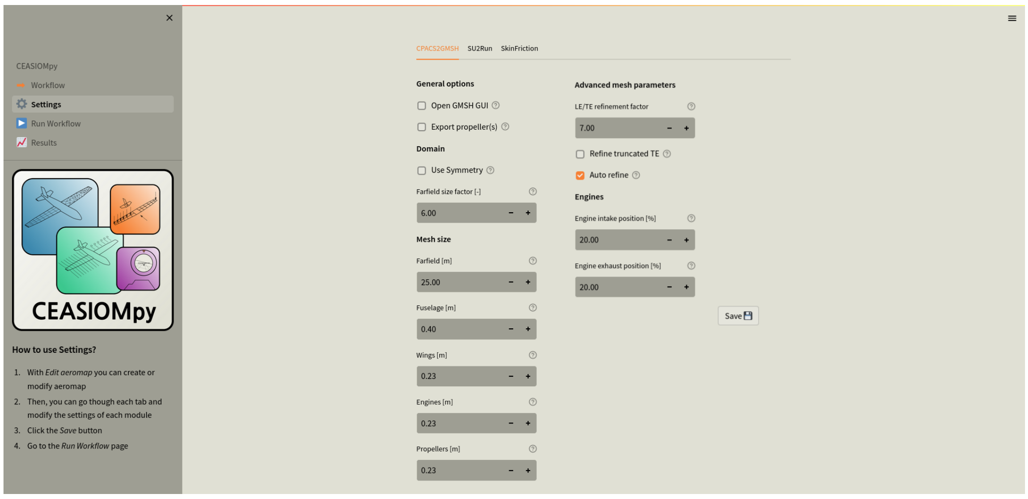
Create new aeromap

Aeromap uid:

Aeromap description: Created with CEASIOMpy Graphical user interface

Figure 3.6: *Aeromap*

In the Mesh module, the user can select the size of the far field and the mesh refinement for each part of the aircraft. It is possible to decide how much the automatic meshing tool should refine the leading edge, trailing edge, fuselage, engine, wing and propeller.



How to use Settings?

1. With *Edit aeromap* you can create or modify aeromap
2. Then, you can go though each tab and modify the settings of each module
3. Click the *Save* button
4. Go to the *Run Workflow* page

General options

- ☐ Open GMSH GUI
- ☐ Export propeller(s)

Domain

- ☐ Use Symmetry

Farfield size factor [-]: 6.00

Mesh size

Farfield [m]: 25.00

Fuselage [m]: 0.40

Wings [m]: 0.23

Engines [m]: 0.23

Propellers [m]: 0.23

Advanced mesh parameters

LE/TE refinement factor: 7.00

- ☐ Refine truncated TE
- ✓ Auto refine

Engines

Engine intake position [%]: 20.00

Engine exhaust position [%]: 20.00

Figure 3.7: *CPACS2GMSH settings*

After the mesh has been generated, it is possible to open GMSH, check the mesh quality and, if necessary, modify the parameters.

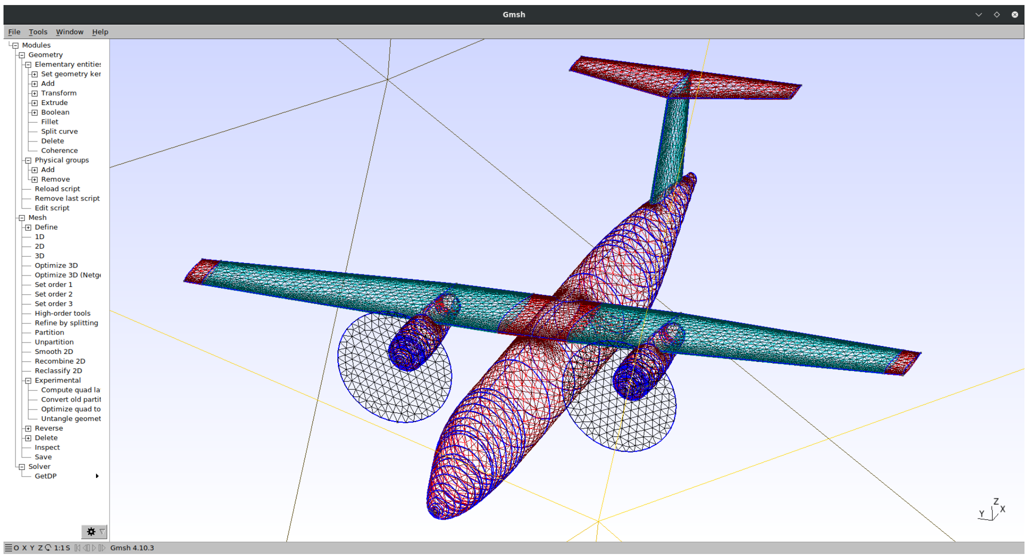
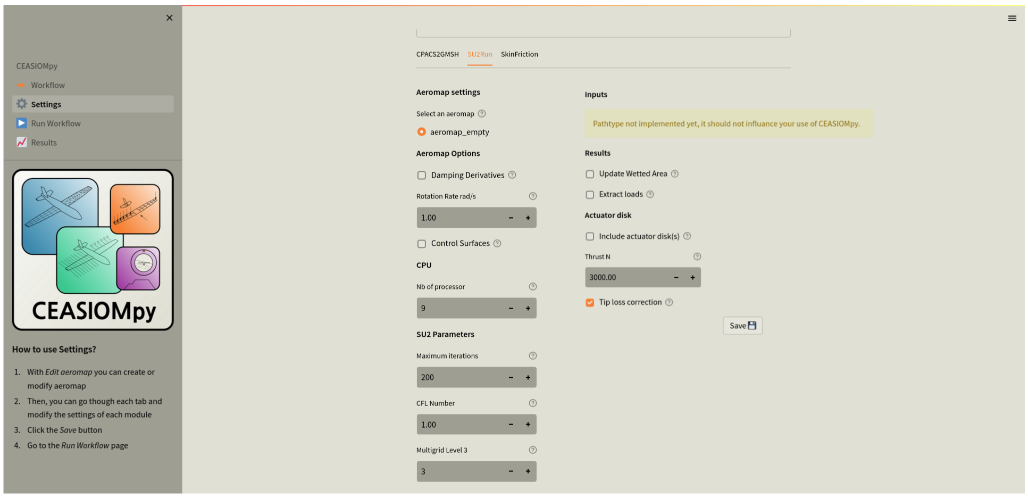


Figure 3.8: *GMSH interface*

In this project, the actuator disk calculation has been added to the CEASIOM environment, so in order to perform the actuator disk calculation using the CPACS2GMSH module, the meshing of the disk has to be enabled in the Settings GUI. At the moment, the only way to mesh the disc is to use the CPACS2GMSH module, as this is not yet possible in SUMOAutoMesh.

Once the mesh has been set up, the next step is to modify the SU2Run parameter, i.e. to set the number of cores to be used in the simulation, checking the CFL number and the number of multigrid levels in order to reach convergence sooner during the calculation.



How to use Settings?

1. With *Edit aeromap* you can create or modify aeromap
2. Then, you can go though each tab and modify the settings of each module
3. Click the *Save* button
4. Go to the *Run Workflow* page

Aeromap settings

Select an aeromap: [✓ aeromap_empty]

Aeromap Options

☐ Damping Derivatives

Rotation Rate rad/s: 1.00

☐ Control Surfaces

CPU

Nb of processor: 9

SU2 Parameters

Maximum iterations: 200

CFL Number: 1.00

Multigrid Level 3: 3

Inputs

Pathtype not implemented yet, it should not influance your use of CEASIOMpy.

Results

☐ Update Wetted Area

☐ Extract loads

Actuator disk

☐ Include actuator disk(s)

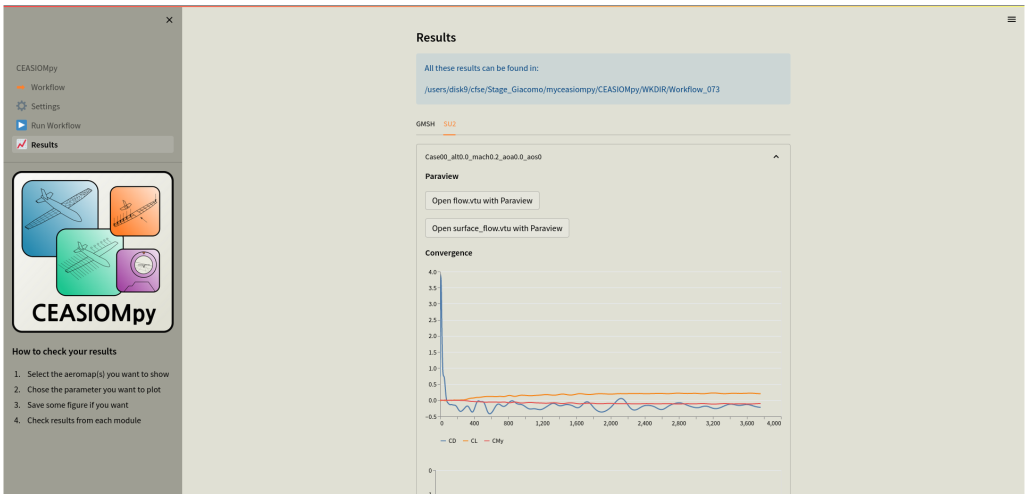
Thrust N: 3000.00

☑ Tip loss correction

Figure 3.9: *SU2Run settings*

When the ActuatorDisk module was incorporated into CEASIOMpy, the option to enable the Actuator Disk model was included in the SU2Run module. This feature allows the user to specify the desired thrust output of the propeller engine.

This is just one way of performing a calculation, and many other modules can be added.
A particularly useful one is the *SkinFriction module*, which allows the presence of viscosity to be taken into account in an inviscid calculation. The only thing you have to do to set it up is to select the aeromap to be used during the calculation.


CEASIOMpy
Workflow
Settings
Run Workflow
Results
CEASIOMpy
Results
GMSH
SU2
Case00_alt0.0_mach0.2_aoa0.0_aos0
Paraview
Open flow.vtu with Paraview
Open surface_flow.vtu with Paraview
Convergence
CD
CL
CMy

All these results can be found in:

/users/disk9/cfse/Stage_Giacomo/myceasiompy/CEASIOMpy/WKDIR/Workflow_073

How to check your results

1. Select the aeromap(s) you want to show
2. Chose the parameter you want to plot
3. Save some figure if you want
4. Check results from each module

Figure 3.10: *Results visualization in the GUI*

Once all the settings are configured, the workflow can be initiated which involves performing a calculation for each module. During the simulation, the residuals can be monitored to track their evolution. Upon completion, the results are automatically stored in the *"result repository"* within a folder named *Workflow00**. As each calculation is performed, the folder changes to create a record of all the actions taken.

4 Actuator Disk calculation procedure

The computational procedure for aerodynamic simulations of aircraft in this project involves taking into account the presence of the propeller. This requires the use of specialised modelling techniques to accurately represent the effect of the propeller on the surrounding airflow.

In order to achieve this, a module is introduced into the simulation software to specifically calculate the effect of the propeller. One common technique used in such modules is the actuator disk model, which involves representing the propeller as a disk that generates thrust and interacts with the surrounding airflow.

4.1 Actuator Disk module

The integration of the *ActuatorDisk* module into the CEASIOMpy environment represents a significant advancement in the CEASIOMpy aircraft design and simulation tool. This module allows for more accurate and detailed modelling of the propeller, which in turn provides a better understanding of the overall aircraft performance.

As seen, there are several different theories that can be used to describe the presence of the propeller. Two of the most commonly used are the actuator disk model and the blade element theory. Both approaches allow the calculation of the thrust generated by the propeller.
The thrust output can be modelled as a constant along the blade or as a variable distribution along the radius of the propeller. The choice of modelling approach depends on factors such as the accuracy required for the simulation, the complexity of the propeller design and the specific performance characteristics being investigated.

4.2 Thrust distribution calculation

To calculate the thrust in the actuator disk module, it has been decided to model it using a *variable load thrust distribution*, which means that the thrust changes along the radius. A way of calculating the distribution of the thrust along the radius was investigated and three methods were explored:

- myBET
- myBEM
- OptimalProp

myBET & myBEM

An initial attempt was made to distribute the total thrust along the blade radius by writing a Python script named *"myBET"*. The script is based on the Blade

Element Theory (BET) and takes input parameters such as the blade geometry, aircraft velocity, and rotational velocity of the blades.

The thrust and power coefficient distributions along the adimensionalized radius are calculated by *"myBET"*. Although the model is simple, it provides an initial estimation of the thrust distribution. However, the results could be affected by errors due to the simplicity of the model.

To obtain more reliable results, a second Python script named *"myBEM"* was written, which incorporates a correction on both the axial and rotational velocities. This script utilizes the Blade Element Momentum (BEM) theory, which combines the Actuator Disk Theory and the Blade Element Theory.

Both theories at the beginning used the theoretical definition of C_L, $C_L = 2\pi\alpha$, then to reach a better result it has been added the *pyfoil library* [19], that exploit XFLR5 [21], in order to have a better lift coefficient evaluation.

However, due to the limited availability of open-source data, finding the chord and twist distributions along the blade radius proved to be a challenge. The only available information was on small blades used in drones, model aircraft, and outdated blades.

OptimalProp

"OptimalProp", unlike the previous two, is a script that is not based on geometrical blade data, but takes as input flow data and just the radius of the propeller, which is easily accessible and gives as a result the thrust distribution along the radius.

This Python script was written by the TARG (Theoretical Aerodynamic Research Group) from the University of Naples [8], the script is based on general momentum theory and blade element theory, to consider the variation of velocity in tangential direction due to the blade movement.

The original program has been analyzed and reorganized, modifying some parts, with the purpose of integrating it into the CEASIOM environment, hence different functions have been created to lighten the script. These functions are called in the main one, *thrust calculator*, thanks to which thrust distribution along the radius is obtained.
Inside the thrust calculation function, a preliminary calculation is done to get *advanced ratio*, and rotational velocity ω, thereafter the iterative process begins. The goal of the process is to obtain the right interference factor that gives the optimal thrust distribution, in fact in the differential definition of thrust, seen in Section 2.2.1, it can be possible to write it as an interference factor function.

$$dT = 4\pi\rho V_\infty^2(1+a)ardr \tag{4.1}$$

The same thing can be done for power which is also calculated in the script

$$dP = 4\pi\rho[V_\infty^3(1+a)^2ardr + \Omega^2 V_\infty(1+a)a'^3r^3dr] \tag{4.2}$$

To get a and a' the theory of optimal axial and rotational induction is used, which is based on solving the minimum constraint problem mentioned in chapter 2, in particular, Eq. (2.30) will be solved.
The solution to the problem is:

$$\frac{1+a}{\chi^2(1-2a')} + \frac{a'}{1+2a} = constant \tag{4.3}$$

with $\chi = \Omega r/V_\infty$.
To have the optimal load distribution, a(r) has to be constant along the radius of the blade, if this happens, the maximum efficiency can be reached, which means having the minimum power to obtain the requested thrust.

Script comparison

In order to choose the best way to perform the calculation, the results were compared. As a first try a simulation was done considering an old wood propeller.

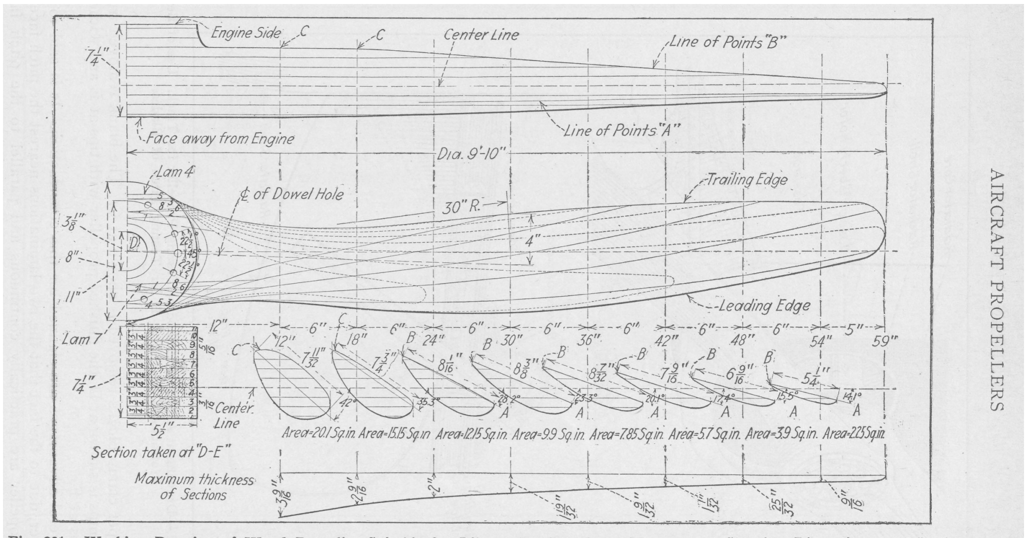


Fig. 221.—Working Drawing of Wood Propeller Suitable for Liberty Engine of 400 Horsepower, Showing Dimensions and Sections of Blade at Various Stations.

Figure 4.1: *Data of the first propeller tested*

The data of the tested blade are as follows, some of them, such as the free stream velocity and the propeller rotation velocity, have been estimated.

Propeller operating data		
Free stream velocity	20	[m/s]
Rotational velocity	27	[1/s]
Radius	2.743	[m]
Blades number	2	[/]

Table 1: *Propeller operating data*

Propeller geometrical data			
element	r/R	x/C	θ
1	0.185	0.116	39.7
2	0.222	0.121	37.6
3	0.333	0.132	32.3
4	0.444	0.134	27.8
5	0.555	0.131	24.4
6	0.666	0.119	21.8
7	0.777	0.100	19.9
8	0.888	0.075	18.3
9	0.944	0.061	17.6

Table 2: *Propeller geometrical data*

Then an analysis was made with the two python script to calculate the thrust distribution along the radius.

