Post Flight Drag Coefficient Analysis of a Model Rocket Using Ansys Fluent

Tommaso Bocchietti

A.Y. 2023/24 - W24

Personal project report.

Instructor: Prof. Fue-Sang Lien
Course: ME663 - Computational Fluid Dynamics
Term: Winter 2024

Contents

List of Figures

List of Tables

Listings

1 Introduction

Model rocketry is a popular hobby that involves the design, construction, and launch of small rockets powered by solid fuel engines.
Even though, at this scale, models are significantly simplified compared to full-scale rockets, they still can be used to study the principles of aerodynamics, propulsion, and flight dynamics. One of the key parameters that affect the performance of a model rocket is the drag coefficient C_d.
The drag coefficient is a dimensionless quantity that characterizes the aerodynamic drag of an object moving through a fluid. It is defined as the ratio of the drag force acting on the object to the dynamic pressure of the flow around the object.

$$C_d = \frac{F_d}{\frac{1}{2}\rho v^2 A} \tag{1}$$

The drag coefficient of a model rocket depends mainly (but not only) on its shape, flow conditions, and properties of the fluid. In general, it cannot be considered a constant value and may vary significantly during the flight of the rocket.
However, for a small model rocket with a 'classical design', it has been observed that the drag coefficient typically ranges between 0.7 and 0.8. This value is based on empirical data and is used as a rule of thumb during the design and analysis phases.
In this project, we aim to analyze the drag coefficient of one of our model rockets using `Ansys Fluent`, a popular commercial software used for computational fluid dynamics (CFD) simulations. Due to inexperience with the software and in general the world of simulations, we will have the necessity to make some strong assumptions and simplifications that might affect the accuracy of the results.
At the end of the project, we will try to validate the CFD simulation results against predictions made using another software for model rocket design and simulation, and against actual flight data collected during a previous launch.
The model rocket analyzed in this project is shown in Figure 1.

Figure 1: Picture of the model rocket analyzed in this project. Photo of the author (July 21, 2023).

2 Methodology

This project consists of three main steps: CAD modeling, CFD simulation, and analysis of the results. In this section, we will briefly describe each step and explain the tools and techniques used to carry out the analysis.

CAD modeling The first step in the analysis is to create a CAD model of the rocket. The model is used to define the geometry that will be analyzed in the CFD simulations.
For this purpose, we opted for `CATIA V5`, a computer-aided design (CAD) software package from `Dassault Systèmes`.

CFD simulation The second step in the analysis is to set up and perform the CFD simulations of the model rocket. In this project, CFD simulations are used to calculate the drag coefficient (C_d) and obtain insights into the aerodynamic performance of the rocket. It has to be noted that simulations will be performed in a steady-state regime, and considering as velocity of the fluid 85% of the peak one that the rocket reaches during the real flight. Considering the drag coefficient as a constant value across the whole range of velocities it's a strong assumption, and should be avoided in a real-world scenario.
The CFD simulations were performed using `Ansys Fluent`, a computational fluid dynamics (CFD) software package from `Ansys Inc.`. This was our first time using a commercial CFD software package, and we found it to be a powerful tool for simulating complex fluid flow problems.

Analysis and validation of the results The final step is to try to validate the results of the CFD simulations. To do so, we will use `MATLAB` to code a simple physics model of the rocket's flight and compare it against two different sources of data:

- `OpenRocket` simulations
- Actual flight data of the rocket, collected during a previous launch

`OpenRocket` is an open-source software for model rocket design and simulation. It allows to select the engine and set the material properties of the rocket. By considering a mean value of the drag coefficient, the software is then able to predict the flight and the apogee altitude.
The actual flight data of the rocket were collected using a barometric altimeter. Unfortunately, the data are very a few, due to the short duration of the flight and the low sampling rate of the altimeter. However, as we will see, they still can be useful to validate the CFD simulations.
The `MATLAB` code used to perform the analysis is both available in the appendix (Appendix 6) and can also be found among the file attached to this report.

3 CAD modeling

The first step in the analysis is to create the CAD of the model rocket.
Any CAD software can be used to create the model but for this project we used `CATIA V5`.
To simplify the analysis, some details of the rocket were omitted from the CAD model, such as the launch lug or the nozzle of the engine. We believe that these simplifications do not affect the accuracy of the study, as the focus is on the nose cone, body, and fins of the rocket.
In Figure 2, the technical drawing of the model is shown, along with the main dimensions.

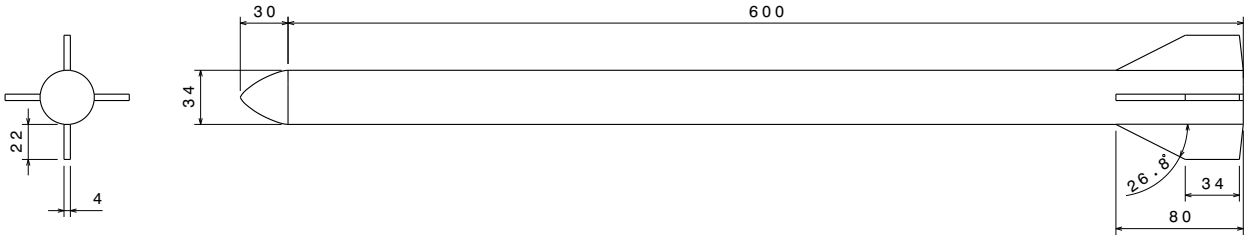

Figure 2: CAD model of the model rocket analyzed in this project. Measurements are in millimeters.

Probably the most important part of the CAD model is the choice of the nose cone shape. For this project, we decided to adopt the **Haack series** nose cone, which is a common choice for model rockets. In particular, instead of being geometrically defined, the Haack series nose cones are mathematically derived to minimize

drag. The shape is controlled by a parameter C, which can be set to different values to obtain different nose cone shapes. The equations that defines the Haack series nose cone are:

$$\theta(x) = \arccos\left(1 - \frac{2x}{L}\right) \tag{2}$$

$$y(\theta, C) = \frac{R}{\sqrt{\pi}}\sqrt{\theta - \frac{\sin 2\theta}{2} + C\sin^3\theta} \tag{3}$$

$$\tag{4}$$

Where L is the length of the nose cone, R is the radius of the base, and C is the parameter that controls its shape.
In our case, we opted for a Haack series nose cone with $C = 0$, which is known to minimize the drag for a given length and diameter. This choice is common in model rocketry, and can also be found under the name of **LD-Haack (Von Kármán)** nose cone.

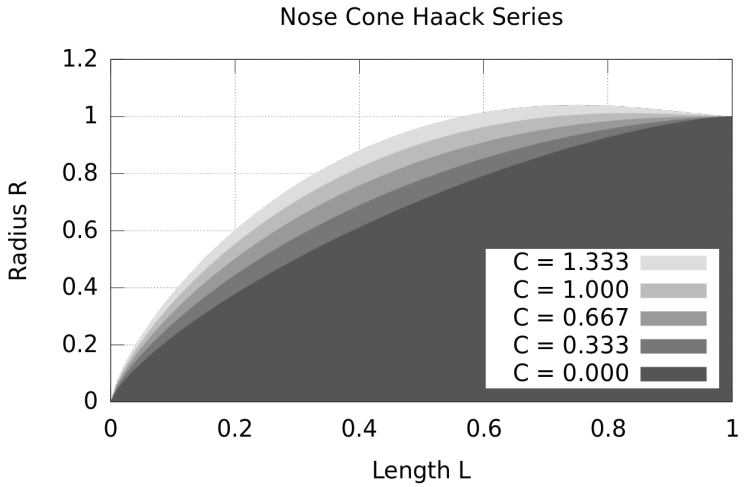

Figure 3: Haack series nose cone with $L = 1$, $R = 1$, and different values of C.

4 CFD simulation

The second step in the analysis is to perform the CFD simulations of the model rocket.
Before running the actual simulations however, we need to perform a series of preliminary steps to prepare the model.

4.1 Geometry preparation

The first step in the CFD simulation is to prepare the geometry of the model rocket and import it into `DesignModeler` of `Ansys Fluent`.
Once the main geometry is imported, we need to create the computational domain around the rocket. For this study, we opted for a simple rectangular domain, with a length of $1.5m$ ($0.5m$ in front of the rocket and $1m$ behind it) and a height and width of $0.7m$ (see Figure 4).

