Universidade Federal de Santa Catarina
Campus Florianópolis
Departamento de Engenharia Mecânica

Caio Dias Fernandes

Combining Semi-Empirical Correlations and Partially Converged CFD Data Through Machine Learning Based Reduced Order Models in a Conceptual Design Framework for Rocketry Aerodynamics

Florianópolis
2022

Caio Dias Fernandes

Combining Semi-Empirical Correlations and Partially Converged CFD Data Through Machine Learning Based Reduced Order Models in a Conceptual Design Framework for Rocketry Aerodynamics

Final paper submitted in partial fulfillment of the requirements for the degree of BEng. in Mechanical Engineering of the Universidade Federal de Santa Catarina.
Advisor: Prof. Dr. Amir Antônio Martins de Oliveira Jr.

Universidade Federal de Santa Catarina
Campus Florianópolis
Departamento de Engenharia Mecânica

Florianópolis
2022

Ficha Catalográfica

Dias Fernandes, Caio

Combining Semi-Empirical Correlations and Partially Converged CFD Data Through Machine Learning Based Reduced Order Models in a Conceptual Design Framework for Rocketry Aerodynamics.

Florianópolis 2022.

88 p.

Trabalho de Conclusão de Curso - Universidade Federal de Santa Catarina, Centro Técnológico, Graduação em Engenharia Mecânica.

Orientador: Amir Antônio Martins de Oliveira Jr.

Inclui referências.

Palavras-Chave: 1. Otimização de Projeto Conceitual. 2. Fluído Dinâmica Computacional. 3. Aerodinâmica Semi-Empírica para Mini-Foguetes. 4. Modelos de Ordem Reduzida. 5. Aprendizagem de Máquinas.

Caio Dias Fernandes

Combining Semi-Empirical Correlations and Partially Converged CFD Data Through Machine Learning Based Reduced Order Models in a Conceptual Design Framework for Rocketry Aerodynamics

Final paper submitted in partial fulfillment of the requirements for the degree of BEng. in Mechanical Engineering of the Universidade Federal de Santa Catarina.

Examination Committee

Prof. Dr. Amir Antônio Martins de Oliveira Jr.
Universidade Federal de Santa Catarina
Orientador

Prof. Diogo Nardelli Siebert
Universidade Federal de Santa Catarina

Prof. Dr. Saulo Güths
Universidade Federal de Santa Catarina

Florianópolis, July 27, 2022

Resumo

Este trabalho propõe um framework paramétrico para a otimização conceitual do projeto aerodinâmico e de estabilidade de um mini-foguete. Com o objetivo de desenvolver uma ferramenta de projeto que considera condições de capacidade computacional restrita; o Método de Barrowman Modificado para aerodinâmica semi-empírica de mini-foguetes, assim como modelos automatizados de CFD em OpenFOAM, foram desenvolvidos e implementado em uma rotina em python. Além disso, o uso de Modelos de Ordem Reduzida tradicionais e baseados em Aprendizagem de Máquinas, construídos a partir de resultados parcialmente convergidos de CFD, também foram investigados como alternativas de baixo custo computacional para o design paramétrico. Especificamente, Redes Neurais de Feed-forward empregando Transferência de Aprendizagem são propostas como alternativas possíveis de ROM, combinando ambos os bancos de dados analíticos (MBM) e numéricos (CFD) visando reduzir o esforço computacional total requerido para o problema de otimização do projeto conceitual de aerodinâmica.

Palavras-Chave: 1. Otimização de Projeto Conceitual. 2. Fluido Dinâmica Computacional. 3. Aerodinâmica Semi-Empírica para Mini-Foguetes. 4. Modelos de Ordem Reduzida. 5. Aprendizagem de Máquinas

Abstract

In this work, a parametric framework for the conceptual design optimization of a model rocket's aerodynamic and flight stability configuration is proposed. To achieve the goal of developing a suitable design tool that takes restricted computational capabilities into account; the semi-empirical Modified Barrowman Method for rocketry aerodynamics, as well as automated OpenFOAM CFD models, were developed and implemented within a python routine. Furthermore, the use of traditional and Machine Learning based Reduced Order Models constructed with partially converged CFD data was also investigated as a low computational effort parametric design alternative. Specifically, Feed-forward Neural Netwroks employing Transfer Learning are proposed as a feasible ROM approach, combining both the analytical (MBM) and numerical (CFD) data-sets with the goal of reducing the total computation effort required for aerodynamics conceptual design optimization problems.

Keywords: 1. Conceptual Design Optimization. 2. Computational Fluid Dynamics. 3. Semi-empirical Rocketry Aerodynamics. 4. Reduced Order Models. 5. Machine Learning.

List of figures

List of tables

List of Acronyms

CFD	Computational Fluid Dynamics
CAE	Computer Aided Engineering
CAD	Computer Aided Drawing
NS	Navier Stokes
RANS	Reynolds Averaged Navier Stokes
ROM	Reduced Order Model
MBM	Modified Barrowman Method
DoE	Design of Experiments
LHS	Latin-hypercube Sampling
RBF	Radial Basis Functions
GEK	Gradient-Enhanced Kriging
PLS	Partial Least Squares
GEKPLS	Gradient-Enhanced Kriging with Partial Least Squares
ML	Machine Learning
TL	Transfer Learning
FNN	Feed-forward Neural Networks
MLP	Multi Layer Perceptron

List of Symbols

x_{cp}	Center of pressure coordinate	[m]
x_{cg}	Center of gravity coordinate	[m]
SM	Static Margin	[cal]
M_a	Mach number	[-]
α	Angle of attack	[°]
C_l	Lift coefficient	[-]
C_n	Normal force coefficient	[-]
C_d	Drag coefficient	[-]
C_a	Axial force coefficient	[-]
C_m	Pitching moment coefficient	[-]
C_{m_α}	Longitudinal stability derivative	[-]
$C_{m\dot{\alpha}}$	Pitch damping stability derivative	[-]
ξ	Damping ratio	[-]
Re	Reynolds number	[-]
h	Fin height	[mm]
s	Fin sweep length	[mm]

Table of contents

1 Introduction

In October of 2018, I and other colleagues from the Technology Center of the Universidade Federal de Santa Catarina (UFSC) started Apex Rocketry, UFSC's first rocket design team based on the university's main Florianópolis campus. The team's main goal is designing, developing and testing different types of low apogee rockets to compete in newly created international rocket design competitions in Latin America. In 2020 Apex Rocketry developed the Armação Project [1] for the 2_{nd} edition of the *Latin American Space Challenge*, achieving the overall 4_{th} best design in the competition.

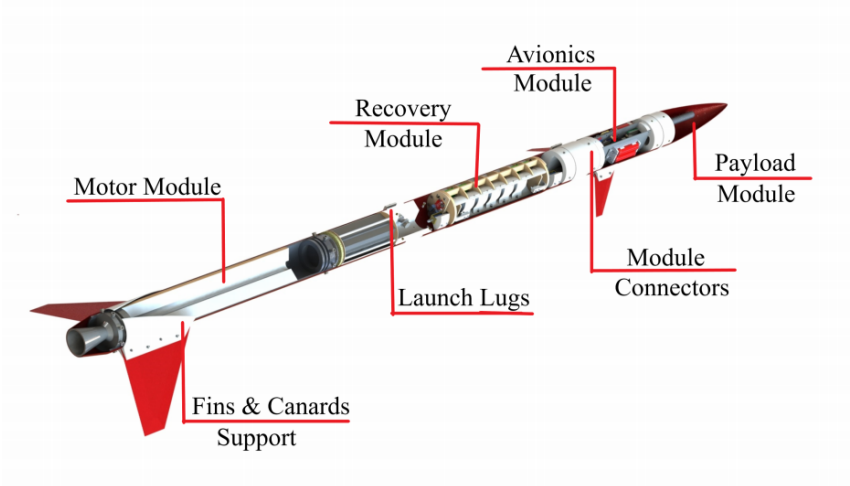


Figure 1 – Fully integrated Armação-A22 rocket. Source [1].

The Armação-A22 rocket is Apex'es third fully designed rocket, and contains four main modules — namely, the Motor, Recovery, Avionics, and Payload modules — depicted in Figure 1. The rocket is configurable for 2 different target apogees, and the flight's main objective is safely flying to and measuring the trajectory's apex, with as little deviation from the target as possible. Apex fully developed a class L motor for the Armação Project, equipped with a Potassium Nitrate and Mono Hydrated Dextrose based solid fuel. Table 1 displays some of the flight's principal parameters, while Figure 2 illustrates the rocket's aerodynamic and stability characteristics.

The conceptual design of a model rocket's aerodynamic surfaces, especially in regards to its flight stability performance, is a critical stage during the project's initial phase, which provides crucial design inputs to other project areas, guiding the vehicle's general design. Computer Aided Engineering (CAE) methods are typically used in this design phase, given their faster analysis cycle when compered to more conventional experimental approaches. CAE methods also present themselves as favorable engineering tools for situations where prototyping is both costly and potentially dangerous.

Table 1 – Armação A-22 principal flight characteristics (adapted from [1]).

	1km target	**2km target**
Liftoff weight (kg)	12.55	8.55
Thrust-to-Weight ratio	6.38	9.82
Motor average thrust (kN)	0.92	1.07
Motor impulse (kN.s)	2.94	3.26
Maximum velocity (m/s)	138	243
Nominal static margin (cal)	1.6	1.3
Predicted apogee (m)	973 ± 38	1973 ± 42

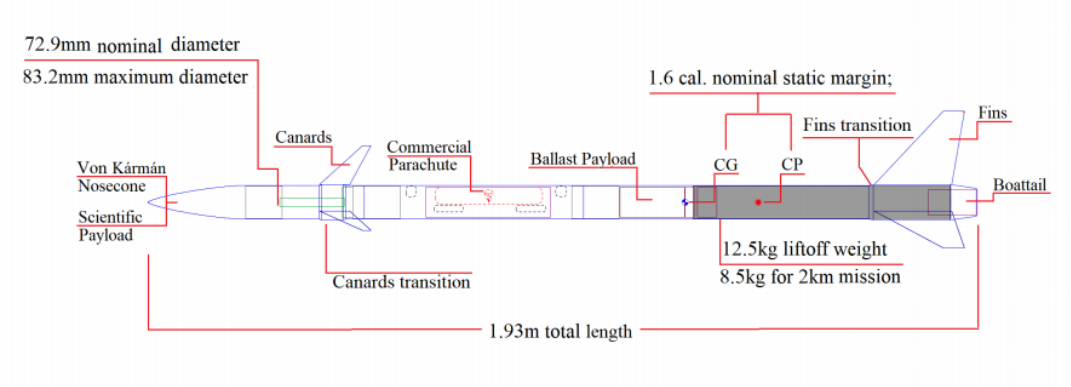


Figure 2 – Detailed *OpenRocket* schematic of the Armação A-22 rocket. Source [1].

The main focus of this work is in studying the Armação A-22 rocket stabilizer's conceptual aerodynamics design; with an specific emphasis in optimizing the fins' geometry to meet certain stability standards while minimizing aerodynamic drag forces. Previous works regarding the Armação A-22 rocket's nosecone aerodynamics have already been carried out [9], with a heavier focus on the use of experimental aerodynamics techniques. During the Armação Project's conceptual design phase, extensive and labor intensive stability and aerodynamics analyses were performed to assure the rocket was safe and capable of fulfilling the mission's requirements. Most of the aforementioned analyses had to be performed manually either using analytical or CFD methods to evaluate the rocket's aerodynamic and stability characteristics, for many different and iterative geometric configurations.

Such a scenario does not lend itself well to design optimization problems (where many successive design alternatives must be evaluated to investigate a sufficiently ample design space), and the main objectives outlined for this work are focused on addressing these issues, accelerating the conceptual aerodynamics and flight stability design phase. Particular focus was put on evaluating a technique elaborated in [10], where Reduced Order Models constructed with partially converged CFD data were created for the optimization of an aerofoil profile. The present work aims to combine this partially converged CFD approach with semi-empirical rocketry aerodynamics methods, by means of Machine Learning based Reduced Order Models employing Transfer Learning.

1.1 Objectives

The general objective of this work is: Proposing a parametric framework for the conceptual aerodynamics and stability optimization of a high power model rocket, using Machine Leaning based Reduced Order Models created by fusing semi-empirical low fidelity data with partially converged CFD results.

Specific objectives include:

- Creating a parametric implementation of the semi-empirical Modified Barrowman Method (lower fidelity model) to estimate the aerodynamic and stability coefficients of a model rocket (with the programming language python);
- Establishing an automated and parametric CFD workflow (higher fidelity model) for transonic high power rocketry external aerodynamics applications (with the open-source CFD software OpenFOAM);
- Designing traditional response surface Reduced Order Models (ROM) built with variable data-sets of partially converged CFD results, as a means to interpolate between discrete data points (with the pySMT toolbox);
- Training Feed-forward Neural Networks with low fidelity, quickly generated data, and using Transfer Learning to train the same networks on higher fidelity data;
- Outlining a suitable design approach combining CFD and Machine Learning to accelerate the solution of conceptual design optimization problems in high power model rocketry aerodynamics, taking restricted computational capabilities into account.

1.2 Work Structure

This work is divided into 5 chapters, with the first two introducing and revising the work's topics, and the latter three expanding upon the engineering techniques implemented and the specific results thereof. In general three main topics are dealt with throughout this work, namely: semi-empirical correlations for rocketry aerodynamics, computational fluid dynamics (CFD), as well as reduced order modelling with machine learning.

Chapter 2 in particular is dedicated exclusively to reviewing relevant technical concepts for the comprehension of this work, with extensive attention being paid to elaborate on the topics of semi-empirical rocketry methods (Section 2.1) and CFD (Section 2.2), with the more complex topics related to Machine Learning and Reduced Order Modelling (Section 2.3) only being tangentially investigated in this study. It is also the author's hope that these introductory chapters can be of use to readers who are beginning their studies in computational methods for rocketry applications.

Chapter 3 exposes the engineering methods related to the three main topics of this Bachelor work. Section 3.1 presents the mathematical formulation of the Modified Barrowman Method used as low fidelity model, while Section 3.2 outlines the parametric computational fluid dynamics workflow taken as a high fidelity model. Again, these two sections comprise most of this work's methodology, and Section 3.3 simply outlines the practical software implementation of chosen ROM and ML models without delving into their mathematical workings.

Finally, Chapter 4 presents the results of the outlined engineering methods. Results for the analytical Modified Barrowman Method are presented in Section 4.1, and the numerical CFD results are discussed in Section 4.2. These latter results were then used in the creation of traditional ROMs exposed in Section 4.3, and both the numerical and analytical results were combined through Transfer Learning in the Machine Learning based ROMs discussed in Section 4.4. Finally Section 4.5 presents the combined finset optimization results. General conclusions about the complete results are drawn in Chapter 5, suggesting possible setups for the proposed conceptual aerodynamics design framework.

2 Technical Review

2.1 Rocketry Fundamentals

Three main forces dictate the dynamical behavior of a rocket during flight: the thrust produced by it's motor, the inertial forces associated with the local gravity and the rocket's mass distribution, and the aerodynamic forces and induced moments developed as the rocket's structure interacts with the flow-field. The present work occupies itself mainly with the last of the aforementioned effects, seeking to determine the rocket's stability and aerodynamic characteristics through means of computational fluid dynamics and semi-empirical correlations. This section aims to provide an overview of some fundamental model rocketry aerodynamics and flight stability concepts and terminology, leaving the full presentation of proper engineering models to later sections of this work.

Evaluating the aerodynamics of high power rockets is a complicated task, and methods of varying complexity and accuracy (both analytical and numerical) have been proposed since the 1950s to achieve such a goal. Ultimately, any fluid mechanics problem can be accurately modeled by the Navier-Stokes set of partial differential equations (reviewed on Section 2.2.1), though such models can often times be too complex and computationally costly to provide meaningful design conclusions during a project's conceptual phase. Many simplified aerodynamic models have therefore been developed, such as the Barrowman method (discussed in Section 2.1.3).

2.1.1 Aerodynamic Forces and Moments

Aerodynamic forces arise as a consequence of the interaction between the rocket and the airflow. As the rocket flies, distributed pressure, shear stress, and temperature fields are developed on all of it's surfaces; largely due to conservation of momentum and energy - as well as viscosity effects - modeled by the Navier-Stokes equations. These distributed fields are often times integrated to determine localized forces, as depicted in Figure 3. Analytical aerodynamic models are normally designed to estimate these localized forces, while CFD techniques are capable of resolving the complete distributed fields.

For axisymmetric flight vehicles, the resultant aerodynamic force acts in a point along the vehicle's center-line (its longitudinal axis) called the center of pressure (x_{cp}). If the x_{cp} is not coincident with the rocket's center of gravity (x_{cg}), then the resultant aerodynamic force will produce a resultant moment about the x_{cg}. The distance between these two points (the colloquial "moment arm") is denominated the static margin (SM), a value which is commonly normalized by the rocket's nominal diameter (d, located at the base of the rocket's nosecone) and expressed as "stability calibers" (cal).

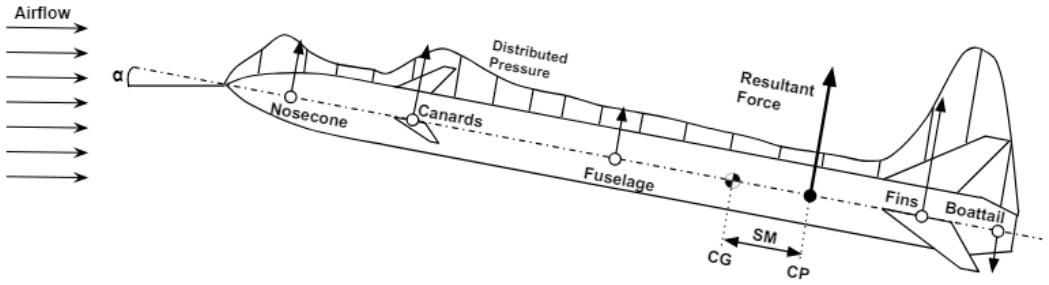
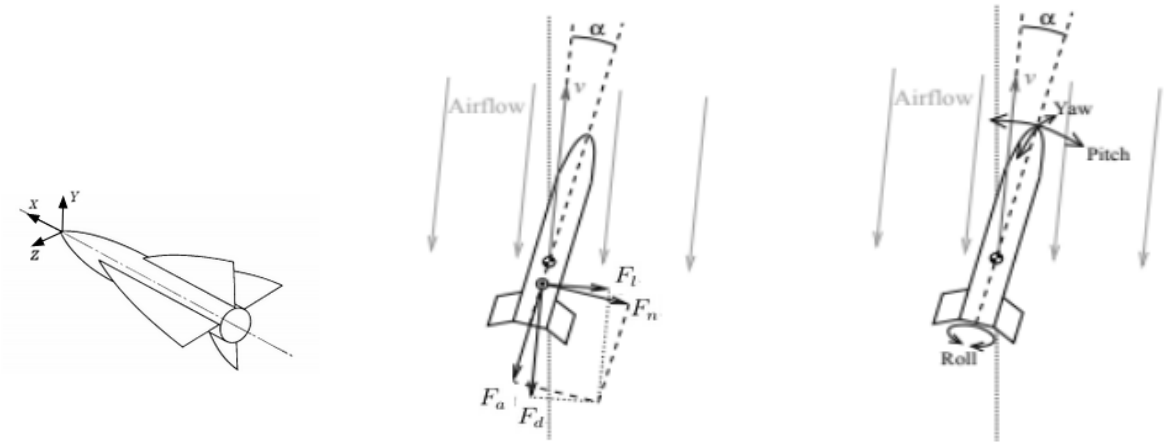


Figure 3 – Simplified schematic illustrating the aerodynamic interaction between a rocket and the surrounding flowfield. Distributed pressure fields are represented by the hatched section, and localized forces are represented by the arrows.

Aerodynamic forces greatly depend on the rocket's velocity, a value which is typically expressed through the Mach number (M_a). The Mach number is a non-dimensional value defined as the ratio between the local speed of sound (c) and the rocket's speed. The local speed of sound is a function of the air's temperature, but it usual to assume a constant value of $c = 340\frac{m}{s}$ for low apogee rockets. M_a is a useful constant for classifying different aerodynamic regimes, with $M_a > 0.3$ typically indicating flow compressibility, and $M_a > 1$ indicating supersonic flows. The Armação A-22 rocket operates within the transonic regime with a maximum M_a around 0.7, indicating that localized supersonic flow can occur.

The aerodynamic forces and moments are also a function of the relative orientation between the rocket and the airflow. The angle of attack (α, also commonly expressed as AoA) can be defined as the angle between the rocket's instantaneous trajectory and prevalent wind direction. α is also defined to lie on one of the rocket's symmetry planes, and is associated with the vehicle's pitching axis. Another important relative angle is the side slip angle (β), associated with the rocket's yawing motion. These angles are close to zero for nominal flight conditions, and their magnitudes only increase drastically during maneuvers or when the rocket's velocity is on the same order of magnitude as the ambient wind velocity, which can happen during launching procedures. Previous Apex Rocketry works [9] sought to develop experimental procedures to measure α during flight.

It is usual to decompose the resultant aerodynamic force in components parallel and perpendicular to the airflow orientation, resulting the vehicle's drag (F_d) and lift (F_l) forces, respectively. Some references prefer to deal with normal (F_n) and axial (F_a) force components, which is an equally valid approach, given that both representations constitute independent basis (as indicated in Figure 4b). A variety of factors are responsible for creating the lift and drag forces, but for this brief review it is sufficient to note that lift forces are normally associated with the pressure gradient along the rocket's stabilizers (and commonly have a linear dependency with α), while the drag forces are associated



(a) Reference coordinate frame. (b) Aerodynamic force decomposition. (c) Reference stability axis.

Figure 4 – Standard coordinates, force decomposition, and stability axis used in this work. Adapted from [2] and [3].

with viscosity and interference effects (typically showing quadratic dependencies with α).

It is also typical to normalize the aerodynamic forces and moments acting on the rocket, in order to extract meaningful non-dimensional coefficients used in many design decisions. This is done by diving the forces by the rocket's nominal dynamic pressure (q_∞). The coefficients for the lift (F_l) and drag (F_d) forces, as well as the pitching moment (M), can be defined as:

$$C_l = \frac{F_l}{q_\infty A_c}, C_d = \frac{F_d}{q_\infty A_c}, C_m = \frac{M}{q_\infty A_c d} \tag{2.1}$$

where $q_\infty = \frac{1}{2}\rho U_\infty^2$, with ρ being the air density, U_∞ the freestream velocity. A_c and d are the reference area and length, typically being the rocket's nominal cross sectional area and it's diameter, respectively. It is important to mention that these coefficient can be derived for the total aerodynamic forces and moments, as well as for each individual rocket component.

2.1.2 Static and Dynamic Stability

The static stability of a rocket is related to the vehicle's tendency of retaining its original position when subjected to unbalanced forces or moments. A flight vehicle that is capable of producing restoring moments when perturbed is said to be statically stable. Dynamic stability is related to the form of motion a (statically stable) rocket undergoes when it tries to return to its original position. A flight vehicle which, after a perturbation, presents stable and decaying oscillations about the mean flight trajectory is said to be dynamically stable. Figure 5 adapted from [4] illustrates these definitions.

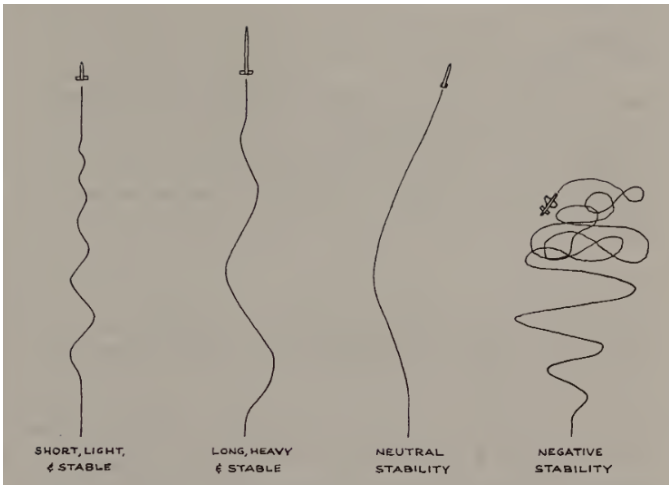


Figure 5 – Sketch depicting the launch trajectory profile for rockets with varying stability characteristics. Source [4].

Nielsen's classic Missile Aerodynamics text [2] defines a rocket's stability derivatives as *"the rates of change of any force or moment coefficient with respect to linear or angular velocity components or time derivatives thereof"*. These quantities are the fundamental aerodynamic inputs for dynamical flight analyses, and are specially important in evaluating a rocket's safety before flight. Stability derivatives are useful in determining both static and dynamic stability criteria. In particular, the static longitudinal stability derivative (C_{m_α}) and the pitch damping stability derivative ($C_{m_{\dot{\alpha}}}$) are of fundamental importance, being analogous to the "spring" and "damper" elements of an idealized mass-spring-damper system. Equation 2.2 enunciates the definitions for the aforementioned stability derivatives given in [2].

$$C_{m_\alpha} = \frac{\partial C_m}{\partial \alpha}, C_{m_{\dot{\alpha}}} = \frac{\partial C_m}{\partial \dot{\alpha}} \tag{2.2}$$

where $\dot{\alpha}$ is the angular velocity (first temporal derivative of α).

These derivatives can be somewhat complex to determine, especially the pitch damping given its dynamic nature (though many recent works propose novel methods of estimating such coefficients through CFD [11]). Therefore, it is useful to expresses these derivatives using simpler definitions which maintain their original physical interpretation.