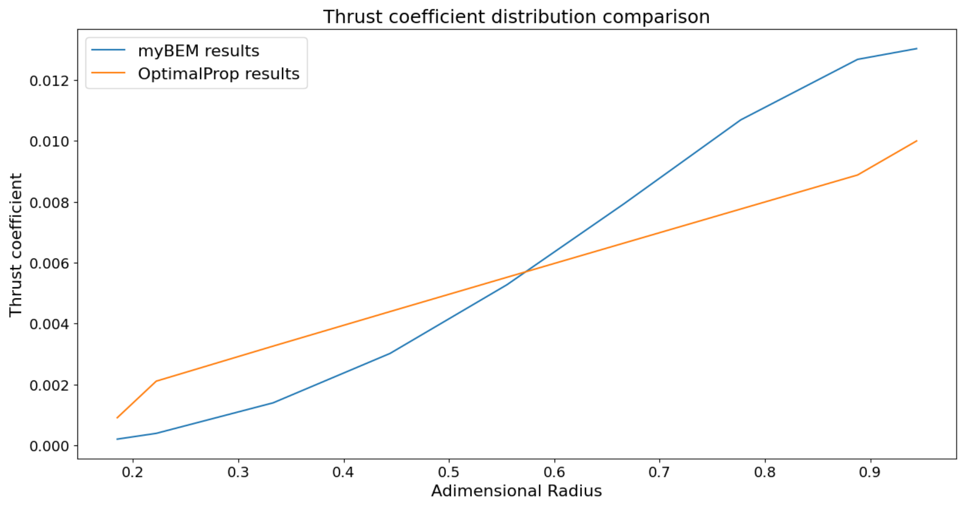
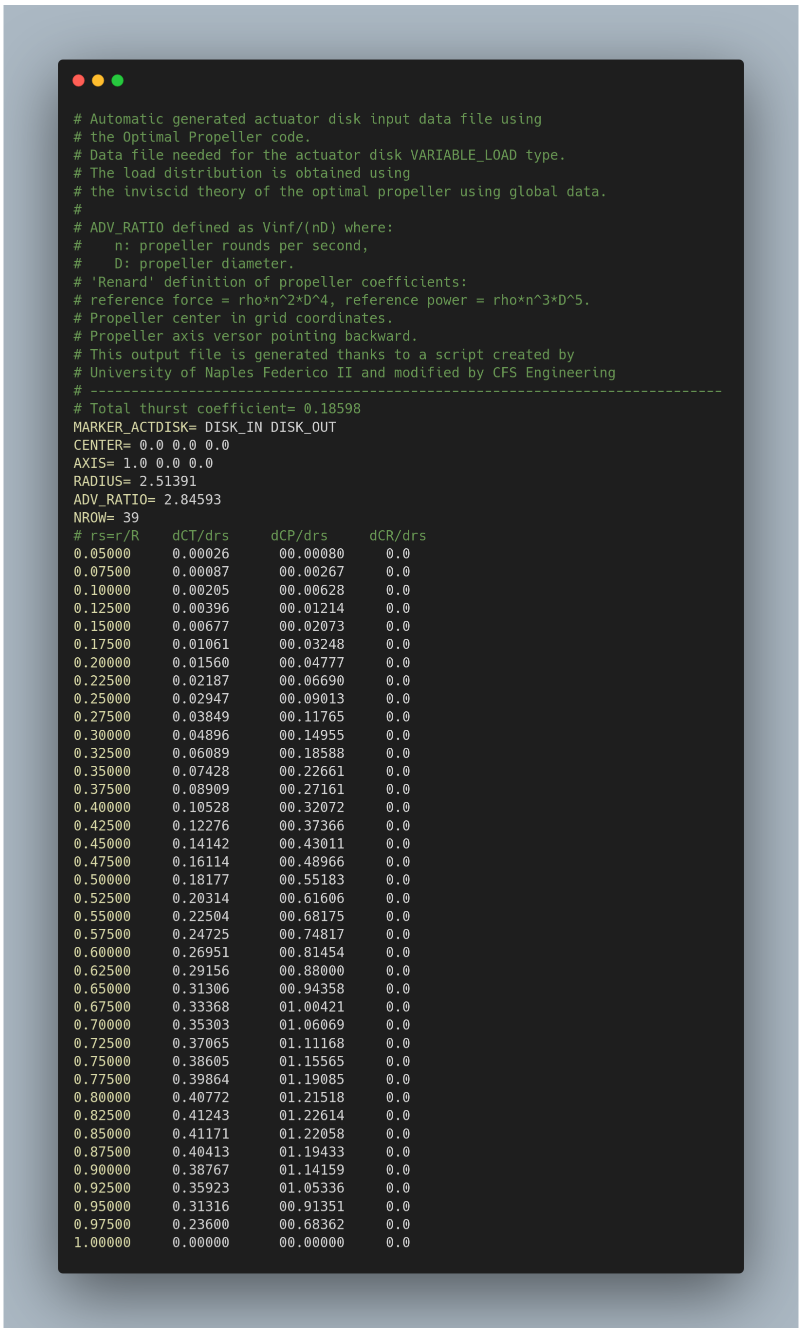


Figure 4.2: *Comparison between the two scripts*

The results are comparable, despite a slightly different trend, but then, trying to change the free stream velocity, the program showed some robustness problems. Considering all the factors involved (including, above all, the problem of data repeatability), it was decided not to delve into pyBEM.

4.3 Module integration

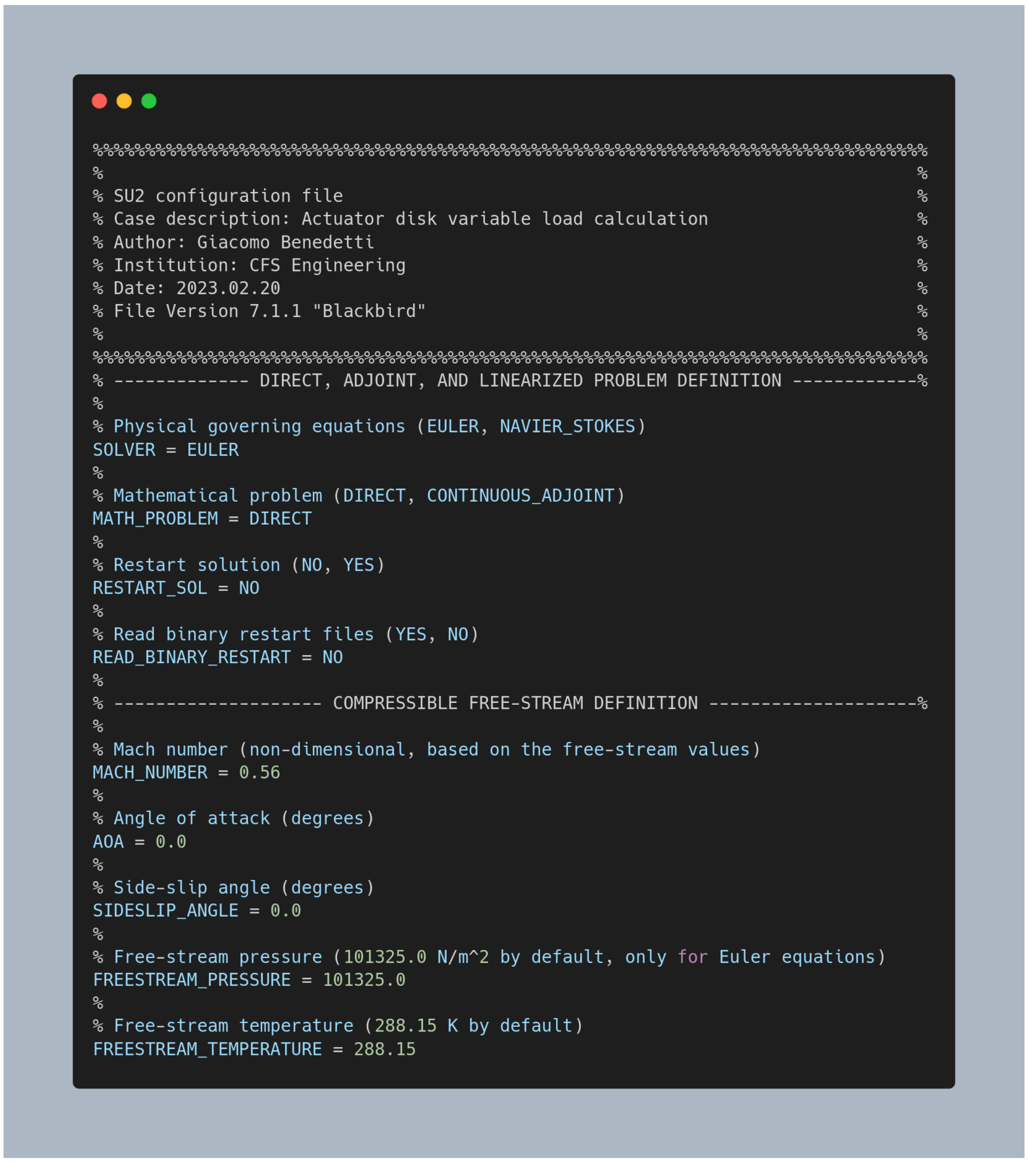
The integration of *OptimalProp* in CEASIOMpy has been done by creating a function called *su2actuatordisk*, which is called inside the main function: *SU2Run*. The purpose of the su2actuatordisk function is to create a .DAT file. This file is very important because it contains the information about the engine position as well as general information about engine performances and the distributions of r/R, $\frac{dC_T}{d(r/R)}$, $\frac{dC_P}{d(r/R)}$, $\frac{dC_R}{d(r/R)}$.



```
# Automatic generated actuator disk input data file using
# the Optimal Propeller code.
# Data file needed for the actuator disk VARIABLE_LOAD type.
# The load distribution is obtained using
# the inviscid theory of the optimal propeller using global data.
#
# ADV_RATIO defined as Vinf/(nD) where:
#    n: propeller rounds per second,
#    D: propeller diameter.
# 'Renard' definition of propeller coefficients:
# reference force = rho*n^2*D^4, reference power = rho*n^3*D^5.
# Propeller center in grid coordinates.
# Propeller axis versor pointing backward.
# This output file is generated thanks to a script created by
# University of Naples Federico II and modified by CFS Engineering
# ---------------------------------------------------------------------------
# Total thurst coefficient= 0.18598
MARKER_ACTDISK= DISK_IN DISK_OUT
CENTER= 0.0 0.0 0.0
AXIS= 1.0 0.0 0.0
RADIUS= 2.51391
ADV_RATIO= 2.84593
NROW= 39
# rs=r/R    dCT/drs     dCP/drs     dCR/drs
0.05000     0.00026      00.00080     0.0
0.07500     0.00087      00.00267     0.0
0.10000     0.00205      00.00628     0.0
0.12500     0.00396      00.01214     0.0
0.15000     0.00677      00.02073     0.0
0.17500     0.01061      00.03248     0.0
0.20000     0.01560      00.04777     0.0
0.22500     0.02187      00.06690     0.0
0.25000     0.02947      00.09013     0.0
0.27500     0.03849      00.11765     0.0
0.30000     0.04896      00.14955     0.0
0.32500     0.06089      00.18588     0.0
0.35000     0.07428      00.22661     0.0
0.37500     0.08909      00.27161     0.0
0.40000     0.10528      00.32072     0.0
0.42500     0.12276      00.37366     0.0
0.45000     0.14142      00.43011     0.0
0.47500     0.16114      00.48966     0.0
0.50000     0.18177      00.55183     0.0
0.52500     0.20314      00.61606     0.0
0.55000     0.22504      00.68175     0.0
0.57500     0.24725      00.74817     0.0
0.60000     0.26951      00.81454     0.0
0.62500     0.29156      00.88000     0.0
0.65000     0.31306      00.94358     0.0
0.67500     0.33368      01.00421     0.0
0.70000     0.35303      01.06069     0.0
0.72500     0.37065      01.11168     0.0
0.75000     0.38605      01.15565     0.0
0.77500     0.39864      01.19085     0.0
0.80000     0.40772      01.21518     0.0
0.82500     0.41243      01.22614     0.0
0.85000     0.41171      01.22058     0.0
0.87500     0.40413      01.19433     0.0
0.90000     0.38767      01.14159     0.0
0.92500     0.35923      01.05336     0.0
0.95000     0.31316      00.91351     0.0
0.97500     0.23600      00.68362     0.0
1.00000     0.00000      00.00000     0.0
```

Figure 4.3: *File .dat of the engine*

This file is called in the .cfg file, which is the configuration file read by the SU2 CFD solver.



```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                              %
% SU2 configuration file                                                       %
% Case description: Actuator disk variable load calculation                    %
% Author: Giacomo Benedetti                                                    %
% Institution: CFS Engineering                                                 %
% Date: 2023.02.20                                                             %
% File Version 7.1.1 "Blackbird"                                               %
%                                                                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ------------- DIRECT, ADJOINT, AND LINEARIZED PROBLEM DEFINITION ------------%
%
% Physical governing equations (EULER, NAVIER_STOKES)
SOLVER = EULER
%
% Mathematical problem (DIRECT, CONTINUOUS_ADJOINT)
MATH_PROBLEM = DIRECT
%
% Restart solution (NO, YES)
RESTART_SOL = NO
%
% Read binary restart files (YES, NO)
READ_BINARY_RESTART = NO
%
% -------------------- COMPRESSIBLE FREE-STREAM DEFINITION --------------------%
%
% Mach number (non-dimensional, based on the free-stream values)
MACH_NUMBER = 0.56
%
% Angle of attack (degrees)
AOA = 0.0
%
% Side-slip angle (degrees)
SIDESLIP_ANGLE = 0.0
%
% Free-stream pressure (101325.0 N/m^2 by default, only for Euler equations)
FREESTREAM_PRESSURE = 101325.0
%
% Free-stream temperature (288.15 K by default)
FREESTREAM_TEMPERATURE = 288.15
```

Figure 4.4: *.cfg file*

The .cfg file for SU2 contains all the settings for the CFD simulation. It is a configuration file that is read by the SU2 CFD solver and defines every aspect of the simulation, such as the mesh file, flow conditions, turbulence models and numerical schemes. The .cfg file is a critical component of the SU2 CFD software and must be properly configured for accurate and reliable simulation results.

5 Results analysis

Before performing calculations on the geometry of interest, it is necessary to validate the tool. Once validated, the potential and limitations of the program were investigated by analysing a simple rotor created in CPACS, followed by a more complex geometry.

5.1 Actuator disk data file validation

To validate the actuator disk data file generated by CEASIOMpy, one of the compressible case tutorials named "Actuator Disk With Variable Load" available on GitHub was used. The thrust distribution file was generated using CEASIOMpy and was used as an input file along with the geometry and mesh provided by the test case on the SU2 website. A simulation was performed using SU2 with RANS equations, as in the tutorial, but the most significant simulation was carried out using the Euler solver, as non-viscous calculations with Euler equations can be performed using CEASIOMpy.

So the configuration file has been set similarly in the two simulations, the surface markers are the same in both cases, instead, the boundary conditions change a bit.

The geometry consists of a rotor with a semi-infinite spinner and the markers that were given are:

- Actuator Disk Inlet
- Actuator Disk Outet
- Spinner
- Farfield
- Inlet
- Outlet

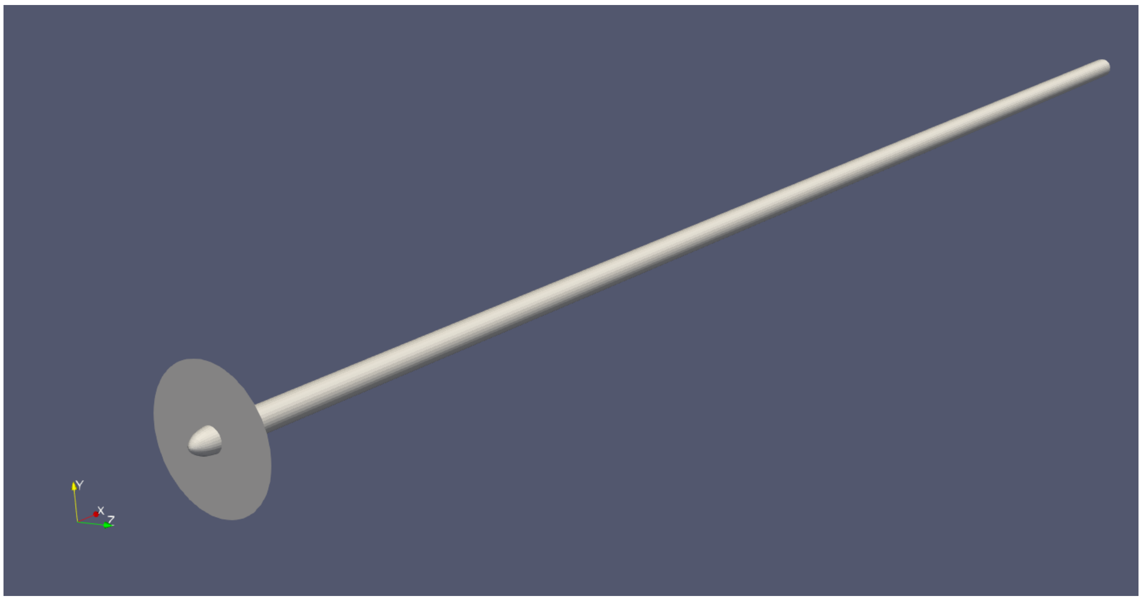


Figure 5.1: *SU2 test case, semi-infinite spinner*

The mesh used is the one that can be found on the SU2 website, is not very fine, but it is sufficient for the validation study discussed here.

