UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS

Guillermo Alexander Krauch Caballero

Development of a program for the multidisciplinary optimization of a single stage rocket

São Carlos

2021

Guillermo Alexander Krauch Caballero

Development of a program for the multidisciplinary optimization of a single stage rocket

Monografia apresentada ao Curso de Engenharia Aeronáutica, da Escola de Engenharia de São Carlos da Universidade de São Paulo, como parte dos requisitos para obtenção do título de Engenheiro Aeronáutico.

Supervisor: Prof. Dr. Alvaro Martins Abdalla

São Carlos

2021

Ficha catalográfica elaborada pela Biblioteca Prof. Dr. Sérgio Rodrigues Fontes da EESC/USP com os dados inseridos pelo(a) autor(a).

```
        Krauch, Guillermo Alexander
K91d       Desenvolvimento de um programa para otimização
        multidisciplinar de um foguete de um estágio /
        Guillermo Alexander Krauch; orientador  Alvaro Martins
        Abdalla. São Carlos, 2021.


           Monografia (Graduação em Engenharia Aeronáutica)
        -- Escola de Engenharia de São Carlos da Universidade
        de São Paulo, 2021.


           1. otimização multidisciplinar. 2. arrasto
        supersônico. 3. propulsão de foguetes. I. Título.
```

Eduardo Graziosi Silva - CRB - 8/8907

FOLHA DE APROVAÇÃO

Candidato: Guillermo Alexander Krauch Caballero
Título do TCC: Desenvolvimento de um programa para otimização multidisciplinar de um foguete de um estágio
Data de defesa: 10/09/2021

Comissão Julgadora	**Resultado**
Professor Doutor Hernan Dario Ceron Muñoz	APROVADO
Instituição: EESC - SAA	
Professor Associado Paulo Celso Greco Júnior	Aprovado
Instituição: EESC - SAA	

Presidente da Banca: Professor Doutor Hernan Dario Ceron Muñoz

[illegible]

(assinatura)

ACKNOWLEDGEMENTS

Firstly I would like to thank my parents, my brothers and all of my family for their unconditional love and support, their faith in my capacity to reach higher challenges and their great examples to follow. I surely would not be able to be half of who I am today if it was not for you.

I would also like to thank professor Abdalla, for presenting me to the idea of the project when it was only at embryonary stage, for his support, guidance and patience during its development and for his input on the different complex topics throughout the project. This dissertation would certainly be a lot shorter and less structured if it wasn't for him.

I'd also like to thank my girlfriend, Mariana, whose support throughout the project and loving patience gave me the focus and drive to get to this point.

To all classmates and friends at the Aeronautical Engineering course and to all my teammates at EESC-USP Aerodesign. Thank you for showing me what it is to truly work as a team, to help each other sharing glories and falls, to persevere and thrive against the adversity and to do something with love giving one's best. I surely learned some of the most valuable lessons with you while working and learning until morning, sharing moments and experiences that will forever mark me.

To everyone that loves air and space, to everyone who dares go beyond what was previously thought possible, to everyone who is brave enough to challenge our believes and to help humanity reach higher limits. Thank you for inspiring me, thank you for giving your grain (or sometimes truckload) of salt to grow humanity's future. Ad Astra!

In the loving memory of Juan Rafael Caballero Morínigo, Pastora Riera de Krauch and Axel Perán Schupmann.

ABSTRACT

Krauch, G. A. **Development of a program for the multidisciplinary optimization of a single stage rocket**. 2021. 113p. Monografia (Trabalho de Conclusão de Curso) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2021.

The space economy is a rapidly growing and developing market, with players entering it at a fast pace. Smaller and medium players rely on nano-satellites as an entering platform, but there's a growing concern for higher reliability and future regulation. Sounding rockets prove a useful platform to safely test and develop these technologies, as well as to conduct other research. In this work, we study the creation of an optimization algorithm for a single-stage sounding rocket that carries a payload to a given altitude, simulated using a one degree of freedom system. Modules of propulsion and aerodynamic drag are the main focus, using simplified analyses to minimize execution time. A solution obtained for a test case of launching a 3U CubeSat sized payload to a 150km altitude is presented and discussed. Other applications of the software are briefly discussed.

Keywords: multidisciplinary optimization, rocket propulsion, supersonic drag

RESUMO

Krauch, G. A. **Development of a program for the multidisciplinary optimization of a single stage rocket**. 2021. 113p. Monografia (Trabalho de Conclusão de Curso) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2021.