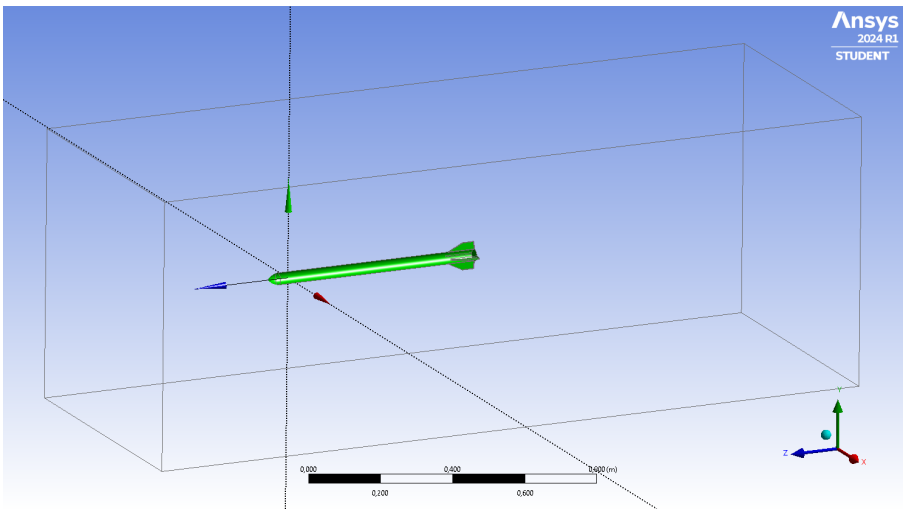

Figure 4: Geometry of the model rocket and the surrounding domain in `DesignModeler`.

The choice in the size of the domain was driven by the idea of having enough space around the rocket to allow the flow to develop and avoid any boundary effects that could affect the results. In fact, the domain distance in the radial direction from the rocket is about 10 times ($0.7/2 = 0.35 \approx 0.34 = 10 * 0.034$) the diameter of the rocket, which is a common rule of thumb for the size of the domain in CFD simulations.

4.2 Mesh generation

The next step in the CFD simulation is to generate the mesh for the computational domain.
Given that our focus is strictly related to the rocket itself and not the flow field around it, we opted for a relatively coarse mesh at the boundary of the domain, and a progressively finer mesh around the rocket. Having the possibility to do so, we also opted for an unstructured mesh, which allows for a more accurate representation of the geometry at the cost of a higher computational time (manageable in our case given the relatively small size of the domain).
The mesh was generated using `Mechanical` module. Default parameters were used for the mesh generation, except of the *Inflaction Layer Mesh* around the rocket.
In Table 1 we report the parameters that were changed from the default values for the mesh generation around the rocket.

Parameter	**Value**
Use Automatic Inflation	Program Controlled
Inflation Option	First Layer Thickness
First Layer Height	$1.5^{-3}m$
Maximum Layers	20
Growth Rate	1.2
Inflation Algorithm	Pre
View Advanced Options	No

Table 1: Mesh *Inflation* parameters used.

In Table 2 we report the statistics of the generated mesh.

Parameter	**Value**
Number of Nodes	73473
Number of Elements	177218

Table 2: Mesh statistics.

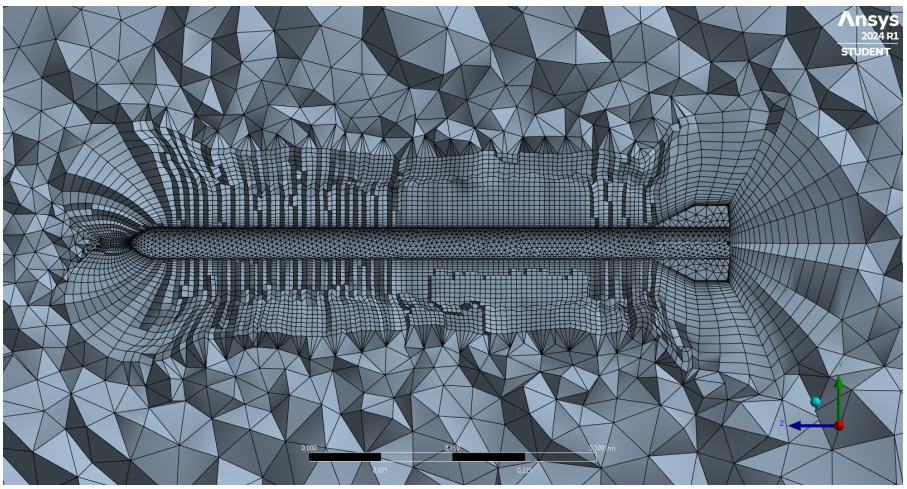

Figure 5: The inflation layer mesh around the rocket body is clearly visible.

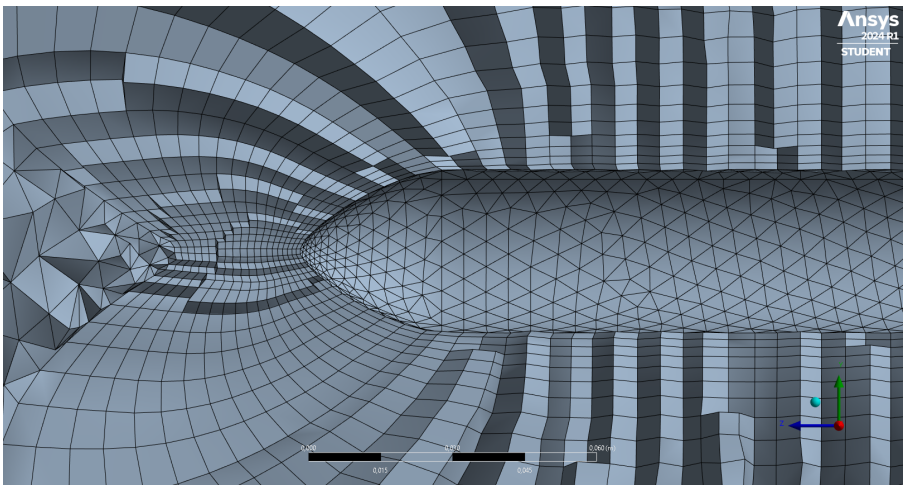

Figure 6: Mesh details around the rocket nose cone. Due to the relatively high curvature of the nose cone, the generated mesh is finer with respect to the linear element of the rocket body.

4.3 Fluent setup

The next step in the CFD simulation is to set up `Fluent` solver.
Here, many configurations could be set, such as the turbulence model, the solver type, the discretization scheme, etc. Given our inexperience with CFD simulations, we opted for the default settings for most of the parameters with some exception that are reported in the following subsections.

4.3.1 Turbulence model

One of the most important parameters to set in the CFD simulations is the turbulence model.
The turbulence model is a mathematical model used to predict the effects of turbulence on the flow field. In our case, we opted for the k-ω *SST* model, which is a widely used model for external aerodynamics simulations. The SST k-ω model is a two-equation eddy-viscosity model that basically combines both the k-ω and k-ϵ models. The use of this model allows for a good prediction of the flow field around the rocket, especially in the near-wall regions and transitional flows.
To do so, we use a k-ω model in the inner parts of the boundary layer, making the model directly usable all the way down to the wall through the viscous sub-layer, hence the SST k-ω model can be used as a Low-Re turbulence model without any extra damping functions. The SST formulation also switches to a k-ϵ behaviour in the free-stream, avoiding the common k-ω problem that the model is too sensitive to the inlet free-stream turbulence properties.
Refer to Menter [1] for more details on the SST k-ω model.

4.3.2 Report definition

In the `Fluent` solver, it is possible to define the reports that will be generated during the simulation.
In our case, we have defined a custom report for the computation of the drag coefficient. The only parameter that we had to modify to enhance the accuracy, was the reference area.
For a simple model rocket, the reference area can be considered as the largest cross-sectional area of the body tube. In our case:

$$A_{\text{ref}} = \pi \cdot \left(\frac{D}{2}\right)^2 = 0.0009079203 \quad [m^2] \tag{5}$$

4.4 Compressibility effects

Before moving on, we need to decide whether to take into account the compressibility effects in the simulations or not.
It's well known from literature and experimental data that compressibility effects start to become important when the flow speed reaches a significant fraction of the speed of sound. In particular, the Mach number of the flow is a good indicator of the importance of the compressibility effects. The condition that distinguishes between incompressible and compressible flows is given by:

$$Ma = \frac{v}{c} = \begin{cases} < 0.3 & \text{Incompressible flow} \\ > 0.3 & \text{Compressible flow} \end{cases} \tag{6}$$

Where Ma is the Mach number, v is the flow speed and c is the speed of sound in the fluid.
To determine whether to take into account the compressibility effects or not, we need to make some hand calculations based on the whether condition present when the rocket was flying. In Table 3 we report the relevant parameters.

Parameter	**Value**
Altitude	$1444m$
Temperature	$15°C$

Table 3: Conditions present during the rocket flight.