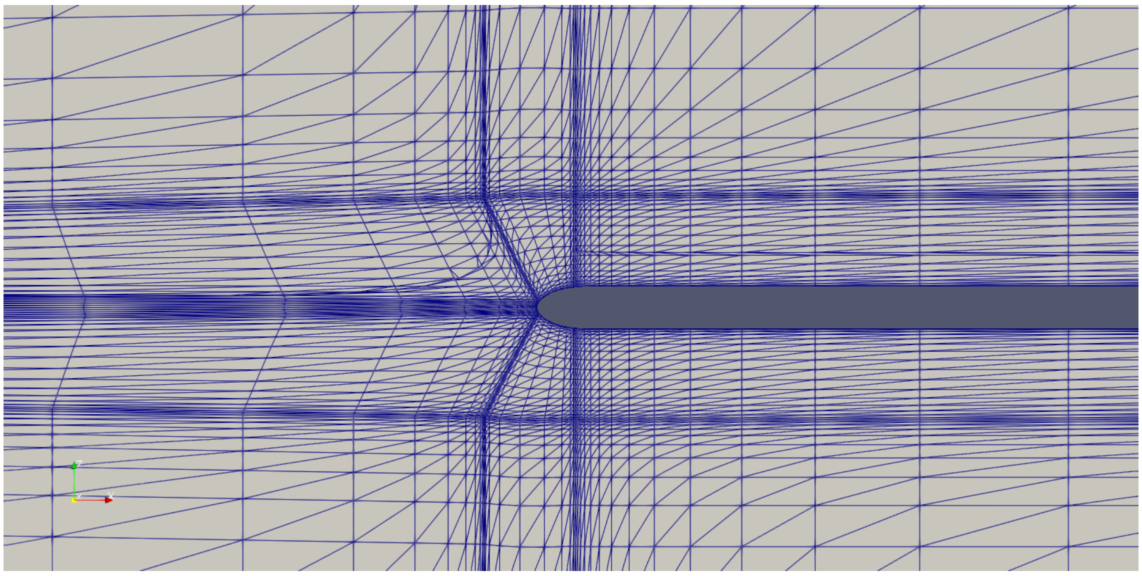


Figure 5.2: *Structured test case mesh*

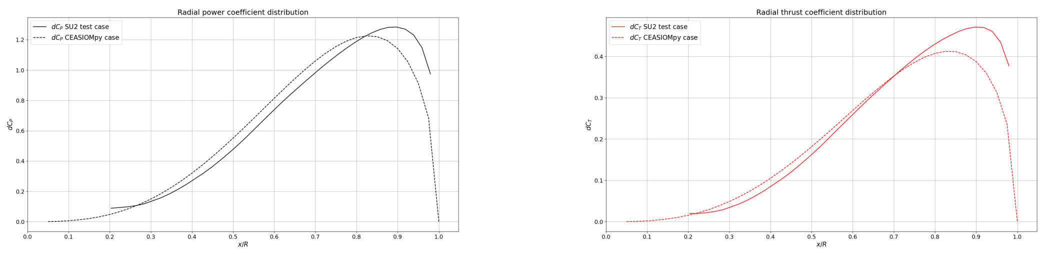
The calculation parameters are geometric and operational, and are set the same as for the SU2 test case:

	Data	**U.o.M.**
Cruise Mach number	0.56	[/]
Rotational velocity	13.53	[1/s]
Thrust	25600	N
Altitude	0	[m]
Propeller radius	2.5146	[m]
Number of blades	5	[/]

Table 3: *Initial parameter of simulation*

This setup has been used for both RANS and Euler calculations.

The radial distribution of thrust and power coefficients are the following



(a) *Radial power distribution along the radius* **(b)** *Radial thrust distribution along the radius*

Figure 5.3: *Comparison of radial distribution of power and thrust along the radius*

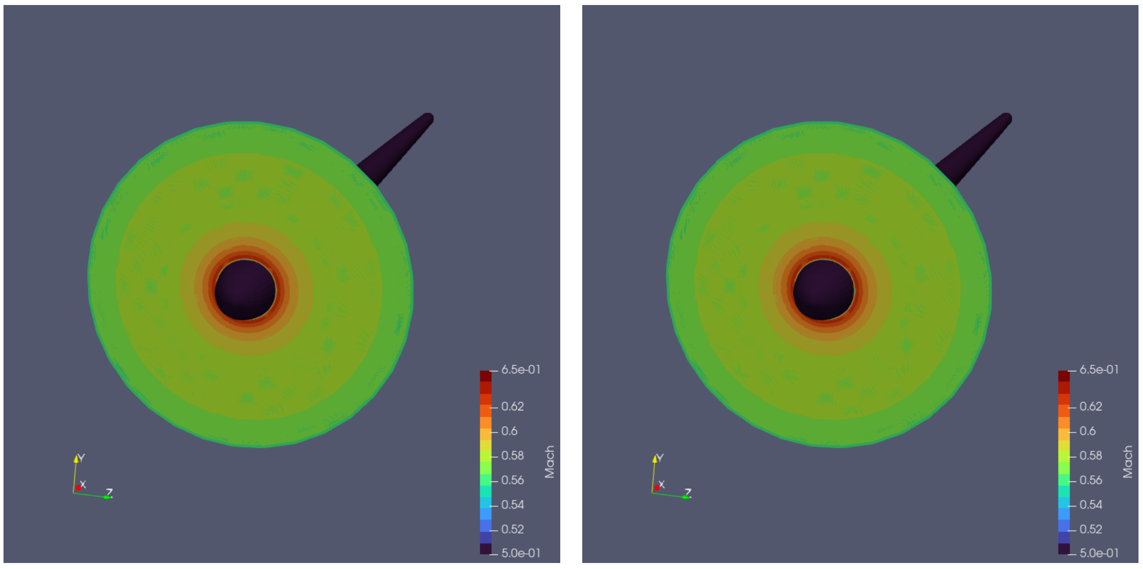
In Fig. 5.3, we compare the results obtained from the original program with the .DAT file generated in CEASIOMpy. We observe some differences between the two coefficients. Specifically, in CEASIOMpy, the minimum values of dC_T and dC_P occur at x/R = 0.025, instead of x/R = 0.2 as in the original program. This discrepancy is due to a limitation in the CPACS module, which affects the interface between the spinner and the hub radius. Additionally, we have chosen to set $dC_T(x/R = 1) = 0$.

Both Euler and RANS simulations use the previous distribution of thrust and power coefficients.

The results obtained from the simulation have been compared to check the accuracy of the tool.

RANS comparison

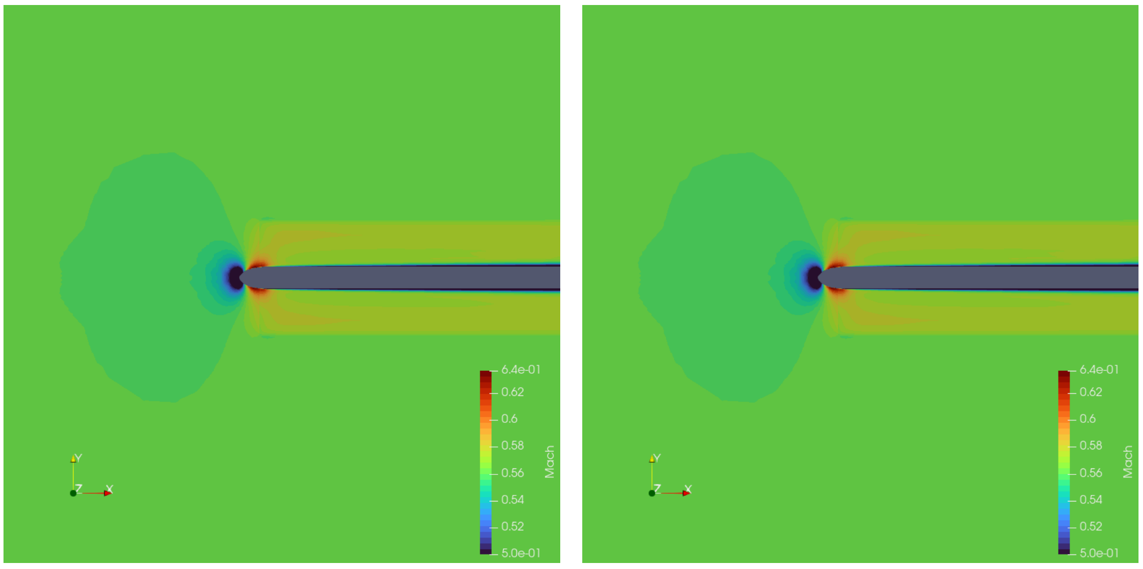
The configuration file used was the one given by the test case, instead the .DAT file was obtained from CEASIOMpy. The RANS calculation was then started and the results were compared with SU2 tutorial.



(a) *Velocity contour from the SU2 test case* **(b)** *Velocity contour calculated with RANS*

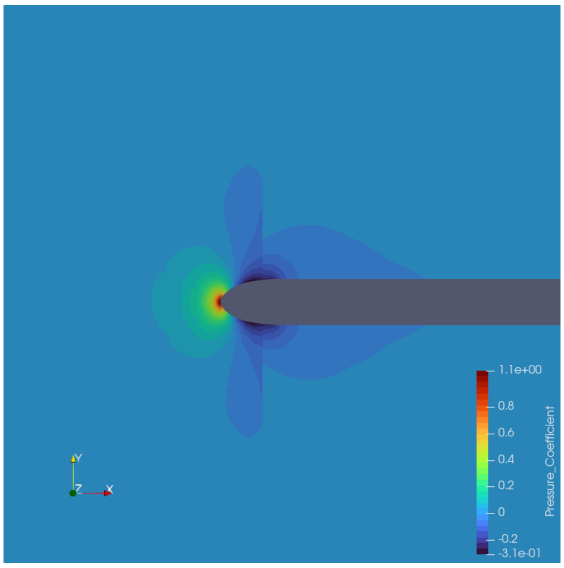
Figure 5.4: *Velocity contour comparison between the tutorial and the case calculated with RANS*

It can be observed that the distributions of the velocity on the disk are the same, which means the *ActuatorDisk.DAT* file generates the same thrust distribution. Contours of the velocity and pressure in a plane parallel to the axis of the spinner are shown in the following pictures.
As can be seen, the results using the original actuator disk and the CEASIOMpy actuator disk are comparable. The main difference is in the pressure distribution downstream of the actuator disk.

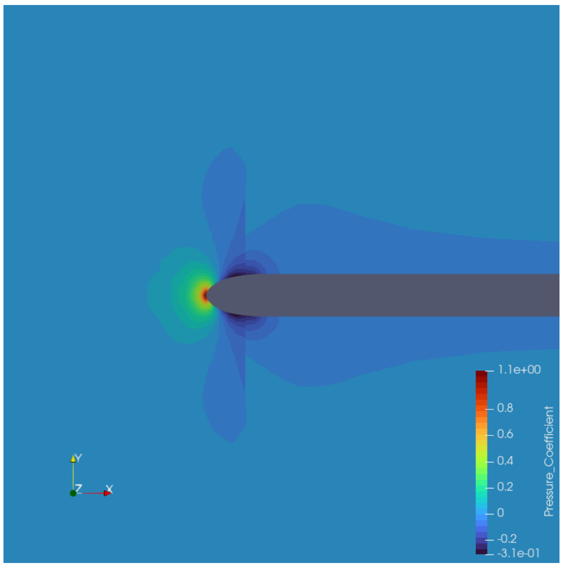


(a) *Velocity contour from the tutorial* **(b)** *Velocity contour calculated with RANS*

Figure 5.5: *Velocity contour comparison between the test case and the case calculated with RANS*



(a) *Pressure coefficient contour from the tutorial*



(b) *Pressure coefficient contour calculated with RANS*

Figure 5.6: *Pressure coefficient contour comparison between the tutorial and the case calculated with RANS*

Upstream of the disk, the pressure rises and downstream of it is reduced, and this corresponds to the theory. It should be noted that the scale has been modified to better show the local pressure variation across the disk and it may appear that it decreases slowly until the free stream pressure, in reality, when taking a closer look at the values, the pressure variations downstream of the disk are minimal.

The behaviour can be also visualized by plotting the pressure variation along a line through the disk, in particular, the line was plotted at 75% of the radius.

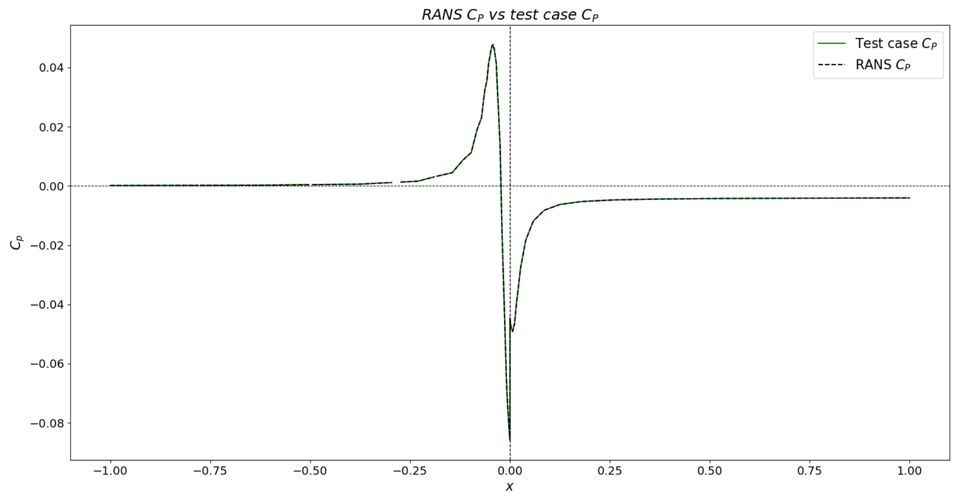


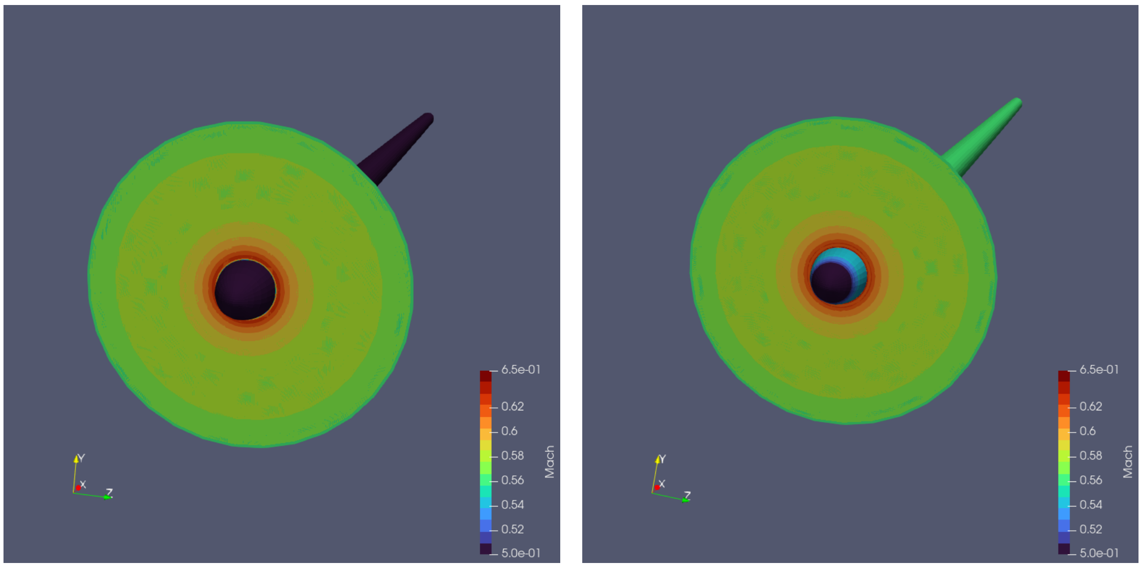
Figure 5.7: *Pressure coefficient plot RANS comparison*

The pressure rises initially, but this is due to the presence of the hub, which creates a stagnation point after the C_P follows the theoretical trend explained above.

Euler comparison

A second calculation was carried out, but this time the results of the test case were compared with the results of a simulation using the Euler equations. The .cfg file was modified to correctly perform the Euler calculation.

The boundary conditions have been modified and new markers have been introduced. In particular, the *Euler marker* has been included to indicate the presence of a wall for inviscid flow.

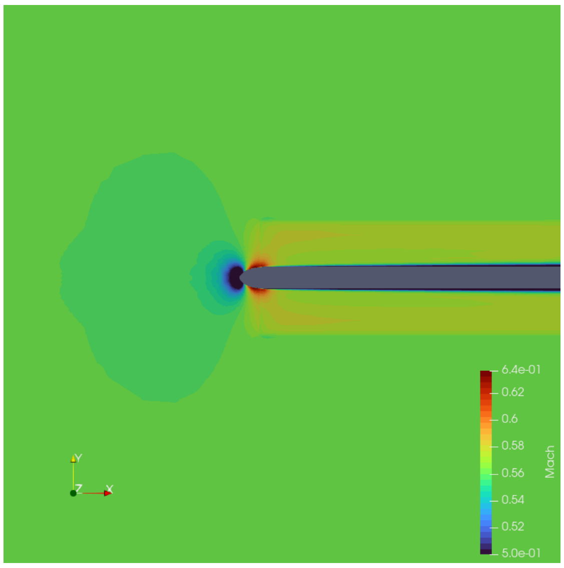


(a) *Velocity contour from the tutorial*

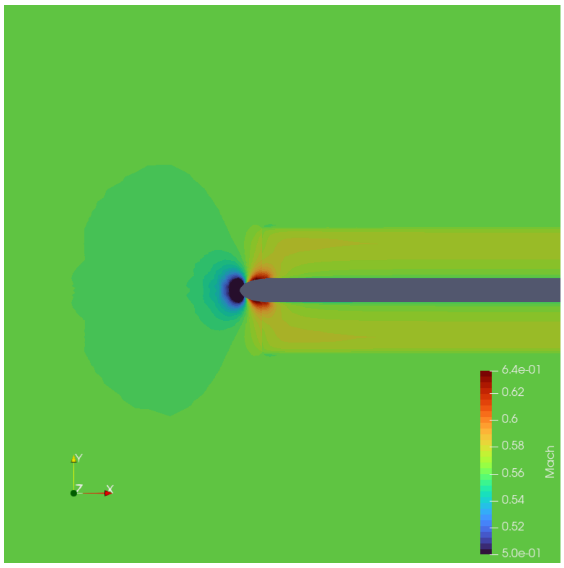
(b) *Velocity contour calculated with Euler*

Figure 5.8: *Velocity contour comparison between the tutorial and the case calculated with Euler*

In the Euler calculation it is possible to observe the absence of the boundary layer, that because it is based on a non-viscous theory, but if you look closely, you can see that this is the only difference between the two simulations.