The static margin (SM) - previously discussed in Section 2.1.1 - is the principal measure of a rocket's static stability, and is defined in Equation 2.3 [2, 3], where the distances (x) are relative to the rocket's nosecone tip and $C_{m,cg}$ is the coefficient of the pitching moment around the rocket's center of gravity. A rocket is defined as statically stable if $SM > 0$, meaning that its aerodynamic forces will produced a restoring moment when a perturbation removes the rocket form its original flight orientation. Values of SM close to zero indicate that the rocket will be neutrally stable (with no restoring moment), and $SM < 0$ implies that the rocket is statically unstable (moments destructively increase

the oscillations). The static margin is very closely related to the static longitudinal stability derivative, as indicated by the latter equality of Equation 2.3.

$$\text{SM} = \frac{(x_{CG} - x_{CP})}{d} = -\frac{\frac{\partial C_{m,cg}}{\partial \alpha}}{\frac{\partial C_l}{\partial \alpha}} = -\frac{\partial C_{m,cg}}{\partial C_l} \tag{2.3}$$

The damping ratio (ξ) is a dimensionless number closely associated with the pitch damping stability derivative. It is essentially an indicator of the nature of the corrective motion of statically stable rocket. $\xi < 1$ indicates that the rocket is under-damped, meaning that it will smoothly oscillate back-and-forth until eventually returning to the flight's original direction. $\xi \approx 1$ indicate that the rocket is critically damped and its motion is not oscillatory. Over-damped rockets ($\xi > 1$) are dynamically unstable, meaning that they cannot return to the original flight path (presenting a similar behavior to statically neutral rockets). Mandell's seminal MIT research (publicized in model rocketry handbooks and hobbyist magazines during the 1960s [12]) defines ξ as:

$$\xi = \frac{C_2}{2\sqrt{I_L C_1}} = \frac{C_2}{2\omega_n I_L} \tag{2.4}$$

where $\omega_n = \sqrt{\frac{C_1}{I_L}}$ is the natural frequency of oscillation, I_L is the rocket's longitudinal moment of inertia, and C_1 and C_2 are referred to respectively as the corrective and damping moment coefficients and can be defined as:

$$C_1 = \frac{\rho A_c U_\infty^2}{2} C_{n_\alpha}(x_{cp} - x_{cg}) \tag{2.5}$$

$$C_2 = \frac{\rho A_c U_\infty}{2} \sum C_{n_\alpha,i}(x_{cp,i} - x_{cg})^2 + \dot{m}(x_{nozzle} - x_{cg})^2 \tag{2.6}$$

where C_{n_α} refers to the total normal force coefficient derivative and $C_{n_\alpha,i}$ are the same coefficient derivatives in a per-component basis (likewise for the location of the centers of pressure $x_{cp,i}$). The first term of Equation 2.6 is the aerodynamic damping, while the second term ($\dot{m}(x_{nozzle} - x_{cg})^2$) is the jet damping, a propulsive phenomenon in which the exhaust gasses' massflow rate ($\dot{m}$) is responsible for added damping in the dynamic system.

Both SM and ξ are fundamental criteria in determining a rocket's safety and flight readiness. It is commonplace for rocketry competitions to regulate these stability parameters (especially the SM), normally demanding minimum SM between values 1 - 1.5. Competition regulators also often indicate that over-stable rockets ($SM > 3$) can be dangerously susceptible to adverse weather effects, especially weathercocking, where a rocket can align itself with the prevalent horizontal wind direction. Literature [12] also indicates that some level of under-damping is beneficial, given that the rocket is subjected to dynamically shifting perturbations from the ambient weather conditions. Values of ξ

above $\frac{\sqrt{2}}{2}$ are deemed excessively close to critical damping (and prone to weathercocking), and it is recommended that the value of ξ lay between 0.05 and 0.2.

2.1.3 Semi-empirical Rocketry Aerodynamics

The Barrowman method [13] is a semi-empirical procedure to estimate a rocket's aerodynamic and stability coefficients. The method is derived from analytical potential flow aerodynamic formulations, as well as different empirical test data common to the first half of the 20^{th} century. Since then, it has been amply used and improved upon by model rockeeters. OpenRocket is a widely used open source trajectory simulation software for model rockets developed by Niskanen [3]. It uses a modification of the Barrowman method proposed by Galejs [14] as a basis for the software's aerodynamics module.

The Barrowman method and it's many modifications have the main goal of estimating the aerodynamic coefficients of a high power model rocket, ultimately correlating the rocket's aerodynamic and stability coefficients with its geometrical characteristics. In particular, most formulations of the Barrowman method seek to determine the rocket's axial and normal force coefficients (C_a and C_n, respectively). As discussed in Section 2.1.1, C_n and C_a constitute an independent basis and can be easily converted to the more general C_l and C_d coefficients. Barrowman's formulation also determines the location of the center of pressure (x_{cp}) for each of the rocket's components, therefore being capable of estimating both the SM and ξ.

This section of the text will be dedicated to enunciating the main hypothesis and structure of the modified Barrowman method implemented in this work. The mathematical formulation presented in this research was adapted from [2, 3, 15] and is fully described in Section 3.1. The orginal Barrowman method adopts the following hypotheses:

1. The rocket is an axisymmetric rigid body;
2. The air flow over the rocket is smooth and does not change rapidly;
3. The fins are thin flat plates;
4. The nose of the rocket comes smoothly to a point;
5. The rocket is thin compared to it's length;
6. The angle of attack of the rocket is small at all times ($\alpha < 10°$);
7. Lift forces on the rocket body tube can be neglected;
8. Viscous forces are negligible;
9. The effects of compressibility can be neglected ($M_a < 0.3$).

Hypotheses 1 through 5 are necessary for the potential flow aerodynamics calculations. Hunt et al [15] and Niskanen [3] still partially consider hypotheses 4 thru 9, but propose some corrections to extend the validity of the original Barrowman equations.

Barrowman's original equations assume that the rocket's normal force is linear. With this consideration, it is possible to fully determine the normal force coefficients only by determining it's derivative (C_{n_α}). This is only valid for small AoA values and if the fuselage's lift forces are neglected. Galejs [14] proposals generalize Barrowman's original linearizations of C_n, accounting for non-constant (AoA dependent) values of C_{n_α}. This technique ultimately weakens the restrictions imposed by hypothesis 6 and 7 of the original Barrowman method.

Barrowman's method proposes analytical correlations to determine the normal force derivatives ($C_{n_\alpha,i}$) and the center of pressure locations ($x_{CP,i}$) for each i^{th} component of the rocket. The method then derives the total rocket coefficients as the sum of each component's contribution. Section 3.1 details how $C_{n_\alpha,i}$ and $x_{CP,i}$ are calculated for each component, and used to determine the rocket's lift coefficient (C_l), static margin (SM), and dynamic stability parameters (ξ and ω_n).

The modified Barrowman method implemented in [3, 15] is also capable of estimating the rocket's drag forces in a per-component basis. To do this, hypothesis 8 is corrected to estimate viscous forces through empirical formulae derived from test data. Section 3.1 illustrates how these drag terms can be estimated in a per-component basis for the modified Barrowman method developed in this work.

To extend the Mach number range of validity of the Barrowman equations and correct hypotesis 9, Galejs [14] proposes the use of the Prandtl-Glauert correction (valid for $M_a < 1$):

$$C'_\lambda = \frac{C_\lambda}{\sqrt{1 - M_a^2}} \tag{2.7}$$

where C_λ is an arbitrary aerodynamic coefficient, and C'_λ is it's compressibility-corrected counterpart.

2.2 Computational Fluid Dynamics

In broad terms, computational fluid dynamics (CFD) refers to an ample collection of numerical techniques used in the modeling of fluid mechanics problems common to many engineering fields. In the context of high power rocketry aerodynamics, CFD engineering models are commonly employed to estimate the aerodynamic coefficients of low apogee launch vehicles [16, 17]. Static and dynamic stability coefficients, such as those described in Section 2.1.2, have also been estimated through CFD in the context of tactical missiles and fighter aircrafts [11, 18].

This brief literature review aims to succinctly present a physical interpretation of the mathematical formulation behind the Reynolds-averaged Navier-Stokes (RANS) modeling framework, which is widely used in industrial applications and implemented in all of the mainstream CFD softwares. Emphasis is placed on illustrating the main concepts of the CFD model later employed in the optimization framework of the Armação rocket's finset. The principal characteristics and nomenclature associated with the open source CFD software *OpenFOAM* is also described in this section, so as to facilitate the comprehension of the CFD models developed in Section 3.2. The mathematical review presented here was adapted from [19].

2.2.1 Reynolds-Averaged Navier-Stokes Equations

The Navier-Stokes (NS) set of partial differential equations governs the motion of fluids. This system of equations is described by the mass conservation (continuity) equation 2.8, the momentum conservation equations 2.9, the energy conservation equation 2.10, and an algebraic equation of state relating pressure and internal energy (as exemplified by Equation 2.11, for an ideal gas).

The NS set of equations correlate the state variables that define the fluid with both the spatial (x) and temporal (t) dimensions of the fluid domain. Both scalar quantities (such as the pressure p, temperature T, density ρ, and specific total mechanical energy E) and the vector quantity of velocity (u) are modeled in the NS equations. In Equations 2.8 thru 2.10 the vector quantities are written following Einstein's notation, with the indexes (i, j, k) representing the three spatial dimensions of the fluid domain.

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u_i}{\partial x_i} = 0 \tag{2.8}$$

$$\left(\frac{\partial \rho u_i}{\partial t} + \frac{\partial \rho u_j u_i}{\partial x_j}\right) = -\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}\left[\mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) - \frac{2}{3}\mu\frac{\partial u_k}{\partial x_k}\delta_{ij}\right] + \rho F \tag{2.9}$$

$$\left(\frac{\partial \rho E}{\partial t} + \frac{\partial \rho u_j E}{\partial x_j}\right) = -\frac{\partial p u_j}{\partial x_j} + \frac{\partial u_i \tau_{ji}}{\partial x_j} + \frac{\partial}{\partial x_j}\left(k\frac{\partial T}{\partial x_j}\right) + S_E \tag{2.10}$$

$$p(\rho, T) = \rho R T \tag{2.11}$$

In brief terms; Equation 2.8 states that the time derivative of the fluid's density must be equal to the divergent of its linear momentum, indicating that the expansion or contraction of the fluid domain must alter its density in order to satisfy the conservation of mass. Equation 2.9 follows the same principle of conservation. It states that the transient and advective terms (on the left hand side) representing the material (or total) derivative - combining both temporal and spatial derivatives - of the fluid's momentum must satisfy Newton's Second Law and be equal to the forces acting on a fluid, modeled by: the pressure gradient imparted on the fluid ($-\frac{\partial p}{\partial x_i}$), the diffusive terms associated

with the shear stress tensor ($\tau_{ij} = \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) - \frac{2}{3}\mu\frac{\partial u_k}{\partial x_k}$), and any external forces (F). Similarly, Equation 2.10 follows from the First Law of thermodynamics in stating that the change in total energy (its material derivative on the left hand side) must be a result of the work done by the pressure ($\frac{\partial p u_j}{\partial x_j}$) and shear stress ($\frac{\partial u_i \tau_{ji}}{\partial x_j}$) terms, as well as the heat flux ($\frac{\partial}{\partial x_j}\left(k\frac{\partial T}{\partial x_j}\right)$) and generation ($S_E$) in the fluid domain.[1]

For turbulent unsteady flows, any generic flow state variable ϕ can be decomposed into its mean (time-averaged) $\overline{\phi}$ and fluctuating $\phi^{'}$ parts. For compressible flows a further decomposition (known as Favre-averaging) must be performed; in which variables are decomposed so as to take density variations into account, namely: $\phi = \tilde{\phi} + \phi^{''}$, where $\tilde{\phi} = \overline{\rho\phi}/\overline{\rho}$ and $\phi^{''}$ represents both turbulent and density fluctuations. By applying standard time-averaging and Favre-averaging to the NS equations (Eqs. 2.8, 2.9, 2.10) one obtains the standard form of the Reynolds-averaged Navier-Stokes (RANS) equations:

$$\frac{\partial \overline{\rho}}{\partial t} + \frac{\partial \overline{\rho}\tilde{u}_i}{\partial x_i} = 0 \tag{2.12}$$

$$\left(\frac{\partial \overline{\rho}\tilde{u}_i}{\partial t} + \frac{\partial \overline{\rho}\tilde{u}_j\tilde{u}_i}{\partial x_j}\right) = -\frac{\partial \overline{p}}{\partial x_i} + \frac{\partial}{\partial x_j}\left(\overline{\tau}_{ij} - \overline{\rho u_i^{''} u_j^{''}}\right) \tag{2.13}$$

$$\left(\frac{\partial \overline{\rho}\tilde{E}}{\partial t} + \frac{\partial \overline{\rho}\tilde{u}_j\tilde{E}}{\partial x_j}\right) = -\frac{\partial \overline{\rho}\tilde{u}_j}{\partial x_j} + \frac{\partial \overline{u_i\tau_{ij}}}{\partial x_j} - \frac{\partial}{\partial x_j}\left(\overline{q}_j + \overline{u_j^{''}p} + \overline{\rho u_j^{''}E^{''}}\right) \tag{2.14}$$

where the Favre-averaged shear stress tensor is $\overline{\tau}_{ij} = \tilde{\tau}_{ij} + \overline{\tau_{ij}^{''}}$, and q_j in Equation 2.14 refers to the heat flux (also present in Equation 2.10). This Favre-averaged time dependent form of the RANS equations is the formulation implemented in many *OpenFOAM* [20] solvers (including the *rhoSimpleFoam* solver used in the CFD models developed in this work).

It is worth commenting here that OpenFOAM (and most of the common CFD codes) use the finite volume method to transform the partial differential RANS equations into a system of matrix equations that can mostly be resolved through linear algebra. In doing so, the finite volume method allows for the quick numerical resolution of matrix equations by computational methods, being therefore crucial to the formulation implemented in the standard CFD softwares.

The main idea behind the Reynolds-averaged Navier-Stokes equations is that the mean flow ($\overline{\phi}$) variables are easier to model and quantify than the real, time-depended, variables (ϕ) of the standard Navier-Stokes equations. However, Equations 2.12 thru 2.14 still have many unresolved turbulent fluctuation ($\phi^{''}$) terms. Therefore, in order to properly model the fluid, some closure model must be assumed to define these turbulent terms in the RANS equations.

[1] A complete description of the conservation principles associated with the NS equations is freely available (in Portuguese) in Professor Maliskas' post graduate online lectures at UFSC: https://www.youtube.com/playlist?list=PLIPfGy5ZylmMiZ58EeuNLDfjQn5xdQVey.

2.2.1.1 Turbulence Modeling

Turbulence models act as closure models to the RANS equations by modeling the turbulent fluctuations of the flow variables in terms of the variable's mean (time-averaged) value. Many turbulence models have been proposed in the last decades to facilitate industrial CFD applications. In the context of external aerodynamics, literature often proposes the Spalart-Allmaras [21] and the $k-\omega$ SST [22] as good general closure models for industrial applications. This work will focus on the $k-\omega$ SST model as it is readily available in *OpenFOAM* and showed promising results in similar case studies [23].

The $k-\omega$ SST model is a hybrid formulation using the $k-\omega$ model in the near-wall region (to capture the flow's boundary layer formation), and the $k-\epsilon$ method on the freestream, fully turbulent regions. Only the $k-\omega$ equations will be presented here, but a similar derivation is done with $k-\epsilon$. Menter et al [22] models the momentum and thermal turbulent boundary layer by correlating the turbulent kinematic viscosity (ν_t) and thermal diffusivity (α_t) of the fluid with two modeling variables: the turbulent kinetic energy (k), and the rate of dissipation of the turbulent eddies (ω).

$$\mu_t = \rho\frac{k}{\omega} \implies \nu_t = \frac{k}{\omega} \tag{2.15}$$

$$\alpha_t = \frac{\nu_t}{Pr_t} = \frac{1}{Pr_t}\frac{k}{\omega} \tag{2.16}$$

where Pr_t is the fluid's turbulent Prandtl number. The dynamic viscosity (μ_t) of the flow is closely associated with the shear stress tensor (τ_{ij}) responsible for many of the turbulent fluctuation terms in the RANS equations:

$$\tau_{ij} = -\rho\overline{u_i'u_j'} = \mu_t\left(\frac{\partial\overline{u}_i}{\partial x_j} + \frac{\partial\overline{u}_j}{\partial x_i}\right) - \frac{2}{3}\rho k\delta_{ij} \tag{2.17}$$

The transport equations for k (Eq. 2.18) and ω (Eq. 2.19) are:

$$\frac{\partial\rho k}{\partial t} + \frac{\partial\rho k\overline{u}_i}{\partial x_i} = \frac{\partial}{\partial x_i}\left[\left(\mu + \frac{\mu_t}{\sigma_k}\right)\frac{\partial k}{\partial x_i}\right] + 2\mu_t\left(\frac{\partial\overline{u}_i}{\partial x_j} + \frac{\partial\overline{u}_j}{\partial x_i}\right)^2 - \frac{2}{3}\rho k\frac{\partial\overline{u}_i}{\partial x_j}\delta_{ij} - \beta^*\rho k\omega \tag{2.18}$$

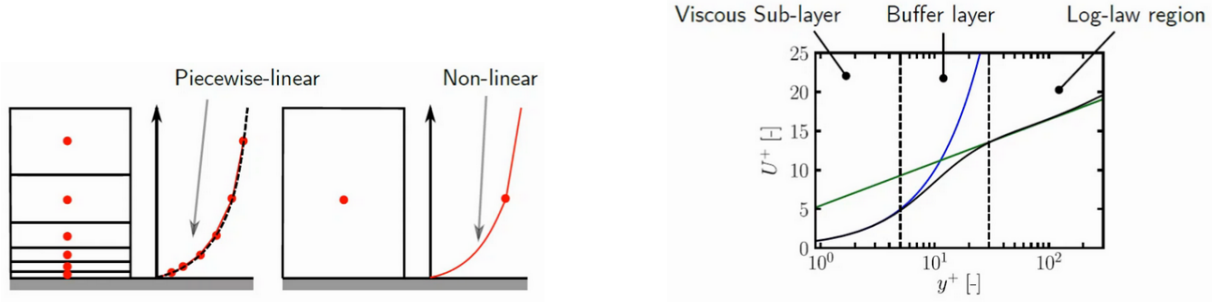
$$\begin{aligned}\frac{\partial\rho\omega}{\partial t} + \frac{\partial\rho\omega\overline{u}_i}{\partial x_i} = \frac{\partial}{\partial x_i}\left[\left(\mu + \frac{\mu_t}{\sigma_{\omega,1}}\right)\frac{\partial\omega}{\partial x_i}\right] + \gamma_2\left[2\rho\left(\frac{\partial\overline{u}_i}{\partial x_j} + \frac{\partial\overline{u}_j}{\partial x_i}\right)^2 - \frac{2}{3}\rho\omega\frac{\partial\overline{u}_i}{\partial x_j}\delta_{ij}\right] \\ -\beta_2\rho\omega^2 + 2\frac{\rho}{\sigma_{\omega,2}\omega}\frac{\partial k}{\partial x_k}\frac{\partial\omega}{\partial x_k}\end{aligned} \tag{2.19}$$

where σ_k, β^*, β_2, $\sigma_{\omega,1}$, $\sigma_{\omega,2}$ and γ_2 are all model constants; and δ_{ij} is the Dirac delta function.

Further elaborations on the relationship between the turbulence closure models and the spatial and temporal domain discretization (i.e.: the meshing algorithms and strategies)

are beyond the scope of this review; tough it is still relevant to make some concluding remarks about the effects of different meshing boundary layer approaches on the RANS formulation.

In meshing the boundary layer for any given aerodynamic flow, the CFD model can follow one of two general approaches. In the first method the mesh must be fully resolved, that is the discrete volumes of the fluid domain close to the solid region must be sufficiently small to accurately model the developed boundary layer in a picewise-linear fashion (Figure 6a, left side). This method is referred to as the "low Reynolds" (low-Re) approach, due to the fact that the Re number at the fluid elements near the wall will be very small. The second approach is referred to as the "high Reynolds" (high-Re) approach, in which the mesh is purposefully bigger than the low-Re mesh elements near the wall region (Figure 6a, right side). In this latter method the boundary layer profile is estimated through the use of a non-linear wall function. Nilsson [5] details how these turbulence model approaches are implemented in *OpenFOAM*.



(a) Meshing strategies for low-Re (left) and high-Re (right) modeling approaches.

(b) $y+$ and $U+$ graph illustrating different flow regions.

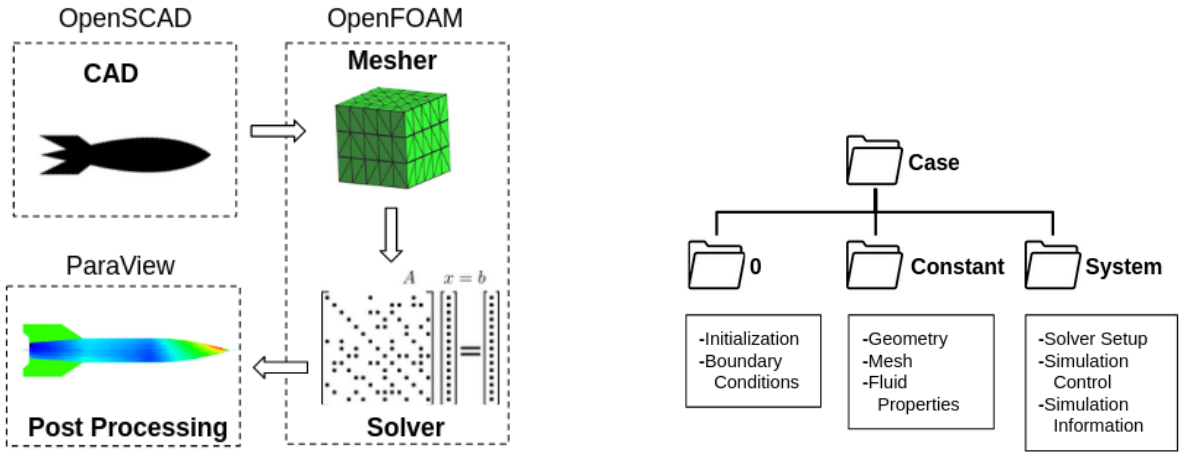
Figure 6 – Schematics illustrating the difference between fully resolved boundary layer (low-Re) and wall function (high-Re) turbulence modeling approaches. Source [5].

The $y+$ parameter shown in Figure 6b is the non-dimensional wall distance related to the flow parameters ate the mesh's element nearest to the wall. Likewise, $U+$ is the non-dimensional velocity near the wall. The $y+$ parameter is fundamental in establishing the range of validity of a given turbulence modeling approach, with low-Re approaches requiring $y+$ in the viscous sub-layer ($y+ < 5$ and preferably close to 1) and high-Re approaches requiring $y+$ values in the log-law layer ($y+ > 30$).

Depending on the meshing approach implemented, the boundary conditions for k and ω at the wall are changed. In the low-Re method k and ω are imposed as fixed values, with k usually set to a small numerical value tending to zero, and ω set to a large value tending to infinity (or calculated with correlations from [22]). In the wall function (high-Re) approach, non-linear correlations estimate the values of k and ω at the wall.

2.2.2 CFD modeling with *OpenFOAM*

OpenFOAM (Open-source Field Operation And Manipulation) is a C++ library for the development of customized numerical solvers for continuum mechanics problems, specially focused in computational fluid dynamics. Being an open-source code, different frameworks or "flavors" of the source code are distributed, maintained, and constantly updated by different organizations. In this work, OpenFOAM v2012 was used, being the latest release (December of 2020) by OpenCFD.



(a) General CFD model construction workflow.

(b) General OpenFOAM case structure.

Figure 7 – Schematics illustrating some basic concepts and terminology used in the construction of CFD models with OpenFOAM.

OpenFOAM follows the same modeling workflow as standard, more widely used, commercial CFD softwares. It consists of: constructing a geometric model of the flow domain through CAD methods; discretizing (meshing) this domain into finite fluid volumes; numerically solving the RANS equations through the matrix system obtained by the finite volume method; analyzing the converged results determined for each discrete volume of the fluid domain. This workflow is illustrated in Figure 7a. OpenFOAM is capable of performing all of the aforementioned steps, though it is typically used only in the meshing and solving stages of the CFD model construction. In this work, third party softwares were used to construc the geometry (OpenSCAD) and analyze the simulation results (ParaView).

Being primarily a C++ code library, OpenFOAM does not have a graphical user interface through which a user can pass the necessary inputs for a given CFD model. Instead, OpenFOAM relies exclusively on a text-based approach structured by a specific hierarchy of folders containing executable C++ files. This folder structure (referred to as the case folder structure) is exemplified in Figure 7b. This text-based execution approach lends itself extremely well to automation. All of the files necessary to execute a CFD

model in OpenFOAM can be easily accessed and modified by external tools, simplifying the development of parametric CFD analysis.

2.3 Reduced Order Modeling

Reduced order models (ROMs, also commonly known as surrogate models, meta models, or response surface models) are simplifications of high-fidelity, complex models, aimed at capturing the behavior of these source models so that engineers can quickly study a system's dominant effects using minimal computational resources. ROMs typically act as engineering "black-boxes", mathematically mapping a set of input design parameters to an output metric space. Such models can themselves also vary in mathematical complexity, ranging from simple linear regression models, to computationally advanced artificial intelligence algorithms.

Owning to their almost real time evaluation speed, reduced order models play a crucial role in parametric evaluations, being one of the central mathematical tools used in numerical or experimental design of experiments (DOE) analysis. Another benefit of this engineering "black-box" modelling approach is the potential to perform combined, multidisciplinary evaluations, essential for the integrated analysis of a flight vehicle [24]. In the field of external aerodynamics for flight vehicles, model order reduction techniques have long been implemented both as tools for multi-disciplinary design optimization, and for predicting complete flow features by accelerating (or simplifying) computationally intensive CFD analyses [25].

The general implementation of any reduced order model requires the prior existence of a robust data-set, functionally correlating input and output parameters. The creation of this arbitrary data-set (through experimental or numerical means) is the most labor intensive aspect of the entire process, and the creation of ROMs is meant to precisely expedite the time consuming repetition of these expensive analysis. ROMs can therefore be often described as multi-dimensional fitting tools, helping to interpolate between disparate data points within a sampled design space.

Another statistical tool that perfectly fits this description are machine learning (ML) models, which have recently also been extensively used to assist model order reduction algorithms for optimization problems in aerospace applications [26]. Machine learning for fluid dynamics is a very current and expanding research field, with many works outlining ML applications to complete fluid flow modeling, as well as to surrogate design models [8] [27]. This works aims to use readily available open source packages using both traditional and machine learning assisted reduced order models, and this brief review will focus on presenting the implemented tools in the automated conceptual aerodynamic design framework discussed here.

2.3.1 Traditional ROMs with pySMT

Python Surrogate Modelling Toolbox (pySMT [28]) is a open source python library that contains a collection of reduced order modeling methods, sampling techniques, and benchmarking functions. SMT is different from existing surrogate modeling libraries because of its emphasis on derivatives, which play an important role in optimization, because problems with a large number of optimization variables require gradient-based algorithms for efficient scalability.

PySMT offers a range of non-random sampling functions, crucial in defining the number of analysis (in this case CFD simulations) points required for the construction of a ROM. The Latin Hypercube Sampling (LHS) algorithm in particular is among the most popular sampling techniques in computer experiments thanks to its simplicity and projection properties with high-dimensional problems. LHS is built as follows: each dimension space representing a variable is partitioned into n sections, where n is the number of sampling points, and only one point is put in each section [6]. This, in effect, means that the design space is more thoroughly and evenly sampled then it would have been with a random sampling of points constrained by the design boundaries, as illustrated by an arbitrary 2D sampling of a square grid shown in Figure 8.