Considering air as an ideal gas, the definition of the speed of sound is given by the following equation:

$$c = \sqrt{\gamma \frac{p}{\rho}} = \sqrt{\gamma \bar{R} T} \tag{7}$$

Where γ is the specific heat ratio, $\bar{R}$ is the specific gas constant and T is the temperature of the fluid.
The specific heat ratio for air is $\gamma = 1.4$ and the specific gas constant considering dry air can be computed as:

$$\bar{R} = \frac{R}{M_{mol}} = \frac{8.314}{\frac{28.96}{1000}} = 287.05 \frac{J}{kg \cdot K} \tag{8}$$

Where R is the universal gas constant and M_{mol} is the molar mass of dry air.
Plugging the values into Equation 7, we get:

$$c = \sqrt{1.4 \cdot 287.05 \cdot (273.15 + 15)} = 340.3 \frac{m}{s} \tag{9}$$

From the rocket flight data, we know that the rocket reached a maximum speed of around $130m/s$.
Plugging the values into Equation 6, we get:

$$Ma = \frac{130}{340.3} = 0.382 \tag{10}$$

The maximum Mach number is 0.382, which is above the threshold of 0.3. For this reason, we decided to take into account the compressibility effects in the simulations. To do so, we modified the *Material* properties in `Fluent`, setting the model of the air to *Ideal Gas* instead of *Incompressible*.
However, we believe that this choice is not critical for the results of the simulations, given that the maximum Mach number was reached only for a short period of time during the flight, while for the rest of it Ma was below the threshold.

4.5 Boundary conditions

We now need to set the boundary conditions for the simulations.
The boundary conditions types used for the simulations are reported in Table 4.

Boundary	Condition type
Inlet	Velocity Inlet
Outlet	Pressure Outlet
Rocket Surface	Wall
Domain Surface	Wall

Table 4: Boundary conditions type used for the CFD simulations.

Velocity inlet Since our target here is to obtain a drag coefficient that is representative of the average flight conditions, we decided to reduce the speed to 85% of the maximum one. Based on the data collected from the performed flight, we know that the rocket reached a maximum speed of $\approx 127 m/s$. By computing 85% of this value, we get a speed of $\approx 110 m/s$.
This is a strong approximation, given that the drag coefficient C_d is also function of the speed of the flow (see Figure 7).

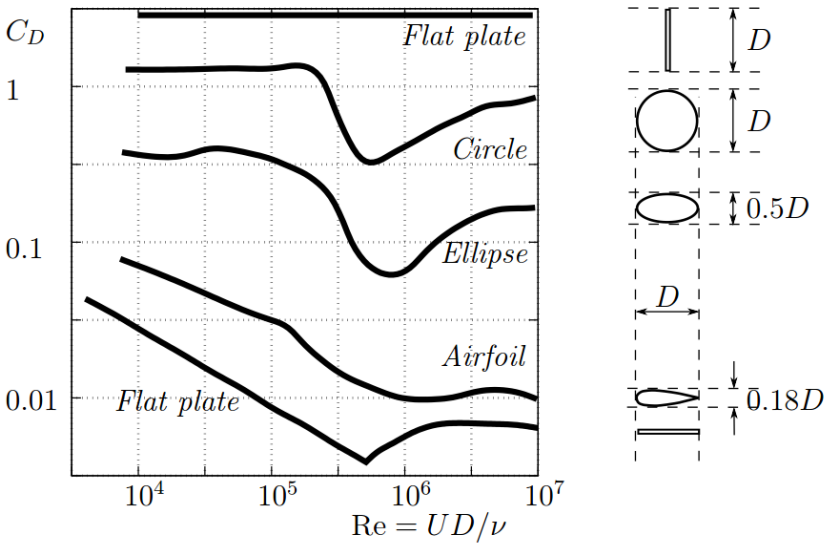

Figure 7: Drag coefficient C_d as a function of the Reynolds number Re for different shapes of the object.

From Figure 7 we can see that the drag coefficient C_d changes significantly at low Reynolds numbers and has a drop across the transition from laminar to turbulent regime for non-slender bodies.
From a quick and approximated calculations we can clearly see that the rocket operate in the turbulence region for most of the flight duration, where the rate of change of the drag coefficient is less pronounced. In particular, considering a set of velocities we can compute the Reynolds number Re as:

$$Re = \frac{\rho v L}{\mu} = \begin{cases} v = 110 m/s \rightarrow Re = 4.9 \cdot 10^6 \\ v = 50 m/s \rightarrow Re = 2.2 \cdot 10^6 \\ v = 10 m/s \rightarrow Re = 4.5 \cdot 10^5 \end{cases} \tag{11}$$

Where ρ (density of the fluid) and μ (dynamic viscosity of the fluid) have been considered constant and equal to $1.225 kg/m^3$ and $1.789 \cdot 10^{-5} kg/(m \cdot s)$ respectively.

Pressure outlet To set the pressure outlet inside the `Fluent` solver, we need to specify it as a gauge pressure. For simplicity (and lack of experience with the software), we set the gauge pressure to $0 Pa$. This is equivalent to say that at the outlet boundary, the pressure is equal to the atmospheric pressure.
However, if we suppose that the atmospheric pressure considered by the software is the standard atmospheric pressure at sea level ($101325 Pa$), this might result in a slight approximation of the simulation.

In fact, given that as stated in Table 3 the altitude of the starting point of the rocket is $1444m$ (meters above sea level), the atmospheric pressure is lower than the standard atmospheric pressure at sea level. In particular, if we consider ideal gas law for the air flow, we can compute the atmospheric pressure at a given altitude as:

$$\frac{p}{\rho} = \frac{RT}{M_{mol}} \tag{12}$$

$$\frac{\nabla p}{p} = \nabla \ln(p) = -\frac{gM_{mol}}{RT}\nabla \tilde{z} \tag{13}$$

$$p = p_0 \exp\left(-\frac{gM_{mol}(\tilde{z} - \tilde{z_0})}{RT}\right) \tag{14}$$

Where p is the pressure, ρ is the density of the fluid, R is the universal gas constant, T is the temperature of the fluid, M_{mol} is the molar mass of the fluid, g is the acceleration due to gravity, $\tilde{z}$ is the altitude, $\tilde{z_0}$ is the altitude at the sea level and p_0 is the standard atmospheric pressure at sea level.
Given Equation 14, we can compute the atmospheric pressure at the launch site, which is at an altitude of $1444m$, as:

$$p_{launch} = 101325 \exp\left(-\frac{9.81 \cdot \frac{28.96}{1000} \cdot 1444}{8.31 \cdot (273.15 + 15)}\right) = 85378Pa \tag{15}$$

We can see that at the launch site the atmospheric pressure is almost 16% lower than the standard atmospheric pressure at sea level.
The difference is even more accentuated if we consider the altitude at the apogee of the rocket, which is around $1979m$. In this case, the atmospheric pressure is:

$$p_{apogee} = 101325 \exp\left(-\frac{9.81 \cdot \frac{28.96}{1000} \cdot 1979}{8.31 \cdot (273.15 + 15)}\right) = 80101Pa \tag{16}$$

In this case, the atmospheric pressure is almost 21% lower than the standard atmospheric pressure at sea level.
Given these considerations, we can imagine that the approximation of the atmospheric pressure at the outlet boundary of the domain might have a slight impact on the results of the simulations, forcing a small extra force from the bottom of the domain, resulting in a slight decrease of the velocity of the flow. However, we believe that this approximation is not critical for the results of the simulations.

4.6 Solver method

The last step in the setup of the `Fluent` solver is to set the solver method.
Thanks to knowledge acquired during the course, at this phase we were able to take informed decisions and we opted for the following settings:

Parameter	**Value**
Pressure-velocity scheme	Coupled
Momentum	Second Order Upwind

Table 5: Solver method settings used.

All the other parameters were left to the default settings, which for most of spatial discretization or interpolations schemes were set to *Second Order Upwind*.
A complete list of the settings used for the simulations is reported in the `XML` file in the Appendix 6.

4.7 CFD results

The simulation was run for 100 iterations, which was enough to reach a good level of convergence considering the residuals and a converged value of the drag coefficient.
In particular, the final value of the drag coefficient was computed as $C_d = 0.86$. This value is generally considered as almost out of the range of the drag coefficient for a rocket, which is usually between 0.7 and 0.8. However, given the shape of the rocket and in particular the four fins that have a considerable impact on the drag coefficient, this value can be considered reasonable[1].

[1]We have runned multiple simulation varying of some perceptual point the velocity at the inlet, and we notice that the drag coefficient was extremely affected. We believe that this has something to do with an error in the simulation setup rather than a physical phenomenon.

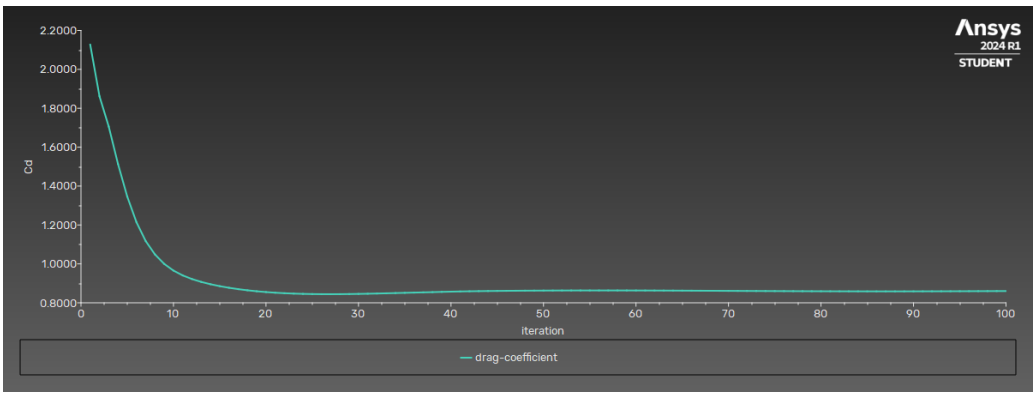

Figure 8: Drag coefficient as function of the number of iterations.