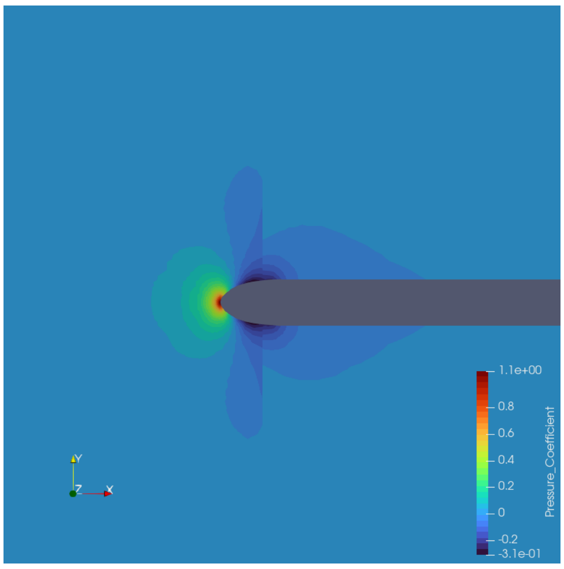


(a) *Velocity contour from the tutorial*

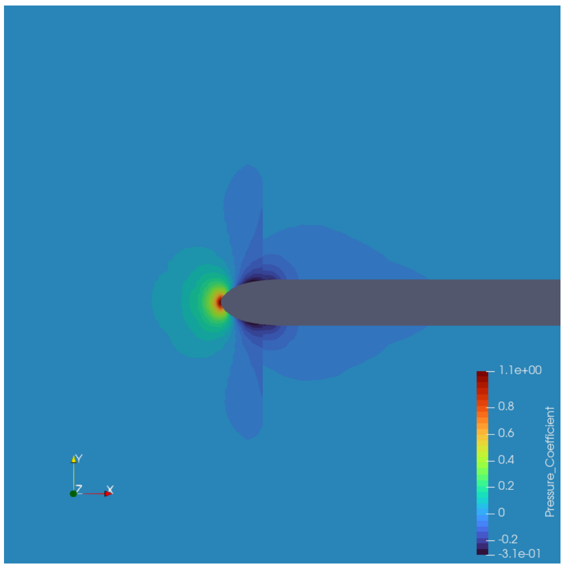


(b) *Velocity contour calculated with Euler*

Figure 5.9: *Velocity contour comparison between the tutorial and the case calculated with Euler*



(a) *Pressure coefficient contour from the tutorial*



(b) *Pressure coefficient contour calculated with Euler*

Figure 5.10: *Pressure coefficient contour comparison between the tutorial and the case calculated with Euler*

Again a distribution of the pressure coefficients was plotted along a line placed at 75% of the radius (in this case it is important to avoid the presence of the boundary layer, which could distort the results).

The results of the two calculations are very similar, the two lines almost overlap.

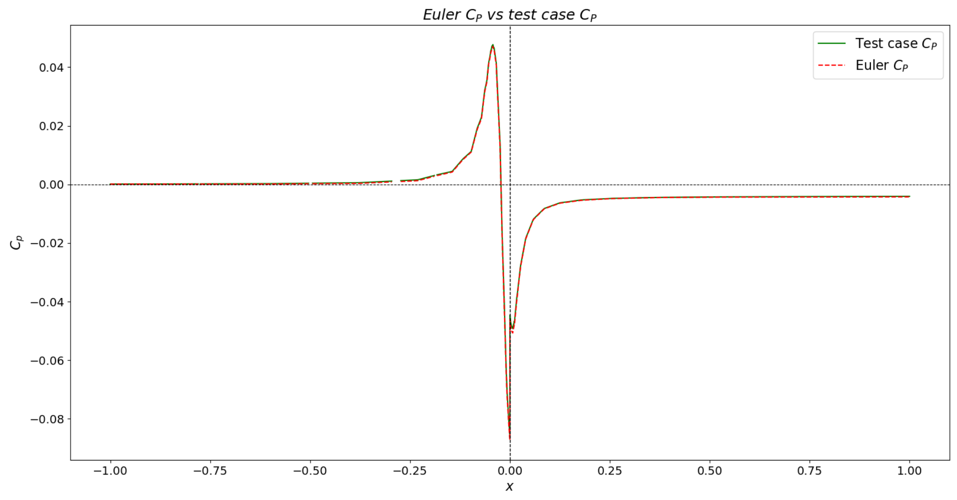


Figure 5.11: *Pressure coefficient plot Euler comparison*

The pressure coefficient in both cases increases near the leading edge of the spinner due to the flow shutdown and because of the high free stream velocity (M=0.56) the effect is greater. As expected from theory, the pressure coefficient decreases after the stagnation point. The coefficient then increases again at $x = 0$, where the disk is located, before returning to the free stream pressure value.

5.2 Simple rotor

After validating the Actuator Disk module, the initial step involved repeating the computation of a basic rotor similar to the SU2 test case but using CEASIOMpy to generate the actuator disk. This was made possible by the integration of the Actuator Disk module into the SU2_Run module. The simulation was done by following these steps:

1. The CPACS geometry was imported into CEASIOMpy. CPACS, as already mentioned, is a conceptual CAD designer, thanks to which it is possible to design an aircraft. For example, the presence of the wing is mandatory, so the propeller was designed with a small wing.
2. The workflow has been chosen, in particular, the modules activated were $CPACS2GMSH$ and $SU2\_RUN$
3. Parameter settings
4. Calculation

The mesh settings were the following:

Mesh Data	
	Refinement [m]
Engine	0.05
Fusulage	/
Propeller	0.05
Farfield	1
Wings	0.05

Table 4: *Mesh sizing*

The mesh obtained can be observed in the Fig. 5.12.

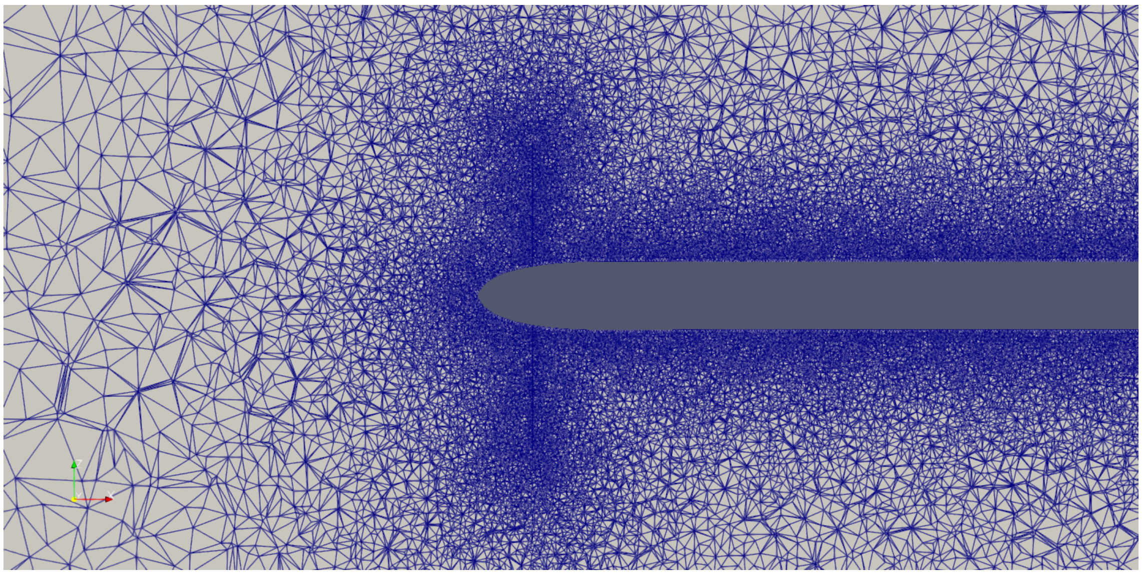
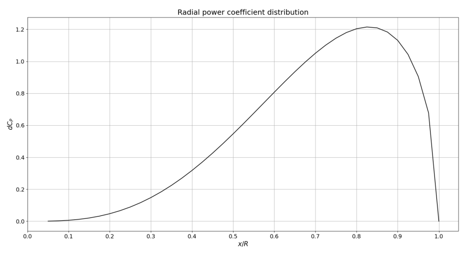


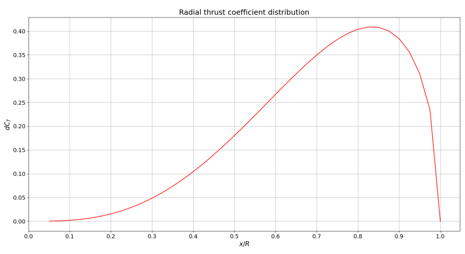
Figure 5.12: *Unstructured mesh of the disk made with GMSH*

The resultant mesh is not the best, considering that it is unstructured and, above all, that it is generated automatically, with only the possibility of selecting the degree of refinement of the various parts.

Once the mesh has been generated, the file containing the distribution of thrust and power coefficients is generated for use in the actuator disk calculation.



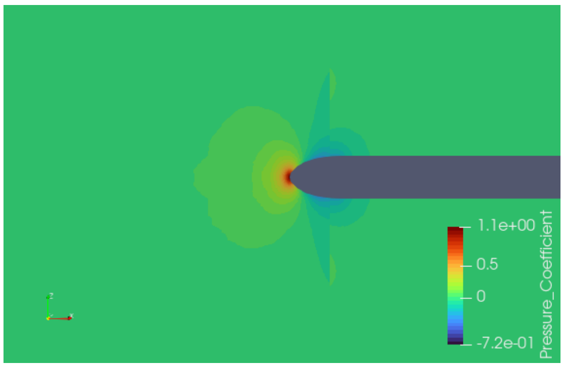
(a) *Radial power coefficient distribution*



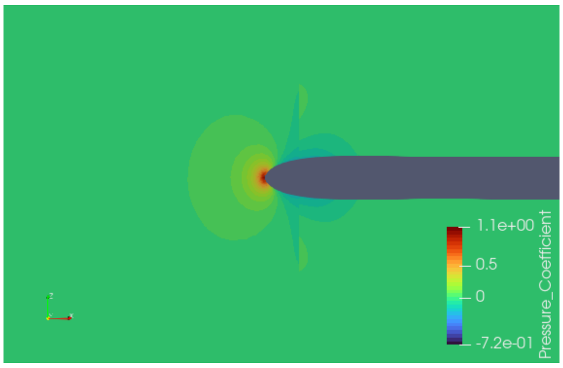
(b) *Radial thrust coefficient distribution*

Figure 5.13: *Radial distributions of power and thrust coefficients*

Once the thrust distribution is know the CFD calculation can be started.



(a) *Pressure coefficient contour from SU2 test case*



(b) *Pressure coefficient contour from CEASIOMpy*

Figure 5.14: *Pressure coefficient comparison*

The pressure coefficient contours, as can be observed, are very similar, the main differences can be noted downstream the disk, where the distribution of C_P is a bit different. This is probably due to the imperfect convergence of the calculation, as can be observed from the following plot.

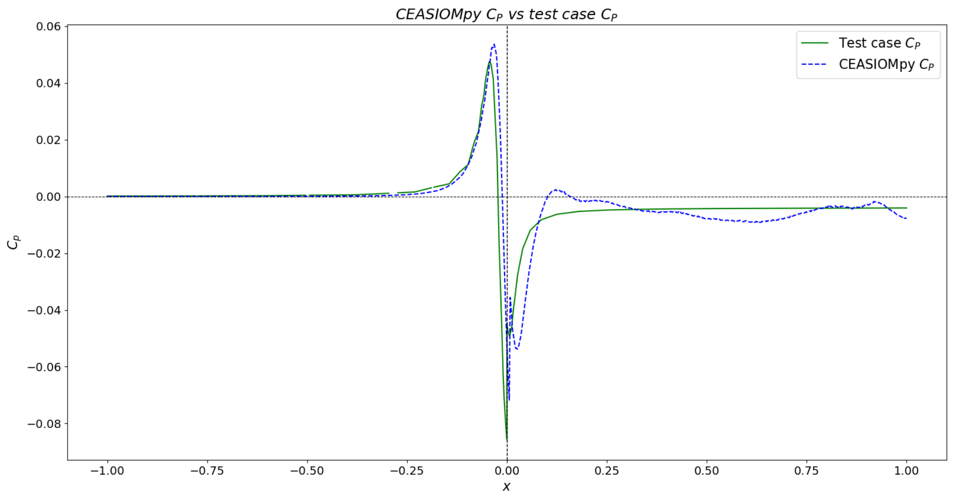


Figure 5.15: *Pressure coefficient comparison between SU2 original test case and CEASIOMpy*

In this second plot, the mesh generated by GMSH is compared with the one from the test case, the main difference from the previous case being that in both cases the .dat file was generated using CEASIOMpy.

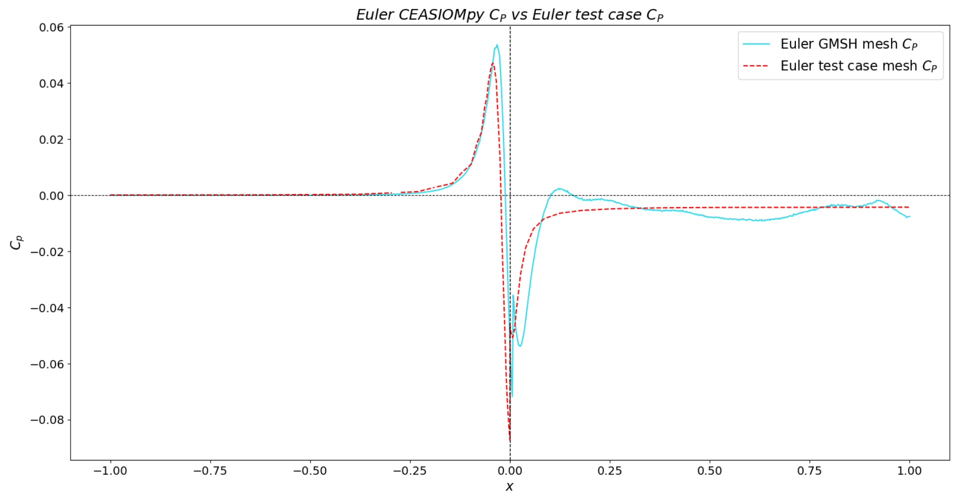


Figure 5.16: *Pressure coefficient comparison between SU2 Euler test case and CEASIOMpy*

Downstream of the disk some oscillations of the pressure coefficient can be noted, but the mean value remains consistent with the test case trend.

Overall, the results obtained can be considered reliable and the tool can therefore be said to be quite accurate.

5.3 Wing

For a more in-depth analysis, a simulation was carried out on a rotor with its associated wing to observe the behaviour of the model taking into account the presence of other parts, such as the wing.

The geometry of the aircraft was created using CPACSCreator. The design consists of a simple tapered wing with an engine positioned at the end of the wing. The design was initially created for a single wing, but symmetry was applied to obtain the geometry for both wings.

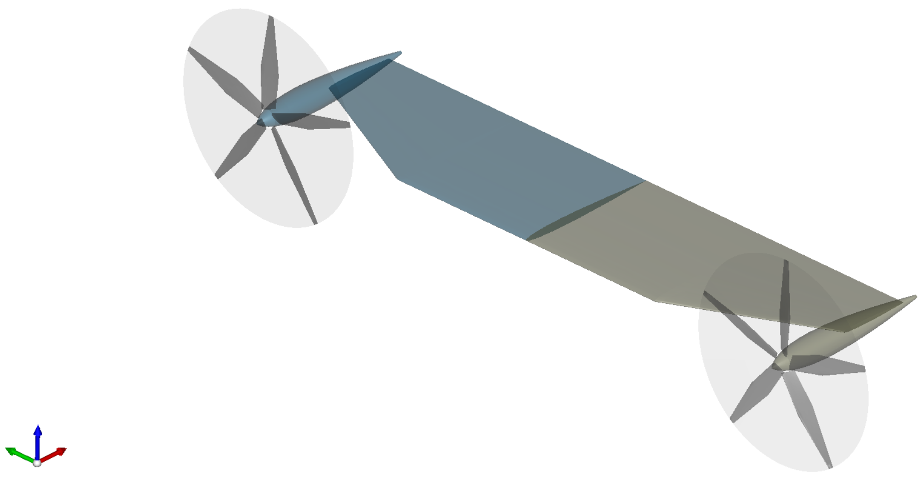


Figure 5.17: *Wings and rotors*

It has been considered to choose the following parameters:

	Data	**U.o.M.**
Cruise Mach number	0.3	[ ]
Rotational velocity	55	[1/s]
Thrust	4000	N
Altitude	0	[m]
Propeller radius	0.66	[m]
Number of blades	5	[ ]

Table 5: *Initial parameter of simulation*

For the sake of simplicity, an altitude of 0 m has been considered, which by convention implies a density of $\rho = 1.25$, then a Mach number of 0.3 has been chosen, given the average speed of a turboprop aircraft.

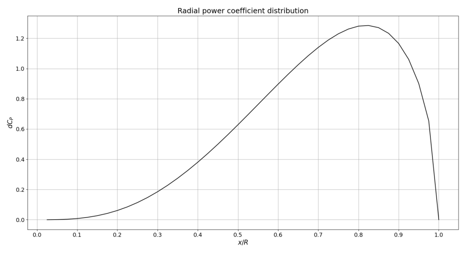
The mesh settings chosen were the following:

Mesh Data	
	Refinement [m]
Engine	0.05
Fusulage	/
Propeller	0.05
Farfield	1
Wings	0.05

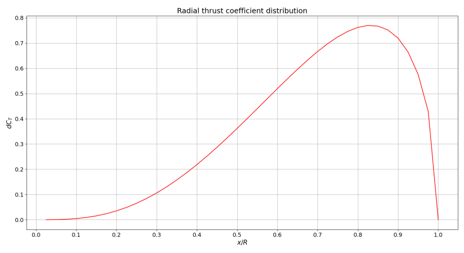
Table 6: *Mesh sizing for the wing case*

In this case the mesh is set similarly to the single rotor case, taking into account the acceptable results obtained in the previous single rotor case.