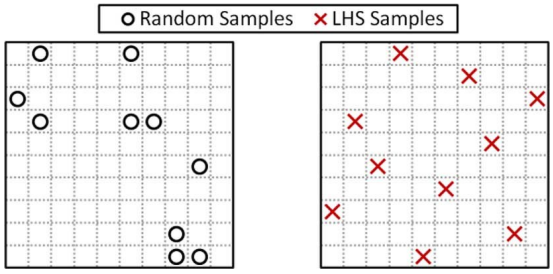


Figure 8 – Illustration of the LHS sampling strategy performance against random sampling methods. Adapted from [6]

Many traditional surrogate modelling functions are available in pySMT, such as linear least-squares models, second-order polynomial approximators, or inverse distance weighting algorithms. Out of these traditional models, Radial Basis Functions (RBF) have been extensively used in reduced order modelling for engineering applications [29]. The RBF reduced order model represents the interpolating function as a linear combination of basis functions, individually assigned for each training point. RBFs are named as such because the basis functions depend only on the distance from the prediction point to the training point for the basis function; essentially being an equation of the form $\Phi(\mathbf{x_e}) = \hat{\Phi}(||\mathbf{x_e} - \mathbf{x_t}||)$, where $\mathbf{x_e}$ is an multidimensional evaluation point, and $\mathbf{x_t}$ is a fixed training point. PySMT in particular implements Gaussian Basis functions of the form shown in equation 2.20, where the only model hyper-parameter is the Gaussian scaling factor d_o. In general, RBF models are simple (only requiring one tuning parameter),

show fast training for small number of training points, but are susceptible to oscillations and numerical issues when points are too close to each other.

$$\Phi(\mathbf{x_e}, \mathbf{x_t}) = e^{\left(\frac{||\mathbf{x_e}-\mathbf{x_t}||^2}{d_o}\right)} \tag{2.20}$$

Another interesting class of traditional ROM methods implemented by pySMT are the Kriging-family models, which are closely related to regression analysis. Kriging is an interpolating model that linearly combines a known function with the result of a stochastic process, predicting the value of a function at a given point by computing a weighted average of the known values of the function in the neighborhood of the point. Kriging models can be extended to utilize gradient information when available. Such methods are known in the literature as gradient-enhanced kriging (GEK) [7], and usually more accurate than Kriging, however, not being computationally efficient when the number of inputs, the number of sampling points, or both, are high.

To allow for a larger number of sampling points or design parameters, pySMT in implements GEKPLS, a gradient-enhaced kriging with partial least squares (PLS) approach. The key idea of the GEKPLS method consists in generating a set of approximating points around each sampling point, by using the first order Taylor method, approximating the gradient information between all adjacent sampled points. Then, the PLS method is applied several times, each time on a different number of sampling points, providing a set of coefficients that gives the contribution of each variable nearby the associated sampling point to the output. Finally, an average of all PLS coefficients is computed to get the global influence to the output. Figure 9 illustrates a simple 1-D ROM created with the Kriging and GEK methods, so as to highlight how the inclusion of gradient data can increase the prediction accuracy, especially for low dimensional problems with small sample spaces. A thorough mathematical formulation of the GEKPLS method is presented in [7].

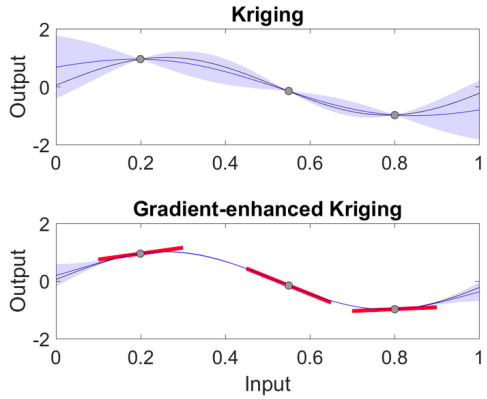


Figure 9 – Illustration of the standard and gradient enhanced Kriging approximation methods. The dots represent the measured (sampled), the black curve represents the true data, while the blue curve represents the ROM constructed with both models, with a shaded accuracy region highlighted in blue. Red lines symbolize the region where first order Taylor approximations are included in the GEK method. Adapted from [7]

.

2.3.2 Machine Learning based ROMs

Much like the traditional reduced order models discussed so far, Machine learning algorithms also build mathematical models based on sample data, known as training data, in order to make predictions or decisions, but crucially without being explicitly programmed to do so. In the field of computational fluid mechanics in particular, Machine Learning can provide a modular and agile modeling framework that can be tailored to address many challenges, such as reduced-order modeling, experimental data processing, shape optimization, turbulence closure modeling, and flow control technologies.

Deep Feed-forward Neural Networks in particular have shown promising results as ROM tools for fluid dynamics problems, as outlined in [8] [30]. Feed-forward Neural Networks (FNN) (often referred to as single or multi-layer perceptrons), are the simplest form of artificial neural network, in which the information only flows in one direction inside the algorithm. In concrete terms this means that different nodes of the model are unidirectionally connect in between each model layer, as depicted in Figure 10.

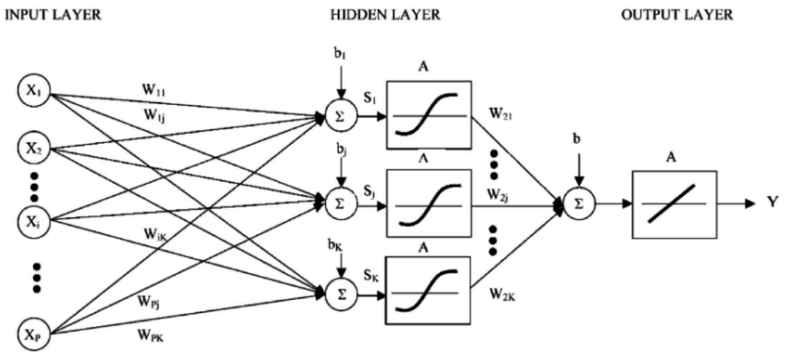


Figure 10 – Example of a Feed-forward Neural Network architecture. **X** and **Y** denote the input and output parameters, while **W** and **b** represent the weights and biases associated with each synapse between layers. **A** stands for the activation function applied to the neural output signal (**S**) resulting from the sum of all input **W** and **b**. Adapted from [8]

.

In summary, an FNN architecture (as depicted for a single hidden layer FNN in Figure 10) typically follows the most general working principles of Machine Learning. The values of an input vector **X** are multiplied by arbitrary weights **W** and fed forward thru the model. Each node in the next layer sums up all incoming **WX** values, adding whatever arbitrary bias **b** that may be applied to this hidden node. The resulting neural signal **S** is evaluated by an Activation function **A**, which feeds the signal forward thru the model, in the case **S** meets a predefined threshold, essentially only selecting the most important nodes in predicting the final output **Y**. Finally, a final activation function is applied in the output layer to convert the numerical values back to original input dimensions.

In such an ML architecture, the learning process is controlled by a back-propagation algorithm. The back-propagation algorithm works by computing the gradient of the loss function (correlating a predicted output **Y** with its real, ground truth, training counterpart $\mathbf{Y}^*$) with respect to each weight by the chain rule, dynamically computing the gradient one layer at a time. By using gradient based optimization techniques during this computational step, back-propagation algorithms can essentially fine-tune the weights of a neural net based on the error rate (i.e. loss computed by the chosen loss function) obtained in the previous iteration (also referred to as epoch). Proper tuning of the weights ensures lower error rates, making the model reliable by increasing its generalization. Many hyper-parameters can be tuned in an Feed-forward Neural Network architecture, such as the number of hidden layers, the number of nodes (or neurons) per layer, and the mathematical formulation of the activation and loss functions.

Activation functions typically take the form of piece-wise continuous functions centered at the origin and regularized within an [-1,1] interval, meaning that if the sum of all input signals to a given neuron results in a positive value, this neuron will be activated. Some common activation functions are depicted in Figure 11. Loss functions can be classified into two major categories depending upon the type of learning task. In classification

problems that try to predict a categorically labelled (discontinuous) output given a set of inputs, Cross Entropy (CE) loss functions (Eq. 2.23) are commonly used. In regression analyses, the goal is predicting a continuous output from the input data, and Mean Square or Mean Absolute errors (Eqs. 2.21 and 2.22), as well as their root values, are common loss functions alternatives for these problems.

$$MSE = \frac{\Sigma_{i=1}^{n}(\mathbf{Y}^* - \mathbf{Y})^2}{n} \tag{2.21}$$

$$MAE = \frac{\Sigma_{i=1}^{n}|\mathbf{Y}^* - \mathbf{Y}|}{n} \tag{2.22}$$

$$CE = -(\mathbf{Y}_i log(\mathbf{Y^*}_i) + (1 - \mathbf{Y}_i) log(1 - \mathbf{Y^*}_i)) \tag{2.23}$$

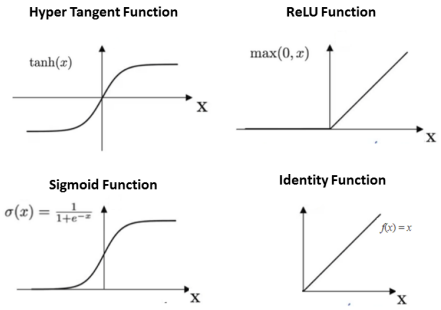


Figure 11 – Common activation functions for Feed-forward Neural Networks. Adapted from [8]

.

Other numerical artifacts can be used to further improve an FNN performance, such as applying regularization techniques. Drop-out is one such technique, in which some pre-defined percentage of all neurons are deactivated, lowering the overall FNN complexity, with avoids over-fitting the training data. Another interesting technique is batch normalization, in which the outputs of each active neuron in the hidden layers are normalized using their mean and variance, resulting in faster and more stable training results. Initializing the neural network with a pre-defined set of weights and biases is another effective approach to improve FNN performance, as it accelerates the convergence of the backpropagation algorithm.

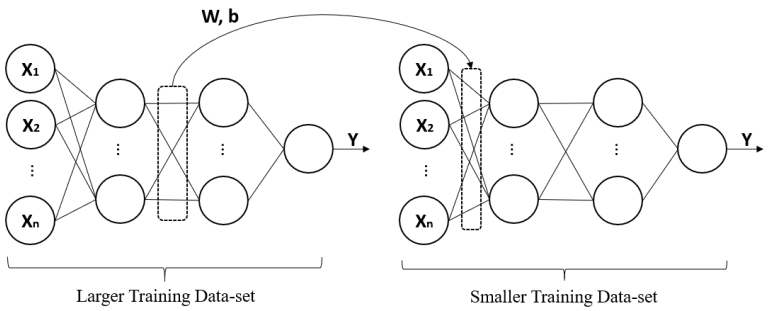


Figure 12 – Example of a simple Transfer Learning method applied to a set of Multi Layer Perceptrons.

Controlling the weights and biases between different Neural Networks is precisely the idea behind Transfer learning (TL), a Machine Learning architecture that focuses on storing knowledge gained while solving one problem and applying it to a different but

related problem. In the context of aerospace engineering, TL techniques with Deep Neural Networks have been successfully applied to generate ROMs for design optimization [26], among other applications. This knowledge transfer can be accomplished in many ways, but one particularly simple approach (illustrated in Figure 12) is storing the outputted weights and biases of a hidden layer from a initial model (for which there is an ample training data-set), and loading these saved parameters back in the initial layer (or layers) of a new model, to be trained with a smaller data set.

3 Method

The principal objective of this work is to propose an analysis framework for the parametric conceptual design of a high power model rocket's aerodynamic and stability configuration. Taking into consideration that many of the computational tools required for such CAE analyses are rather computationally expensive, particular attention has been paid to propose a *light-weight* conceptual aerodynamics design framework, taking restricted computational capability into account. In fact, one of the intended goals is that the proposed framework could be used in regular personal computers or laptops, readily available to most university students interested in experimental rocket design.

Figure 13 outlines the methodology implemented to propose the aforementioned conceptual design framework. Firstly, a set of $\mathbf{X}$ geometric characteristics from the Armação A-22 rocket are defined, out of which a subset $\mathbf{x}$ are chosen as design parameters, which are then discretely sampled within a constrained design space, crating a vector of design points. Each design point is then analyzed by lower and higher fidelity aerodynamic modelling strategies, namely by the semi-empiric Modified Barrowman Method, and by parametric CFD models, respectively. The output (larger data-set) from the lower fidelity method can be directly optimized to estimate the best design point, while the output (smaller data-set) from the higher fidelity method needs to be interpolated by some reduced order modelling technique, such as the enhanced Kriging model, before optimization. Finally, the proposed framework also evaluates combining both the lower and higher fidelity outputs in an machine learning based ROM (by means of Transfer Learning in Multi Layer Perceptrons).

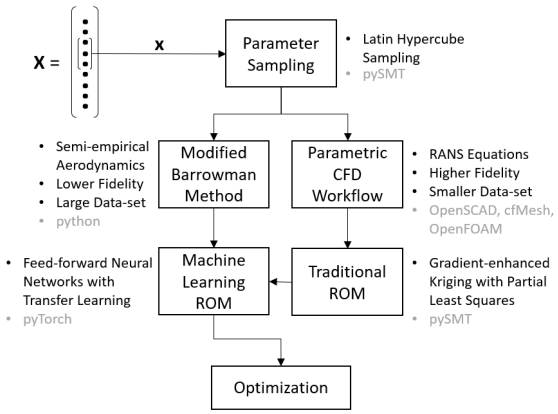


Figure 13 – Schematic organization of the framework for automated conceptual aerodynamic design analysis. Bullet points highlight the implemented methods and utilized tools for each stage of the workflow.

Figure 14 illustrates the set $\mathbf{X}$ of geometric characteristics from the Armação A-22

rocket under consideration in this study. These geometric values act as crucial inputs for the semi-empirical aerodynamics model, as well as for the CAD procedure adopeted in the parametric CFD workflow. To explicitly enunciate, the most relevant geometric characteristics are: 1) l_n, l_b, l_t: nosecone, body tube, and boattail lengths; 2) d, d_b: nosecone (nominal) and base diameters; 3) x_c, x_f, x_{cg}: position of the canards, fins, and center of gravity; 4) c_r, c_t: root and tip chord lengths; 5) h, s, Γ: height, sweep length, and mean-chord angle for a stabilizer. Other important geometric parameters (such as the fin-set's aspect ratio, AR) can be derived from the inputs illustrated in Figure 14.

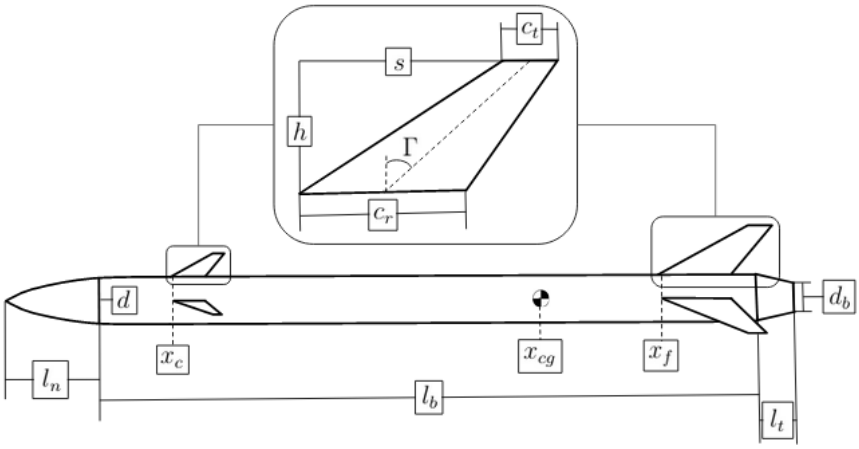
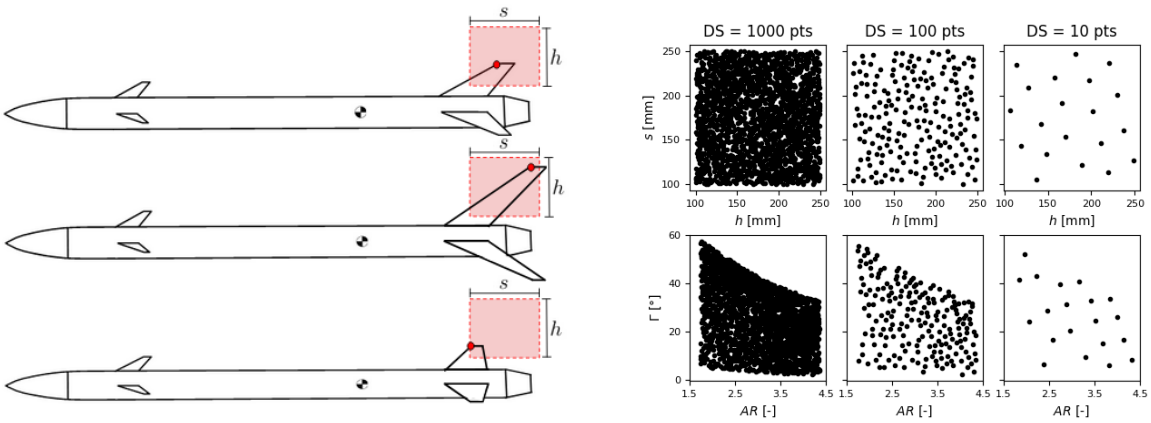


Figure 14 – Geometric parameters of the Armação A-22 rocket taken as possible input design variables.

To reduce the optimization problem's complexity, this work focuses on exclusively analysing the design of the Armação rocket's fin-set, altough the proposed design framework could be expanded to include more design parameters, if enough computational resources are available. Out of the vector $\mathbf{X}$ of geometric variables, two stabilizer parameters were selected as design inputs ($\mathbf{x}$), namely the fin's height (h), and the fin's sweep length (s). All other geometric quantities were fixed, according to the Armação A-22 dimensions presented in [1]. Choosing two design parameters not only lowers the computational expense of the optimization problem, but also allows for the immediate visualisation of the results in a 2D space. This makes the design changes and optimization results immediately clear, while still being sufficient for the construction and testing of the proposed conceptual aerodynamic design framework. Figure 15a highlights some possible fin-set configurations within the chosen $h - s$ design space.

The chosen design parameters were sampled using the Latin Hypercube Sampling (LHS) algorithm available in pySMT, using the standard hyper-parameters provided within the open source library. Both the fin's height h and sweep length s were constrained within the interval [100, 250]mm. The design space sample division number (DS, i.e.: the number of h and s divisions) varied depending on the model fidelity under con-

sideration. In the parametric CFD workflow, for example, sampling numbers between 2 and 12 were investigated (totaling between 4 and 24 total design points); while the semi-empirical Modified Barrowman Method could handle sampling numbers of more than 1000, without incurring on long solution times.



(a) Possible fin-set configuration within the chosen design space.

(b) Representation of the constrained design space for various sampling numbers.

Figure 15 – Selected input design parameters for the fin-set parametric analysis.

3.0.1 Optimization Problem

All of the analysis methods to be presented in this chapter share the same objective: being used as adequate modelling tools in design optimization problems. The main goal of this work is combining all of the methods outlined in Figure 13 into a unified parametric framework for the conceptual aerodynamics and stability analysis of a high power model rocket. As previously mentioned, to expedite iterative testing of the software architecture, as well as to reduce computational expenses, the present work focused on optimizing the Armação A-22 rocket's stabilizers, selecting only two design parameters, the fin's height h and sweep length s.

In effect, such a low dimensional design space can be trivially optimized, as the design coefficients (outputs) can be represented by surface functions, where local minima or maxima values are easily visually identifiable. Computationally speaking, these optimum points can also be found through brute force methods iterating all data points, given that the resulting output space isn't too large. Many of the software tools discussed in this chapter are, however, able to implement much more complex optimization functions; such as typical Newton (gradient-based) methods, or genetic optimization algorithms. Even though the optimization itself is trivialized in the particular setup under investigation in this work, it is still important to properly define its objectives. Equation 3.1 enunciates this constrained optimization problem. In summary, the optimization goal is

minimizing the aerodynamic drag at null angle of attack ($\alpha = 0°$), while achieving fin-set configurations that observe stable results for the static margin and damping ratio.

$$\begin{aligned} \text{minimize} \ : \quad & C_d = f(x)|_{\alpha,M} \\ \text{subject to} : \quad & SM = g_1(x)|_{\alpha,M} \in [1.5, 2] \text{ cal} \\ & \xi = g_2(x)|_{\alpha,M} \in [0.05, 0.2] \\ \text{with} \ : \quad & x(h, s) \in \mathbb{R}^2 \\ & h \in [100, 250] \text{ mm} \\ & s \in [100, 250] \text{ mm} \end{aligned} \tag{3.1}$$

3.1 Modified Barrowman Method

This section formally enunciates the equations of the modified Barrowman method (MBM) developed in this study. A brief introduction of semi empirical aerodynamics and stability methods in given in Section 2.1.3. The model developed here is largely based on the original Barrowman work [13], considering the modifications established by Galejs [14] for high α flights. This formulation of the Barrowman method is amply used in modern high power rocketry commercial simulation softwares, such as OpenRocket (Niskanen [3]) and Cambridge Rocketry Toolbox (Box et al. [15]). The present modified Barrowman method also includes semi-empirical equations from both of the aforementioned software packages, as well as Fleeman [31], LaBudde [32], and Mandell [12]. Throughout this section, the equations adapted from these works have footnotes referring back to the original material.

3.1.1 Lift and Drag Forces

The modified Barrowman method assumes that the rockets total lift and drag coefficients (C_l and C_d, respectively) can be described as a sum of the aerodynamic coefficients in a per-component basis.

To estimate the normal force coefficient (C_n), a pseudo-linear relationship with α is assumed (Equation 3.2). Barrowman's original method enforces this linear relationship by asserting that the total normal force coefficient derivative (C_{n_α}) be constant, but the MBM allows for other non-linear functional relationships between C_{n_α} and α.

$$C_n = C_{n_\alpha}\alpha \tag{3.2}$$

$$C_{n_\alpha} = \sum C_{n_\alpha,i} \tag{3.3}$$

The rockets C_{n_α} can be estimated through a sum of the contributions of each component (Equation 3.3). For the model developed here only the nosecone (N), body tube

(B), transition (T, as in the boattail), and stabilizers (S, as in the fins and canards) were considered as relevant components in the rockets lift production. Table 2 enunciates the analytical expressions for the normal force coefficient derivative of these components. Notice that it is possible to define varied rocket configurations through combinations of these few expressions.

Table 2 – Normal force coefficient derivatives

Nosecone[a]	$C_{n_\alpha,N} = 2\left(\frac{\sin(\alpha)}{\alpha}\right)$	(3.4a)
Body tube[b]	$C_{n_\alpha,B} = K\frac{A_c}{A_p}\left(\frac{\sin^2(\alpha)}{\alpha}\right)$	(3.4b)
Stabilizer[c]	$C_{n_\alpha,S} = \left[\left(\frac{\pi AR}{2\alpha}\right)\lvert\sin(\alpha)\cos(\alpha)\rvert + \frac{2\sin^2(\alpha)}{\alpha}\right]\left(\frac{A_s}{A_c}\right)$	(3.4c)
Transition (boattail)[d]	$C_{n_\alpha,T} = \left[\frac{2}{A_c}\left(\frac{\sin(\alpha)}{\alpha}\right)\right](A_c - A_o)$	(3.4d)

a: Openrocket [3], derived from Eq. 3.19, p.22. **b**: Openrocket [3], derived from Eq. 3.26, p.24. **c**: Fleeman [31], derived from p.31 (unnumbered equation). **d**: Openrocket [3], derived from Eq. 3.19, p.22.

where A_c, A_p, A_s, and A_o are the nominal cross-sectional area (at the nosecone's base), the rocket's fuselage planform area (excluding the stabilizers and including nosecone and boattail), the planform area of the stabilizer, and the cross-sectional area at the boattail's base. AR is the stabilizer's aspect ratio ($AR = \frac{2h^2}{A_s}$), and K is a constant value typically between 1 and 1.5 ($K = 1.1$ was used here, as recommended in [33]).

The drag force coefficient can likewise be estimated through a sum of the contributions of each component in the total C_d (Equation 3.5). The MBM developed here considers only the body tube (B), base (O, as in the boattail), and stabilizers (S, as in the fins and canards) as relevant components in the drag generation, as well as the interference effect (I) between the stabilizers and the body tube. The present model also neglects any potential effect of the angle of attack on the drag forces, therefore only being capable of estimating the C_d for $\alpha = 0$.

$$C_d = \sum C_{d,i} \tag{3.5}$$

Table 3 – Drag force coefficients

Body tube[a]	$C_{d,B} = C_f\|_B \left[1 + \frac{60}{(\frac{L}{d})^3}\right] \left[2.7\frac{l_n}{d} + 4\frac{l_b}{d} + \left(\frac{2l_t}{d}\right)\left(1 - \frac{d_b}{d}\right)\right]$	(3.6a)
Base (boattail)[b]	$C_{d,O} = 0.029\frac{\left(\frac{d_b}{d}\right)^3}{\sqrt{C_{d,B}}}$	(3.6b)
Stabilizer[c]	$C_{d,S} = 2C_f\|_S \left(1 + \frac{2\delta}{s}\right)\left\{\left(\frac{2n}{\pi d^2}\right)\left[h(c_r + c_t) + dc_r\right]\right\}$	(3.6c)
Interference[d]	$C_{d,I} = 2C_f\|_S \left(1 + \frac{2\delta}{s}\right)\left(\frac{2nc_r}{\pi d}\right)$	(3.6d)

a: Box et al. [15], derived from Eq. 41. **b**: Box et al. [15], derived from Eq. 42. **c**: Box et al. [15], derived from Eq. 43. **d**: Box et al. [15], derived from Eq. 44.

where L is the total rocket length, δ is the stabilizer thickness, n is the number of stabilizers, and s is the stabilizer span length ($s = \frac{l_s}{\cos(\Gamma)}$). The value C_f refers to the friction coefficient of the specified component, and is described by Equation 3.7.

$$C_f(Re) = \begin{cases} \frac{1.328}{\sqrt{Re}}, & \text{if} Re \leq Re_c \\ \frac{0.074}{Re^{0.2}} - \frac{B}{Re}, & \text{if} Re \geq Re_c \end{cases} \tag{3.7}$$

where Re is the Reynolds number, $B = Re_c\frac{0.074}{Re^{0.2}} - \frac{1.328}{\sqrt{Re}}$, and Re_c is the critical (transitional) Reynolds number (in the MBM a value of $Re_c = 5 \cdot 10^5$ was used, as suggested by [12]). The Reynolds number is a common non-dimensional quantity in aerodynamics defined as:

$$Re(x) = \frac{\rho U_\infty x}{\mu} \tag{3.8}$$

where ρ, μ, and U_∞ are the air's density, dynamic viscosity, and freestream velocity, respectively. x is a characteristic length that describes the object. For Equation 3.6a Re is evaluated with $x = L$, and for Equations 3.6c and 3.6d Re is evaluated with $x = s$.