A economia do espaço é um mercado em rápida expansão e desenvolvimento, com atores entrando em um ritmo acelerado. Pequenos e medianos atores apostam em nano-satélites como a plataforma de entrada, mas existe uma preocupação crescente sobre confiabilidade e futuras regulações de espaçonaves. Os foguetes de sondagem se mostram uma plataforma útil para testar e desenvolver estas tecnologias, assim também como conduzir várias outras pesquisas. Neste trabalho, estudamos a criação de um algoritmo de otimização multidisciplinar para um foguete de estágio único que leva uma carga até uma determinada altitude, simulado usando um sistema de um grau de liberdade. Os módulos de propulsão e arrasto aerodinâmico são o foco principal, utilizando análises simplificadas para minimizar o tempo de execução. Uma solução obtida para o caso modelo de lançar um satélite de 3U a uma altitude de 150km e apresentada e discutida. Outras aplicações para o programa desenvolvido também são apresentadas.

Palavras-chave: otimização multidisciplinar, propulsão de foguetes, arrasto supersônico

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

CG	Center of Gravity / Center of Mass
CFD	Computational fluid dynamics
DOF	Degrees of Freedom
LEO	Low Earth Orbit
MAC	Medium Aerodynamic Chord
USD	United States Dollars

LIST OF SYMBOLS

A_B	Burning area
A_c	Combustion chamber's cross-sectional area
A_e	Nozzle's exit cross-sectional area
A_t	Nozzle's throat cross-sectional area
A_i	Nozzle's cross-sectional area at a point i
A_n	Fourier series n^{th} grade sine coefficient
a_r	Burning rate a coefficient
C_D	Total drag coefficient
C_D^{SUB}	Profile drag coefficient
C_D^{SUB}	Wave drag coefficient
$C_{DL\&P}$	drag coefficient due to leakages and protuberances
C_{Dmisc}	Drag coefficient due to miscellaneous components
C_D^{PAR}	Parachute's drag coefficient
C_{fc}	Components friction coefficient
c_{sound}	Speed of sound
D	Total aerodynamic drag
d_{MAX}	Objects maximum diameter
d_C	Combustion chamber's diameter
dt	Time finite differential
F	Rocket Thrust
FF_c	Component's form factor
F^*	Part of the rocket thrust that only considers effects of mass ejection
f_c	factor in calculations of form factor
f_s	safety factor for motor calculation

g	Local gravity
g_0	Gravity at sea level
h	Altitude variable
$\dot{h}$	Altitude’s first derivative over time / Speed in the altitude axis
$\dot{h}$	Altitude’s second derivative over time / Acceleration in the altitude axis
h_{ref}	Reference altitude, goal of the optimization
h_{max}	Maximum altitude reached by rocket
I_t	Total impulse
I_{sp}	Specific impulse
K_i	Runge-Kutta’s method i coefficient
k_c	component’s finishing roughness
l_C	Length of the cylinder composing the combustion chamber
M_R	Mass ratio
M_{gas}	Gas’ molar weight
M_{ach}	Mach number
M_{ach}^{cr}	Critical Mach number
M_{ach}^{dd}	Divergence Mach number
MAC_{fin}	Fin’s Medium aerodynamic chord
MS_{min}	Minmum static margin
m	Mass (at any given time)
$\dot{m}$	Mass flow
m_f	Final mass / Empty mass
m_p	Propellant mass
n_r	Burn rate n coefficient
P_{AMB}	Medium’s ambient pressure
P_e	Nozzles exhaust pressure

$P_{i,x,y}$	Nozzles pressure on a given point i, x or y
P_0	Nozzles stagnation pressure
P_C	Combustion chamber's pressure
p_{mut}	
Q_c	Component's interference factor
R_u	Ideal gas universal constant
R_{spec}	Ideal gas specific constant
R_e	Reynolds number
$R_{ecutoff}$	Cutoff Reynolds number
r_B	Burning rate
r_{earth}	Earth's radius
r_p	point's radial coordinate
S_{wetc}	Component's wetted surface
S_{REF}	Reference area
$S(x)$	area intersected by a M_{ach} plane crossing the object's main axis on the x coordinate projected in the flow direction
$T_{i,x,y}$	Nozzle's temperature on a given point i, x or y
T_0	Nozzle's stagnation temperature
T_C	Combustion chamber's temperature
ToW	Thrust-over-Weight ratio
t_0	time of ignition
$t_{burnout}$	time when the combustion stops
t_{wall}	motor's casing wall thickness
$\frac{t}{c}$	Airfoil's maximum thickness relative to chord
t	time variable
th_{fin}	Dimensional fin thickness