The distributions of the thrust and power coefficients along the radius are generated again.



(a) *Radial power coefficient distribution*



(b) *Radial thrust coefficient distribution*

Figure 5.18: *Radial distributions of power and thrust coefficients*

Then the CFD calculation is performed.

It is possible to visualise the velocity contour from a frontal point of view, as can be seen in the Fig. 5.19, the velocity trend follows the theoretical and the previous simulation trend.

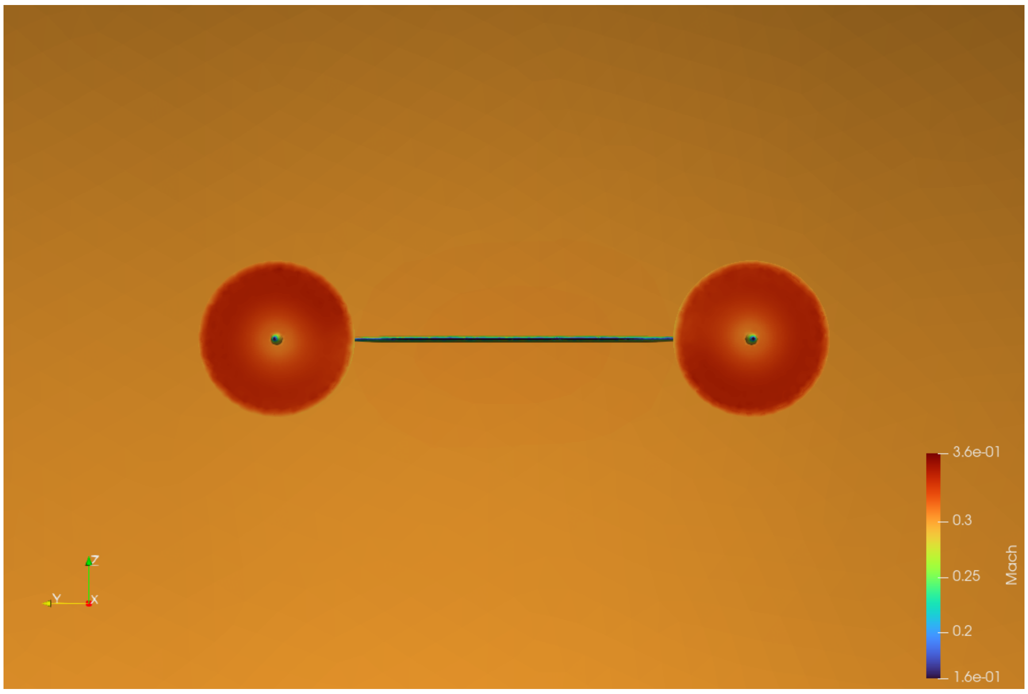


Figure 5.19: *Velocity contour of the wing with rotor*

Doing a zoom on the disk and changing the scale it is possible to better visualize how the velocity is distributed along the radius

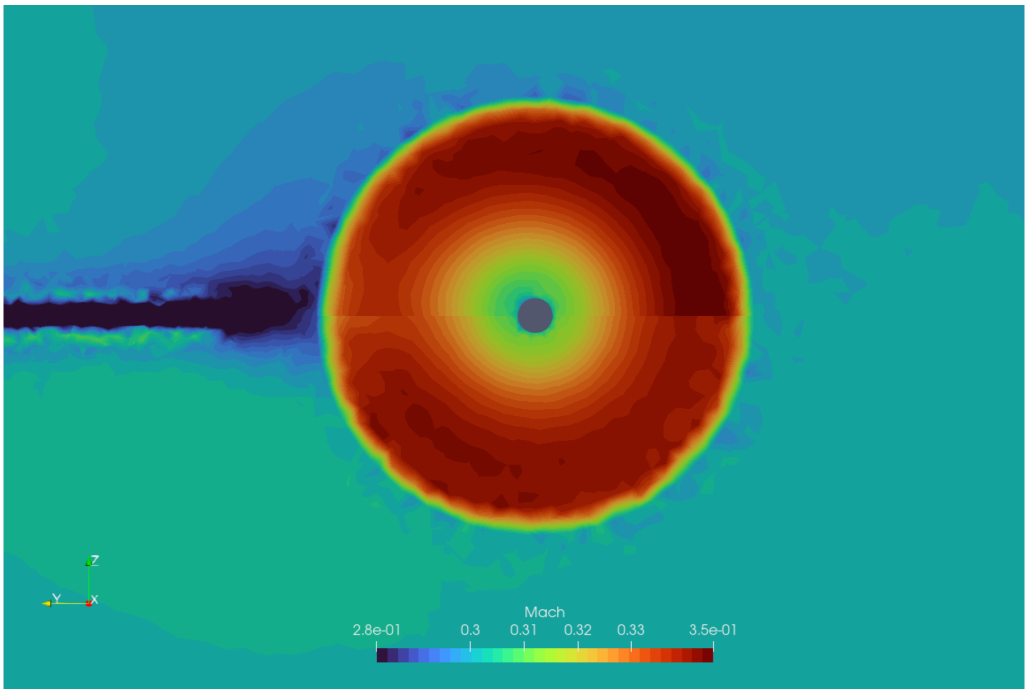
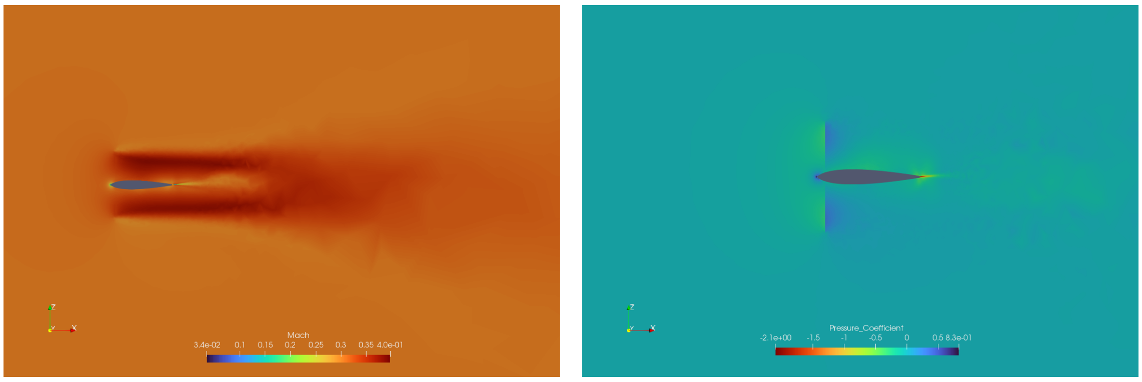


Figure 5.20: *Velocity contour with a zoom on the disk*

If instead a plane parallel to the axes of the pylons is cut, it is possible to see how the velocity and C_P vary upstream and downstream of the disk.



(a) *Velocity contour* **(b)** *Pressure coefficient contour*

Figure 5.21: *Contours of velocity and pressure of the wing with two disk*

Then a contour of the momentum along the z-direction can be visualized below, from Fig. 5.22 and Fig. 5.23 the swirl of the flow can be observed.

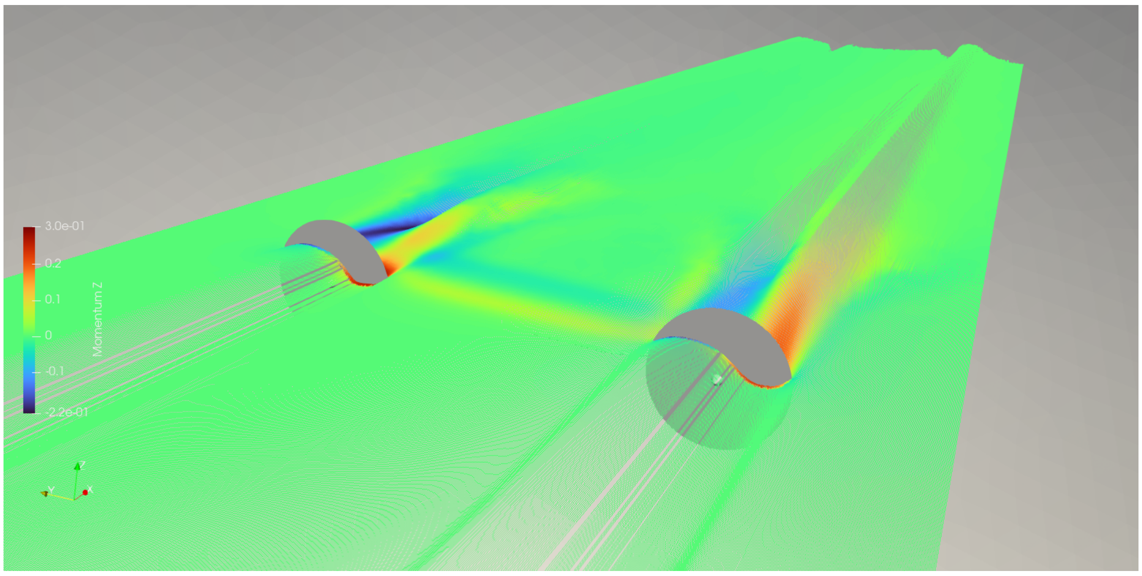


Figure 5.22: *Z Momentum of wing*

To better visualize the rotation is possible to look at the single rotor from a frontal view:

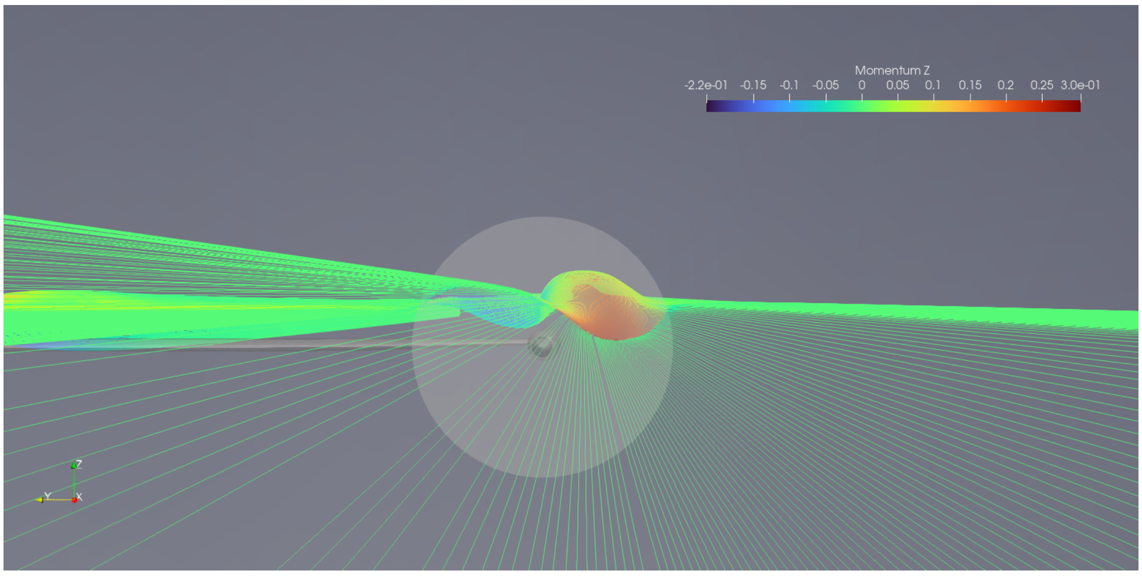


Figure 5.23: *Z Momentum*

The disk is rotating counterclockwise

Regarding the pressure coefficient trend:

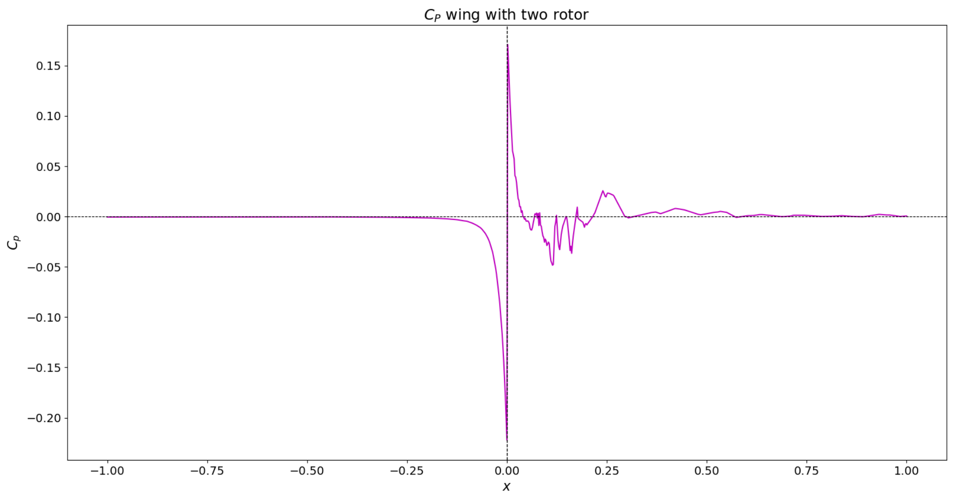
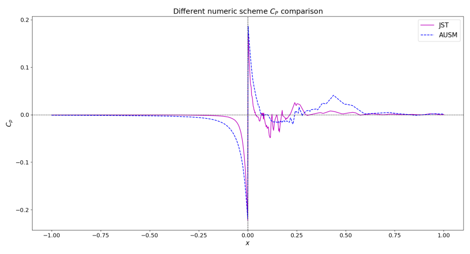
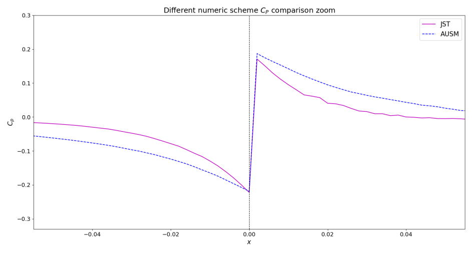


Figure 5.24: *C_P trend at $y = 0.3$*

The simulation has converged, but not optimally, as can be seen from the plot of the C_P values. In particular, there are some oscillations downstream of the disk, probably due to either the unstructured mesh or the numerical scheme used.
A more detailed analysis was therefore carried out. First, the JST (Jameson-Schmidt-Turkel) scheme was used as the space discretization method for flow simulation. Then the AUSM (Advection Upstream Splitting Method) was used to see if this method produced better results.



(a) *Pressure coefficient contour numeric scheme comparison*



(b) *Zoom of the pressure coefficient near the disk*

Figure 5.25: *C_P comparison using two different numeric schemes*

So from what can be seen in Fig. 5.25 the trend is similar and when looking at the position of the disk, at $x = 0$, it can be seen that the pressure jump is almost the same. From these results it can be concluded that the oscillations are probably due to the quality of the mesh.

5.4 Plane section

After adding the wings, the next step was to add a fuselage section. The CAD model of Fig. 5.26 was made by DLR as a basic study case, and a simulation was done to study the behaviour and interaction between the different parts.

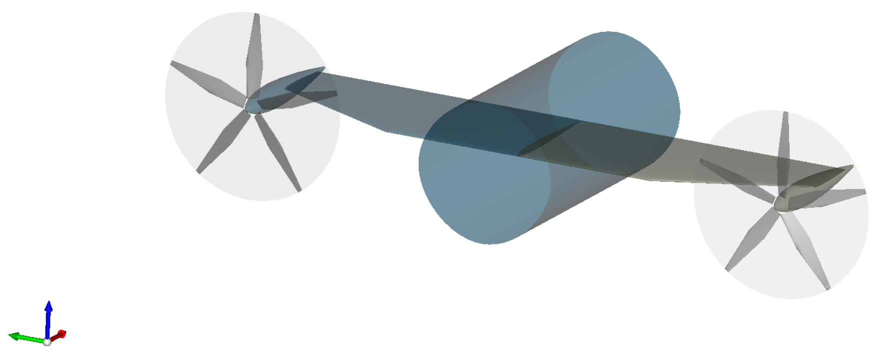


Figure 5.26: *Aircraft section*

Since the fuselage is cut sharply, it is possible to expect that the calculation will not converge optimally.

The simulation conditions used are shown in Tab.7, it has been chosen to keep the same condition as the previous simulation so that the results can be compared.

	Data	**U.o.M.**
Cruise Mach number	0.3	[/]
Rotational velocity	55	[1/s]
Thrust	4000	N
Altitude	0	[m]
Propeller radius	0.66	[m]
Number of blades	5	[/]

Table 7: *Initial parameter of simulation*

The distribution of thrust coefficient and the distribution of power coefficient are the same of Fig. 5.18

To ensure optimal convergence of the simulation, we set it to run up to 10000 iterations. However, due to the added complexity of the fuselage, this alone was not sufficient. Therefore, we implemented a CFL number adaptation technique to achieve better convergence.