Once normal force (C_n) and drag force (C_d) coefficients are defined for the entire rocket using Equations 3.2 and 3.5 evaluated with each component contribution (Tables 2 and 3), the corresponding lift force (C_l) and axial force (C_a) coefficients can be obtained through simple coordinate frame transformations (as depicted in Figure 4b) given by Equations 3.9 and 3.10.

$$C_l = C_n\cos^{-1}(\alpha) - C_d\tan(\alpha) \tag{3.9}$$

$$C_a = C_d\cos(\alpha) - C_l\sin(\alpha) \tag{3.10}$$

The estimated coefficients can also be corrected to account for variations in the Mach number (M_a) through the Prandtl-Glauert correction given by Equation 2.7, where λ denotes a general aerodynamic coefficient, and C'_λ is the velocity-corrected coefficient.

$$C'_\lambda = \frac{C_\lambda}{\sqrt{1 - M_a^2}} \tag{2.7, revisited}$$

3.1.2 Static and Dynamic Stability

The original Barrowman method estimates the rocket's static margin (SM) by determining the position of its center of gravity (x_{cg}) and center of pressure (x_{cp}). The center of gravity can be easily obtained experimentally, or estimated for simple and complex geometries given enough information about the rocket's materials. Therefore, the modified Barrowman method presented here considers that x_{cg} is a given fixed parameter, and preoccupies itself only in determining the location of the center of pressure.

The MBM obtains the resultant location of the center of pressure (x_{cp}) by combining the effect of each of the rocket's components, considering that only the normal force component is relevant in determining the x_{cp}, as shown in Equation 3.11:

$$x_{cp} = \frac{\sum C_{n_\alpha,i} x_{cp,i}}{C_{n_\alpha}} \tag{3.11}$$

The semi-empirical aerodynamics method presented here only considers some components as relevant in producing lifting (and normal) forces. To satisfy Equation 3.11, the MBM estimates the x_{cp} for the same components as those in Table 2 (namely: the nosecone, body tube, stabilizers, and boattail), as shown in Table 4.

Table 4 – Center of pressure locations

Nosecone[a]	$x_{cp,N} = \frac{1}{A_c}\int_0^{l_n} \pi r^2(x)dx$	(3.12a)
Body tube[b]	$x_{cp,B} = x_B + \frac{l_b}{2}$	(3.12b)
Stabilizer[c]	$x_{cp,S} = x_S + \frac{l_s}{3}\left(\frac{c_r + 2c_t}{c_r + c_t}\right) + \frac{1}{6}\left[\frac{(c_r^2 + c_t^2 + c_r c_t)}{c_r + c_t}\right]$	(3.12c)
Transition (boattail)[d]	$x_{cp,T} = x_T + \frac{V_T}{A_c}$	(3.12d)

a: Approximation cited in LaBudde [32]. **b**: x_{cp} is approximated as the center of area [32]. **c**: Openrocket [3], adapted from Eq. 3.34, p.27. **d**: Approximation cited in LaBudde [32].

where x_B, x_S, and x_T represent the positions of the body tube, stabilizers, and boattail (measured from the nosecone tip). Equations 3.12a and 3.12d approximate the x_{cp} as the ratio of the component's volume and reference area. In Equation 3.12d V_T is the volume

of revolution of the boattail, whereas the latter term of Equation 3.12a is the expression for the volume of revolution of a nosecone whose profile is described by a general function $r(x)$ defined in the range $x = 0$ to $x = l_n$ (nosecone's length). The Armação rocket's nosecone has a Von Kármán profile, which can be described by:

$$\theta(x) = \cos^{-1}\left(1 - \frac{2x}{l_n}\right) \tag{3.13a}$$

$$r(\theta) = \frac{d}{2\sqrt{\pi}}\sqrt{\theta - \frac{\sin(2\theta)}{2}} \tag{3.13b}$$

Once Equation 3.11 is satisfied by evaluating all of the normal force coefficient derivatives and center of pressure locations are evaluated following Tables 2 and 4; the rocket's static margin (SM) can be computed by revisiting the first half of Equation 2.3:

$$\text{SM} = \frac{(x_{CG} - x_{CP})}{d} \tag{2.3, 1st half, revisited}$$

To estimate the rocket's dynamic stability parameters the modified Barrowman method makes use of the previously calculated C_{n_α} and x_{cp} coefficients. Revisiting some of the Equations from Chapter 2.1, it can be observed that at this point the MBM already has determined most of the parameters necessary to estimate the rocket's natural frequency (ω_n) and pitch damping ratio (ξ):

$$\xi = \frac{C_2}{2\sqrt{I_L C_1}} = \frac{C_2}{2\omega_n I_L} \tag{2.4, revisited}$$

$$C_1 = \frac{\rho A_c U_\infty^2}{2} C_{n_\alpha}(x_{cp} - x_{cg}) \tag{2.5, revisited}$$

$$C_2 = \frac{\rho A_c U_\infty}{2} \sum C_{n_\alpha,i}(x_{cp,i} - x_{cg})^2 + \dot{m}(x_{nozzle} - x_{cg})^2 \tag{2.6, revisited}$$

In a similar fashion to what was done with the center of gravity (x_{cg}), the MBM considers that the rocket's longitudinal moment of inertia (I_L) is a given fixed parameter. This quantity can also be estimated through simple methods given enough information about the rocket's geometry and materials.

The latter term of Equation 2.6 (i.e.: the jet damping) is also considered as a given fixed quantity. Both the jet damping and the longitudinal moment of inertia are quantities that vary in time as the rocket's fuel is consumed, altering the massflow rate of the exhaust gases ($\dot{m}$) and the distribution of mass along the rocket's longitudinal axis. In the MBM, however, an average value of these parameters was deemed sufficient for the estimation of ξ and ω_n.

3.2 Parametric CFD Workflow

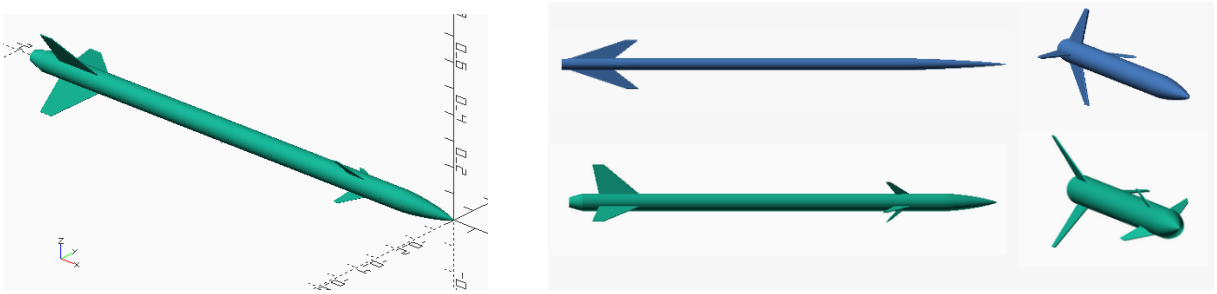
Computational fluid dynamics models for the aerodynamic optimization of the Armação rocket's main stabilizers were considered in this work as the high fidelity approach, especially in comparison with the semi-empirical Modified Barrowman Method. These CFD models were developed with OpenFOAM and other open source tools available for the Linux / UNIX operating system, allowing for the automatization of the design workflow through python and bash scripting.

3.2.1 Pre-processing

3.2.1.1 Parametric CAD

OpenSCAD [34] is an open source CAD scripting language. Unlike most commercial engineering technical drawing softwares, OpenSCAD operates entirely through a text-based user interface. The main advantage of this UI approach is allowing the end user to have more control over the geometry creation, especially in regards to batch running and process automation.

Often times it can be difficult to design complex geometries with OpenSCAD's text-based interface. This, however, is not a major problem for the Armação rocket and other simple rocket configurations, which can be largely described with simple mathematical surfaces of revolution or planar extrusions.



(a) OpenSCAD Rendering of the Armação rocket.

(b) Arbitrary rocket configurations parametrized with OpenSCAD.

Figure 16 – Illustrations of the parametric CAD models. The lower left image on Figure 16b show the Armação rocket with it's canards pitched downwards.

The script developed in OpenSCAD automatically constructs an Armação-like rocket consisting of four modules, namely: nosecone, body tube, body transition, and stabilizers. The first three modules create the rocket's parts through surfaces of revolutions, using simple lines to create the fuselage (body tube) and boattail (conical body transition), and using the Von Kàrmàn profile (Equation 3.13) for the nosecone. Both the fins and canards (stabilizers) are modeled through a point list defining a planar surface, and are

later positioned along the fuselage, linearly extruded, and symmetrically rotated along the body tube.

The previously described script is run for each module, generating stl (standard tessellation language) files for each of the rocket's parts. The stl files are essentially a collection of coordinates points describing a triangular (or tessellated) surface mesh of the rocket geometry. The individual stl files for each rocket part are then combined into a single stl file for the entire rocket using linux's bash scripting language, simply by reading each module stl file and automatically copying its coordinate list into an new file. This procedures preserves the identification of each of the rocket's modules, allowing OpenFOAM to later extract the aerodynamic and stability coefficients for each component individually, as well as for the entire rocket.

Though the focus of this work is only on parametrizing the rocket's main fins, the scripted CAD program developed allows for the complete parametrization of the entire geometry of the Armação rocket. Figure 16a displays the original (reference) geometry for the Armação rocket (following the coordinate system of Figure 4a), while the upper images of Figure 16b show a completely different configuration effortlessly generated with the same code. This parametric CAD procedure also allows for individual control of the rocket's stabilizers rotation, which can be especially useful in evaluating control laws for actively stabilized rocket configurations employing movable canards.

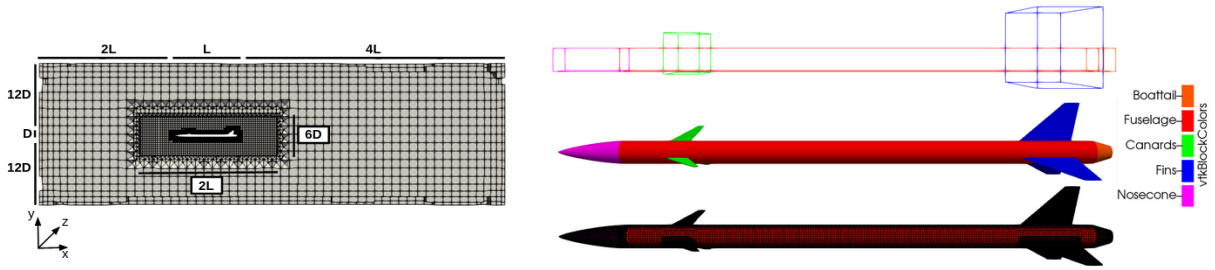
3.2.1.2 Automatic Meshing

OpenFOAM supports many different meshing algorithms, both commercial and open source. The foam-extend distribution maintained and developed by OpenCFD and used in this work has three main options for meshing: blockMesh, snappyHexMesh, and cfMesh. BlockMesh and snappyHexMesh are both native to OpenFOAM's source code, while cfMesh is a third party freely available meshing toolbox integrated with OpenFOAM and developed by Creative Fields [35].

All meshing options are capable of handling hexahedral (or cartesian) meshes. BlockMesh is designed for very simple structured grids, while snappyHexMesh and cfMesh can handle unstructured, geometry-confirming, grids. Moreover, cfMesh is also capable of handling tetrahedral and polyhedral meshes. In [30], the snappyHexMesh and cfMesh algorithms were extensively compared, arriving at the conclusion that cfMesh was capable of generating meshes much more quickly (on the order of 10 times faster) than snappyHexMesh, though the latter was capable of handling more complex geometries. Due to the faster turnout time reported in [30], the cfMesh algorithm was used to generate the parametric rocket meshes in this work.

Unlike OpenFOAM's standard snappyHexMesh algorithm, cfMesh is capable of generating boundary layer cells fully covering the geometry. During preliminary testing, this boundary layer generation appeared to be much faster and consistent for the cartesian

mesh option than for the tetrahedral and polyhedral options, so the former setup was selected.



(a) Rectangular flow domain with cartesian mesh.

(b) Standard surface mesh generated for the Armação rocket.

Figure 17 – Fluid domain and surface mesh crated with cfMesh. L and D in Figure 17a denote the rocket's total length and nominal diameter. Figure 17b highlights the variable mesh refinement for each of the rocket's parts.

A rectangular flow domain containing the rocket surrounded by a mesh refinement box was modeled following the best practice guidelines mentioned in [36]. Figure 17a illustrates a slice of the meshed rectangular domain, with the missing z-dimension having the same discretization as the y-dimension. The parametric CAD procedure described in Section 3.2.1.1 maintained the separation of the rocket into its modules, allowing for separate surface mesh refinements for each of the rocket's parts. Figure 17b illustrates this variable mesh refinement.

Another important setup in cfMesh'es algorithm is the capability of extracting features from the geometry. This is done through the definition of feature edges, as shown in the upper right image of Figure 18. This procedure is especially useful for regions with sharp angles on the geometry, such as the nosecone's tip, the stabilizer's leading and tailing edges, and the region connecting the stabilizers to the fuselage body.

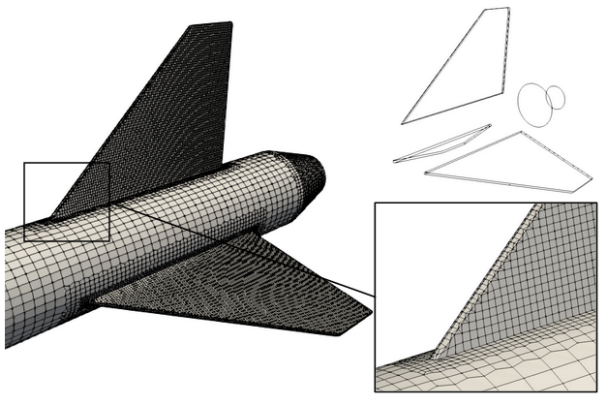


Figure 18 – Surface mesh refinement and feature edges obtained with cfMesh.

CfMesh allows for great control over the boundary layer discretization. As mentioned in Section 2.2, two meshing approaches are to be considered for the boundary layer: the

high Reynolds (high $y+$) and the low Reynolds (low $y+$) methods. These methods differ specifically in the height of the first mesh cell directly in contact with the geometry's surface. For the high R_e case large elements are used alongside empirical wall-functions modeling the boundary layer flow, while in the low R_e case small elements are used to fully resolve the boundary layer.

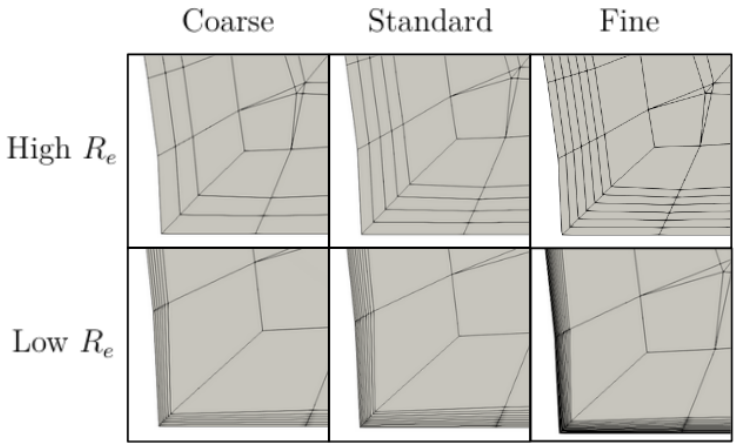


Figure 19 – Boundary layer discretization, focusing on the contact region between the fins and fuselage.

Figure 19 illustrates the different boundary layer meshing approaches. For each method three boundary layer refinements were evaluated (coarse, standard, fine). For the high R_e approach, the meshed boundary layer extends up to 50% of the standard cell height (roughly equal to half of the rocket's diameter), namely being about 26mm. The elements in the high R_e boundary layer all have the same fixed height, and the refinement only increases the total number of elements in the boundary layer. The low R_e boundary layer extends to roughly 20% of the standard cell height (about 8mm in total), however, in this case, the elements have a much smaller first cell height and exponentially grow outwards. Table 5 presents an overview of the boundary layer and local surface refinement previously discussed.

Table 5 – Overview of the cfMesh meshDict file setup for the Armação rocket. The numbers in parenthesis represent the refinement (ref.) levels. All dimensional values are reported in meters.

	High R_e			Low R_e		
	Coarse	Standard	Fine	Coarse	Standard	Fine
Max. cell size	2.56E-1 (1)			2.28E-1 (1)		
Box ref.	1.28E-1 (2)			1.14E-1 (2)		
Nosecone ref.	4.26E-2 (6)			3.80E-2 (6)		
Body ref.	5.16E-2 (5)			4.56E-2 (5)		
Boattail ref.	4.26E-2 (6)			3.80E-2 (6)		
Stabilizers ref.	3.66E-2 (7)			3.26E-2 (7)		
Nosecone edges	3.66E-2 (7)			3.26E-2 (7)		
Stabilizers edges	3.20E-2 (8)			2.85E-2 (8)		
Layers	2	4	6	6	10	14
Growth ratio	1	1	1	1.1	1.1	1.1
1*st cell height*	1.2E-3	8.0E-4	4.0E-4	3.0E-5	2.0E-5	1.0E-5

Like OpenSCAD, cfMesh can be easily ran in an automated fashion. The main mesh file (meshDict.c) is incorporated into the OpenFOAM case's system folder, containing the previously described setup for the surface, edge, and boundary layer refinements. Additional files containing refinement surfaces and edges are generated by running cfMesh commands through bash scripting, and later fed into the main meshDict file along side the stl files generated by OpenSCAD. For parallel runs with multiple cores, OpenFOAM easily incorporates mesh partitioning algorithms to decompose the meshed domain for individual core solution (via the decomposeParDict.c file in the system folder).

3.2.2 CFD Models Setup

3.2.2.1 rhoSimpleFoam

RhoSimpleFoam [20] is a steady state compressible RANS solver for turbulent flow available in OpenFOAM. The flow conditions incorporated in this solver are well suited to model the transonic flow of the Armação rocket flying at Mach 0.7, though other solver alternatives such as the transient rhoPimpleFoam [20] or HiSA (High Speed Aerodynamic) solver developed by [37] have been shown to be more accurate and stable than the more simple formulation of rhoSimpleFoam.

The SIMPLE (Semi-Implicit Method for Pressure Linked Equations) is the standard solver algorithm implemented in rhoSimpleFoam. Being a pressure-based compressible solver, rhoSimpleFoam solves an equation for the pressure term and derives the flow's variable density via thermodynamic equations of state. RhoSimpleFoam is a segregated solver, meaning that the equations characterizing the flow (such as Equations 2.12, 2.13, 2.14, 2.18 and 2.19 for the RANS formulation with the $k-\omega$ SST turbulence model) are

solved sequentially, i.e.: the solution of the preceding equations is inserted in the subsequent equation. In simplistic terms, the rhoSimpleFoam solver with the SIMPLE solution method employs the solution approach enunciated in Algorithm 1.

Algorithm 1: rhoSimpleFoam solver with SIMPLE method (pseudo-code).

while not converged **do**
  Initialize → velocity (**u**), pressure (**P**), and density (ρ) fields;
    → compute estimated cell-face fluxes ϕ with **u** and ρ;
  Momentum eqs. → estimate non-mass-conservative $\mathbf{u}^*$ and ρ^*;
  Energy eq. → compute specific energy (**e**) or enthalpy (**h**);
    → update the compressibility factor $\mathbf{\Psi}$ with **e** (or **h**);
    → update the temperature field **T** with **e** (or **h**);
  Momentum + Continuity eqs. → compute the mass-conservative pressure **P**;
  Update → compute mass-conservative **u** with **P**;
    → compute mass-conservative ρ with **P** and $\mathbf{\Psi}$;
    → correct ϕ with **u** and ρ;
  Turbulence eqs. → compute turbulent kinetic energy (**k**) and dissipation rate (ω);
  Check → **P**, **u**, ρ, **e** (or **h**), **k** and ω for convergence.
end

The implementation of a CFD case with rhoSimpleFoam follows the standard OpenFOAM case structure presented in Section 2.2, with the *0* folder containing the initialization setup, the *constant* folder containing geometry, mesh, turbulence, and thermophysical properties information, and the *system* folder containing the proper CFD model setup (including pre and post processing options). The following sections discuss the contents of the CFD case folders with regards to the modeled boundary conditions and the solver numerical options.

3.2.2.2 Boundary Layer & Boundary Conditions

A freestream pressure far-field boundary condition was employed to close the mathematical formulation of the RANS + $k-\omega$ SST model. This type of boundary condition uniquely derives a pressure condition at the freestream surfaces located far away from the geometry body, using the freestream Mach number and static atmospheric conditions. The far-field regions where this boundary condition is applied can be clearly observed in Figure 17a. This condition can only be effectively computed for compressible single-phase flows, being therefore adequate, stable and fast for external aerodynamics applications according to [36].

In OpenFOAM, the boundary conditions are specified in the *0* folder by assigning precise setups for each modeled flow variable to be computed at the flow domain's ex-

tremities. In the external aerodynamics application presented in this work the mesh has two extremities: the previously discussed free-stream far-field (where the pressure far-field is enforced), and the rocket's surface (were the boundary layer is developed). This indicates that the boundary layer can be correctly interpreted (and treated) as any other boundary condition).

The CFD models developed here include 7 flow variables that must be specified at the domain's boundaries: the velocity vector $\mathbf{u}$, the pressure scalar $\mathbf{P}$, the temperature derived from the energy equation $\mathbf{T}$, the turbulent model variables $\mathbf{k}$ and ω, and the turbulent kinematic viscosity (ν_t) and thermal diffusivity (α_t) derived from the turbulent variables (as show in Equations 2.15, 2.16). For the free-stream boundary condition most of the flow variables are uniform and constant, with only ν_t and α_t being iteratively calculated. It is important to note that the velocity vector $\mathbf{u}$ only has flow components in the x and y directions (u and v, following the standard coordinate system of Figure 4a), implying that the rocket is only subject to an angle of attack (α) but has a null angle of side-slip (β) due to the null z-velocity component. Table 6 partially illustrates the *0* folder setup for the adopted free-stream pressure far-field boundary condition, and Table 7 illustrates the setup for the wall boundaries.

Table 6 – *0 Folder:* Free-stream.

	High R_e	Low R_e
$\mathbf{u}$	Uniform: $\mathbf{u}_\infty = [u, v, 0]$ m/s	
$\mathbf{P}$	Uniform: $\mathbf{P}_\infty = 1E5$ pa	
$\mathbf{T}$	Uniform: $\mathbf{T}_\infty = 298$ K	
$\mathbf{k}$	Uniform: $\mathbf{k}_\infty = 0.01$	
ω	Uniform: $\omega_\infty = 10$	
$\nu_\mathbf{t}$	Calculated: $\nu_{t_0} = 0$	
$\alpha_\mathbf{t}$	Calculated: $\alpha_{t_0} = 0$	

Table 7 – *0 Folder:* Wall Boundary.

	High R_e	Low R_e
$\mathbf{u}$	no slip	
$\mathbf{P}$	zero gradient	
$\mathbf{T}$	zero gradient	
$\mathbf{k}$	kqRWF: 0.01	kLowReWF: 0
ω	omegaWF: 10	
$\nu_\mathbf{t}$	nutkWF: 0	nutLowReWF: 0
$\alpha_\mathbf{t}$	compressible alphatWF: 0	

For the wall boundary conditions (describing the boundary layer) there are, as already mentioned in Section 2.2 and highlighted in Figure 19, the high Reynolds approach fully using wall functions, or the low Reynolds approach fully resolving the boundary layer flow. OpenFOAM incorporates both approaches in its source code, making it so that only minimal changes are required to switch the formulation (considering, of course, that the mesh is also adequate for the formulation). Table 7 displays the setup for the wall boundary conditions. Wall functions (WF) are only used to model $\mathbf{k}$, $\nu_\mathbf{t}$, ω, and $\alpha_\mathbf{t}$, being that the latter two variables have unique wall functions valid for both approaches. In [5], the authors extensively detail the exact implementation of each wall function, but for this work it is sufficient to note that only $\mathbf{k}$ and $\nu_\mathbf{t}$ have to be altered for the low R_e approach.

3.2.2.3 Solver Numerics

Two numerical setups for the Armação rocket CFD model were investigated in this work, mainly differing in the numerical approach to modeling the pressure (**P**) and energy (**e** or **h**) terms in the RANS equations. The first approach utilizes the GAMG (Geometric Agglomerated Algebraic MultiGrid) solver for **P** and models the energy equation in terms of the specific energy **e**. The second approach uses the PBiCGStab (Preconditioned Bi-Conjugate Gradient Stabilized) solver for **P**, and models the energy with an equation for the specific enthalpy **h**. Throughout this work these numerical approaches will be referred to as GAMG and PBiCGStab, respectively.

The errors (or residuals) accumulated during the numerical solution of the RANS matrix system are closely dependent on the mesh and flow anisotropy (spatial variation). The basic idea behind multi-grid solvers (such as GAMG) is to use a coarse grid with fast solution times to smooth out high frequency errors and generate starting solutions for the finer, original mesh [20]. This procedure is incredibly important for convergence, but can be slow due to the repeated mesh coarsening and refining steps taken for each solver iteration. Preconditioned gradient type solvers (such as the PBiCGStab) do not employ this successive multi-grid resolution approach, and are therefore less stable but faster. To enforce stability, these solvers have to be paired with matrix pre-condtioning (pc) and smoothing algorithms, with can lead to similar solve times as multi grid solvers [20]. OpenFOAM allows for the full control of the numerical solver setup for each modeled flow variable (through the fvSolution.c file in the system folder). Table 8 highlights this setup for both numerical approaches previously mentioned.

For the GAMG setup the pressure term is solved through the GAMG algorithm with Gauss Seidel (GS) matrix smoothing. The energy equation is modeled by the specific total energy (**e**) including the pressure work ($\frac{\partial \mathbf{P}}{\partial t}$) unsteady term. The majority of the flow variables (momentum, turbulence, and energy) are modeled with the PBiCGStab solver with DiLU (Diagonal-based Incomplete LU) matrix pre-conditioner (pc). The flow's density, however, is solved with simple matrix back substitution (diagonal solver), given that it can be obtained from the thermodynamic algebraic state equations. Table 9 displays the relaxation factors adopted for this setup. The described configuration is identical to OpenFOAM's compressible external aerodynamics tutorial (in version v2012). This numerical setup is stable and quickly resolves each iteration, but the higher overall relaxation factors can lead to slower convergence.