T_{AMB}	Medium temperature
v_e	Nozzle exhaust velocity
$v_{i,x,y}$	Nozzle flow speed in a given point i, x or y
V	Object's velocity
V_{ter}	Object's terminal velocity
$x^rel_{th_{max}}$	fraction of chord where the maximum thickness of the airfoil occurs
$\overline{x}$	.
x_0	.
x_{plane}	x coordinate of the intersection of a M_{ach} plane with the rocket's main coordinate axis
x_p	point's coordinate on the rocket's main axis
y	check
γ	Specific heat ratio
θ	angle of rotation of the M_{ach} plane
ρ	check
ρ_{AMB}	Medium's density
ρ_p	Propellant density
μ_{AMB}	Medium's viscosity
Λ_m	Sweep angle of the maximum thickness line on an aerodynamic surface
λ_{thrust}	thrust correction factor
σ_y	material's yield stress
ℓ_c	components characteristic length

CONTENTS

1 INTRODUCTION

According to NASA the term "sounding rocket" derived from the analogy to maritime soundings made of the ocean depths. It describes rockets that carry instruments into the upper atmosphere to investigate its nature and characteristics, gathering data from meteorological measurements at altitudes as low as 32 km to data for ionospheric and cosmic physics at altitudes up to 6400 km. Sounding rockets also flight-test assemblies that will be sent to space (WELLS; WHITELEY; KAREGEANNES, 1975).

The focus of this project will be to analyze the different factors taken into account on the initial phases of the conceptual project of a sounding rocket, in order to create an algorithm capable to simulate different configurations and optimize the rocket for a desired goal. Throughout this project, special care will be taken so that the resulting algorithms are easy to adapt for future uses and expansions.

1.1 Motivation

With the growth of the space economy, many private players (large, medium and small) have started to flow into this market. The space economy is rapidly growing and could achieve a yearly revenue of over 1 trillion USD, compared to its current 350 billion USD. (MORGAN-STANLEY, 2020)

With these new smaller players, a new wave of component suppliers focused in this segment can be expected to rise. A common concept in current small players in the space economy is the use of a scalable NanoSat network, growing steadily as the service gains momentum [ref]. These new components and assemblies will need test to be deemed safe and deployable. While there are no currently widespread regulations for the launch of small satellites, market best practices encourage extensive testing campaigns and many players are already calling for a stricter test regulation in order to avoid the Chain Crash Orbital effect. (ROETGEN; DESCH, 2020)

In this context the mission of this project can be shortly defined as "to create an optimization program that helps give the first steps on the conceptual project of a suborbital launch vehicle, with a target group on the test of nano satellites and nano satellite’s components".

1.2 Mission Definition

A usual approach for sounding rocket optimization research is defining a reference altitude that must be achieved by the rocket and then optimizing the desired variables to minimize one specific magnitude (ADAMI; MORTAZAVI; NOSRATOLLAHI, 2016)

(OKNINSKI, 2017) (BARBOSA; GUIMARãES, 2012). Thus, the mission will be defined as to reach a target altitude h_{ref} optimizing the total launch mass of the vehicle.

Considering the wide range of altitudes for sounding rocket operations described on (WELLS; WHITELEY; KAREGEANNES, 1975), considering a realistic value for h_{ref} will make a significant difference in project complexity. ESA defines LEO as an orbit that is relatively close to Earth's surface. It is normally at an altitude of less than 1000 km but could be as low as 160 km above Earth (ESA, 2020). Most new players in the sounding rocket market focus their attention to the altitude range between 100km and 300km(OKNINSKI, 2017). Thus, a reference altitude of 150km will be defined as the target in order to test LEO components' launch and environmental resistance. As a secondary objective, an estimation of the time of the trajectory spent in micro-gravity will be given in order to study possible scientific payloads. While micro-gravity can be defined, using a strict definition, as an acceleration lower than one millionth of the gravity on Earth's surface. 1% of the gravity on Earth's surface, which is considered enough to do many experiments, will be the loose definition considered (ROGERS; VOGT; WARGO, 1997).

1.3 Payload Definition

In order to define the payload that will be used as reference a study on the frequency of use of different payload standards was conducted. According to the NanoSat and CubeSat Database (KULU, 2021), the number of registered NanoSatellites in the past 18 years, over 90% of those are some variation of the CubeSat basic model. As shown in Figure 1 the predominant design is the 3U Cubesat totaling almost 54% of the produced units. The open-source approach, small volumes and the growing modularity make the CubeSat the design standard for small enterprises looking to enter the space economy. The use of the CubeSat concept as a platform for small players on the space industry is on the rise (NAGEL; NOVO; KAMPEL, 2020).