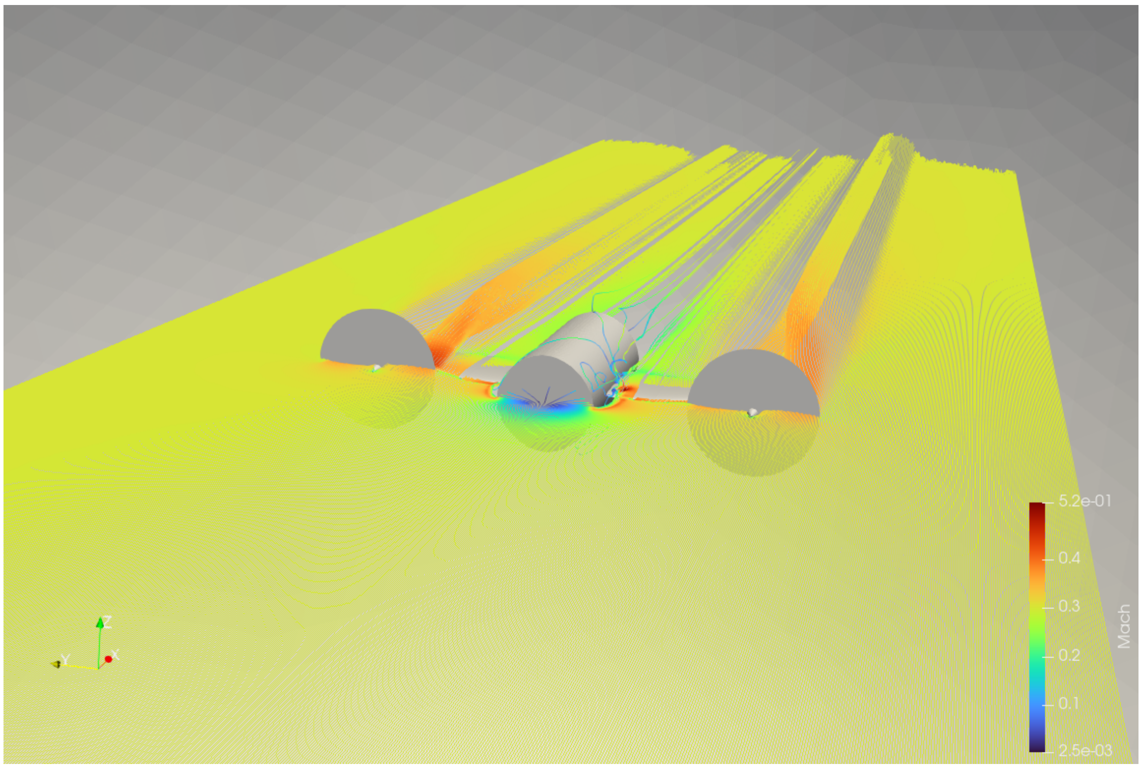


Figure 5.27: *Mach streamline*

In Fig. 5.27, it is possible to observe the streamline with Mach variation. The swirl after the disk can also be noted, and as predicted, the flow around the fuselage is not optimally converged.
Instead, observing the Mach contour of the disk, it can be seen that there are no problems.

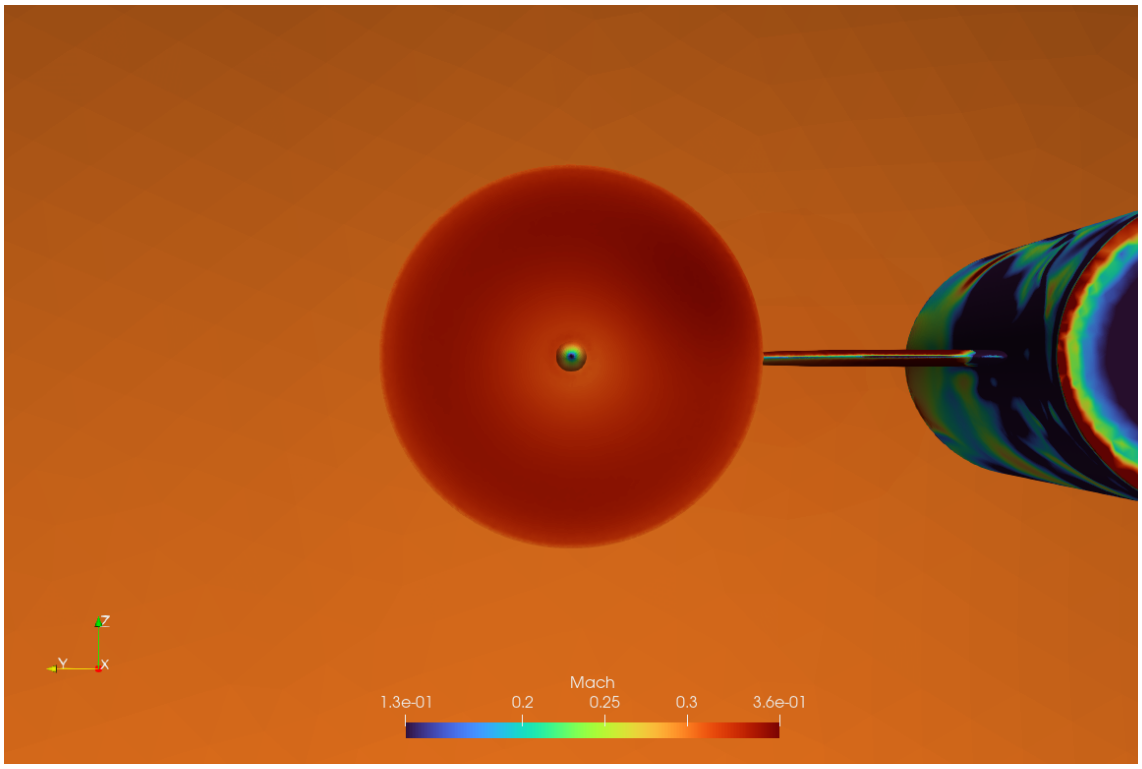
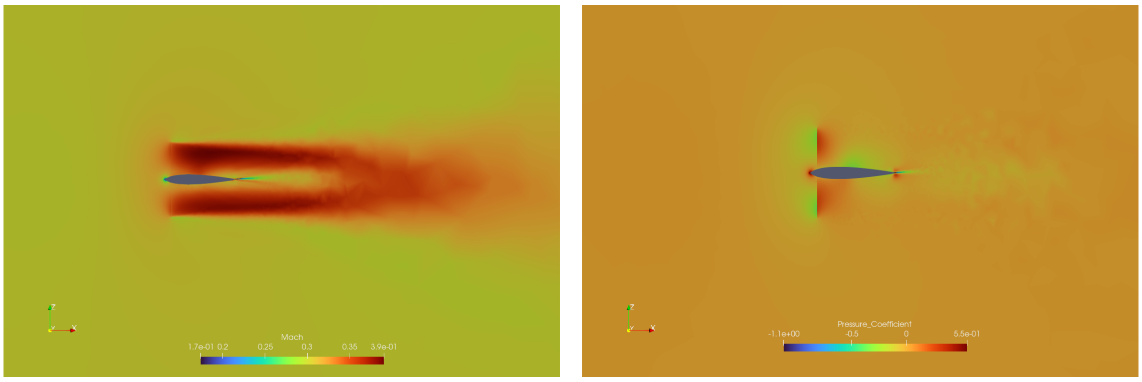


Figure 5.28: *Mach contour of the disk*

To visualize better the presence of the disk, the following contours can be seen.



(a) *Mach contour* **(b)** *Pressure coefficient contour*

Figure 5.29: *Mach and pressure coefficient contours*

The propeller induces a significant change in velocity, as shown by the Mach number increasing from M=0.3, the free stream velocity, to approximately 0.4. Additionally, the pressure drop resulting from the propeller's action can be easily observed from the contour shown above, but also from the Fig. 5.30, where the trend of the pressure coefficient along the x-axis is plotted.

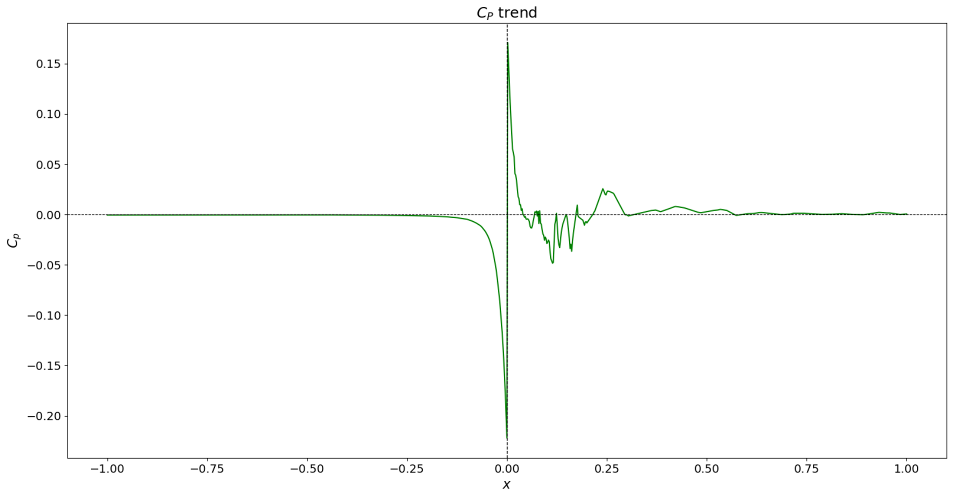


Figure 5.30: *Pressure coefficient plot at* $y = 0.3$

From the z-momentum streamline, it is possible to better observe the rotation of the fluid generated by the propeller.

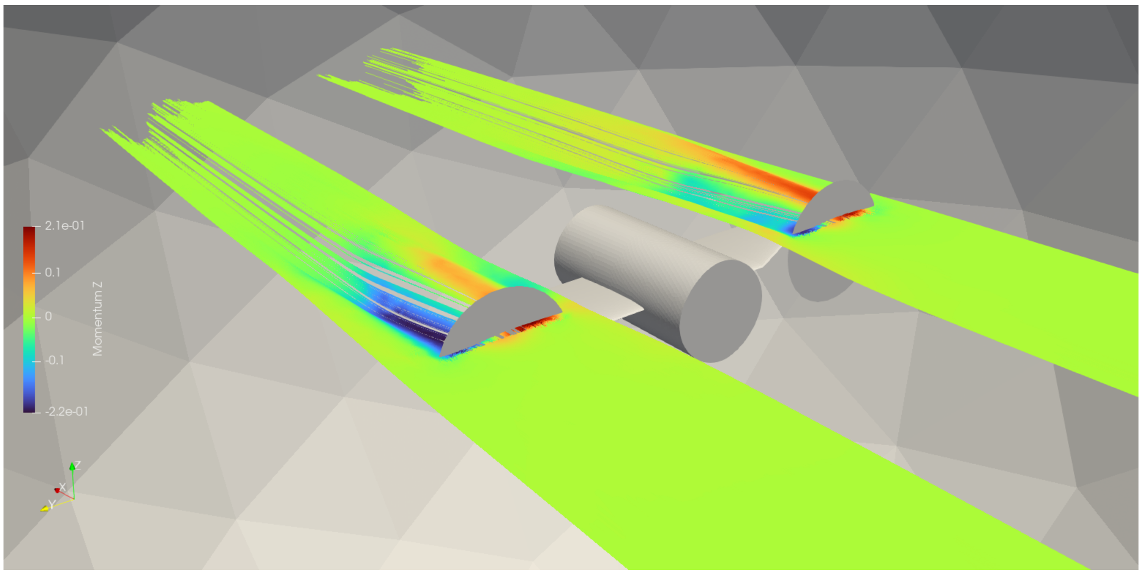


Figure 5.31: *Z Momentum*

A frontal view helps to better visualise the motion of the flow.

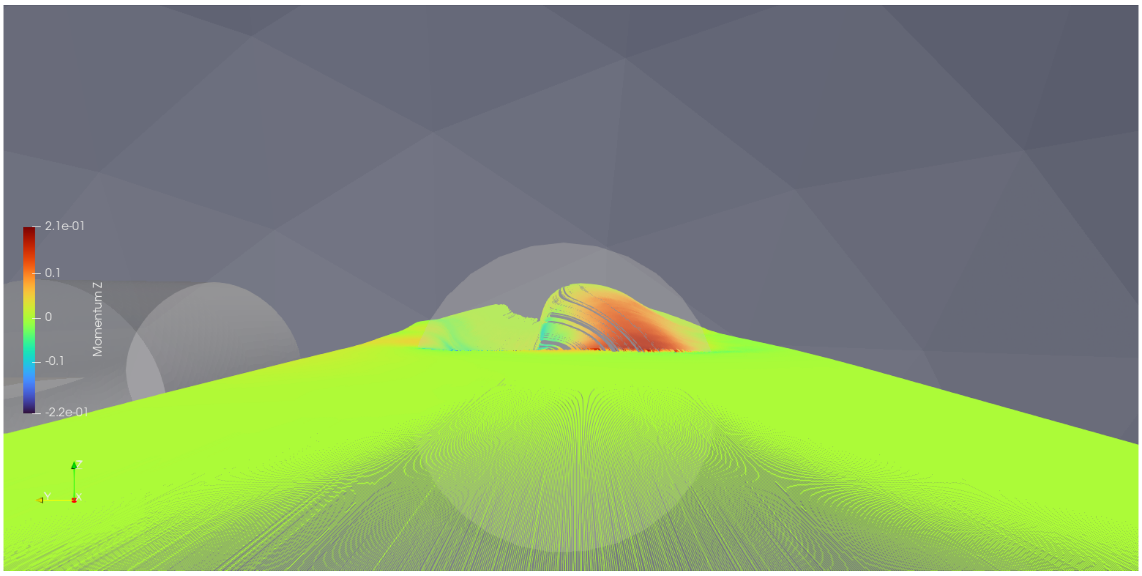


Figure 5.32: *Z Momentum*

5.4.1 Multirotor wing

To push the limits of the model, simulations with multiple rotors on each wing were performed. Two cases of particular interest were considered:

- Two-engine wing
- Five-engine wing

These simulations were conducted in order to investigate the behavior of the aircraft with multiple engines and to determine the impact of engine placement on overall performance.

Two-engine wing

The first simulation that was conducted involved a simple wing with two engines, where the propellers used were of the same size as the ones in the previous simulations.

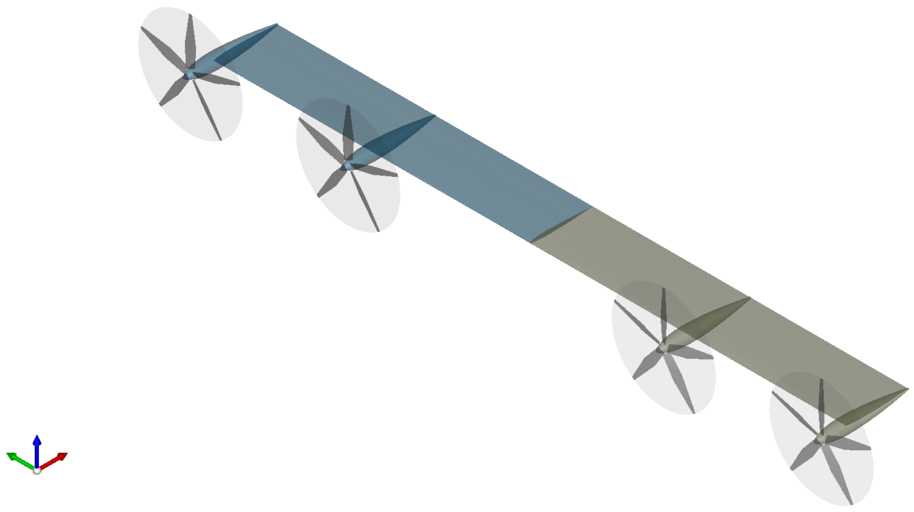


Figure 5.33: *CAD of double rotor wing*

Also the operative conditions were maintained

	Data	**U.o.M.**
Cruise Mach number	0.3	[/]
Rotational velocity	55	[1/s]
Thrust	4000	N
Altitude	0	[m]
Propeller radius	0.66	[m]
Number of blades	5	[/]

Table 8: *Initial parameter of simulation*

The simulation was then initialized, and the following results were obtained. The Mach contour reveals that there is an interaction of the flows from the different engines downstream of the disk.

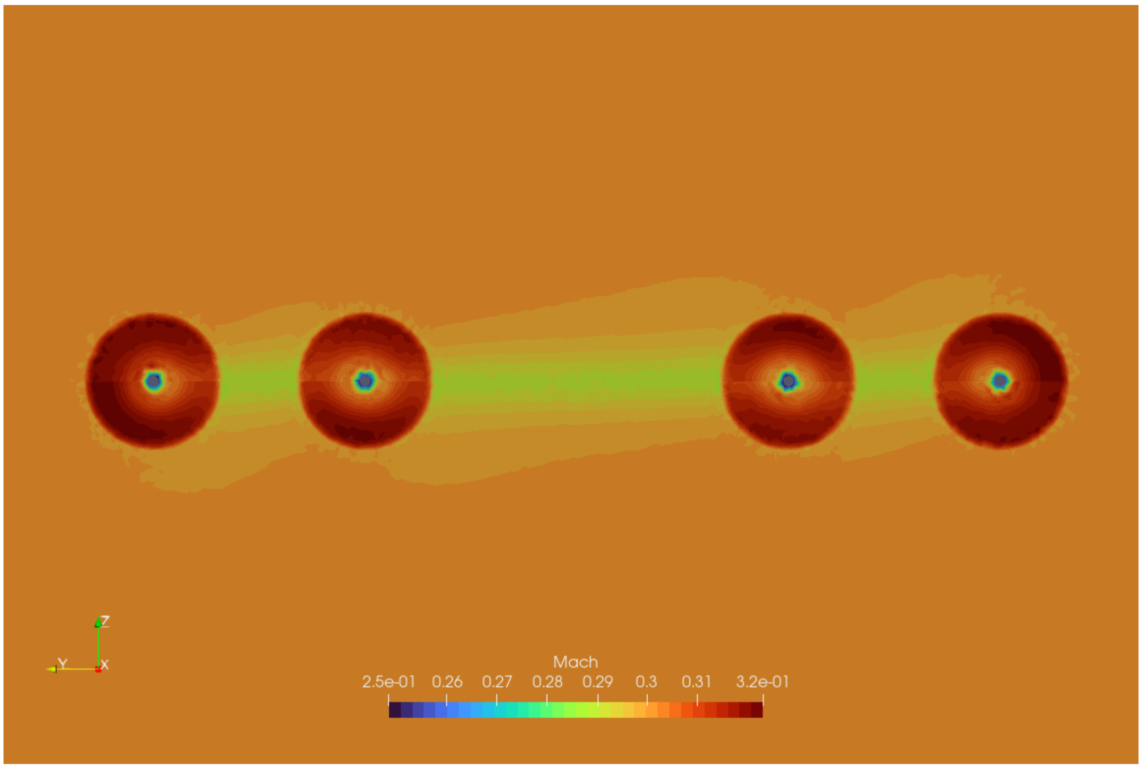
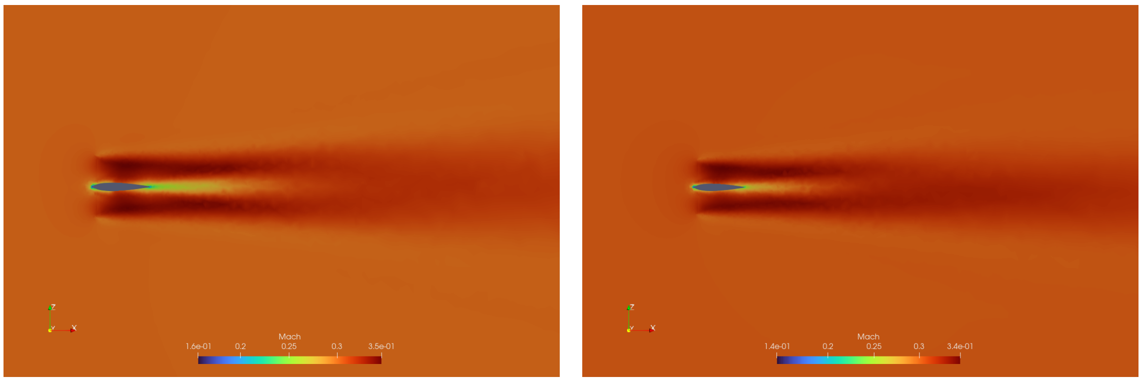


Figure 5.34: *Mach contour of double rotor wing*

In fact, even when observing the same Mach contour, but now on a plane parallel to the flow direction, the velocity field of the two engines are not exactly the same.