The PBiCGStab setup follows the reference GAMG setup very closely. The major change is in modeling all flow variables with the PBiCGStab solver, though the GAMG solver is employed as a pre-conditioner for **P**, employing symmetric Gauss Seidel (symGS) smoothing. Combining both methods can be advantageous for solver accuracy, but is necessary slower per solver iteration given the increased steps of calculation (even though

the GAMG calculation steps are not implemented as extensively in this approach). The second important difference of this numerical setup is that it uses the specific enthalpy for the energy equation and does not include the pressure work term ($\frac{\partial \mathbf{P}}{\partial t}$). This simplification naturally decreases the modeling accuracy of the energy equation, but neglecting the pressure work term has stabilizing effect (especially for steady state solvers) and can be less detrimental when using the specific enthalpy as the computed energy variable [20]. This numerical setup allowed for larger relaxation factors and faster convergence, except for **P** (due to its limited solver accuracy with fewer GAMG iterations).

Table 8 – *system* folder: fvSolution.

	GAMG	PBiCGStab
P solver	GAMG	PBiCGStab
P pc.	-	GAMG
P smoother	GS	symGS
Energy	**e**	**h**
$\frac{\partial \mathbf{P}}{\partial t}$	yes	no
Energy solver	PbiCGStab	
Energy pc.	DILU	
u, **k**, ω solver	PbiCGStab	
u, **k**, ω pc.	DILU	
ρ solver	diagonal	

Table 9 – Relaxation factors.

	GAMG	PBiCGStab
P	0.7	0.3
ρ	0.01	0.01
e	0.5	-
h	-	0.7
u	0.3	0.5
k	0.5	0.7
ω	0.5	0.7

To complete the numerical setup it also necessary to define the numerical schemes for spatial and temporal interpolation and discretization adopted by the solver. This is done by modifying the fvSchemes file in the *system* folder. Table 10 highlights the numerical schemes used for both solution methods (GAMG and PBiCGStab). These schemes follow directly from the standard OpenFOAM v2012 tutorials for external compressible aerodynamics. Table 10 uses nabla-notation for the spatial gradient (∇X), divergent ($\nabla \cdot X$), and laplacian ($\nabla^2 X$). Terms followed by X are valid for all flow variables.

Table 10 – Numerical schemes (partial contents of *system* folder: fvSchemes.c).

$\frac{\partial X}{\partial t}$	steady state
∇X	cell limited Gauss linear
$\nabla \cdot \mathbf{u}$	bounded Gauss linear Upwind limited
$\nabla \cdot \mathbf{P}$	Gauss Upwind
$\nabla \cdot \mathbf{e}$ ($\nabla \cdot \mathbf{h}$)	bounded Gauss linear Upwind limited
$\nabla \cdot \mathbf{k}$, $\nabla \cdot \omega$	bounded Gauss Upwind
$\nabla^2 X$	Gauss linear corrected
cell-to-cell Interpolation	linear

Finally, it is worth quickly describing the CFD models' solution initialization and control. For complex flow scenarios it is common to initialize the flow field with the

solution of a less complex CFD model (often times potentialFoam is used for this purpouse in OpenFOAM). This, however, can create unnecessary difficulties in the automatization of the CFD case. For the models developed here an alternative initialization and control approach was used.

Hard upper and lower limits were set for the calculated temperature (**T**), pressure (**P**), and velocity (**u**) on all the meshes elements. If the solver arrives at numerical results higher (or lower) than this limits, the solution is kept at the limited value. This creates an stabilizing (damping) effect on the flow field, but can lead to incorrect results if a large portion of the cells in the mesh is limited. This setup is described in the fvOptions file on the *system* folder. **P** was limited between $[0.2P_\infty, 2P_\infty]$ (P_∞ is the atmospheric pressure), **T** was contained within $[100, 1000]$K, and **u** within $[0, 1.4u_{in}]$ (u_{in} is the inlet velocity).

3.2.3 Post-processing

OpenFOAM allows for the direct calculation of aerodynamic coefficients during run-time for any defined geometry. The parametric CAD construction described in Section 3.2.1.1 facilitates the calculation of the aerodynamic coefficients for each of the rocket's modules, as it preserves the separate identification of each part (as illustrated in Figure 17b). The lift (C_l), drag (C_d), and pitch moment (C_m) coefficients (among others) can be setup up with additional files in the *system* folder and automatically saved in the *constant* folder (in the post-processing sub-folder). Similarly, forces, moments, and flow dependent mesh information (like the $y+$) can also be calculated and saved during run-time.

A python script was developed and integrated with the main parametric CFD code to read these post-processing files created by standard OpenFOAM applications and then further evaluated the obtained coefficients automatically. C_l, C_d, and C_m information for each of the rocket's modules is read the case folder and used to derive the stability coefficients (namely the static margin SM, damping ratio ξ, and natural frequency ω_n). These aerodynamic and stability coefficients (along with the lift derivative coefficient C_{l_α}) are collectively referred to as design coefficients throughout this work.

Following the conventions presented in Section 2.1, the rocket's cross-sectional area and its nominal diameter (both at the nosecone's base) were used as the reference length and area for the coefficients. Constant values for the air's pressure (101325 pa) and density (1.225 kg/m^3) were adopted. Lift and drag forces were decomposed only the xy plane, enforcing the no side-slip angle condition (null z-velocity). The y axis was considered for the pitching moment (following Figure 4c), and the rocket's center of gravity, longitudinal inertia, and exhaust gases' mass-flow rate were considered as constants for the derivation of the stability coefficients (as was done for the low fidelity modified Barrowman method). Revisiting the static stability margin equations of Section 2.1 enunciates how the stability coefficients can be derived from the aerodynamic coefficients. The dynamic stability

coefficients can similarly be derived from the aerodynamic coefficients of each of the rocket's parts (following Equations 2.4, 2.5, 2.6).

$$SM = \frac{x_{cp} - x_{cg}}{d} = \frac{\partial \frac{\partial C_m}{\partial \alpha}}{\partial \frac{\partial C_l}{\partial \alpha}} = \frac{\partial C_m}{\partial C_l} \qquad \text{(2.3, revisited)}$$

Observing the derivative term in equation 2.3 one limitation of the CFD models becomes apparent, both C_l and C_m have to be defined for an extended range of angle of attacks α to compute their derivatives. Unlike the semi empirical modified Barrowman method, the CFD models are only evaluated at a single α, as well as being much more computationally costly. To overcome this constraint, the parametric CFD setup developed here simulates the rocket at two distinct values of α, mirrors the obtained results, and approximates a curve with these points.

All CFD cases in this work were evaluated at $\alpha = 2°$ and $\alpha = 10°$ at $M_a = 0.7$. To arrive at the aforementioned fitted curves and calculate the aerodynamic coefficients derivatives, C_l was estimated as an linear function, C_d was considered as an quadratic function, and C_m was estimated as an odd cubic function. All coefficients were then fitted with the available 4 data points (two simulated and two mirrored results). Figure 20 illustrates the described fitting procedure.

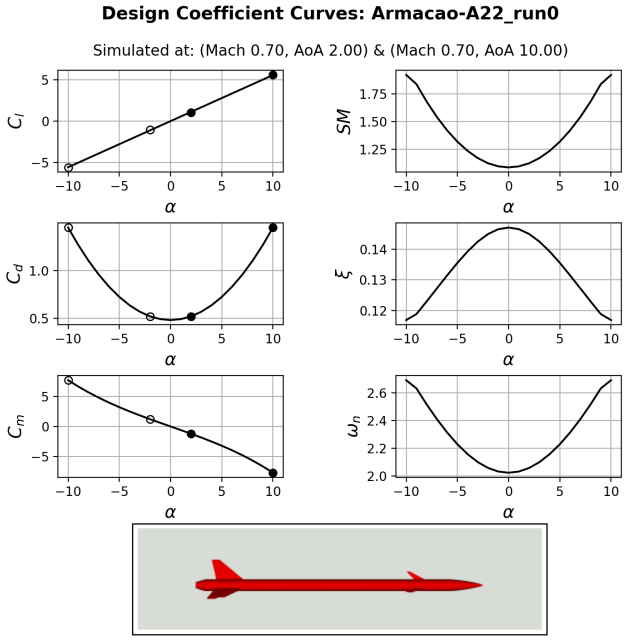


Figure 20 – Automatically generated design report graphics after a complete run (two simulations) for the Armação rocket. The filled points represent the solved results, and the empty points indicate the mirrored results.

The post processing code developed in python automatically generates and saves Figure 20 after both CFD runs (for $\alpha = 2°$ and $\alpha = 10°$) have converged. It also extracts

convergence information for each individual CFD run and reports it in Figure 21. Further post processing operations such as the flow field's visualization were performed in Paraview and were not automated in the parametric CFD procedure.

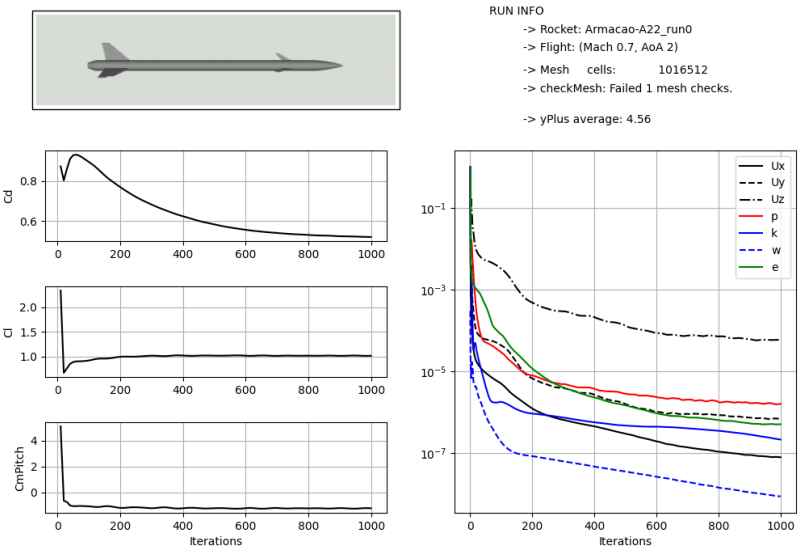
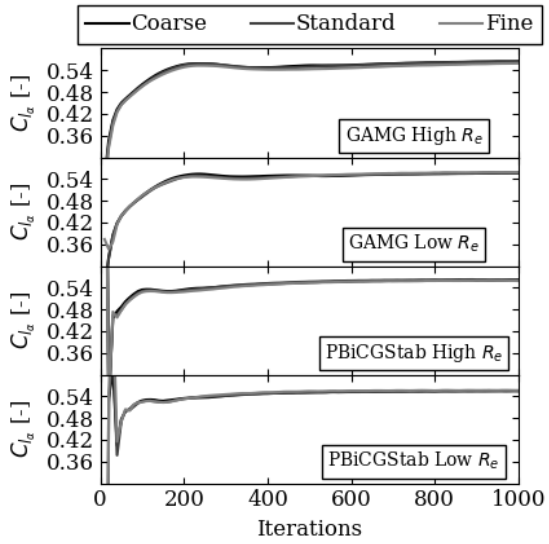


Figure 21 – Solution report created by the post processing function, displaying relevant convergence and mesh information for one individual CFD run.

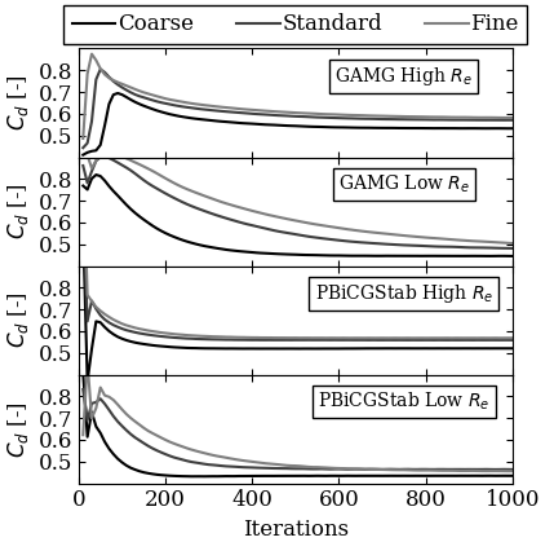
3.2.4 Preliminary Results & CFD Model Selection

In total, 12 CFD models were evaluated as high fidelity models for the transonic aerodynamics of the Armação rocket. These models arise from combinations of the two numerical setups (GAMG and PBiCGStab), two boundary layer modeling approaches (high R_e and low R_e), and three mesh refinement levels (coarse, standard, and fine). Each model was ran twice, following the fitting procedure described in Section 3.2.3 for the determining the design coefficients (C_{l_α}, C_d, SM, and ξ).

The aforementioned design coefficients were taken as the sole convergence criteria, and typical residual convergence was ignored for the purposes of this work. This was done because during preliminary testing the numerical residuals of the RANS equations often times appeared to converge faster than the design coefficients. A conservative condition of less than 1% of relative variation in 100 iterations was adopted as the convergence criteria for all design coefficients. All simulations were performed in parallel with 6 cores on Ryzen 5 3600 CPU, and ran for a total of 1000 iterations (per simulation). Figures 22a, 22b, 23a and 23b display the results for the design coefficients evaluated at null angle of attack ($\alpha = 0°$), meaning that they represent an interpolated result obtained from the simulated $\alpha = 2°$ and $\alpha = 10°$ points.

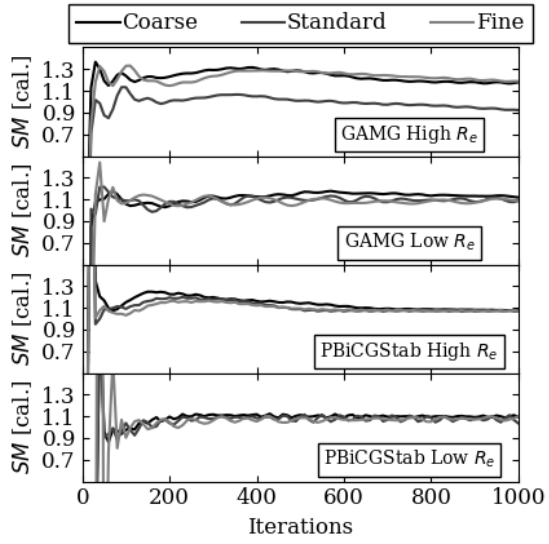


(a) Lift coefficient derivative convergence.

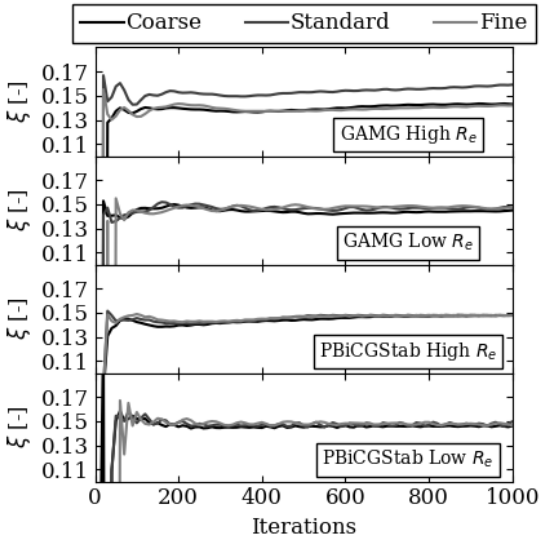


(b) Drag coefficient convergence.

Figure 22 – Aerodynamic design coefficients convergence for the proposed CFD models.



(a) Static stability margin convergence.



(b) Damping ratio convergence.

Figure 23 – Stability design coefficients convergence for the proposed CFD models.

From the preliminary convergence results it is apparent that the stability coefficients present an oscillatory behavior whereas the aerodynamic coefficients do not. This can be attributed to the derivative term in the static margin ($\frac{\partial C_m}{\partial C_l}$ in Equation 2.3), which greatly depends on the fitted $C_l - \alpha$ and $C_m - \alpha$ curves. This oscillation, however, quickly dampens down, and the C_d is the actual coefficient with the slowest convergence. The PBiCGStab numerical setup appears to converge faster than the GAMG alternative, especially for the C_d. However, as mentioned in Section 3.2.2.3, the PBiCGStab takes longer to complete the calculation of each iteration. To properly compare all 12 CFD models, Figure 24 plots

all the converged design coefficient results alongside the elapsed computation time.

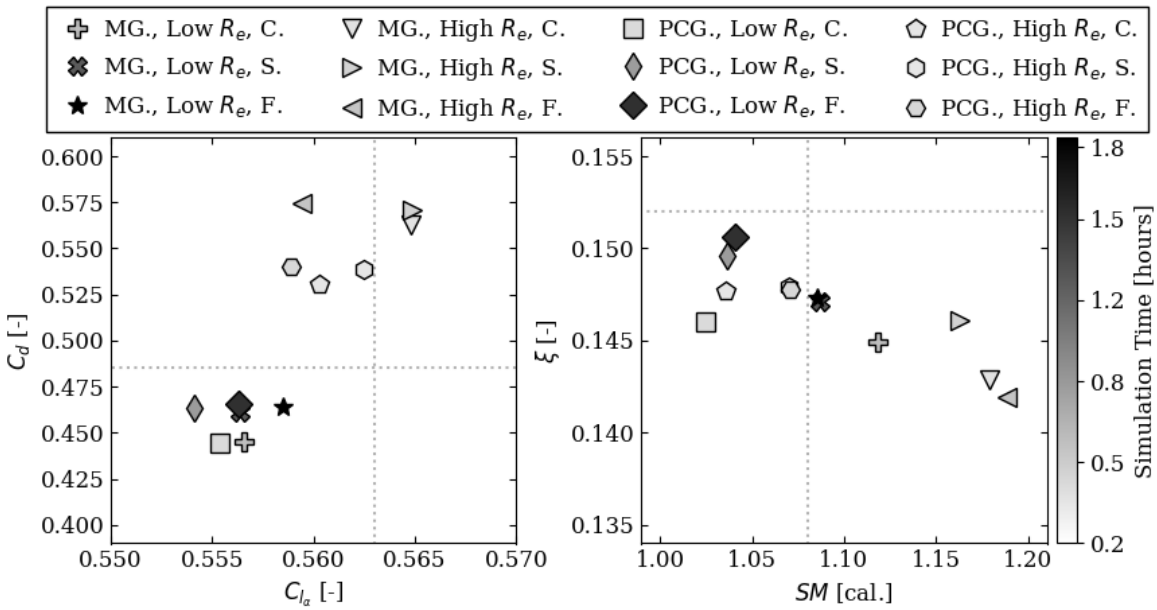
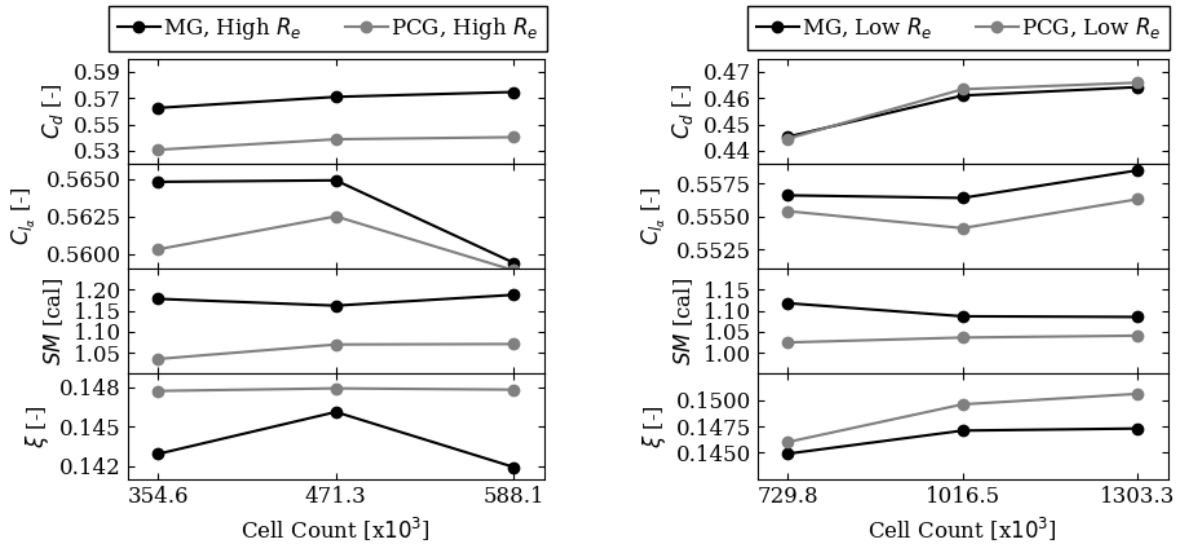


Figure 24 – Design coefficient space comparing all proposed CFD models. The axis denote the design coefficients, the symbols represent the CFD model setup, and the colors indicated the elapsed computation time until convergence. The opaque dashed lines represent reference values for each design coefficient. MG and PCG stand as abbreviations for the GAMG and PBiCGStab setups.

Observing the limits of the design coefficient axis in Figure 24, it is noticeable that all CFD models arrived at very similar results for C_{l_α} and ξ. C_d and SM, however, presented more disparate results. Opaque dashed lines are present in both plots to help visualize reference values of this design coefficients for the Armação rocket. These reference values were obtained from ANSYS Fluent simulations reported in [1]. A distinction can be observed in Figure 24 between the high R_e and low R_e boundary layer approaches, with the former leading to higher values of C_d than the latter.

Given the absence of experimental results to validate this data, it is hard to quantify which formulation best models the aerodynamic scenario, tough the low R_e models do appear to be closer to the ANSYS CFD reference. Seeking to further quantify the usefulness of the proposed models - besides only observing their computation cost and proximity to reference results - the Grid Convergence Index (GCI) strategy proposed by Roache [38] was also analyzed. This technique investigates the dependence of the numerical results on the mesh, by extrapolating the data from a series of simulations of increasing mesh refinement to a theoretical continuum solution (infinite mesh refinement). The GCI metric is a percentile deviation between the current result and this theoretical continuum result. Figure 25 shows the results of this analysis for all evaluated CFD models.

Table 11 combines all CFD performance metrics outlined thus far, namely: the Grid Convergence Index (GCI, Figure 25), the relative percentile error between the CFD model



(a) High R_e models.

(b) Low R_e models.

Figure 25 – Grid convergence analysis for the proposed CFD models employing the GCI metric.

and the reference coefficient values from [1] (ϵ, Figure 24), and the total simulation time required to reach less than 1% of relative variation (in 100 iterations) for all design coefficients (t, Figures 22 and 23).

Table 11 – Collected metrics evaluated in the selection of a viable CFD model setup. Entries marked with * indicate oscillating GCI results, not being suitable for the extrapolation to a continuum solution. The highlighted rows indicate the CFD setups selected for further analysis.

Setup	C_d			C_{l_α}			SM			ξ		
	GCI [%]	ϵ [%]	t [min]	GCI [%]	ϵ [%]	t [min]	GCI [%]	ϵ [%]	t [min]	GCI [%]	ϵ [%]	t [min]
MG High R_e C	5.5	16.0	12.7	0.3*	0.3	4.5	9.9*	7.2	2.2	31.5*	4.7	1.2
MG High R_e S	3.8	17.7	19.5	18.5*	0.3	6.3	40.8*	5.7	2.8	55.8*	2.6	3.1
MG High R_e F	-	18.5	25.4	-	0.6	7.6	-	8.0	6.3	-	5.4	3.8
MG Low R_e C	14.8	8.2	27.9	0.4*	1.1	9.7	5.9	1.6	8.0	4.0	3.4	5.7
MG Low R_e S	2.1	4.9	70.33	3.3*	1.2	14.4	0.4	1.2	7.2	0.5	1.9	11.2
MG Low R_e F	-	4.3	109.2	-	0.8	18.6	-	1.3	8.7	-	1.8	7.6
PCG High R_e C	1.3	9.4	6.8	1.6*	0.5	2.5	1.7	5.8	11.20	1.9	1.5	2.5
PCG High R_e S	0.9	11.1	10.9	2.2*	0.1	4.1	0.3	2.7	7.3	1.3	1.4	3.6
PCG High R_e F	-	11.4	17.1	-	0.7	5.3	-	2.6	23.56	-	1.5	5.9
PCG Low Re C	13.3	8.4	14.7	2.8*	1.4	7.3	8.5	6.8	16.1	13.2	2.7	11.7
PCG Low Re S	1.5	4.5	42.5	1.4*	1.6	12.7	0.9	5.8	9.6	1.1	0.3	8.5
PCG Low Re F	-	3.9	88.3	-	1.2	41.4	-	5.4	11.0	-	0.4	16.6

In general the PBiCGStab did prove itself as the faster numerical setup, producing comparable results with the GAMG alternative. In the end the PBiCGStab High R_e coarse and PBiCGStab Low R_e Standard models were selected for further use in the construction of the Reduced Order Models. These models showed a close proximity to

the reference C_{l_α} and C_d values, while having adequate SM and ξ results. Furthermore, these models showed adequate (although higher then desired) GCI values, and excellent total simulation times.

3.3 Reduced Order Modelling Strategies

Reduced Order Models act as center-piece in the parametric framework for the conceptual aerodynamic design of a model rocket rocket proposed in this work. ROMs were investigated as tools for the interpolation of sparse output parameter spaces, resulting from the parametric CFD analysis presented in Section 3.2; as well as tools for merging low and high fidelity data-sets through Machine-Learning based Reduced Order Models. Two open source python libraries were utilized for the purpose of creating ROMs out of varying data-sets; namely the python Surrogate Modelling Toolbox (pySMT) for the creation of traditional response surface models (as well as to implement the sampling strategy mentioned at the beginning of this chapter), and pyTorch, a well established Machine Learning package.

Before the actual construction on the ROMs, an auxiliary function was implemented in the software architecture to store all the outputted results in organized databases. The parametric CFD workflow generated a series of different Design of Experiments (DoE) data-bases, one for each design coefficient, and for each of the sampling numbers in the range [2,4,6,8,10,12] (totaling 24 data-bases). Furthermore, the CFD analysis also stored the convergence history of all aerodynamic (C_l, C_d) and stability (SM, ξ) coefficients for each design point, adding one more dimension to each data-base. This function was also used to created a large data-base for the Modified Barrowman Method results (1000 sample divisions, 2000 design points simulated), even though this particular data-set was not used to created traditional ROMs, only for training the initial Feed-forward Neural Network discussed in Section 3.3.2.

3.3.1 Traditional Response Surface Reduced Order Models

Out of the many Reduced Order Modelling algorithms available in pySMT, two response surface methods mentioned in [25] were investigated in this study, namely: Radial Basis Functions (RBF), and Gradient-enhanced Kriging with Partial Least Squares (GEKPLS). Generally, the aforementioned algorithms were treated as "black-box" tools, and their hyper-parameters were set to the standards documented at [28] for general applications. A brief outline of these models is also discussed in Section 2.3.1.

Implementing any response surface model using pySMT generally follows the procedure exemplified by Algorithm 2. Executing these steps for the RBF ROM requires no more the 5 lines of python code, thanks to pySMT integrated capabilities. The only

hyper-parameter of note is the Gaussian Scaling Factor (d_0 in Eq. 2.20), which by standard is set to 1.

Algorithm 2: General ROM implementation with pySMT (pseudo-code).