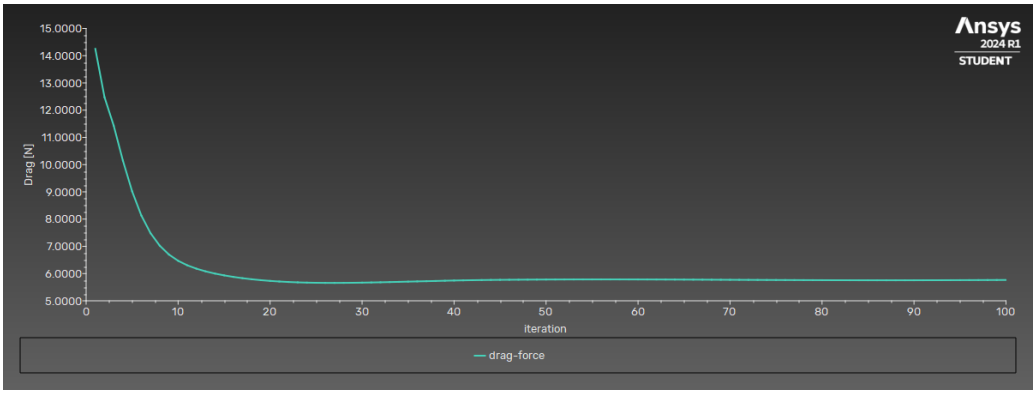

Figure 9: Drag force as function of the number of iterations.

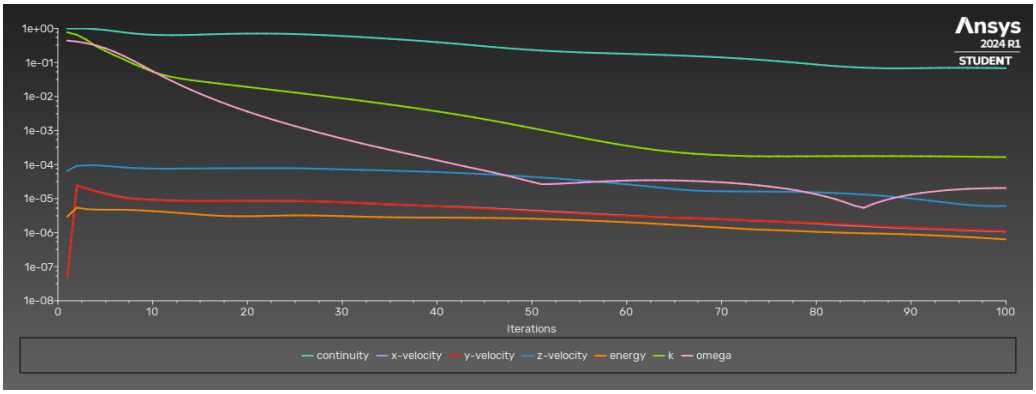

Figure 10: Residuals as function of the number of iterations.

4.7.1 Result visualization

Although the main focus of the study was to compute the drag coefficient of the rocket, we can also proceed to analyze the pressure and velocity field around the rocket which can give us some insights on the flow field.

Pressure field Pressure contours around the rocket are shown in Figures 11 and 12.

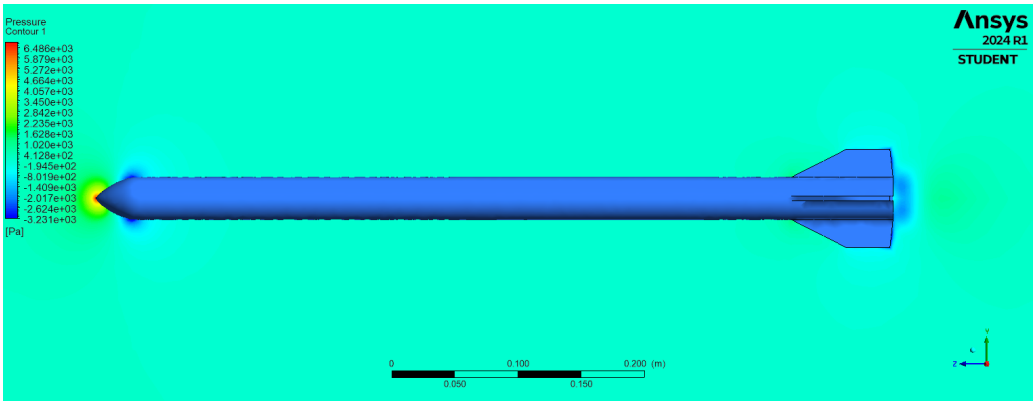

Figure 11: Pressure contours around the rocket. Expected behavior is observed, with the pressure maximum at the bottom of the nose cone and the minimum at both the top of the nose cone. In the rear of the rocket, a low pressure area is formed even if not very pronounced.

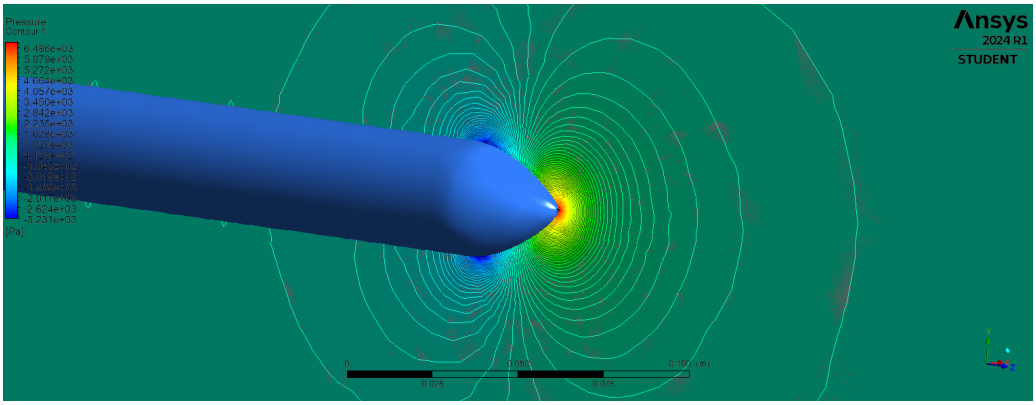

Figure 12: Detail of the pressure field around the nose cone.

Velocity field Velocity contours and streamlines around the rocket are shown in Figures 13, 14 and 15.

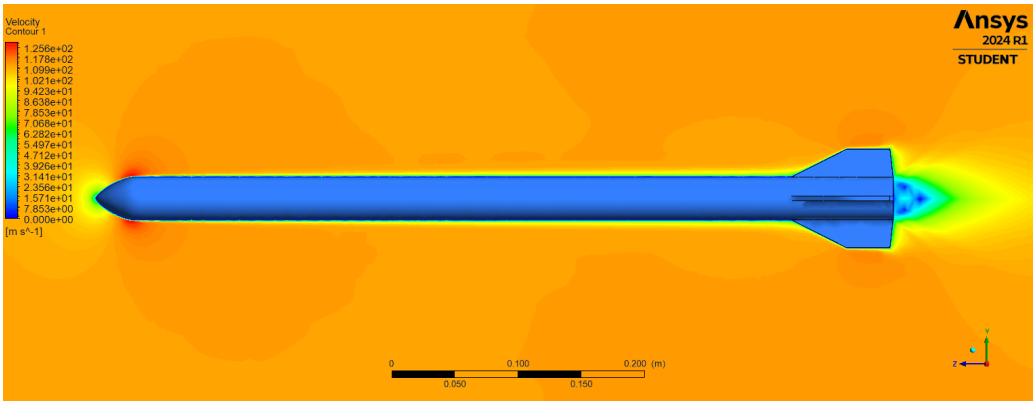

Figure 13: Velocity contours around the rocket. Expected behavior is observed, with the velocity maximum at the bottom of the nose cone and the minimum at both the top of the nose cone and the rear of the rocket. No-slip condition are also respected.