(a) *Mach contour of internal engine* **(b)** *Mach contour of external engine*

Figure 5.35: *Mach contour of engines*

The internal engine has a maximum Mach number of approximately 0.35 and a minimum Mach number of around 0.16. In contrast, for the second engine, the external one, the Mach number range is between 0.14 and 0.34. Therefore, what is significant is the variation in the distribution of the Mach number and the maximum value achieved. In Fig. 5.35a, the maximum Mach number is higher.

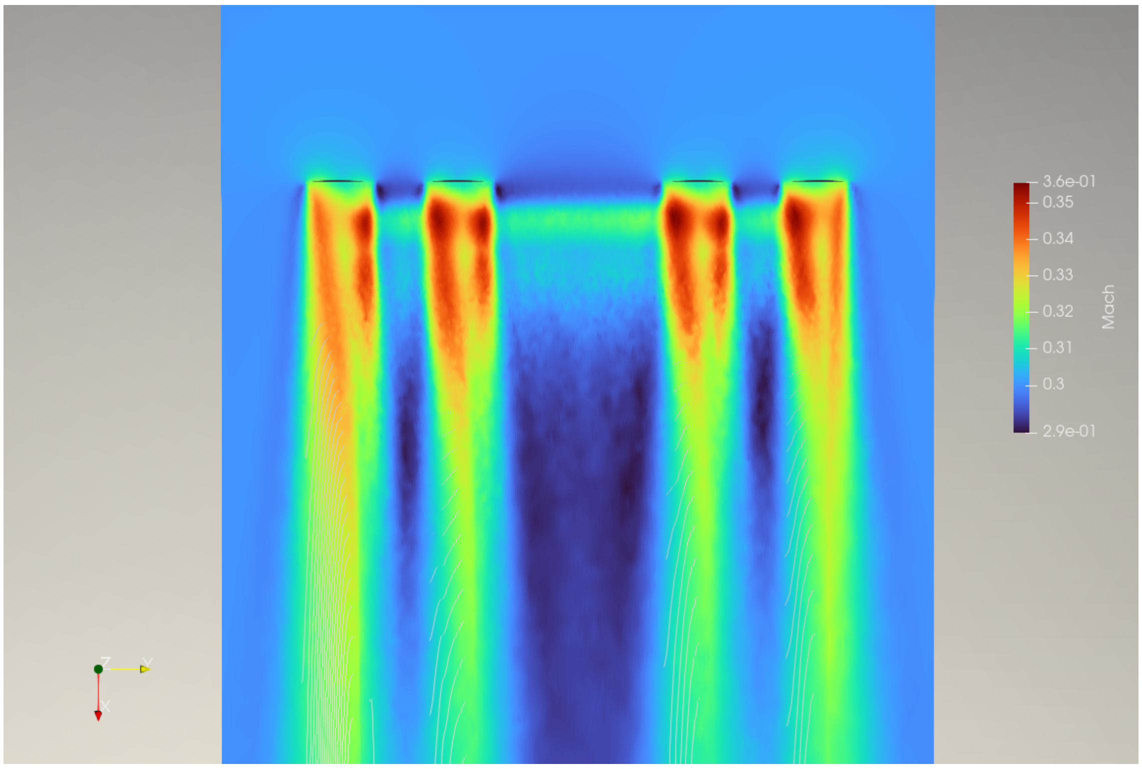
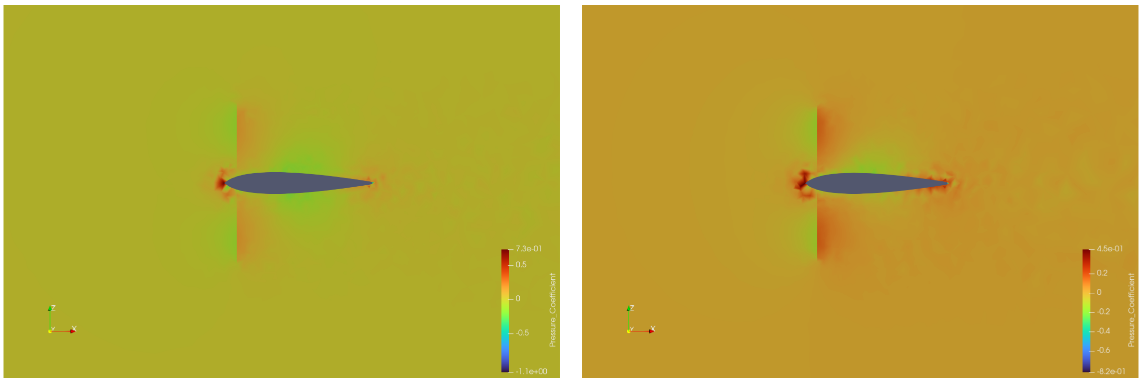


Figure 5.36: *Mach contour from above*

In Fig. 5.36, the differential Mach distribution is displayed, revealing some interesting insights. Overall the engines located on the right side of the aircraft exhibit similar contours to the engine on the left side. On the other hand, the internal engines exhibit a more similar Mach distribution, while the external ones display more differences. This phenomenon may be attributed to the influence of the wake interaction caused by the adjacent engines.

Regarding the pressure coefficient.



(a) *C_P contour of internal engine* **(b)** *C_P contour of external engine*

Figure 5.37: *Pressure coefficient contour of engines*

Again, there are some differences between the two engines. In Fig. 5.37b the pressure range is greater than in Fig. 5.37a, but the maximum pressure is lower. Additionally, the minimum C_P is lower than the pressure coefficient of the internal engine.

It is also important to consider the variation of z-momentum, as with the Mach and C_P contours, in order to fully understand the aerodynamic behavior of the aircraft. In particular, there are noticeable differences in the wake between each engine.

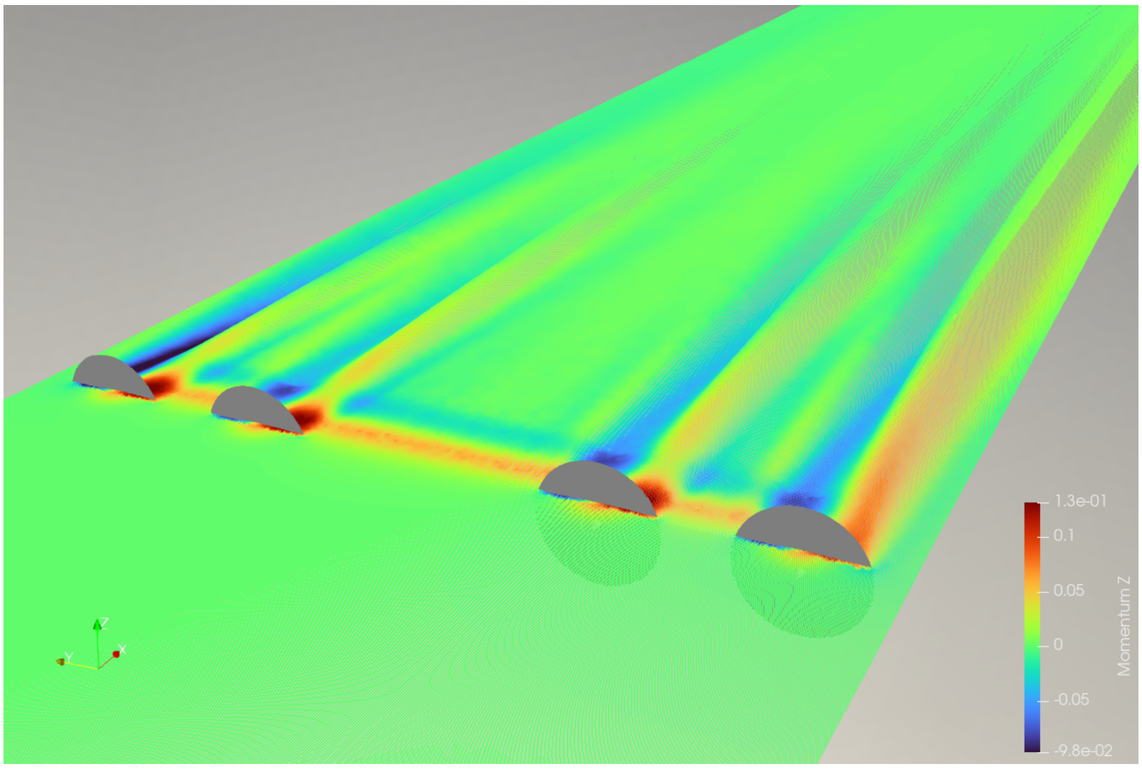


Figure 5.38: *Z momentum double rotor wing*

The plots of Mach and pressure coefficient variation along the x-axis confirm what has just been said.

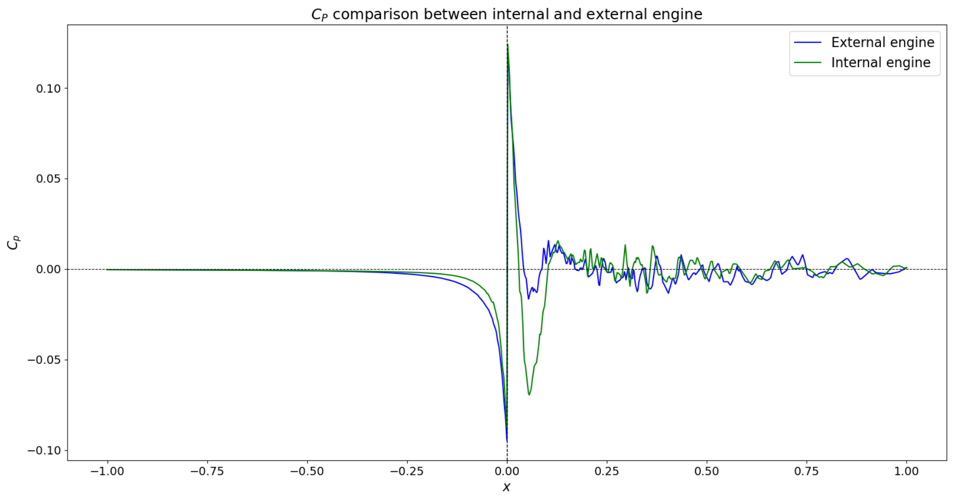


Figure 5.39: *C_P comparison between internal and external engine*

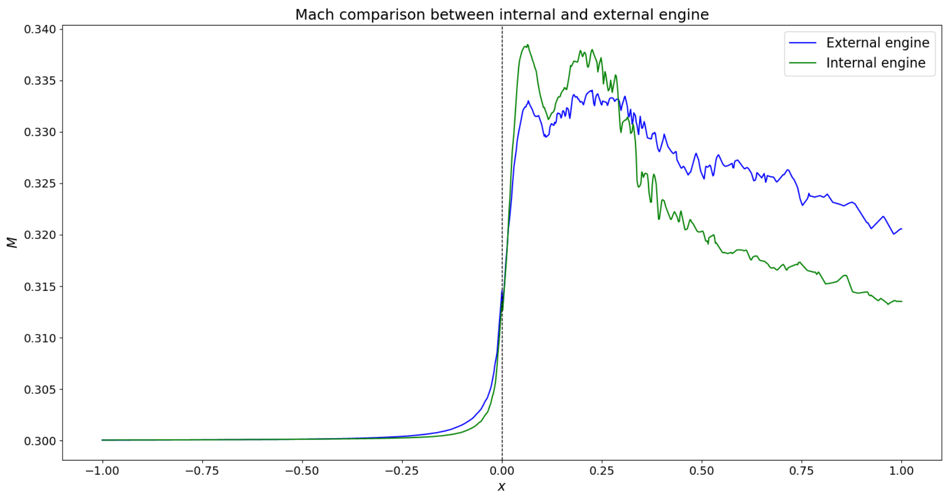


Figure 5.40: *Mach comparison between internal and external engine*

Five-engine wing

Ultimately, it was decided to add three additional engines to the previous wing and perform a simulation of a wing with five engines. This decision was made in line with research efforts focused on *Advanced Urban Mobility*, which involves rethinking traditional aircraft designs. Specifically, the objective is to develop electric planes for short-range distances initially, with the potential to explore additional capabilities in the future.

Expanding on the concept of *Advanced Urban Mobility*, the goal is to develop planes that can be utilized in various contexts. This includes the capability to take off and land on water, as well as the ability to extinguish fires. These planes would be designed to provide more flexible transportation options in urban areas and offer new solutions for emergency situations. By exploring innovative aircraft designs and technologies, it may be possible to revolutionize the way we think about air travel and its potential uses.
This is also the aim of the European COLOSSUS (Collaborative System of Systems Exploration of Aviation Products, Services and Business Models) project, which brings together a group of experts in various fields of aeronautical engineering to design this new type of plane. The focus of the project is on developing seaplanes and firefighting planes, which can be used in a variety of challenging environments and scenarios. By combining expertise from different fields, the project aims to create innovative designs and solutions to meet the complex demands of these applications

(a) *JEKTA electric seaplane*

(b) *Regent electric seaplane*

Figure 5.41: *Electric seaplanes*

Looking at the figures above, it was decided to carry out a simulation for a similar configuration. The design was inspired by the configuration shown in Fig. 5.41a, then a simple wing with 5 rotors was designed using CPACS creator.

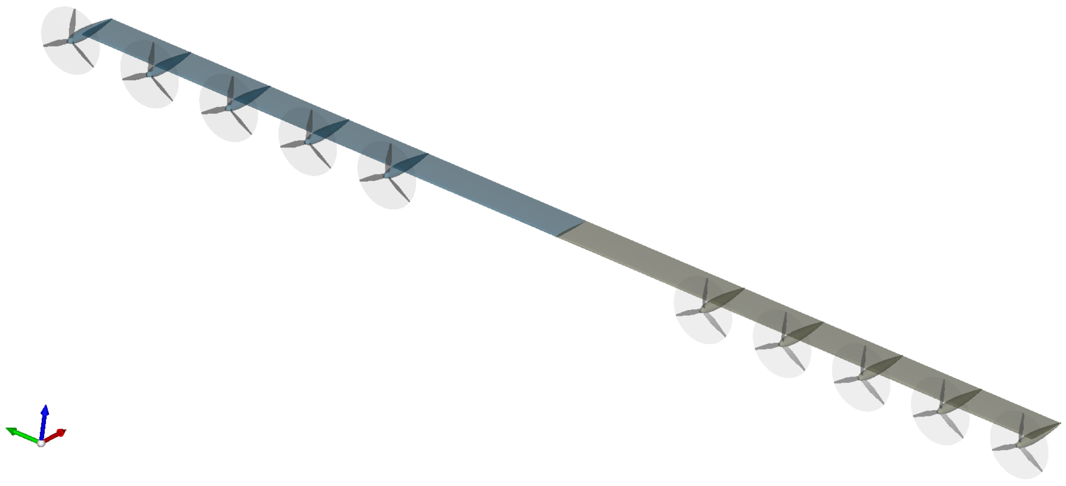


Figure 5.42: *Five-engine wing CAD*

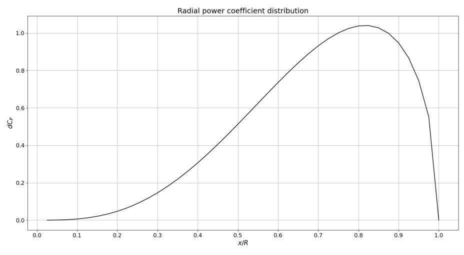
The operational and geometrical characteristics are as follows:

	Data	**U.o.M.**
Cruise Mach number	0.2	[/]
Rotational velocity	28.33	[1/s]
Thrust	3000	N
Altitude	0	[m]
Propeller radius	1	[m]
Number of blades	3	[/]

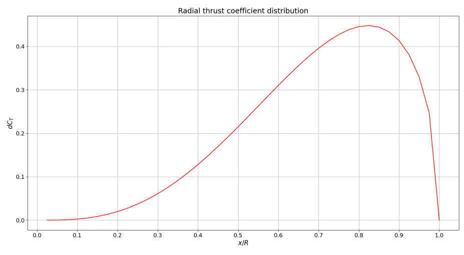
Table 9: *Initial parameter of simulation*

In this case will be important to study also the interactions between the disks, in particular downstream.