Load → data-bases with the CFD model output for each design coefficient (C_l, C_d, SM, ξ);
Set → separate training (one DoE set, unconverged) and validation data-bases (all other DoE sets, converged);
Assign → Reduced Order Model to main function class;
Set → Hyper-parameter tuning;
Train → ROM with discrete design points from unconverged CFD results;
Evaluate → ROM against all converged CFD results;
Derive → final ROM performance metric (R^2).

For both traditional ROMs considered, a "leave-one-out" approach was selected to evaluate the Reduced Order Models performance against new, related, data. During preliminary investigations, the ROMs were trained for one CFD data-set (the DoE 10 set, with 20 design points), evaluated at 50% residual convergence. The generated response surfaces were evaluated against the combined data points from all other remaining data-sets, at full convergence. This means that these ROMs constructed with a few, partially converged, data points, were evaluated against all available data points at the best available convergence resolution. A simple root mean square error function was selected to measured each ROM's performance.

Implementing the Gradient-enhanced Kriging with Partial Least Squares (GEKPLS) model requires one additional step in comparison with the RBF ROM, that is, calculating the gradients of each design coefficient (C_l, C_d, SM, ξ) with respect to the design parameters (h, s). To do this, a central differencing with nearest neighbour points technique was used. Inside an iterative loop for all design points x_i, the data-bases were first sorted according to proximity to x_i, the distance between the x_i and its n closest neighbours is calculated (in this case n was chosen as 4), and the difference between all outputs y_i (related x_i) and the neighbouring outputs y_n (related to x_n) are computed and stored. The local derivatives of the design coefficients at x_i are then estimated by averaging the sum of differences centered at x_i. The partial derivative of C_d with respect to h at some point (h_i, s_i) would be estimated according to Equation 3.14. The estimated gradients are computed into a matrix of the same dimensions as the input matrix for the GEKPLS method, and included alongside the original training data-base during the model's training phase.

$$\frac{\Delta C_{d,i}}{\Delta h_i} = \frac{1}{4}\Sigma_n^4 \frac{C_{d,i} - C_{d,n}}{h_i - h_n} \tag{3.14}$$

Section 4.3 compares the performance of both discussed traditional ROMs. After

this evaluation was performed, the GEKPLS model was selected and included in the main framework for the automated conceptual aerodynamics analysis proposed in this work. The ROM model constructed with partially converged CFD data was evaluated for the same number of design samples as the lower fidelity Modified Barrowman Method, generating comparable data-sets for each model; allowing for the fusion of this variable fidelity data within the Machine Learning Based ROM stage of the proposed framework presented here.

3.3.2 Transfer Learning in Multi Layer Perceptrons

Equipped with both the lower fidelity data-set outputted by the Modified Barrowman Method, and the higher fidelity data-set outputted by the Reduced Order Model constructed with partially converged CFD data, Deep Feed-forward Neural Networks were implemented as tools of data-fusion in the proposed conceptual aerodynamics design framework presented here. PyTorch, a proven open source Machine Learning library for python [39], was extensively used to construct Multi Layer Perceptron models. Again, as with the tradition ROMs discussed in the previous section, the Neural Network models developed here were largely treated as "black-box" engineering tools, but careful evaluations were made to establish a reasonable architecture for the Neural Networks constructed.

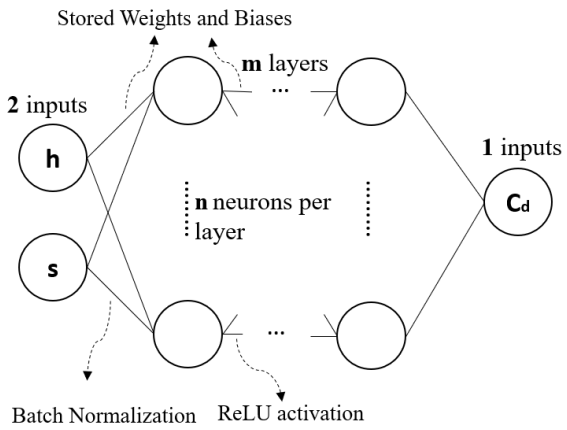


Figure 26 – General configuration and required parameters for a Multi Layer Perceptron correlating inputs of h and s with predictions of the drag coefficient C_d.

From a software development point o view, creating Feed-forward Neural Networks is made extremely simple by pyTorch predefined classes and functions. Designing a Multi Layer Perceptron (MLP), for example, can be accomplished by following structured modular functions extensively detailed in pyTorch's documentation [39]. Figure 26 illustrates the required inputs to define a general MLP architecture in pyTorch, namely: the number of inputs (2, for the fin-set optimization problem), the number of hidden layers **m**, the number of neurons per hidden layer **n**, and the number of outputs (1, for a single design coefficient prediction).

Different structures were investigated for the MLPs developed here, namely with permutations of $\mathbf{m} = [1,3,5]$ and $\mathbf{n} = [16,32,64]$. Rectified Linear (ReLU) activation functions were used in all models created. Batch normalization was applied to the active outputs of all hidden layers, with batch sizes of 64 and 4, for the MLPs trained with the MBM data-set and with the CFD ROM data-set, respectively. All MLP networks employed a Mean Squared Error Loss function (Eq. 2.21), and the SGD (Stochastic Gradient Descent) algorithm was selected to optimize the back-propagation phase of the Neural Networks.

The Multi Layer Perceptrons were trained for 1000 iterations (or epochs) of the back-propagation algorithm, and a "leave-one-out" validation technique was also implemented for the final MLP (CFD ROM data-sets), meaning that a model trained with a DoE set with partially converged CFD data was validated against all other DoE sets with fully converged data. As was done with the response surface ROMs; all hyper-parameters for the Neural Networks, loss function, and back-propagation optimizer, were set to the standard values recommended for general analysis by the pyTorch documentation [39].

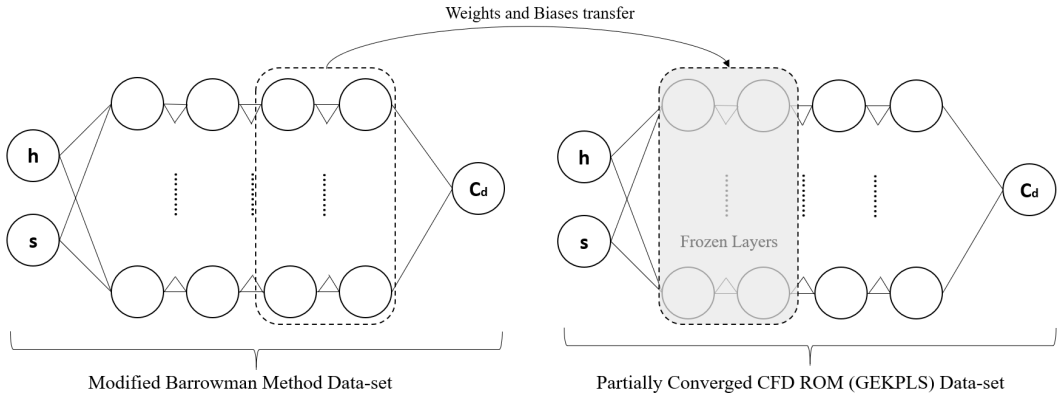


Figure 27 – Schematic of the Transfer Learning step for a MLP architecture with 4 hidden layer and an arbitrary number of neurons per layer.

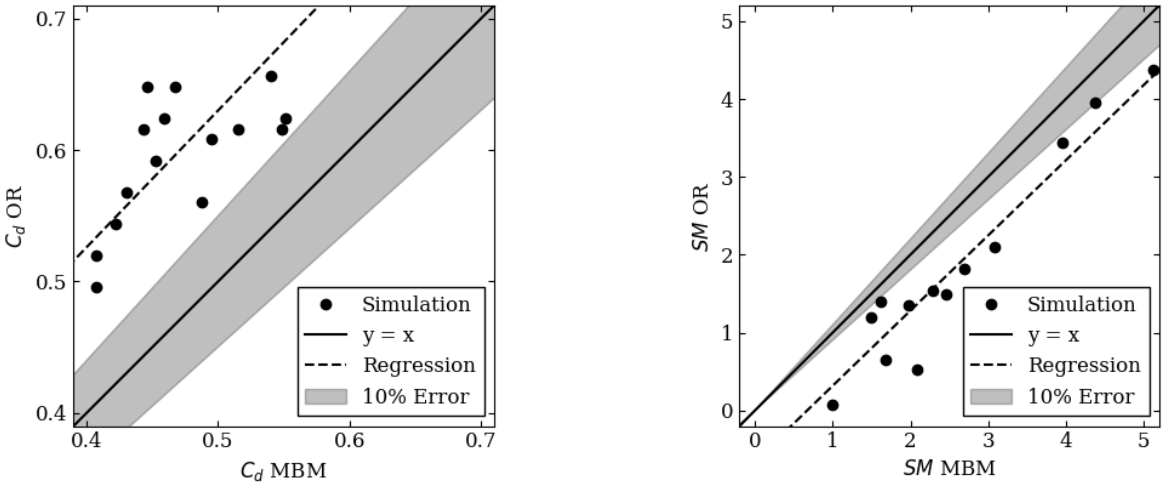
To simplify the Transfer Learning step, the same MLP architecture was used for both the low fidelity (MBM) and high fidelity (CFD ROM) data-sets. Because of this, no extra processing needs to be done with the stored weights and biases vectors, and they can be directly exported from one Neural Network to the next. In the present work, this was simply done by first training the MLP Network with data form the Modified Barrowman Method data-set, and storing the final weights and biases vectors corresponding to the last $\mathbf{m}/2$ hidden layers. Then, these vectors are loaded to the first $\mathbf{m}/2$ hidden layers of the same MLP model, and a new training process is initiated with the partially converged CFD ROM data-set. In effect, these loaded weights and biases remain fixed throughout the training, and only the input signal magnitudes are altered for the first (frozen) layers of this MLP. The last half of the hidden layers evolve normally during the training. Figure 27 illustrates this process.

4 Results & Discussion

4.1 Analytical Model

In Apex Rocketry's previous projects, the rocket's aerodynamic and stability conceptual design typically revolved around iterative testing of different rocket geometries using OpenRocket. This process, though manually repetitive, has thus far been able to consistently produce safe, statically stable rockets. Therefore, this open-source rocket flight simulation software was taken as a reference against which the modified Barrowman method (MBM) developed in this work could be compared.

To compare the low fidelity, semi-empirical MBM with OpenRocket, the Armação A-22 rocket was taken as a baseline model and 16 other rockets were constructed by randomly altering the rockets fins and canards parameters, within physically meaningful constraints. Each randomized geometry was then simulated in OpenRocket and with the MBM. Unlike the MBM, OpenRocket's software was designed to simulate an entire flight, so obtaining data at an specific flight scenario (given α and M_a) is not as straightforward as in the MBM. Therefore, all randomized rockets were equipped with the same motor, had very similar total masses, and were launched at an up-right 90° angle with no ambient horizontal wind, so as to approximately achieve $M_a = 0.7$ with $\alpha = 0°$ at the motor burnout (maximum speed).



(a) Total drag coefficient predictions.

(b) Static margin predictions.

Figure 28 – Analytical aerodynamics comparison between the modified Barrowman method (MBM) and OpenRocket (OR). The "y=x" line indicates an equal prediction between the models.

Figure 28 compares both of the semi-empirical models. For both the C_d and SM, a shaded region is included to indicate a 10% deviation between the predicted values. Figures 28a and 28b show that the data sets do not perfectly agree with each other. This is to be expected, given that the MBM combines equations from references other than OpenRocket [3]. More importantly for the optimization purposes of this work is the tendency (represented by the linear regression shown in Figures 28a and 28b) between the data sets.

Even though constant offset errors are clearly noticeable for the SM and C_d, both data sets still show a good agreement between their linear tendencies. This indicates that similar alterations to the rocket geometry will lead to similar predicted performance outcomes for each model. This preliminary result is a good indicative that rockets optimized with the low fidelity MBM can be expected to follow similar optimization paths as those optimized with the more traditional, labor-intensive, OpenRocket method.

The modified Barrowman method's performance was further evaluated through direct comparisons between the MBM's and CFD model's outcome. The aerodynamic and stability design coefficients curves were estimated for the Armação A-22 rocket using the MBM and the PBiCGStab Low R_e Fine mesh CFD model resolved after 1000 iterations. This specific CFD model setup was selected as it was one of the most computationally expensive and robust models proposed in Section 3.2.

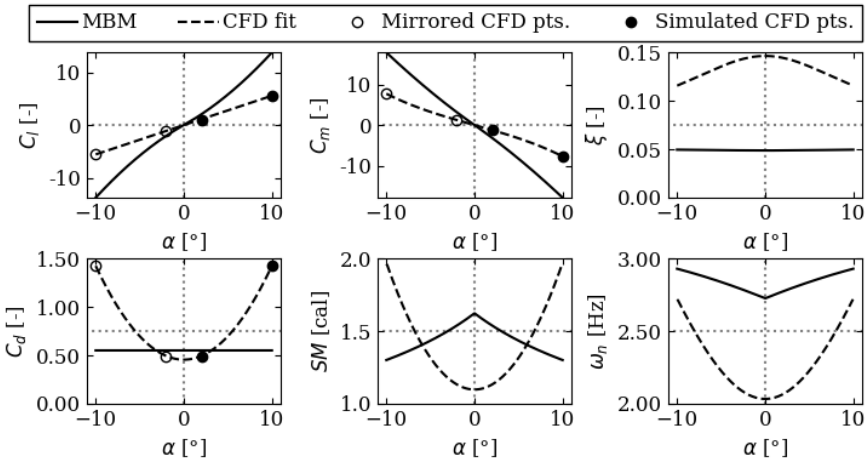


Figure 29 – Design coefficient curves estimated by the modified Barrowman method (MBM) and PBiCGStab Low R_e Fine CFD model (refereed to simply as CFD).

Figure 29 illustrates the aforementioned comparison. The curves representing the MBM results were calculated for each α value, and the CFD model curves were fitted by the simulated and mirrored results (at $\alpha = 2°$ and $\alpha = 10°$), following the procedure described in Section 3.2.

For the aerodynamic coefficients (C_l, C_d, and C_m), a positively good agreement between the analytical and numerical models can be observed, especially considering the

crudeness of the proposed modified Barrowman method implementation. This agreement naturally breaks down at higher angles of attack (especially for the constant prediction of C_d), but it is reasonable to assert that the MBM and CFD are comparable for nominal (low α) flight conditions.

The same conclusions, however, cannot be drawn out for the static and dynamic stability coefficients (SM, ξ, and ω_n). Both methods compute these coefficients through vastly different techniques; with the MBM estimating the x_{cp} location and C_{l_α} for each component using semi-empirical correlations, and the CFD model directly calculating the stability coefficient as the derivatives of the aerodynamic coefficients. Noticing that the SM curve obtained by the MBM does not respect the derivative definition of the static margin (latter half of Eq. 2.3), it is reasonable to conclude that the semi-empirical stability correlations do not adequately compare to the CFD results. However disparate these results are from one another, all coefficients still show reasonable values, leading to admittedly valid rocket designs.

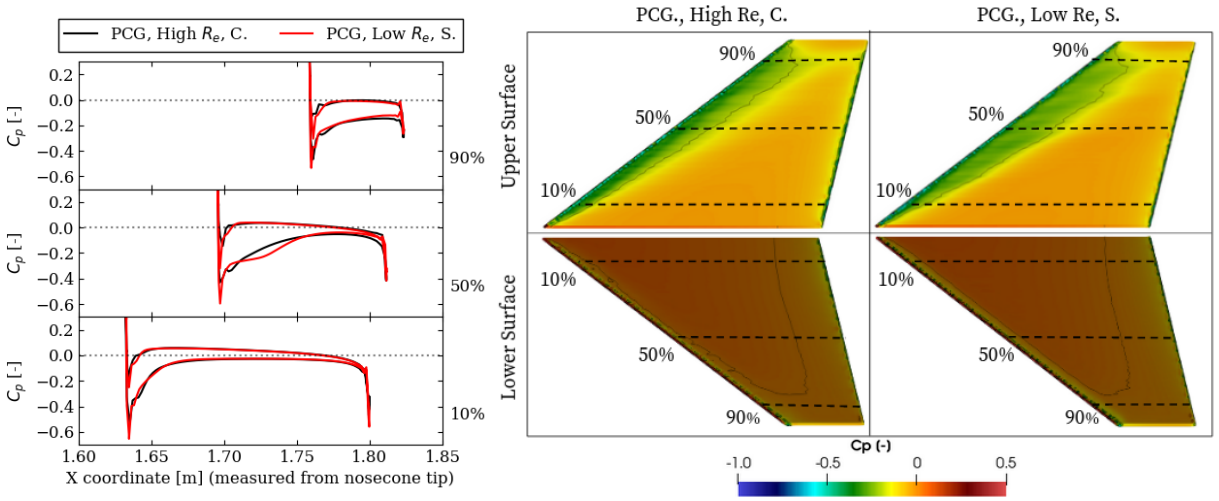
4.2 Numerical Model

After the preliminary investigation of the CFD model setups proposed in Section 3.2, the PBiCGStab High R_e Coarse mesh and PBiCGStab Low R_e Standard mesh models were selected for further evaluation. One of the principal purposes of this work is investigating the use of less computationally expensive (and therefore less robust) CFD models as an alternative to more traditional numerical design approaches. For this reason, the selected CFD models differ specially in their mesh and boundary layer modeling approaches. Further discussions on the differences between these CFD setups are presented in Section 3.2.

The parametric OpenFOAM CFD code was ran for both CFD models using the Armação A-22 rocket as a baseline. The models were simulated until the convergence criteria (less than 1% variation per 100 iterations for all design coefficients) was reached; namely for 360 iterations for the PBiCGStab High R_e Coarse model, and 400 iterations for the PBiCGStab Low R_e Standard model. All simulations were performed on a personal computer with a 6-core 12-thread Ryzen 3 5600 CPU. The calculated flow variable fields were manually post processed using Paraview.

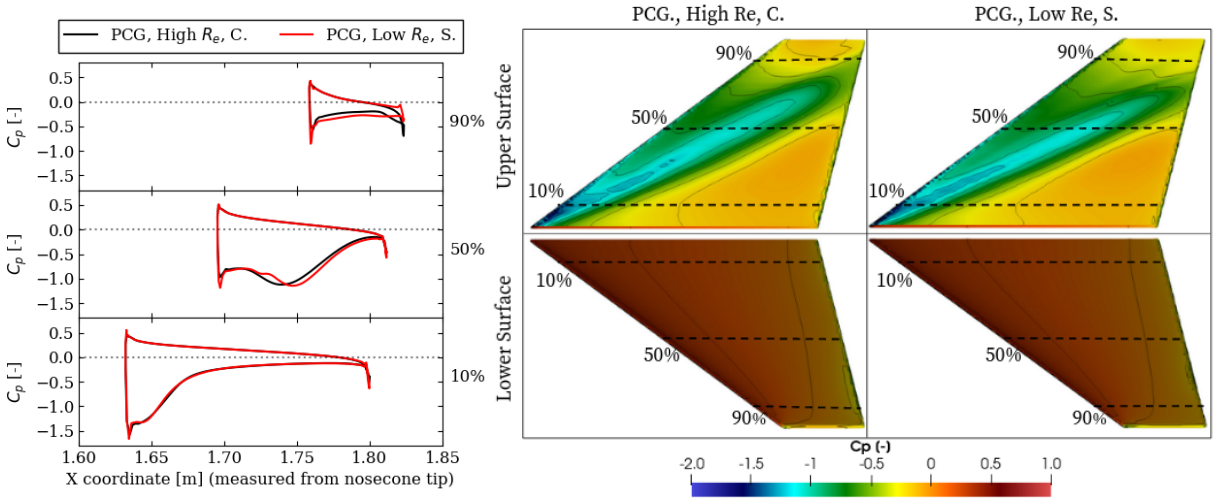
Initially, a comparison between the pressure fields calculated for each CFD model was performed. Figure 30 illustrates the coefficient of pressure (C_p) field particularly focusing on one of the rocket's fins, given that it is the most important aerodynamic surface of the rocket. Curves of the C_p along sample fin chord lengths are presented alongside the distributed C_p field for the upper and lower fin surfaces. At an α of $+2^\circ$ the rocket is undergoing an upwards pitching motion, so the oncoming wind directly hits the fin's lower surface. This leads to a mostly positive manometric pressure field at the fin's lower

surface, with a negative manometric pressure field developing at the opposite surface. This phenomenon can be clearly observed in Figure 30a, with the flatter positive sections of the C_p curves corresponding to the fin's lower surface. It is also interesting to note how the positive C_p values steadily decline for regions closer to the fin's tip chord, possibly due to wing tip effects.



(a) C_p curves for select cross sections. (b) Distributed C_p field along the fin's surfaces.

Figure 30 – Coefficient of pressure for one of the Armação A-22 rocket's aerodynamically loaded fins at $M_a = 0.7$ and $\alpha = 2°$.



(a) C_p curves for select cross sections. (b) Distributed C_p field along the fin's surfaces.

Figure 31 – Coefficient of pressure for one of the Armação A-22 rocket's aerodynamically loaded fins at $M_a = 0.7$ and $\alpha = 10°$.

Figure 31 highlights similar results for the rocket's pressure field at $\alpha = 10°$. At this more intense aerodynamic loading scenario, higher magnitudes and stronger gradients of C_p can be observed. This conditions naturally leads to higher values of resultant aerodynamic forces and moments, resulting in higher magnitudes of C_l, C_d, and SM. A

considerable difference in the C_p field can be observed for Figures 30b and 31b. The C_p gradient present in the $\alpha = 10°$ case, though similar to an oblique shock structure (which could happen, given the rocket's transonic aerodynamics), can be more likely attributed to a recirculating flow detachment phenomena occurring close to the fin's leading edge. This type of flow physics is indeed expected to occur at high angles of attack, especially considering that the fins are flat plates with sharp leading and tailing edges.

RANS CFD solvers, such as the rhoSimpleFoam solver employed here, are not suitable for modeling large scale detached flow; and the low number of simulated α values is certainly not large enough to capture aerodynamic stall effects. Both of these phenomena are, however, secondary for the optimization purposes of this work, and it is more meaningful here to discuss the similarities between the chosen High R_e and Low R_e boundary layer CFD models.

In Figures 30a and 31a a very good agreement between the C_p predictions of both models can be observed, especially for sections close to the fin's root chord. Slight differences between the models can only be observed for the higher gradient regions of the fin's upper surface, especially for regions close to the tip chord. This good comparison of the simulated C_p field is a strong indicative that the less robust High R_e CFD model (employing empirical wall functions to estimate the boundary layer) can produce comparable C_l and C_m (and, therefore their derivative SM value) results to the more costly Low R_e CFD model.

Even though the PBiCGStab High R_e Coarse mesh and PBiCGStab Low R_e Standard mesh show similar results for the C_p at the rocket's surface, the same cannot be said about the flow's coefficient of friction (C_f). Figures 32 and 33 display the simulated C_f field for both of the selected CFD model setups. Only results for $\alpha = 10°$ are shown due to their larger magnitudes and easier visualization of flow phenomena, but the $\alpha = 2°$ condition showed similar results between the models. Different rotations about the rocket's longitudinal axis are show to illustrate the complete flow-field at the rocket's surface.

A considerable difference in C_f can be observed in Figures 32 and 33; which certainly amounts to the differences observed for the C_d prediction between the High R_e and Low R_e models, as the integral of C_f along the rocket's surfaces constitutes an important part of the total drag force. The higher average values of C_f for the High R_e model (especially for the rocket's body tube) can be outlined as the principal difference between the Low and High R_e CFD models proposed in this work. In fact, returning to Fig. 24 equipped with this interpretation, it is noticeable that all the High R_e models show higher values of C_d, in agreement with the reasons more closely observed here.

The differences in the coefficient of friction values follow as a natural consequence of the different boundary layer modeling strategies employed by the High R_e and Low R_e CFD models. C_f is directly dependent on the wall shear stress τ, which in turn is derived

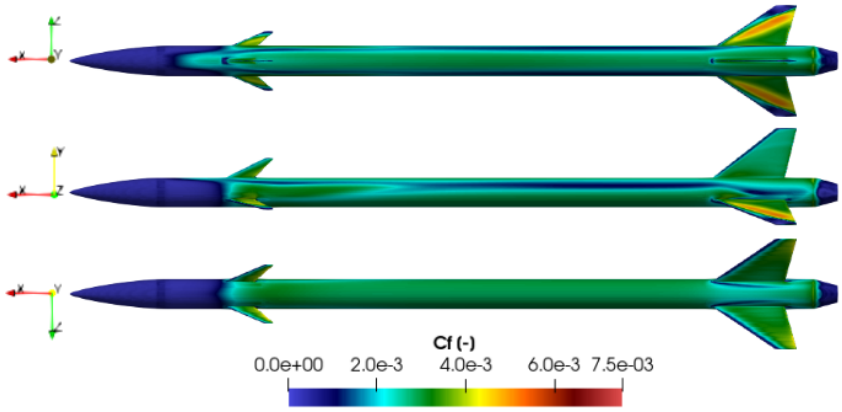


Figure 32 – C_f field at the rocket's surface for the PBiCGStab High R_e Coarse mesh CFD model simulated at $M_a = 0.7$ and $\alpha = 10°$.

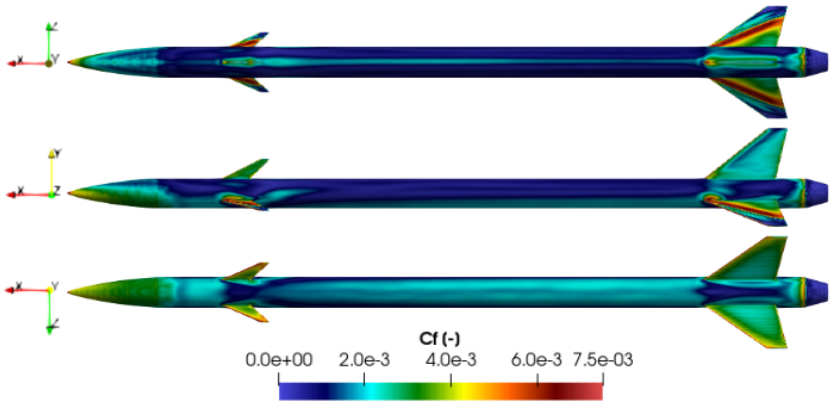


Figure 33 – C_f field at the rocket's surface for the PBiCGStab Low R_e Standard mesh CFD model simulated at $M_a = 0.7$ and $\alpha = 10°$.

from the local velocity gradient at the wall. Low R_e models employ small mesh elements to model this velocity gradient as piece-wise linear functions, whereas High R_e models approximate this velocity gradient through empirical wall functions.