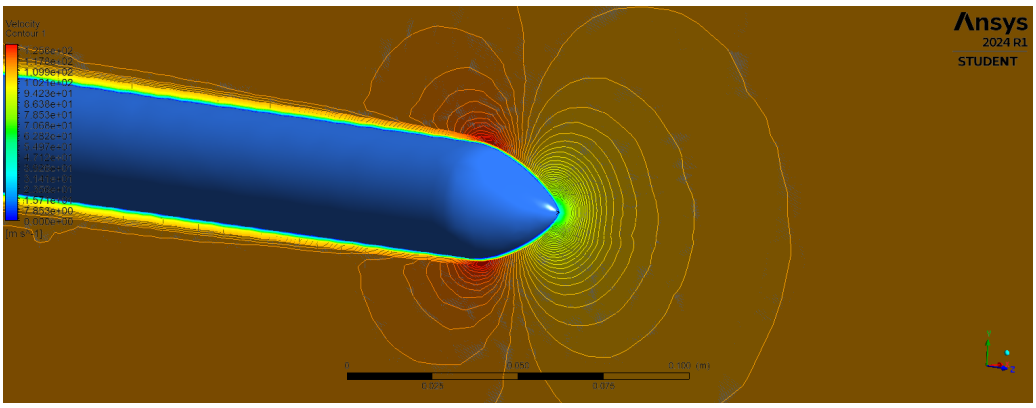

Figure 14: Detail of the velocity field around the nose cone.

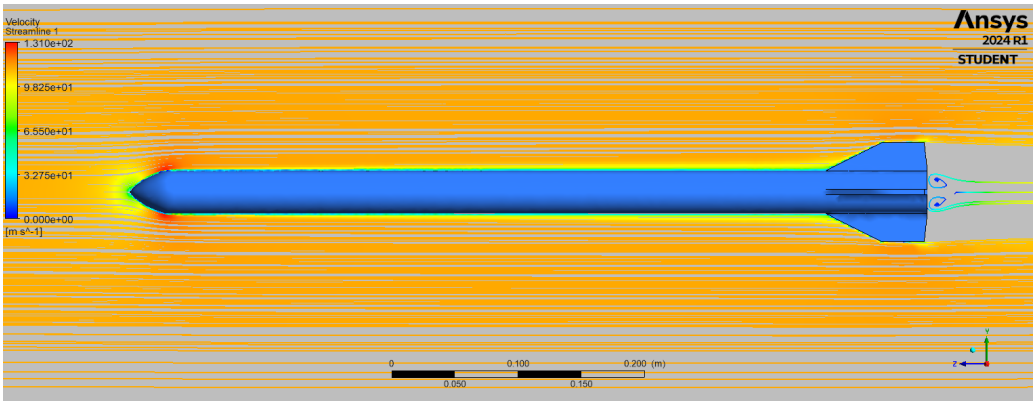

Figure 15: Streamlines around the rocket.

Vorticity field Vorticity streamlines are highlighted in Figure 16.

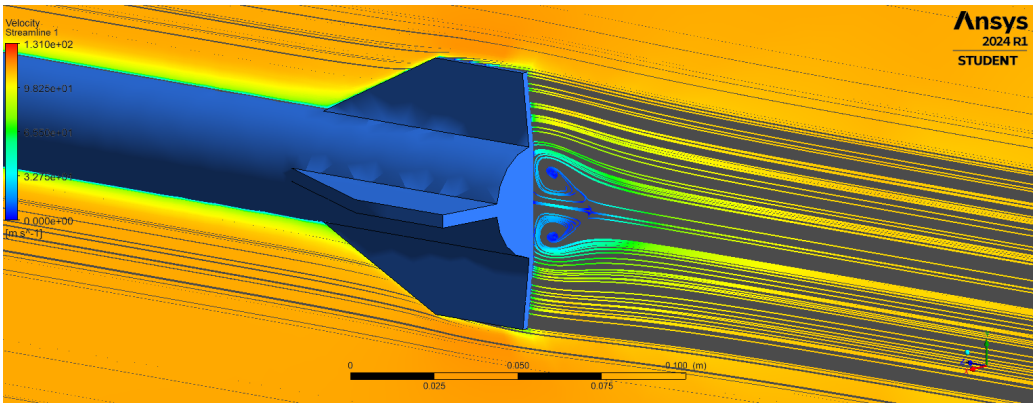

Figure 16: Vortex shedding behind the rocket are clearly visible.

Temperature field Temperature contours around the rocket are shown in Figure 17.

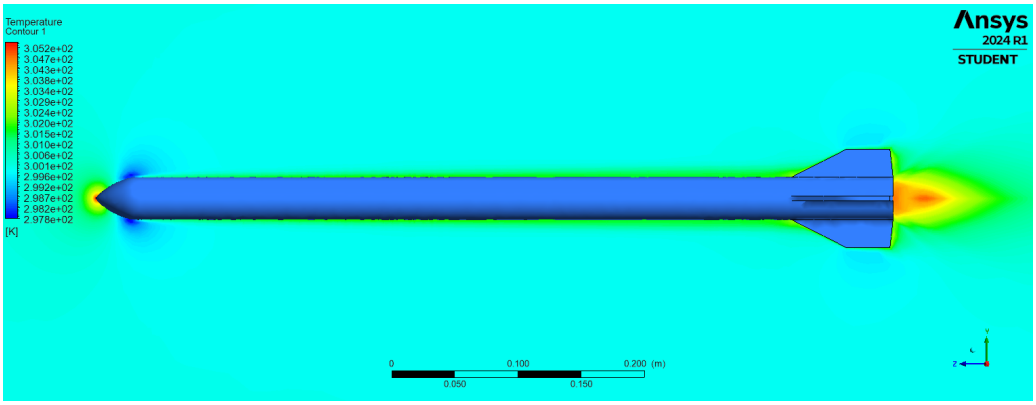

Figure 17: Temperature field around the rocket. The variation is minimal ($+5°C$ at the nose and $-2°C$ at the first full cross-section of the body). It's interesting to see that the temperature rises also at the rear, where a vortex are formed.

5 Analysis and validation of the results

Finally, we can proceed with the validation of the results.
As previously mentioned in Section 2, we are going to compare the results of the CFD simulations with both the flight data collected during the rocket launch and the results of the simulation run in `OpenRocket`.
To do so, we have coded in `MATLAB` a simple model that simulate the dynamics of the rocket flight, which takes into account drag force, gravity, thrust force and mass reduction due to the fuel consumption.

5.1 Rocket flight physics model

To compute a simple flight dynamics model, it's necessary to observe the force acting on the rocket. In Figure 18, the rocket is represented by a rectangle, and the forces acting on it are shown.

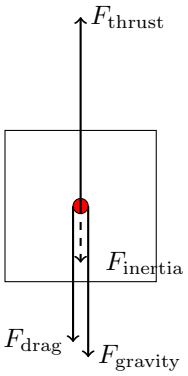

Figure 18: Forces acting on the rocket during the flight.

A simple balance of forces can be written as:

$$F_{\text{thrust}} - F_{\text{drag}} - F_{\text{gravity}} = F_{\text{inertia}} \tag{17}$$

Thrust force F_{thrust} is the thrust force that the engine is supposed to deliver to the rocket during its flight. In particular, knowing that the engine used for the rocket is a `TSP F35` engine, we can retrieve the thrust curve from the manufacturer's website obtaining the data shown in Figure 19.

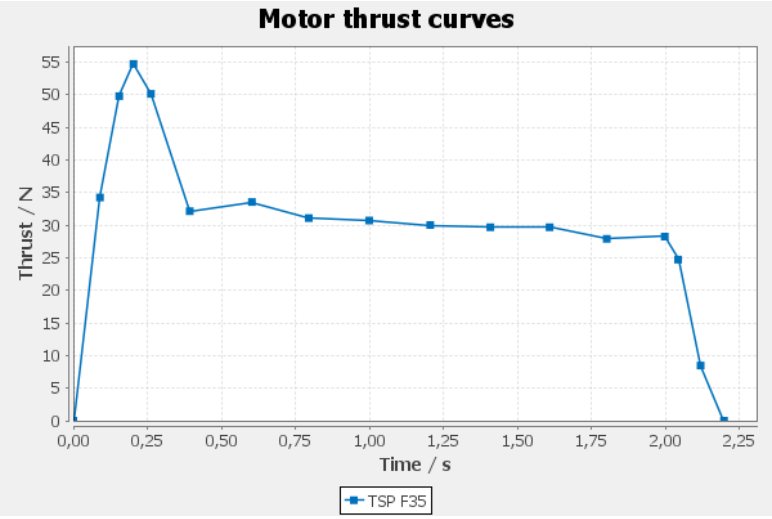

Figure 19: Thrust curve of the `TSP F35` engine.

Equivalently, the thrust points are reported in Table 6.

Time [s]	**Thrust [N]**
0.000	0.000
0.090	34.235
0.155	49.765
0.203	54.706
0.262	50.118
0.393	32.118
0.603	33.529
0.795	31.059
0.999	30.706
1.205	30.000
1.408	29.647
1.608	29.647
1.801	27.882
1.997	28.235
2.042	24.706
2.118	8.471
2.197	0.000

Table 6: Thrust points of the `TSP F35` engine.