This time the conditions are different, so the distribution of thrust and power will be different.



(a) *Radial power coefficient distribution*



(b) *Radial thrust coefficient distribution*

Figure 5.43: *Radial distributions of power and thrust coefficients*

After running the CFD simulation, the results were found to be unsatisfactory. Due to the complexity of the geometry, it was decided not to pursue the analysis further at this time.

5.5 NSMB CFD comparison

It has been chosen of comparing the results given by SU2 and another software that uses a different way to calculate the presence of the propeller; in particular, NSMB was utilized. NSMB (Navier-Stokes Multi Block) is a CFD software developed by CFS Engineering through the years.

This time the CAD drawing for this simulation was created with OnShape [20], seen that some difficulties were encountered when trying to create an accurate mesh of the CPACS file. Consequently, the semi-infinite spinner was redrawn and the mesh was generated with ICEMCFD [17]. This was done because NSMB required a structured mesh, unlike the unstructured mesh used for simulations with SU2. In addition, the purpose of using a structured mesh was to achieve a more accurate calculation since the mesh is no longer generated automatically. The structured mesh was successfully created.

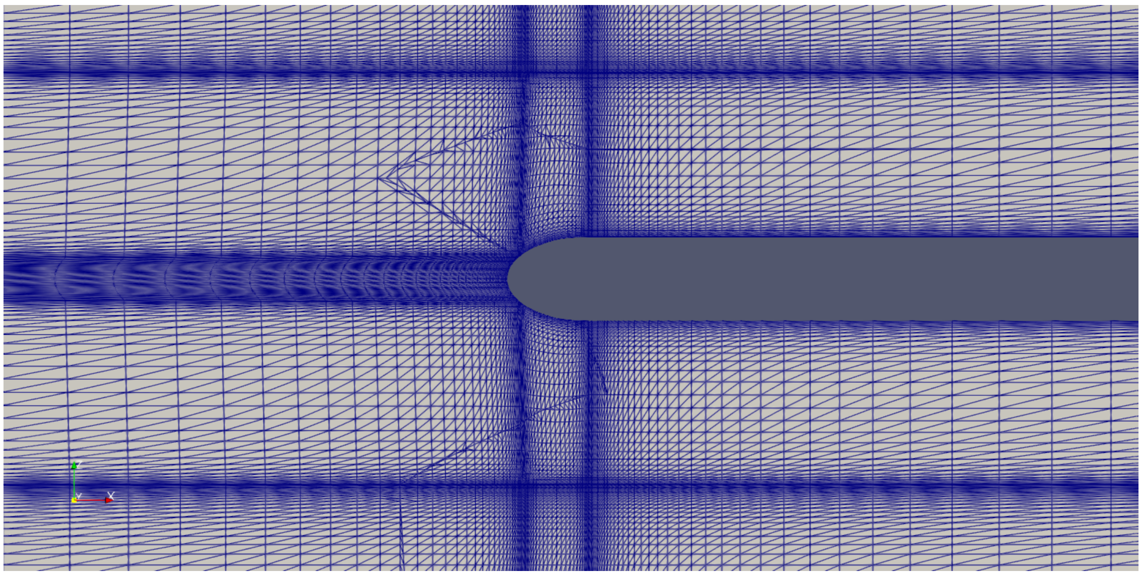
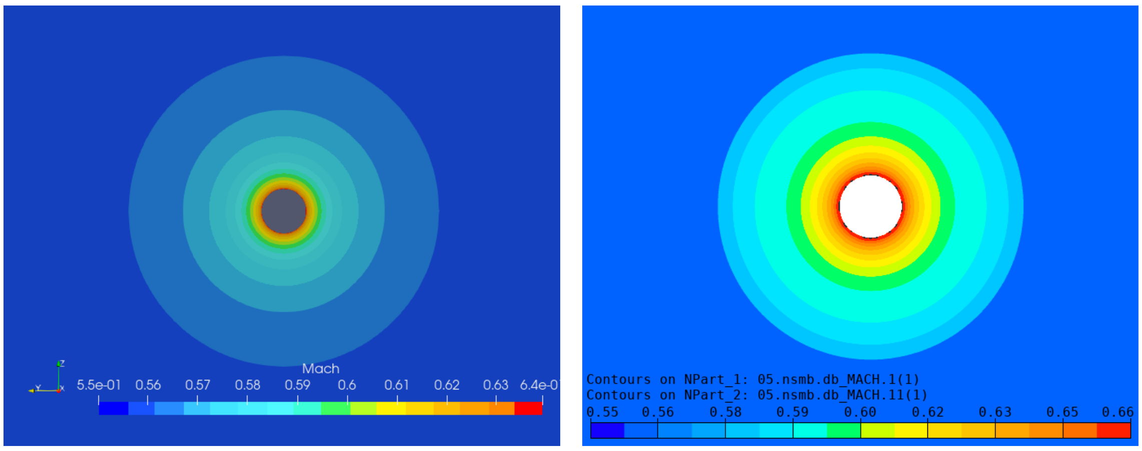


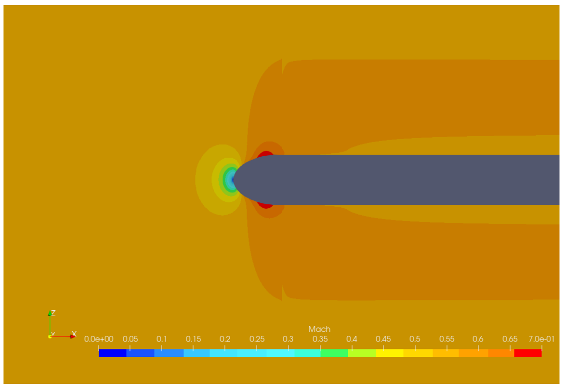
Figure 5.44: *Structured mesh, done with ICEMCFD*

Regarding the simulation in SU2, unlike the case with the GMSH mesh, the convergence is achieved without any difficulty and very quickly (less than 400 iterations).

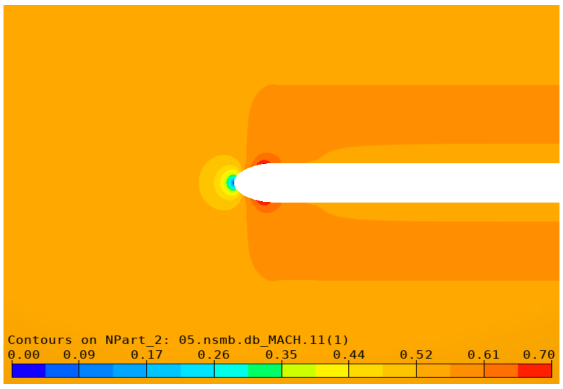


(a) *Front view of Mach contour calculated by SU2* **(b)** *Front view of Mach contour calculated by NSMB*

Figure 5.45: *Front view Mach contour comparison*

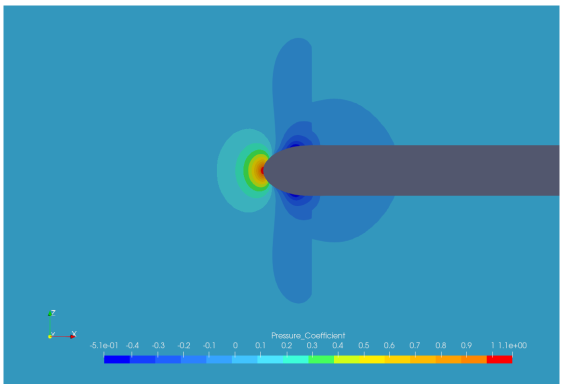


(a) *Side view of Mach contour calculated by SU2*

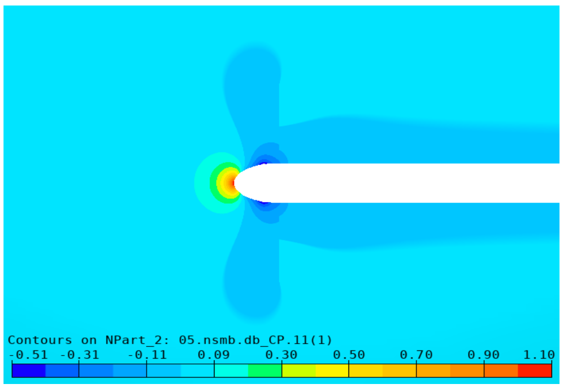


(b) *Side view of Mach contour calculated by NSMB*

Figure 5.46: *Mach contour comparison*



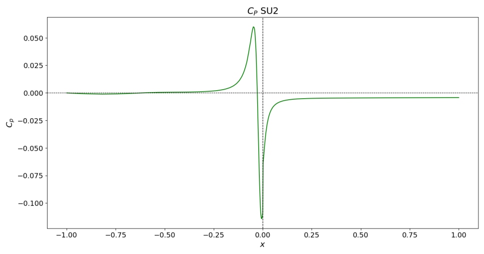
(a) *Side view of C_P contour calculated by SU2*



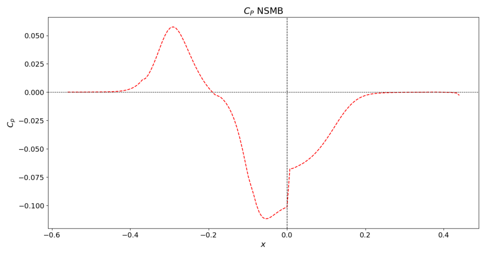
(b) *Side view of C_P contour calculated by NSMB*

Figure 5.47: *Pressure coefficient contour comparison*

Looking at the pressure coefficient plot, it can be noted some differences



(a) *C_P trend of the calculation made with SU2*



(b) *C_P trend of the calculation made with NSMB*

Figure 5.48: *Pressure coefficient plots*

The trend of the two curves is different, but the maximum and minimum values of C_P are the same.

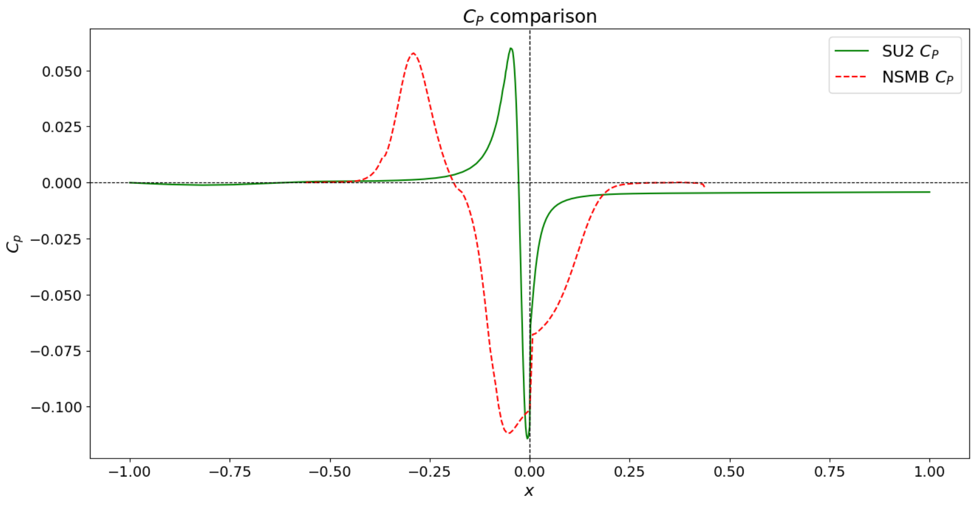


Figure 5.49: *Comparison of pressure coefficient*

When zooming close to x=0, i.e. close to the disk, it can be noticed that the pressure jump is very similar.

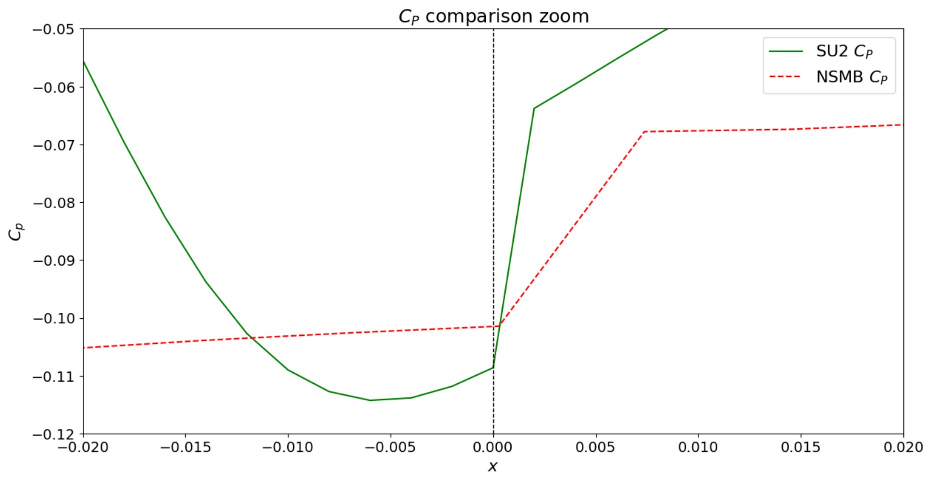


Figure 5.50: *Zoom of the comparison*

Seeing that the comparison gave some good results, also considering that the actuator disk calculator implemented in NSMB is simpler than the one used for the SU2 calculation.
Furthermore, looking at the results in Sect.5.1, it can be seen that there are not many differences between the contours.

6 Conclusion and future developments

The primary objective of this thesis was to integrate the presence of the propeller in CEASIOMpy in order to obtain a more comprehensive aircraft simulation. Several attempts were made to get the thrust distribution along the propeller blade. Initially, a script based on the Blade Element Theory was developed, then, considering the disadvantages of this theory, it was decided to switch to the Actuator Disk Theory and to use an existing script which was modified to integrate it into CEASIOMpy.

Once the integration of the model in CEASIOMpy was done, several CFD simulations were carried out to explore the capabilities and the limitation of the programme. The first results were considered acceptable, and the accuracy of the calculations was not compromised even when using the automatic GMSH tool.

It is important to note that upon further analysis, some of the results obtained were not entirely satisfactory. As a result, a more thorough investigation was carried out, with particular emphasis on the mesh. The automatically generated mesh, using GMSH, was compared with the structured mesh generated by ICEMCFD. In addition, the generation of the structured mesh allowed a calculation to be performed using NSMB, which only works with structured meshes. The results were then compared and significant improvements were observed. This analysis highlights the importance of carefully considering the mesh used for simulations as it can greatly affect the accuracy of the results.

In the future, the first step will undoubtedly be to improve the mesh in order to eliminate the problems associated with the oscillation of the pressure coefficient downstream of the disc. This improvement will have to be made without losing the current advantages such as low computational cost.

COLOSSUS
COLLABORATIVE SYSTEM OF SYSTEMS EXPLORATION
OF AVIATION PRODUCTS, SERVICES & BUSINESS MODELS

Figure 6.1: *COLOSSUS logo*

Looking ahead, the plan is to continue to develop the module to allow more complex and precise calculations in CEASIOMpy, but also to work on and improve the actuator disk model in NSMB as part of the European COLOSSUS project. The aim is to further increase the capabilities of the simulations and to continue to push the boundaries.

References

[1] R.D. Flack, 2005, *"Fundamentals of Jet Propulsion with Applications"*, Cambridge Aerospace Series

[2] Saeed Farokhi, 2014, *"Aircraft Propulsion"*, John Wiley & Sons, second edition

[3] J.D. Mattingly, 2002, *"Aircraft Engine Design"*, AIAA Education Series, second edition

[4] H. Glauert, 1935, *Aerodynamic Theory*, W.F. Durand ed. Springer, Airplane Propellers vol.4

[5] C. Hirsch, 1995, *Numerical Computational of Internal and External Flows*, John Wiley & Sons, vol. 1

[6] C. Hirsch, 1995, *Numerical Computational of Internal and External Flows*, John Wiley & Sons, vol. 2

[7] R. Tognaccini, 2016, "Aerodinamica dell'Ala Rotante"

[8] E. Saetta, L. Russo, R. Tognaccini, 2020, "Implementation and validation of a new actuator disk model in SU2"

[9] Andrea Poyer, 2012, "Methods for propeller simulation", FH Wiener Neustadt

[10] Gerrit-Daniel Stich, Luis Fernandes, Jared Duensing Gaetan Kenway, Jeffrey Housman and Cetin Kiris, 2022, "Validation of Actuator Disk, Actuator Line and Sliding Mesh Methods within the Structured Curvilinear Overset Solver", `https://www.nas.nasa.gov/assets/nas/pdf/ams/2022/AMS_20220825_Stich.pdf`

[11] "Blade Element Theory for Propellers", `http://www.aerodynamics4students.com/propulsion/blade-element-propeller-theory.php`

[12] "Software Components" `https://su2code.github.io/docs/Software-Components/`

[13] "Optimal Propeller" `https://github.com/su2code/SU2/blob/master/SU2_PY/OptimalPropeller.py`

[14] "CEASIOMpy, Open source conceptual aircraft design environment" `https://github.com/cfsengineering/CEASIOMpy`

[15] "Gmsh", Christophe Geuzaine and Jean-François Remacle, 2009, `http://gmsh.info/`

[16] "CPACS, Common Parametric Aircraft Configuration Schema" `https://www.cpacs.de/`

[17] ANSYS Inc. (2019). ICEM CFD 19.2.0 [Computer software]. Houston, TX: ANSYS Inc.

[18] Martin Siggel, Jan Kleinert, Tobias Stollenwerk & Reinhold Maierl, 2019, "TiGL: An Open Source Computational Geometry Library for Parametric Aircraft Design"

[19] pyfoil, `https://pypi.org/project/pyfoil/`

[20] OnShape, `https://www.onshape.com/en/`

[21] XFLR5, `http://www.xflr5.tech/xflr5.htm`

[22] SU2, `https://su2code.github.io/`

[23] Streamlit, `https://github.com/streamlit/streamlit`

[24] Airinnova `https://airinnova.se/`

[25] CFS Engineering, `https://cfse.ch/`

[26] Surface Modeler SUMO, `https://www.larosterna.com/products/open-source`

[27] pyTornado, `https://github.com/airinnova/pytornado`