Carefully observing Fig. 33, it can be noted that the Low R_e boundary layer modeling approach does in fact produce more physically coherent results, capturing finer flow details. This is especially evident when evaluating the calculated C_f field at the nosecone, where a much more meaningful C_f gradient is depicted when compared to the High R_e case.

Both C_p and C_f are important coefficients in determining the resultant aerodynamic forces and moments developed by the rocket. However, at the moderate Mach (0.7) and high Reynolds ($\approx 2.3 \times 10^6$) number flows evaluated here, strong advecction effects are to be expected. This means that, for regions of simple attached flow, the overall boundary layer thickness is reduced when compared to slower free-stream speeds. This, in turn, reduces the total effect that the viscous boundary layer (and therefore C_f) have on the total resultant forces. This indicates that, for this type of external flow application, C_p plays a larger overall role than C_f. In fact, this is also expressed by the magnitudes

of both coefficients, given that both are manometric values non-dimensionalized by the free-stream dynamic pressure.

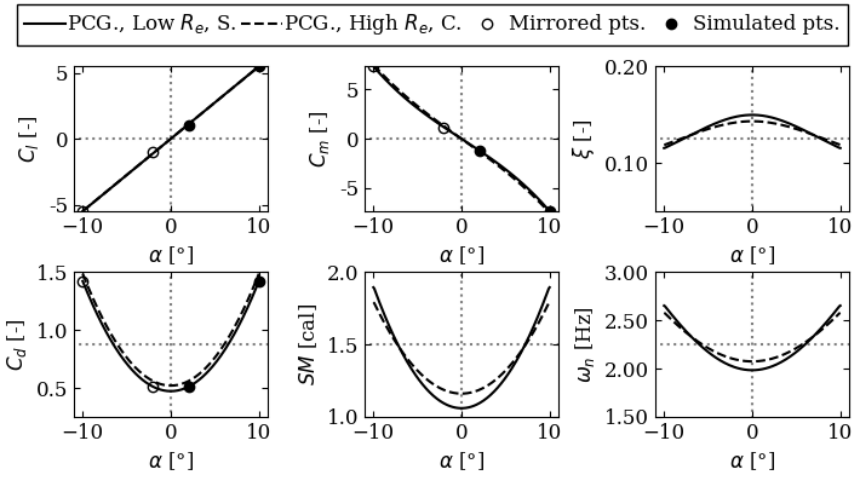


Figure 34 – Design coefficient outcomes for the PBiCGStab High R_e Coarse mesh and PBiCGStab Low R_e Standard mesh CFD model setups.

Finally, the aerodynamic and stability design coefficients estimated by both CFD models were compared. Figure 34 outlines these results. Following the previous considerations on the effects of C_p and C_f, it becomes apparent that the results for C_l and C_m are directly linked to the good C_p agreement between models, and the C_d results are linked to the poor C_f agreement between both models.

Furthermore, the higher impact of the C_p on the total results can once again be observed by noticing that the C_d curves are only slightly offset from one another (due to the differences in C_f), indicating that the C_p contributes to a majority of the estimated C_d magnitude. In fact, larger discrepancies between the Low and High R_e boundary layer modeling approaches are actually observed for the static and dynamic stability coefficients. This, however, naturally follows from the propagation of errors in C_l and C_m, as all stability coefficients are obtained through the derivatives of these two terms.

Considering the in depth comparison between the proposed models, it can be concluded that the less robust High R_e CFD model is a good alternative to the more costly and refined Low R_e approach, especially for conceptual optimization problems. This conclusion is primarily based on the good agreement of the simulated C_p fields between both modeling approaches; but the higher C_d values predicted by the High R_e model (due to larger C_f effects) can also be interpreted as a conservative scenario, given that C_d is the principal optimization variable for most problems. These favorable results for the High R_e model are even more encouraging when noting that each fully converged simulation with this setup demands about 5 minutes of computation, compared with the 30 minutes that the Low R_e model requires (for 6 CPU cores).

4.3 Traditional Reduced Order Models

Once a high fidelity CFD modeling approach was properly defined to tackle the Armação rocket's aerodynamic optimization problem, the parametric design workflow outlined in Section 3.3 could be fully implemented. In so doing, the OpenFOAM script developed in this work was able to create a data set of CFD simulation results for an arbitrary sample of rocket designs, $\underset{\sim}{x}$. Traditional reduced order models (ROMs) in the form of response surfaces were then constructed to interpolate between these design points.

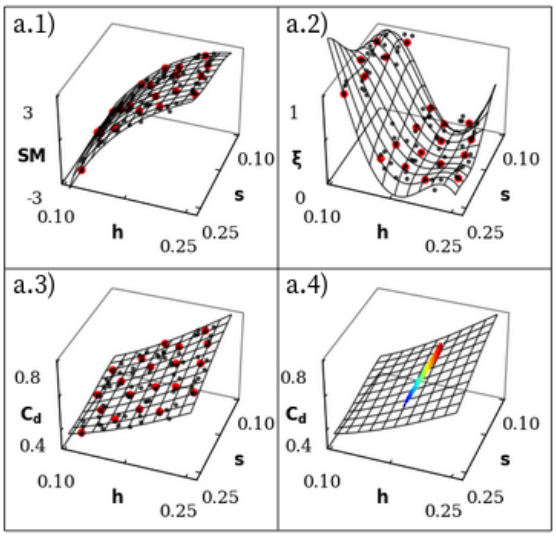
The PBiCGStab High R_e Coarse mesh model presented in Section 4.2 was used to construct the high fidelity model data set. The rocket fin's height (h) and sweep length (s) were parameterized within the interval $[100, 250]$mm, and the rocket's other geometric parameters were kept constant; so as to facilitate the visualization of the optimization results in three dimensions. The Latin Hyper Cube sampling strategy was employed to select the sample $\underset{\sim}{x} = (h, s)$ points within the design constraints. The complete CFD results data set is subdivided into 6 smaller Design of Experiments (DoE) data sets, corresponding to $[2, 4, 6, 8, 10, 12]$ result points per dimension of $\underset{\sim}{x}$; totaling $[4, 8, 12, 16, 20, 24]$ CFD results for each DoE (given that $\underset{\sim}{x}$ has two dimensions, h and s).

Initially, two ROMs strategies were investigated as alternatives for the construction of response surface interpolations: radial basis functions (RBF), and gradient-enhanced Kriging models with partial least squares (GEKPLS). Implementations for both ROMs were readily available in python by the pySMT library, and where promptly integrated within the software design for this work. During preliminary investigations, both ROMs were employed using partially converged CFD estimations of the C_d, SM, and ξ coefficients obtained from the DoE 10 data set. Figure 35 displays the response surface outcomes for the aforementioned ROMs.

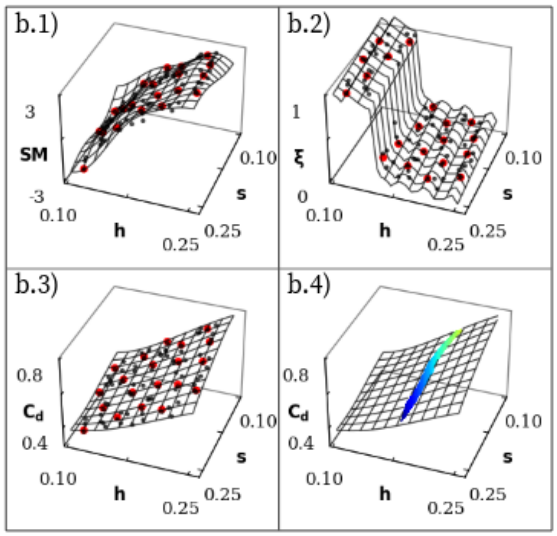
In Figure 35 the 3d-wireframe surface represents the reduced order model for each design coefficient (C_d, SM, and ξ) as a function of the design variable ($\underset{\sim}{x} = (h, s)$). Each ROM was constructed using only the (unconverged) points highlighted in red, and was subsequently compared to all other (converged) results.

Comparing Figures 35a and 35b it is visually clear that both the RBF and GEKPLS ROMs appear to model the continuous C_d and SM CFD results reasonably well, with the RBF model showing smoother results, which could be attributed to the quadratic gaussian mathematical formulation of the model. The damping ratio (ξ) data set, however, highlights the considerable differences between both ROMs. As discussed in Section 3.2, ξ tends toward imaginary values for undamped rockets. To avoid undefined results within the data set, dynamically unstable rockets are artificially set to $\xi = 1$ during the CFD results post processing phase.

The GEKPLS reduced order model appears to capture this discontinuity within the $\underset{\sim}{x}$ space well, but the model's reliance on the partial derivatives of the design coefficients



(a) Reduced order model built with radial basis functions.



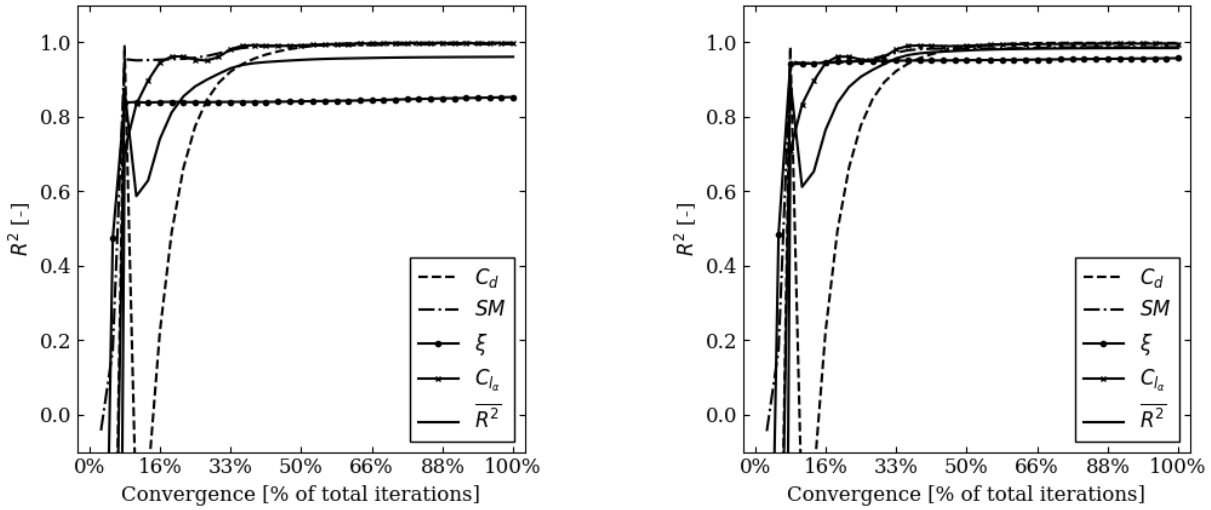
(b) Reduced order model built with gradient-enhanced kriging with partial least squares.

Figure 35 – Response surfaces created by the RBF and GEKPLS ROMs for the DoE 10 50% convergence data set. Red points correspond to the DoE 10 results, and gray points correspond to the results of all other DoE data sets.

in terms of $\underset{\sim}{x}$ leads to excessive oscillations of the response surface interpolation between the design points. The RBF model, though not showing any oscillations, cannot capture the discontinuity in ξ, which severely undermines its validity as an optimization tool in the context of this work. This assertion can be explained by interpreting Figures 35 a.4 and b.4. The colored map in the C_d response curves of both figures corresponds to rocket designs that obey the optimization constrains of $SM = [1.5, 2]$ and $\xi = [0.05, 0.2]$ discussed in Section 3.0.1. Observing both figures it becomes apparent that the differences in the prediction of ξ ultimately lead to different valid optimization spaces and, consequentially, different optimum outcomes.

To adequately compare the performance of the RBF and GEKPLS reduced order models, the coefficient of determination (or R^2 score) between the ROMs constructed for the DoE 10 data set and the converged results from all other DoE data sets was evaluated (following the leave-one-out validation procedure discussed in Section 3.3). Reduced order models were constructed using data from the convergence history of the design coefficients C_{l_α}, C_d, SM and ξ. Figure 36 highlights these model performance results.

Comparing Figures 36a and 36b the previously described conclusions are made even clearer. Both reduce order models show similar R^2 results for all design coefficients other than ξ, with the GEKPLS ROM showing better results. It is also worth noting how the design coefficients' R^2 values evolves in a similar manner to the actual CFD residual's history along the calculated iterations (as indicated in Figures 22 and 23). An average of the R^2 values for all design coefficients is also shown in Figure 36 ($\overline{R^2}$), and this value is than evaluated for ROMs constructed for all DoE data sets and CFD convergence values

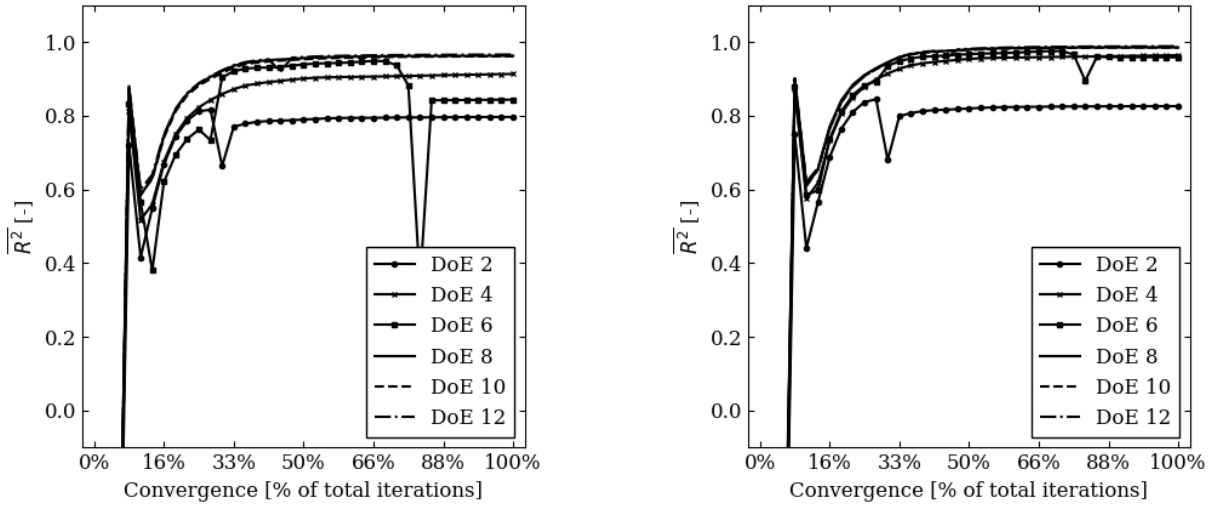


(a) RBF ROM built with the DoE 10 data set.

(b) GEKPLS ROM built with the DoE 10 data set.

Figure 36 – Coefficient of determination (R^2) for ROMs constructed at varying CFD convergence values for the lift derivative (C_{l_α}), drag (C_d), static margin (SM), and damping ratio (ξ) design coefficients.

in Figure 37.



(a) RBF ROM performance.

(b) GEKPLS ROM performance.

Figure 37 – Average coefficient of determination ($\overline{R^2}$) values for ROMs constructed from all design of experiments (DoE) data sets evaluated at varying CFD convergence values.

Figures 37a and 37b once again indicate that the GEKPLS reduced order model performs better than the RBF ROM alternative, with the latter achieving $\overline{R^2}$ scores greater than 0.9 after approximately 40% convergence for all ROMs constructed with 4 or more CFD results per design variable (DoEs). The RBF model, in comparison, shows a poorer

performance for DoE up to 6 CFD results per design variable. Both ROM strategies clearly show that that larger data sets lead to predictions closer to the actual CFD results, and that the DoE 2 data set is too small to produce adequate response surface predictions.

The good performance results for the gradient-enhanced Kriging with partial least squares reduced order model (GEKPLS ROM) constitute a promising indication that this ROM can in fact lead to adequate optimization results using only partially converged CFD data; and for this reason this strategy was selected for further evaluation on the aerodynamic optimization problem of the Armação rocket. For these reasons, this Reduced Order Model was selected and Incorporated into the automated framework discussed in this work.

4.4 Machine Learning Assisted Reduced Order Model

At this stage in the parametric framework for the conceptual aerodynamic design proposed in this work, both data-bases required for the Transfer Learning Reduced Order Modelling approach idealized in Section 3.3.2 are available. The first data-base is populated with design coefficients estimated by the lower fidelity, semi-empirical, Modified Barrowman Method. The second data-base consists of the results from the surface response GEKPLS model created with partially converged CFD data, presented in Section 4.3. The same Multi Layer Perceptron Network was developed for each data-set and trained for 1000 epochs, with the weights and biases from the lower fidelity MLP being loaded into the higher fidelity MLP following the method discussed in Section 3.3.2.

Different architectures for the Multi Layer Perceptron Neural Networks were investigated by evaluating varied combinations between the number of hidden layers and of neurons per layer. All architectures used ReLU activation functions, with batch normalization in between the hidden layers, as well as the stochastic gradient descent backpropagation optimizer with mean square error loss function. A "leave-one-out" validation strategy (comparing the model against all unused converged CFD data) was implemented, using the coefficient of determination R^2 as the validation metric.

Figure 38 presents the aforementioned results. Observing it, it becomes apparent that most architectures can predict the rocket's drag coefficient (C_d) and static margin (SM) relatively well, while the prediction of the damping coefficient (ξ) appears to be harder, especially for MLP architectures with 16 neurons per hidden layer. As was the case for the traditional response surface models reported in Section 4.3, the Neural Network's worst performance for ξ is to be expected, due to the output discontinuity in the design space. Ultimately an MLP architecture with 5 hidden layers (7 total layers) and 32 neurons per hidden layer was selected, as it produced R^2 validation results above 0.9 for all design

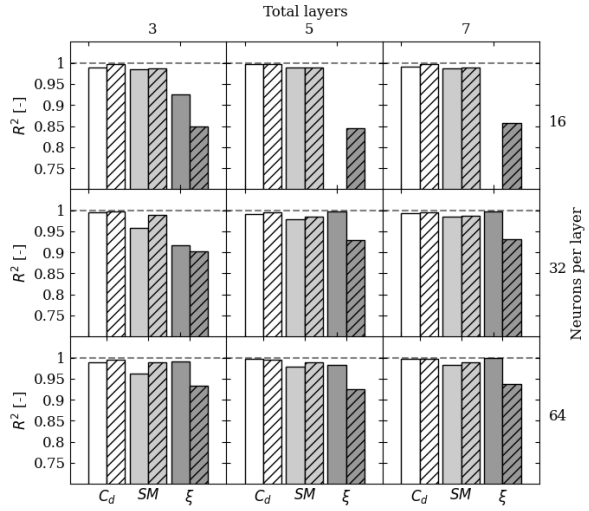


Figure 38 – Parametric investigation on the Multi Layer Perceptron architecture. Dashed bars correspond to the complete Transfer Learning Neural Network, while standard bars indicate the validation metric for the lower fidelity MLP based on the MBM data-set.

variables for the Transfer Learning Multi Layer Perceptron highlighted with the hashed bars.

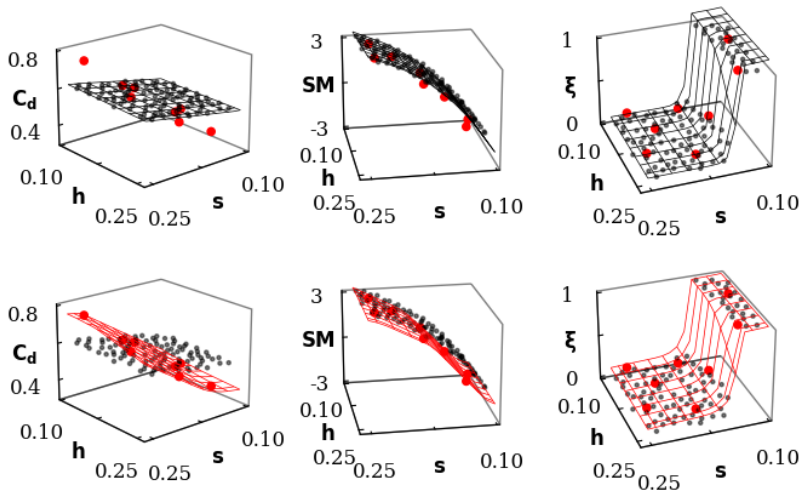
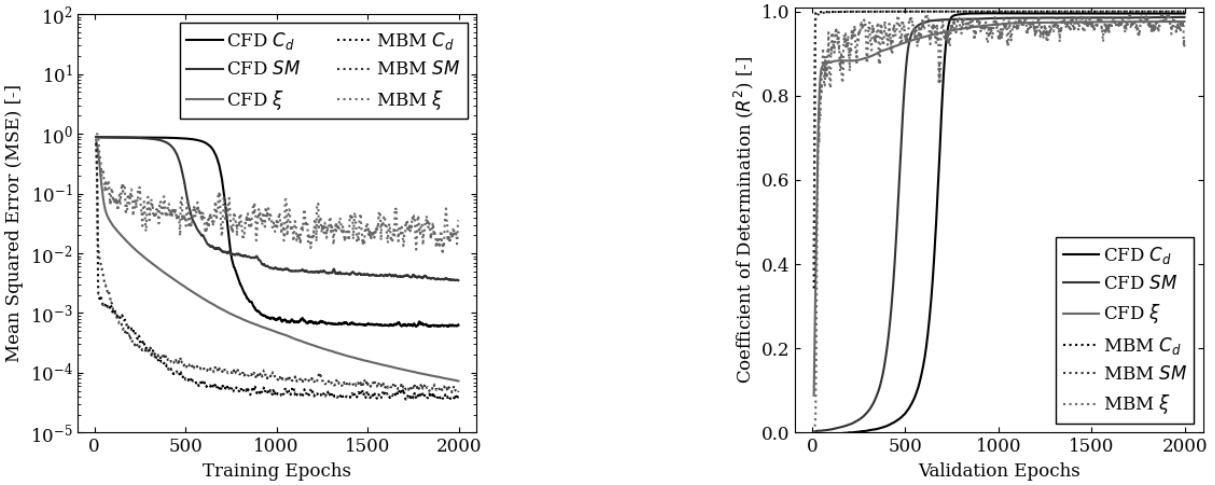


Figure 39 – Multi Layer Perceptron estimation of the aerodynamic and stability coefficients in the design parameter space. Black colors indicate the MLP trained with the MBM data-set, and red colors are assigned to the Transfer Learning MLP trained with both data-sets. Red dots indicate the CFD results for the DoE 4 data-set at 50% convergence, used to train the latter Neural Network.

Figure 39 illustrates the predictions of this final MLP architecture in the fin-set's $h-s$ design space. In this Figure, the Transfer Learning MLP was trained with the CFD DoE

4 data-set at 50% convergence, meaning that there were only 8 CFD data points used to train the model. It is remarkable to notice that, even with so few training data points, the Transfer Learning Neural Network is able to change from a surface fitting the MBM data, to a surface fitting the CFD data. This is especially apparent for the C_d results, were the CFD can predict more realistic quadratic results, while the semi-empirical Modified Barrowman Method only produced linear estimates of the drag coefficient.



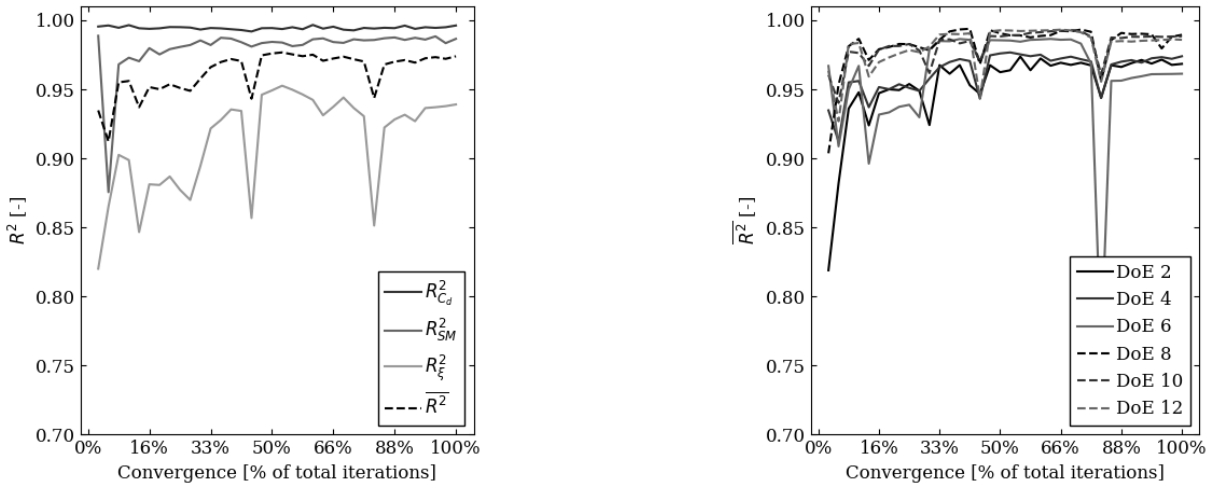
(a) Training results evolution.

(b) Validation results after each training epoch.

Figure 40 – Extended training and validation results for the selected Transfer Learning MLP architecture. Dashed lines represent the initial MLP trained with MBM data, and solid lines represent the final TL MLP trained with partially converged CFD data, and validated against fully converged CFD data.

The good performance of this TL MLP architecture is again noticeable in Figure 40, were the training loop is extended to 2000 iterations. After 1000 epochs, all results showed MSE errors lower than 10^{-2}, apart from the ξ model trained with the MBM data-set, which appears to be over-fitting the data, potentially due to the high number of data points. Immediately after each training epoch, the resulting model was also validated against all other (fully converged) CFD data points not used in the training. After 1000 epochs, R^2 validation results for the TL MLP reach a plateau roughly suppressing 0.95.

Finally, the chosen Transfer Learning Multi Layer Perceptron architecture was evaluated for all CFD Desing of Experiments data-sets, and for all the CFD convergence history stored in each DoE data-set. To visualize the overall performance of the Machine Learning based models, an average of the R^2 metric for all design coefficients was taken as the principal analysis parameter, as illustrated in Figure 41a. Observing this metric in Figure 41b, it is noticeable that most models showed average validation results above 0.95 after 50% of the convergence history has elapsed, and even models trained with very sparse data-sets (DoEs 2, 4, and 6) showed overall good validation results.



(a) Visualisation of the average validation metric derived from all design coefficients.

(b) Average R^2 evaluated for all CFD DoE sets and convergence history values.

Figure 41 – Validation results for the chosen TL MLP architecture trained with varying CFD DoE data-sets at varying convergence stages.

4.5 Optimization Results

After establishing all the variable fidelity models that constitute the aerodynamics conceptual design framework idealized in Figure 13, the crucial (and final) step of solving the optimization problem presented in Section 3.0.1 can be executed. At this stage, three different results data-sets are available to perform the optimization step, namely: the lower fidelity results from the semi-empirical Modified Barrowman Method; the higher fidelity results from the GEKPLS Reduced Order Model constructed with CFD data; and the results from the Transfer Learning Feed-forward Neural Networks combining both previous data-sets.