Given that we are going to solve the system at time step much more smaller than the time step of the thrust curve, we can interpolate the thrust curve to obtain the thrust force at each time step.
In the end, we can write:

$$F_{\text{thrust}} = F_{\text{thrust}}(t_i) \approx F_{\text{thrust, interpolated}}(t) \tag{18}$$

Drag force F_{drag} is the drag force acting on the rocket during the flight. This force can be computed as:

$$F_{\text{drag}} = \frac{1}{2}\rho v^2 A C_d \tag{19}$$

However, to more precise, we can also take into account some non-linearity of the equation such as the variation of the air density with respect to the altitude $\rho(\tilde{z})$ (see Equation 14). If this is the case, then the drag force can be computed as:

$$F_{\text{drag}} = F_{\text{drag}}(\tilde{z}, v) = \frac{1}{2}\rho(\tilde{z}) v^2 A C_d \tag{20}$$

Gravity force F_{gravity} is the gravity force acting on the rocket during the flight. This force can be computed as:

$$F_{\text{gravity}} = mg \tag{21}$$

However, it's important to notice that the mass of the rocket is not constant during the flight, but it's decreasing due to the fuel consumption. In particular, knowing that the mass of the rocket at the beginning and at the end of the flight is $m_0 = 0.316kg$ and $m_f = 0.224kg$ respectively, we can compute the mass of the rocket function of time as:

$$m(t) = \begin{cases} m_0 - \frac{m_0 - m_f}{t_f} t & \text{if } t \leq t_f \\ m_f & \text{if } t > t_f \end{cases} \tag{22}$$

Where t_f is the time at which the engine stops working (see last row of Table 6).
By doing so, the gravity force can be computed (neglecting any variation of the gravity with respect to the altitude) as:

$$F_{\text{gravity}} = F_{\text{gravity}}(t) = m(t)g \tag{23}$$

Inertia force F_{inertia} is the inertia force acting on the rocket during the flight. This force can be computed as:

$$F_{\text{inertia}} = m(t)a(t) \tag{24}$$

Where $a(t)$ is the acceleration of the rocket at time t.

5.2 Equation of motion & Code implementation

By combining all the forces acting on the rocket, we can write the equation of motion as:

$$m(t)\ddot{z} + \frac{1}{2}\rho(z)AC_d\dot{z}^2 + m(t)g = F_{\text{thrust}}(t) \tag{25}$$

Where z is the altitude of the rocket, $\dot{z}$ is the velocity of the rocket and $\ddot{z}$ is the acceleration of the rocket.
As often happen in mechanical problem, the equation of motion is a second-order differential equation. To solve it, we can rewrite it as a system of first-order differential equations as:

$$\begin{cases} \dot{z} = v \\ \ddot{z} = \dot{v} = \frac{F_{\text{thrust}}(t) - \frac{1}{2}\rho(z)AC_d v^2 - m(t)g}{m(t)} \end{cases} \tag{26}$$

The system of equations can be solved using a simple `MATLAB` script, which is reported in Listing 1.

```
%% Rocket physics modelling

funcs.rho       = @(s)    1 / ((8.314 / 28.96 * 1000) * (273.15 + 15)) * 101325 * exp(-(9.81 * 28.96 / 1000 *
      (s)) / (8.31 * (273.15 + 15)));
funcs.mass      = @(t)    ((rocket.m(end) - rocket.m(1)) / (thrust_curve(end).Time - thrust_curve(1).Time) *
     t + rocket.m(1)) .* (t <= thrust_curve(end).Time) + rocket.m(end) * (t > thrust_curve(end).Time);
funcs.F_gravity = @(t)    9.81 * funcs.mass(t);
funcs.F_drag    = @(s, v) 1/2 * rocket.Cd * funcs.rho(s) .* rocket.A .* v.^2;
funcs.F_thrust  = @(t)    interp1([thrust_curve.Time 100], [thrust_curve.Thrust 0] *
     thrust_reduction_coefficient, t, "pchip");


%% Solution of the ODE for the Equation of Motion of the system

[t, z] = ode45(@(t, z) EquationOfMotion(t, z, funcs), ...
    [0 timestamp(end)], ...
    [funcs.velocity(0); funcs.altitude(0)], ...
    odeset('RelTol', 1e-6, 'AbsTol', 1e-6));


%% Functions

function [z_dot] = EquationOfMotion(t, z, funcs)

velocity = z(1,1);
altitude = z(2,1);

```

```
acceleration = (funcs.F_thrust(t) - funcs.F_gravity(t) - funcs.F_drag(altitude, velocity)) / funcs.mass(t);

z_dot(1,1) = acceleration;
z_dot(2,1) = velocity;

end
```

Listing 1: `MATLAB` script to compute the rocket flight model.

The full version of the code can be found in the appendix (see Section 6).

5.3 CFD Simulation vs. Collected flight data & `OpenRocket` simulation

Finally, the results of the simulation are compared with the data collected during the rocket launch and the results of the simulation run in `OpenRocket`.

5.3.1 Flight data

To collect data, we used a BMP280 barometer/thermometer sensor, which was placed inside the nose cone of the rocket and connected to an Arduino Nano board. Because of poor quality SD writer module however, we weren't able to sampling at a high frequency and the number of collected data is limited (but still sufficient for our purpose).
In Table 7 we report all the data collected during the rocket launch:

Time [s]	**Altitude [m]**
494.130	1443
495.233	1443
496.334	1444
497.437	1554
498.540	1678
500.745	1809
501.848	1847
502.953	1903
504.055	1931
505.161	1958
506.263	1973
507.364	1979
508.468	1976
509.570	1965

Table 7: Data collected during the rocket launch.

In Figure 20, both the data of Table 7 and their interpolation are shown.

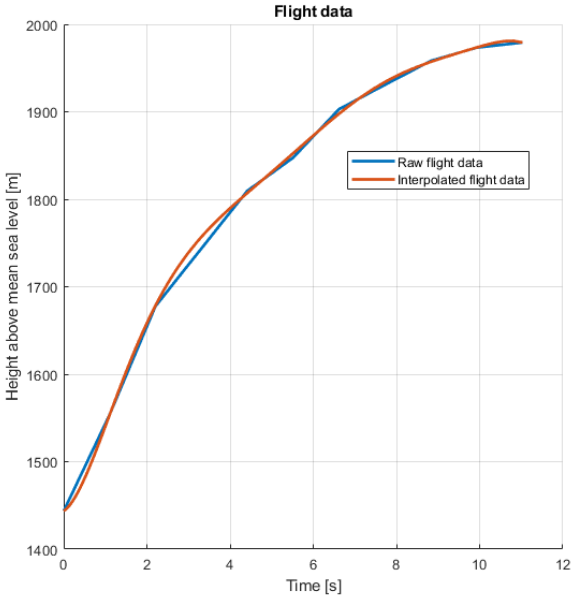

Figure 20: Data collected during the rocket launch.

For the interpolation, we used a Lagrange polynomial of degree 7 (out of 10 points) to obtain a smooth curve that can be used to compute also the vertical velocity of the rocket.

5.3.2 `OpenRocket` simulation

As we have said, `OpenRocket` is a software that allows to both design and simulate the flight of a rocket.
By assigning shape, dimensions, mass, and an hypothetical C_d to each component of the rocket, the software is able to give a raw estimation of the dynamics during the flight. In Figure 21, the results of the simulation run in `OpenRocket` are shown.

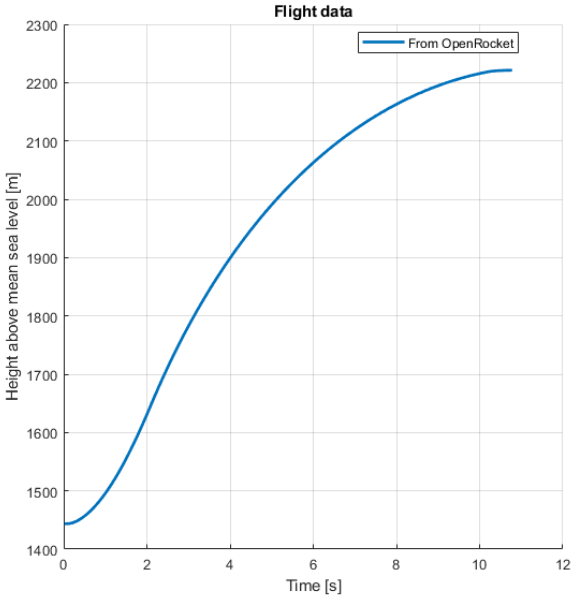

Figure 21: Simulation run in `OpenRocket`.

5.3.3 Final comparison

From Figure 20 and Figure 21, we can see that the two curves (at the least in the apogee altitude) are not in agreement.

This is a common situation, as the simulation in `OpenRocket` are performed considering almost ideal condition and neglecting any possible aerodynamic imperfection of the rocket (which of course are present in the real world).
To overcome this discrepancy between the simulation and the real world, we will decrease the thrust curvature by a given factor until the peak velocity between the two curves are the same. This might be a very strong assumption, but in reality in the world of model rocket is quite common as the effective thrust of the engine is almost always lower than the nominal thrust declared by the manufacturer.
Having said this, we can proceed with the comparison of the results of the CFD simulation with the collected flight data.

CFD simulation vs. `OpenRocket` simulation In Figure 22, the altitude and the vertical velocity of the rocket are shown.