The optimization of the Armação-A22 rocket's fin-set was taken as a case study to formulate the conceptual design framework, and only two design parameters were defined as optimization inputs: the stabilizer's height h and sweep length s. As discussed in Section 3.0.1, this 2D design space can be trivially optimized. Figure 42 reinforces this statement, by plotting the drag coefficient optimization results for MBM data-set and visually highlighting the minimum drag location. C_d is plotted in the $h - s$ space with a colored region outlining the valid optimization space, within the constraints of $SM \in [1.5, 2]$ cal and $\xi \in [0.05, 0.2]$.

Figure 43 uses the previous constrained optimization space visualisation, but displaying the results obtained with the GEKPLS ROM and the Transfer Learning MLP data-sets. Figures 43a and 43b also illustrate the optimization outcomes for models constructed with different DoE data-sets, and with varying levels of CFD convergence. Observing this figures, it is noticeable that the valid optimization region is different between the models,

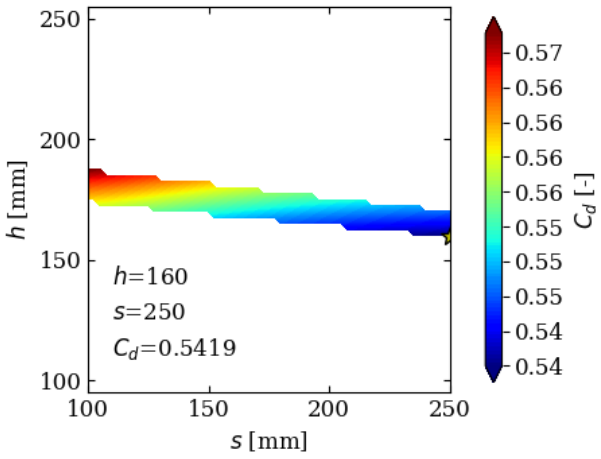
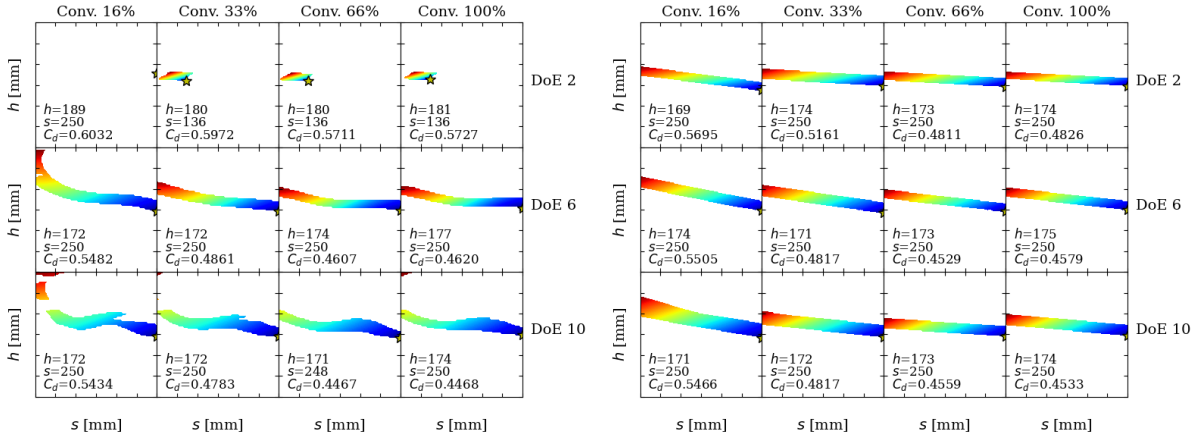


Figure 42 – Constrained optimization results for the MBM data-set. The valid design region is color coded to represent the C_d magnitude, and the optimum point is highlighted with a star icon and explicitly written in the plot.

with the traditional ROM result showing more complex boundary structures. The Transfer Learning model, on the other hand, mimics the generally well-behaved optimization boundaries observed for the MBM model in Figure 42, while achieving optimization results similar to the ROM model, certainly due to the fusion of both data-sets implemented for this approach.



(a) Constrained optimization with the GEKPLS ROM data-set.

(b) Constrained optimization with the Transfer Learning MLP data-set.

Figure 43 – Optimization results for the CFD ROM and TL MLP models, with varied Design of Experiments data-sets at varied convergence points.

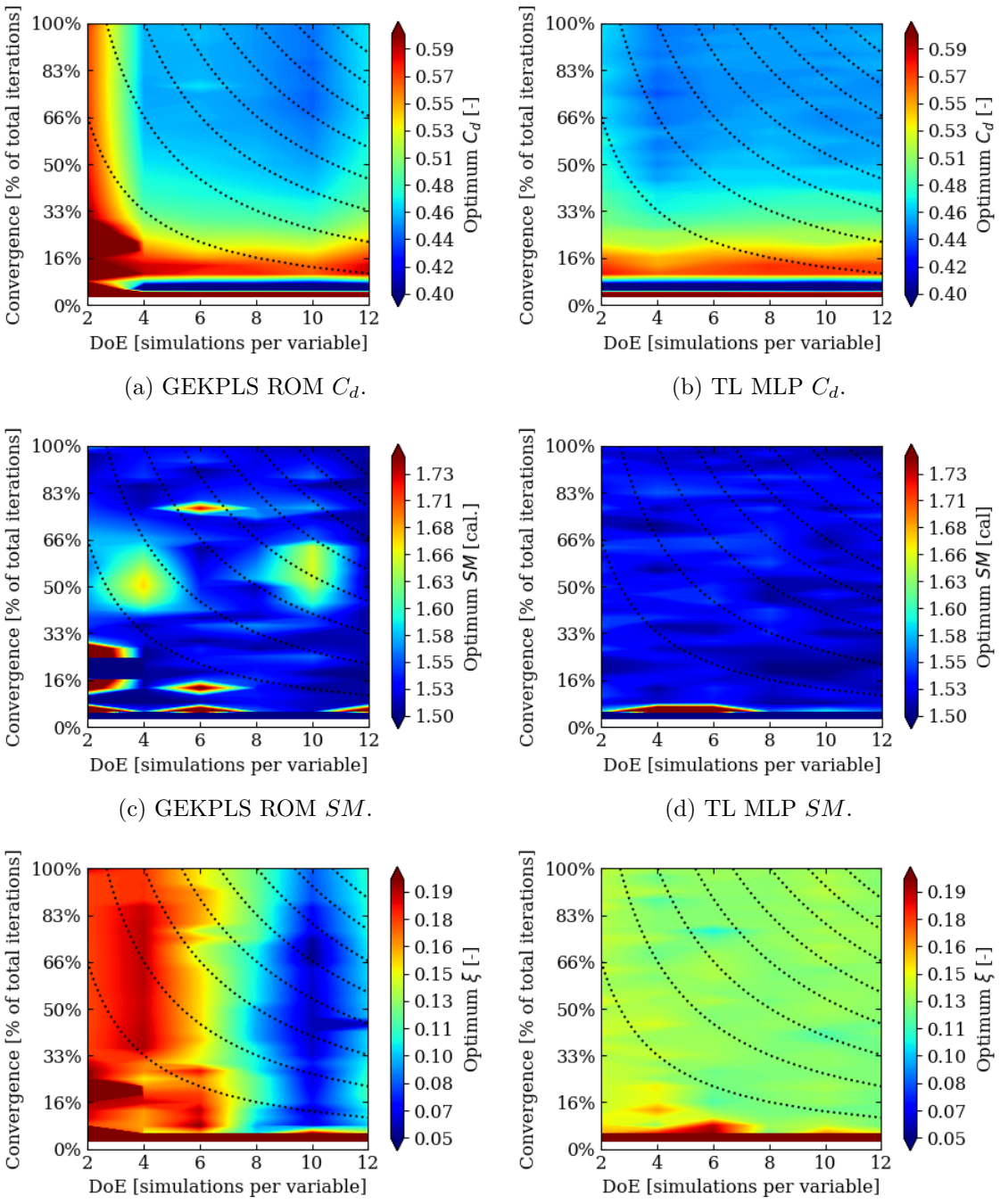
For linear aerodynamic scenarios, highly swept lifting structures with shorter vertical spans are known to generally have lower drag coefficients then rectangular wings. Therefore, the fact that the optimization for most scenarios follows this path of higher s and lower h is a good indication of their physical consistency. Furthermore, as the flow struc-

tures (and the derived stability coefficients) are typically linear for low angles of attack, one would expect that the general linear behaviour observed for the MBM (Figure 42) and TL MLP (Figure 43b) data-sets to also be the more physically reasonable result. The higher order behaviour apparent in the valid optimization space obtained by the GEKPLS ROM data-set could be attributed to the mathematical nature of the response surface created, highly depending on the local gradients in the $h - s$ space.

The difference in the optimum results between the GEKPLS ROM and the TL MLP models is also apparent in Figure 43, especially for the lower DoEs data-sets. To further visualise this, Figures 44 and 45 plot the optimum results of the design coefficients (C_d, SM, ξ) and parameters (h, s) obtained by both models. These combined optimization results are represented as heat-maps in a "computation expense" space, illustrating the convergence history and necessary number of CFD runs (DoE multiplied by the number of design parameters). Iso-lines of computation time are also plotted against the combined results, with longer computation times occurring at the top right corner of the plots (large fully converged DoE sets).

Observing Figures 44 and 45 it is noticeable that the Transfer Learning MLP optimization results are overall smoother then the GEKPLS Reduced Order Model results. In essence this means that, even if trained with a smaller and partially converged data-set, the modelling approach combining semi-empirical and CFD data produces comparable optimum results to the most computationally demanding scenarios (large sets of fully converged CFD points).

Analyzing these combined optimization results, it is clear that the Transfer Learning based approach shows more uniform results along the computation effort spectra. This effect could be attributed to the information coming from the Modified Barrowman Method during the initial training of the Multi Layer Perceptron networks, as this data has no relation with the number of CFD Design of Experiments, nor with the CFD convergence. The TL MLP model does, however, arrive at very similar predictions to the purely CFD based GEKPLS Reduced Order Model, meaning that the information coming form this latter data-set also plays an important role in the Transfer Learning approach. For these reasons, the TL based model seams to be the better alternative for the parametric conceptual aerodynamics design framework developed in this study.



(a) GEKPLS ROM C_d.

(b) TL MLP C_d.

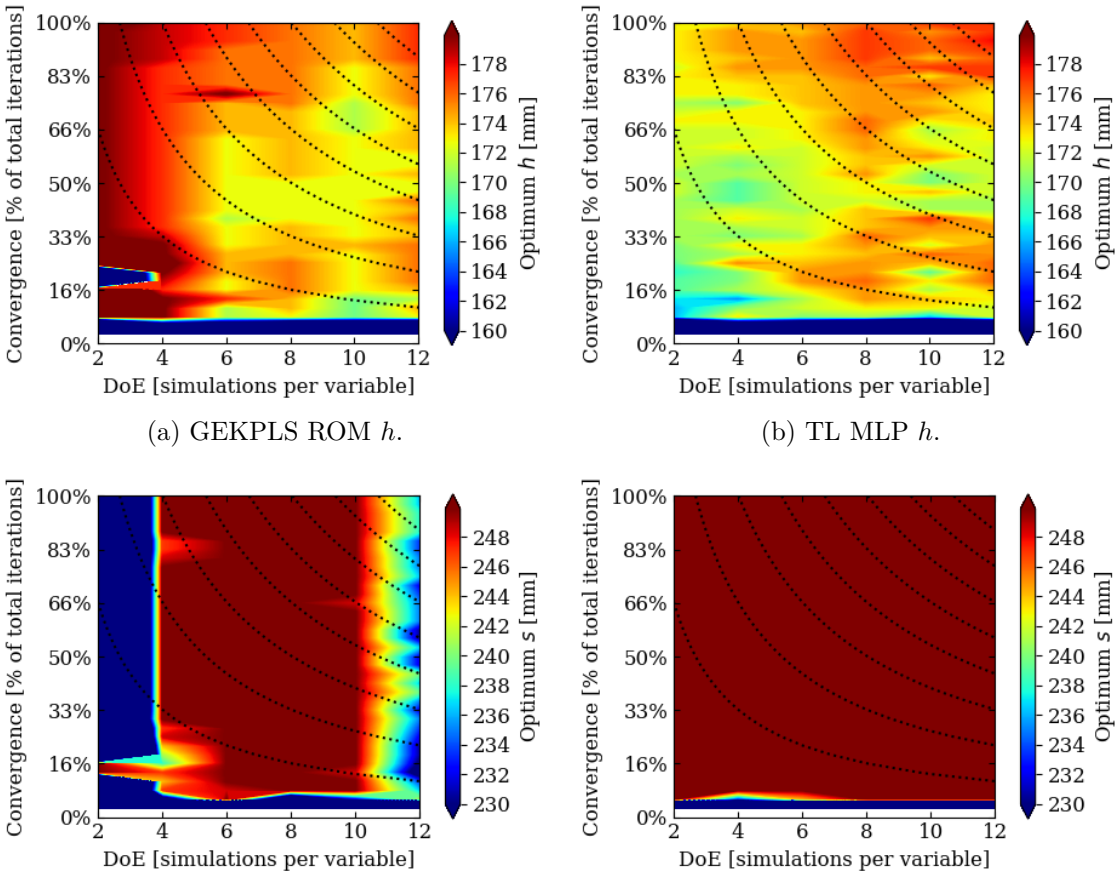
(c) GEKPLS ROM SM.

(d) TL MLP SM.

(e) GEKPLS ROM ξ.

(f) TL MLP ξ.

Figure 44 – Combined design coefficients optimization results for the traditional and Machine Learning based Reduced Order Models created with partially converged CFD and semi-empirical data.



(a) GEKPLS ROM h.

(b) TL MLP h.

(c) GEKPLS ROM s.

(d) TL MLP s.

Figure 45 – Combined design parameters optimization results for the traditional and Machine Learning based Reduced Order Models created with partially converged CFD and semi-empirical data.

5 Conclusions

In this bachelor work a parametric design framework for the conceptual evaluation of a model rocket's aerodynamic and flight stability configuration has been developed. In particular, the use of automated semi-empirical rocketry aerodynamics and of parametric CFD models requiring low computational effort was prioritized; so as to achieve the goal of developing a suitable conceptual design framework that takes restricted computational capabilities into account. In effect, all of the analysis discussed in this work were performed in a readily available personal computer, with 6 processing cores. Furthermore, the use of traditional and Machine Learning based Reduced Order Models constructed with partially converged CFD data was also investigated as a low computational effort parametric design alternative.

A modified version of the Barrowman Method [13] was implemented in python by combining many semi-empirical correlations for a model rocket's aerodynamics and flight stability. This fast, low fidelity, analytical modelling alternative showed a good agreement with reference numerical results, especially for low angles of attack (as shown in Figure 29).

Higher fidelity computational fluid dynamics models were also implemented in the proposed framework. In particular, OpenFOAM parametric CFD models for the transonic aerodynamics of a model rocket were designed. After preliminary investigations of the CFD model's setups in Section 3.2.4, the PBiCGStab numerical scheme (employing High and Low Reynods boundary layer modelling techniques) was further evaluated. The wall-function (High Re) boundary layer model showed comparable results with the more discretized (Low Re) computationally expansive model (as seen in Figures 30 and 34), and was therefore deemed as a suitable CFD approach for the proposed conceptual design framework.

Traditional response surface Reduced Order Models were developed from the parametric CFD results. The open source pySMT library was used to implement the Gradient-enhanced Kriging with Partial Least Squares ROM, built using data-sets with variable CFD points at different convergences. This ROM strategy produced good validation results ($R^2 > 0.95$) for the aerodynamic and stability design coefficients (C_d, SM, and ξ), as seen in Figure 37. A Transfer Learning approach using Multi Layer Perceptrons was also investigated as a means of combining the information from the analytical and numerical models. This Machine Learning based ROM alternative also showed good overall validation results (Figure 41).

Finally, all outlined models were implemented in the aerodynamic optimization problem of the Armação A-22 rocket's fin-set, whose goal was optimizing the h and s parame-

ters to achieve minimum C_d values constrained by stable SM and ξ results. In particular, the TL MLP approach combining both the MBM and partially converged CFD data-sets showed promising optimization results for low computation effort scenarios, as seen in Figures 44 and 45.

Revisiting Figure 13, a feasible suggestion for the outlined framework is presented in Figure 46. Following this framework, the conceptual aerodynamics design for a high power model rocket could be considerably accelerated. If 4 design parameters are selected for optimization, being sampled 4 times per variable (DoE 4), and resolved only to 60% convergence, initial results for the conceptual design phase could be obtained within hours, not days as is currently the case within a usual project life-cycle in Apex Rocketry. This accelerated results can only be achieved thanks to the fusion of the Modified Barrowman Method and partially converged CFD data by means of Transfer Learning.

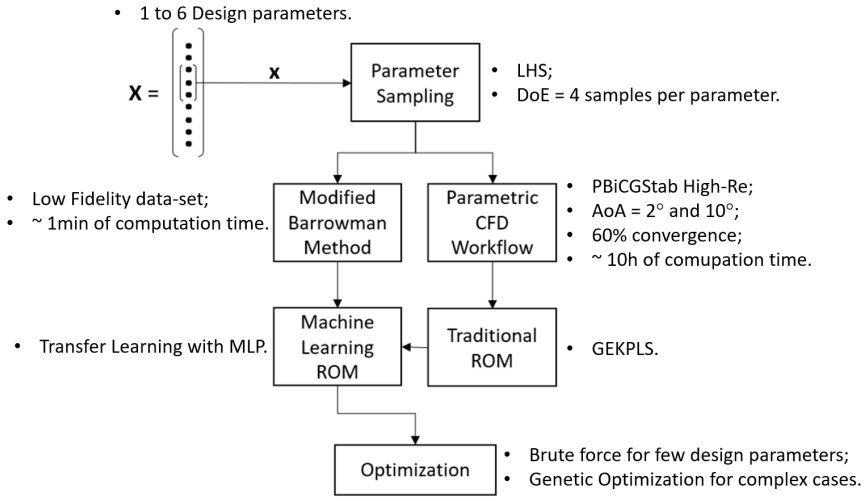


Figure 46 – Suggested implementation of the conceptual aerodynamics design framework combining semi-empirical and partially converged CFD data.

In essence the results presented throughout this work indicate that: Even though it is not possible to precisely ascertain whether or not the outlined procedures are accurate in reaching optimum design parameters; it is observable that the presented techniques can arrive at similar conclusions even with reduced computational effort scenarios. In particular, Transfer Learning ROMs built by fusing MBM and partially converged CFD data were shown to produce uniform optimization results for a large computational effort spectra, meaning that this conceptual design framework could be used with very few, not fully converged, CFD runs, reducing the required project time to arrive at preliminary design conclusions.

References

1 Apex Rocketry. **The Armação Project**: Team 12 project technical report. 2020 LASC Online, 2020. 7, 11, 16, 17, 40, 60, 61

2 NIELSEN, J. N. **Missile Aerodynamics**. McGraw-Hill Book Company, Inc., 1960. 7, 22, 23, 25

3 NISKANEN, S. **Development of an Open Source Model Rocket Simulation Software.** 2009. M.Sc. thesis, Helsinki University of Technology, Available at http://openrocket.sourceforge.net/documentation.html. 7, 22, 23, 25, 26, 42, 43, 45, 67

4 STINE, G. H.; STINE, B. **Handbook of Model Rocketry**, 7^{th} edition. John Wiley & Sons, Inc., 1965. 7, 22, 23

5 FANGQING, L. **A Thorough Description Of How Wall Functions Are Implemented In OpenFOAM**. 7, 30, 53

6 JIN, R.; CHEN, W.; SUDJIANTO, A. **An efficient algorithm for constructing optimal design of computer experiments**. Journal of Statistical Planning and Inference, Elsevier, vol 134, 2005. 7, 33

7 BOUHLEL, M. A.; MARTINS, J. R. R. A. **Gradient-enhanced kriging for high-dimensional problems**. Engineering with Computers, v. 1, n. 35, p. 157–173, jan. 2019. 7, 34, 35

8 BRUNTON, S. L.; NOACK, B. R.; KOUMOUTSAKOS, P. **Machine Learning for Fluid Mechanics**. Annual Review of Fluid Mechanics, Annual Reviews, 2020. 7, 32, 35, 36, 37

9 COUTINHO, B. H. **Aerodynamic Analysis and Testing of Dynamic Pressure Field at a Rocket's Nosecone Surface.** 2020. Trabalho de Conclusão de Curso (Bachalerado em Engenharia Mecânica), Departamento de Engenharia Mecânica, Universidade Federal de Santa Catarina, Florianópolis. 17, 21

10 FORRESTER, A. I. J.; BRESSLOFF, N. W.; KEANE, A. J. **Optimization using surrogate models and partially converged computational fluid dynamics simulations**. Proceedings of the Royal Society A, 2006. 17

11 MI, B.; ZHAN, H. **Review of Numerical Simulations on Aircraft Dynamic Stability Derivatives**. Archives of Computational Methods in Engineering, Springer, 2019. 23, 26

12 MANDELL, G. K. **Model Rocketry Magazine: Fundamentals of Dynamic Stability**. 1968, 1969. Vol. 1, Nos. 1 thru 5. Available at https://www.nar.org/members/magazine-archives/model-rocketry-magazine/. 24, 42, 44

13 BARROWMAN, J. S. **The Practical Calculation of the Aerodynamic Characteristics of Slender Finned Vehicles.** 1967. Msc. Thesis, The

Catholic University of America, Faculty of Aerospace Engineering. Available at https://ntrs.nasa.gov/citations/20010047838. 25, 42, 84

14 GALEJS, R. **Peak Of Flight Magazine: What Barrowman Left Out?** 2018. Issue 470. Available at https://www.apogeerockets.com/education/downloads/Newsletter470.pdf. 25, 26, 42

15 BOX, S.; BISHOP, C. M.; HUNT, H. **Estimating the Dynamic and Aerodynamic Parameters of Passively Controlled High Power Rockets for Flight Simulation**. 2009. Available at http://cambridgerocket.sourceforge.net/AerodynamicCoefficients.pdf. 25, 26, 42, 44

16 HAMMARGREN, K. **Aerodynamics Modeling of Sounding Rockets: A Computational Fluid Dynamics Study**. 2018. Msc. Thesis, Luleå University of Technology, Department of Computer Science, Electrical and Space Engineering. 26

17 ABBAS, L. K.; CHEN, D.; RUI, X. **Numerical Calculation of Effect of Elastic Deformation on Aerodynamic Characteristics of a Rocket**. International Journal of Aerospace Engineering, 2014. 26

18 BARTOWITZ, M. E. **Determination of Static and Dynamic Stability Coefficients Using Beggar**. 2008. Msc. Thesis, Air Force Institute of Technology, Department of Aeronautics and Astronautics. 26

19 WINTER, M. **Benchmark and validation of Open Source CFD codes, with focus on compressible and rotating capabilities, for integration on the SimScale platform.** 2013. Msc. Thesis in Engineering Mathematics Computational Sciences, Chalmers University of Technology, Sweden. 27

20 HOLZINGER, G. **OpenFOAM: A little User-Manual**. 2018. Freely distributed software manual available at: https://usermanual.wiki/Pdf/openFoamUserManualPFM.954010286.pdf. 28, 51, 54, 55

21 PALART, P.; ALLMARAS, S. **A One-Equation Turbulence Model for Aerodynamic Flows**. La Recherche Aeropostiale, 1994. 29

22 MENTER, F. **Two-Equation Eddy-Viscosity Turbulence Models for Engineering Applications**. AIAA Journal, 1994. 29, 30

23 ARAYA, G. **Turbulence Model Assessment in Compressible Flows around Complex Geometries with Unstructured Grids**. Fluids, MDPI, 2019. 29

24 MORINO, L.; BERNARDINI, G.; MASTRODDI, F. **Multi–Disciplinary Optimization for the Conceptual Design of Innovative Aircraft Configurations**. Proceedings of the 2004 International Conference on Computational & Experimental Engineering Science, 2004. 32

25 MENDONçA, G.; AFONSO, F.; LAU, F. **Model order reduction in aerodynamics: Review and applications**. Journal of Aerospace Engineering, Proc IMechE Part G, 2019. 32, 62

26 YAN, X. et al. **Aerodynamic shape optimization using a novel optimizer based on machine learning techniques**. Aerospace Science and Technology, Elsevier, 2019. 32, 38

27 WILLARD, J. D. et al. **Integrating Physics-Based Modeling with Machine Learning: A Survey**. Association for Computing Machinery, abs/2003.04919, 2020. 32

28 BOUHLEL, M. A. et al. **A Python surrogate modeling framework with derivatives**. Advances in Engineering Software, p. 102662, 2019. ISSN 0965-9978. 33, 62

29 BIANCOLINI, M. E. **Fast Radial Basis Functions for Engineering Applications**. 33

30 LUI, H. F. da S. **Construction of reduced order models for fluid flows using deep neural networks**. 35, 48

31 FLEEMAN, E. L. **Tactical Missile Design.** American Institute of Aeronautics and Astronautics (AIAA), Inc., 2001. 42, 43

32 LABUDDE, E. V. **Extending The Barrowman Method For Large Angles Of Attack**. NARCON (National Association of Rocketry Conference), 1999. 42, 45

33 VALLINI, L. **Static and Dynamic Analysis of the Aerodynamic Stability and Trajectory Simulation of a Student Sounding Rocket**. 2013/2014. Final Course Paper, Corso de Laurea Magistrale in Ingegneria Aerospaziale, Facoltà di Ingegneria, Università di Pisa. 43

34 GOHDE, J. **Programming with OpenSCAD: A Beginner's Guide to Coding 3D-Printable Objects**. No Starch Press, 2021. Available at: https://programmingwithopenscad.github.io/,. 47

35 JURETIć, F. **cfMesh User Guide** document version: 1.1. Creative Fields, Ltd, 2015. Available at: https://cfmesh.com/,. 48

36 GOETTEN, F. et al. **A review of guidelines and best practices for compressible aerodynamic simulations using RANS CFD**. Asia Pacific International Simposium on Aerospace Technology, 2019. 49, 52

37 HEYNS, J.; OXTOBY, O. **Modelling high-speed viscous flow in OpenFOAM**. 9th South African Conference on Computational and Applied Mechanics, SACAM 2014, 01 2014. 51

38 ROACHE, P. **Quantification of Uncertainty in Computational Fluid Dynamics**. Annual Review of Fluid Mechanics, v. 29, 1997. 60

39 PASZKE, A. et al. **PyTorch: An Imperative Style, High-Performance Deep Learning Library**. In: WALLACH, H. et al. (Ed.). Advances in Neural Information Processing Systems 32. Curran Associates, Inc., 2019. p. 8024–8035. Disponível em: <http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>. 64, 65