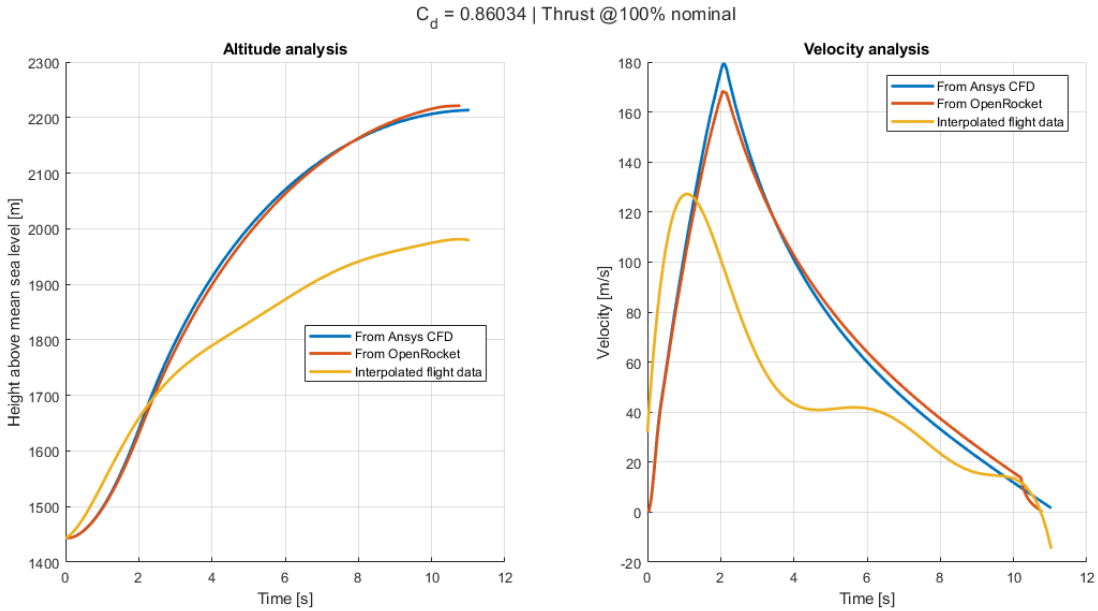

Figure 22: Comparison of the altitude and the vertical velocity of the rocket, considering a full thrust curve and $C_d = 0.86$ (from the CFD simulation). As the legend reports, the blue line represents the prediction based on the C_d computed in the CFD simulation, the orange line represents the simulation run in `OpenRocket`, while the yellow line represents the collected flight data.

As we can see, the curves generated with the CFD simulation and the one from `OpenRocket` are almost identical, while the collected flight data are way off.

CFD simulation vs. collected flight data However, as we have mentioned at the beginning of this section, the thrust curve of the engine are usually not accurate and often turns out to be much less than the nominal thrust declared by the manufacturer. To overcome this discrepancy, we can decrease the thrust curve by a factor of 0.7 until the peak velocity between the flight data and the CFD simulation are the same. In this case, the results are shown in Figure 23.

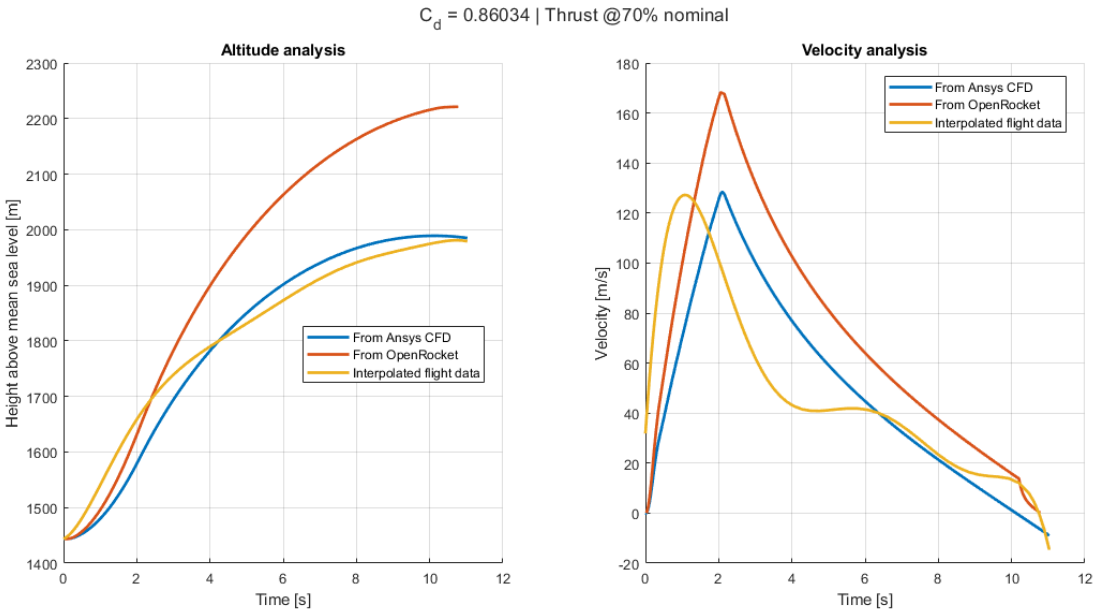

Figure 23: Comparison of the altitude and the vertical velocity of the rocket, considering a thrust curve decreased by a factor of 0.7 and $C_d = 0.86$ (from the CFD simulation).

By decreasing the thrust curve by a factor of 0.7, the peak velocity of the flight data and the CFD simulation are almost the same. Even if collected data are few and of poor quality (see the strong non-physical oscillation in the velocity curve), we can still try to relate the two curves. By doing so, we can see that the CFD simulation is in good agreement with the collected data, while the simulation run in `OpenRocket` of course is not.

`OpenRocket` coefficient of drag estimation One may also ask what's the coefficient of drag used in the `OpenRocket` simulation. Unfortunately, the software doesn't provide this information, but we can try to estimate it by comparing the altitude estimation for different values of C_d around the value computed in the CFD simulation. In Table 8, we report three different values of C_d and the corresponding peak altitude and velocity computed using our simple model and the comparison with the simulation from `OpenRocket`.

C_d	**Altitude @ $t = 10s$ [m]**	**Peak velocity [m/s]**
`OpenRocket`	$2.2166e+03$	168.2810
0.903	$2.1900e+03$	177.6557
0.860	$2.2068e+03$	179.3001
0.817	$2.2271e+03$	180.9773

Table 8: Comparison of the peak altitude and velocity for different values of C_d.

In Table 8, we presented the results for $C_d = 0.903$, $C_d = 0.860$ and $C_d = 0.817$ (that are respectively 5% and 5% lower and higher than the value computed in the CFD simulation). As we can observe, the peak velocity is almost the same for the three values of C_d, while the peak altitude is slightly different. In particular, the best correspondence with the `OpenRocket` simulation is obtained for $C_d = 0.860$. Also, from a shape point of view, the curve obtained with $C_d = 0.860$ is the one that best fits the `OpenRocket` simulation.

6 Conclusions

The present work aimed at investigating the drag coefficient of a model rocket using computational fluid dynamics.
The rocket was at first modeled in an external CAD software and then imported into Ansys Fluent for the CFD simulation. The simulation was carried out using the $k-\omega$ SST turbulence model and in a steady-state regime and the drag coefficient (C_d) was calculated at 85% of the speed maximum speed reached by the rocket during the real world flight.
The results obtained from the CFD simulation were compared to the flight data and the simulation from `OpenRocket`, showing a good agreement with both of them after some adjustments to the thrust curve.

Moreover, the drag coefficient was found to be $C_d = 0.86$, which is in line with the expected values for a model rocket. This value is also in line with the one obtained from the flight data and the one from `OpenRocket`.
The present work showed that CFD simulations can be a powerful tool to investigate the aerodynamic behavior of model rockets and that the results obtained can be potentially used to optimize the design iterating backward and forward between the CAD model and the CFD simulation.

References

[1] Florian R. Menter. Two-equation eddy-viscosity turbulence models for engineering applications. *AIAA Journal*, 32:1598–1605, 1994.

A Ansys Fluent settings for the CFD simulation

```
<FluentXMLReport>
<version>
Fluent
Version: 3d, dp, pbns, sstkw (3d, double precision, pressure-based, SST k-omega)
Release: 24.1.0
Title:
</version>

<Models>

Models
------

   Model                         Settings
   ------------------------------------------------------------
   Space                         3D
   Time                          Steady
   Viscous                       SST k-omega turbulence model
   Heat Transfer                 Enabled
   Solidification and Melting    Disabled
   Radiation                     None
   Species                       Disabled
   Coupled Dispersed Phase       Disabled
   NOx Pollutants                Disabled
   SOx Pollutants                Disabled
   Soot                          Disabled
   Mercury Pollutants            Disabled
   Structure                     Disabled
   Acoustics                     Disabled
   Eulerian Wall Film            Disabled
   Potential/Electrochemistry    Disabled
   Multiphase                    Disabled

</Models>

<MaterialProperties>
Material Properties
-------------------

   Material: air (fluid)

      Property               Units      Method      Value(s)
      ------------------------------------------------------------
      Density                kg/m^3     ideal-gas   #f
      Cp (Specific Heat)     J/(kg K)   constant    1006.43
      Thermal Conductivity   W/(m K)    constant    0.0242
      Viscosity              kg/(m s)   constant    1.7894e-05
      Molecular Weight       kg/kmol    constant    28.966

   Material: aluminum (solid)

      Property               Units      Method     Value(s)
      ------------------------------------------------------
      Density                kg/m^3     constant   2719
      Cp (Specific Heat)     J/(kg K)   constant   871
      Thermal Conductivity   W/(m K)    constant   202.4

</MaterialProperties>

<CellZoneConditions>
Cell Zone Conditions
--------------------

   Zones

      name    id   type
      ------------------
      solid   2    fluid

   Setup Conditions

      solid

         Condition       Value
```

```
        --------------------
        Frame Motion?   no
        Mesh Motion?    no

</CellZoneConditions>

<BoundaryConditions>
Boundary Conditions
-------------------

   Zones

      name     id   type
      ----------------------------
      inlet    5    velocity-inlet
      outlet   6    pressure-outlet
      wall     7    wall
      rocket   8    wall

   Setup Conditions

      inlet

         Condition                  Value
         -------------------------------
         Velocity Magnitude [m/s]   112

      outlet

         Condition   Value
         ----------------

      wall

         Condition                  Value
         ------------------------------------------
         Thermal BC Type            Heat Flux
         Wall Motion                Stationary Wall
         Shear Boundary Condition   No Slip
         Wall Surface Roughness     rough-bc-standard

      rocket

         Condition                  Value
         ------------------------------------------
         Thermal BC Type            Heat Flux
         Wall Motion                Stationary Wall
         Shear Boundary Condition   No Slip
         Wall Surface Roughness     rough-bc-standard

</BoundaryConditions>

<SolverSettings>
Solver Settings
---------------

   Equations

      Equation     Solved
      -------------------
      Flow         yes
      Turbulence   yes
      Energy       yes

   Numerics

      Numeric                          Enabled
      ---------------------------------------
      Absolute Velocity Formulation    yes

   Pseudo Time Explicit Relaxation Factors

      Variable                    Relaxation Factor
      ---------------------------------------------
      Density                     1
      Body Forces                 1
```

```
      Turbulent Kinetic Energy    0.75
      Specific Dissipation Rate   0.75
      Turbulent Viscosity         1
      Energy                      0.75
      Explicit Momentum           0.5
      Explicit Pressure           0.5

   Linear Solver

      Variable                    Solver Type   Termination Criterion   Residual Reduction Tolerance
      ------------------------------------------------------------------------------------------------
      Flow                        F-Cycle       0.1
      Turbulent Kinetic Energy    F-Cycle       0.1
      Specific Dissipation Rate   F-Cycle       0.1
      Energy                      F-Cycle       0.1

   Pressure-Velocity Coupling

      Parameter                              Value
      ----------------------------------------------
      Type                                   Coupled
      Pseudo Time Method (Global Time Step)  yes

   Discretization Scheme

      Variable                    Scheme
      ----------------------------------------------
      Pressure                    Second Order
      Density                     Second Order Upwind
      Momentum                    Second Order Upwind
      Turbulent Kinetic Energy    Second Order Upwind
      Specific Dissipation Rate   Second Order Upwind
      Energy                      Second Order Upwind

   Solution Limits

      Quantity                                Limit
      ----------------------------------------------
      Minimum Absolute Pressure [Pa]          1
      Maximum Absolute Pressure [Pa]          5e+10
      Minimum Static Temperature [K]          1
      Maximum Static Temperature [K]          5000
      Minimum Turb. Kinetic Energy [m^2/s^2]  1e-14
      Minimum Spec. Dissipation Rate [s^-1]   1e-20
      Maximum Turb. Viscosity Ratio           100000

</SolverSettings>

</FluentXMLReport>
```

Listing 2: Settings applied to the CFD simulation in `Ansys Fluent`.

B `MATLAB` code used for the validation phase

```
clc
clear variables
close all

%% Section 0: Settings & Parameters

rocket = struct( ...
    'A', (34e-3/2)^2 * pi, ...
    'm', [0.316 0.224], ...
    'Cd', 0.86034355);

output = struct('plot', true);

funcs = struct();

thrust_reduction_coefficient = 1.0;


%% Section 1: Data import

```

```
flight_data = table2struct(readtable('Flight data.xlsx'));
thrust_curve = table2struct(readtable('Thrust curve.xlsx'));
openrocket = table2struct(readtable('OpenRocket.csv'));


%% Section 2: Flight data interpolation

liftoff_idx = 3;
[~, apogee_idx] = max([flight_data.Altitude]);

timestamp = [flight_data(liftoff_idx:apogee_idx).Time_s_] - flight_data(liftoff_idx).Time_s_;
altitude = [flight_data(liftoff_idx:apogee_idx).Altitude];

altitude_poly_coefficient = polyfit(timestamp, altitude, 7);
velocity_poly_coefficient = polyder(altitude_poly_coefficient);
acceleration_poly_coefficient = polyder(velocity_poly_coefficient);

funcs.altitude = @(t) polyval(altitude_poly_coefficient, t);
funcs.velocity = @(t) polyval(velocity_poly_coefficient, t);
funcs.acceleration = @(t) polyval(acceleration_poly_coefficient, t);


%% Section 2: Rocket physics modelling & Thrust data interpolation

funcs.rho        = @(s)    1 / ((8.314 / 28.96 * 1000) * (273.15 + 15)) * 101325 * exp(-(9.81 * 28.96 / 1000 *
      (s)) / (8.31 * (273.15 + 15)));
funcs.mass       = @(t)    ((rocket.m(end) - rocket.m(1)) / (thrust_curve(end).Time - thrust_curve(1).Time) *
     t + rocket.m(1)) .* (t <= thrust_curve(end).Time) + ...
    rocket.m(end) * (t > thrust_curve(end).Time);
funcs.F_gravity = @(t)    9.81 * funcs.mass(t);
funcs.F_drag     = @(s, v) 1/2 * rocket.Cd * funcs.rho(s) .* rocket.A .* v.^2;
funcs.F_thrust  = @(t)    interp1([thrust_curve.Time 100], [thrust_curve.Thrust 0] *
     thrust_reduction_coefficient, t, "pchip");


%% Section 3: Solution of the ODE for the Equation of Motion of the system
% The EoM is based on the anonymus functions defined above

[t, z] = ode45(@(t, z) EquationOfMotion(t, z, funcs), ...
    [0 timestamp(end)], ...
    [0; funcs.altitude(0)], ...
    odeset('RelTol', 1e-6, 'AbsTol', 1e-6));


%% Plots

[~, apogee_idx] = max([openrocket.Altitude_m_]);
OR_time = [openrocket(1:apogee_idx).x_Time_s_];
OR_altitude = [openrocket(1:apogee_idx).Altitude_m_] + altitude(1);
OR_velocity = [openrocket(1:apogee_idx).VerticalVelocity_m_s_];

if (output.plot)

    figure_data_validation = figure('Name', 'Data validation', 'NumberTitle', 'off', 'Position', [100 200
     1200 600]);
    tile = tiledlayout(1, 2);
    title(tile, ['C_d = ' num2str(rocket.Cd) ' | Thrust @' num2str(thrust_reduction_coefficient * 100) '%
     nominal']);

    % Altitude vs. time
    nexttile(1);
    hold on
    grid on

    plot(t, z(:, 2), ...
        OR_time, OR_altitude, ...
        t, funcs.altitude(t), ...
        'LineWidth', 2);

    xlabel('Time [s]');
    ylabel('Height above mean sea level [m]')
    title('Altitude analysis');
    legend('From Ansys CFD', 'From OpenRocket', 'Interpolated flight data', 'Location', 'best');


    % Velocity vs. time
```

```
92      nexttile(2);
93      hold on
94      grid on
95
96      plot(t, z(:, 1), ...
97          OR_time, OR_velocity, ...
98          t, funcs.velocity(t), ...
99          'LineWidth', 2);
100
101     xlabel('Time [s]');
102     ylabel('Velocity [m/s]')
103     title('Velocity analysis');
104     legend('From Ansys CFD', 'From OpenRocket', 'Interpolated flight data', 'Location', 'best');
105
106 end
107
108
109
110 %% Functions
111
112 function [z_dot] = EquationOfMotion(t, z, funcs)
113
114 velocity = z(1,1);
115 altitude = z(2,1);
116
117 acceleration = (funcs.F_thrust(t) - funcs.F_gravity(t) - funcs.F_drag(altitude, velocity)) / funcs.mass(t);
118
119 z_dot(1,1) = acceleration;
120 z_dot(2,1) = velocity;
121
122 end
```

Listing 3: MATLAB code used for validating the CFD simulation result against flight data and simulation from OpenRocket.