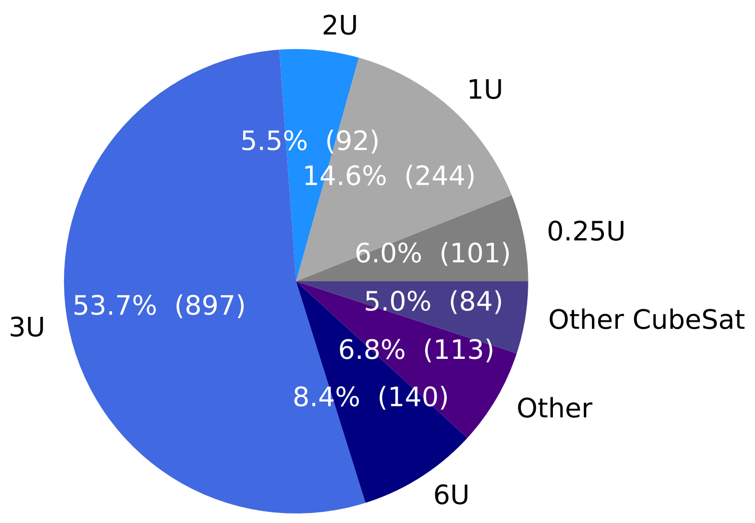


Figure 1 – Configurations of Nano-Satellites launched between 2003 and 2021, synthesized from data available in (KULU, 2021)

The CubeSat standard was created by California Polytechnic State University, San Luis Obispo and Stanford University's Space Systems Development Lab in 1999 to facilitate access to space for university students. Since then the standard has been adopted by hundreds of organizations worldwide. The CubeSat standard facilitates frequent and affordable access to space with launch opportunities available on most launch vehicles.(THE CUBESAT PROGRAM,, 2021?)

The usual CubeSat configurations are specified as 1U, 1.5U, 2U, 3U, 6U e 12U, as can be seen in Figure 2. The masses and tolerances for *CG* for each configuration can be found in table 1.

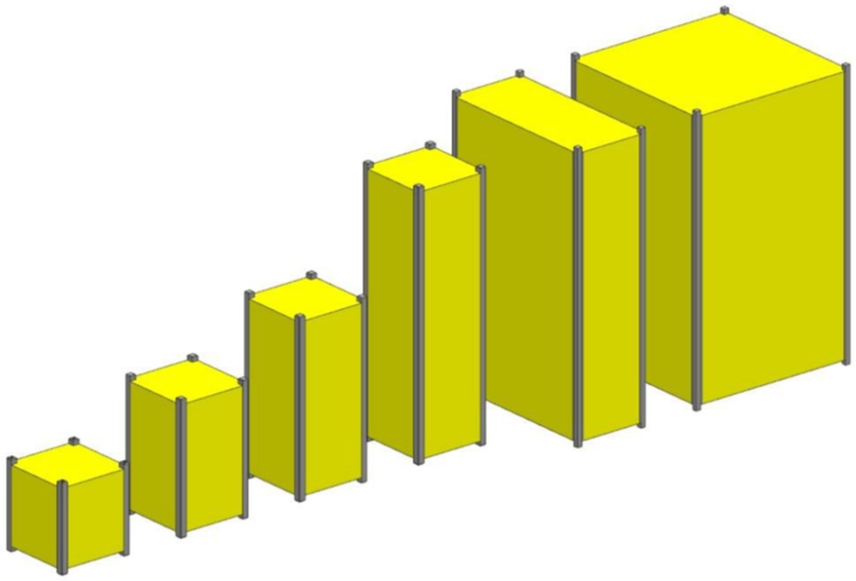


Figure 2 – Different U Configuration of the CubeSat Concept, extracted from (THE CUBESAT PROGRAM,, 2020)

U Configuration	Mass [kg]	Ranges of acceptable center of gravity		
		X Axis	Y Axis	Z Axis
1U	2	+ 2 cm / -2 cm	+ 2 cm / -2 cm	+ 2 cm / -2 cm
1.5U	3	+ 2 cm / -2 cm	+ 2 cm / -2 cm	+ 3 cm / -3 cm
2U	4	+ 2 cm / -2 cm	+ 2 cm / -2 cm	+ 4.5 cm / -4.5 cm
3U	6	+ 2 cm / -2 cm	+ 2 cm / -2 cm	+ 7 cm / -7 cm
6U	12	+ 4.5 cm / -4.5 cm	+ 4.5 cm / -4.5 cm	+ 7 cm / -7 cm
12U	24	+ 4.5 cm / -4.5 cm	+ 4.5 cm / -4.5 cm	+ 7 cm / -7 cm

Table 1 – Typical Mass for each U configuration with Ranges of acceptable center of gravity locations as measured from the geometric center on each major axis, extracted from (THE CUBESAT PROGRAM,, 2020)

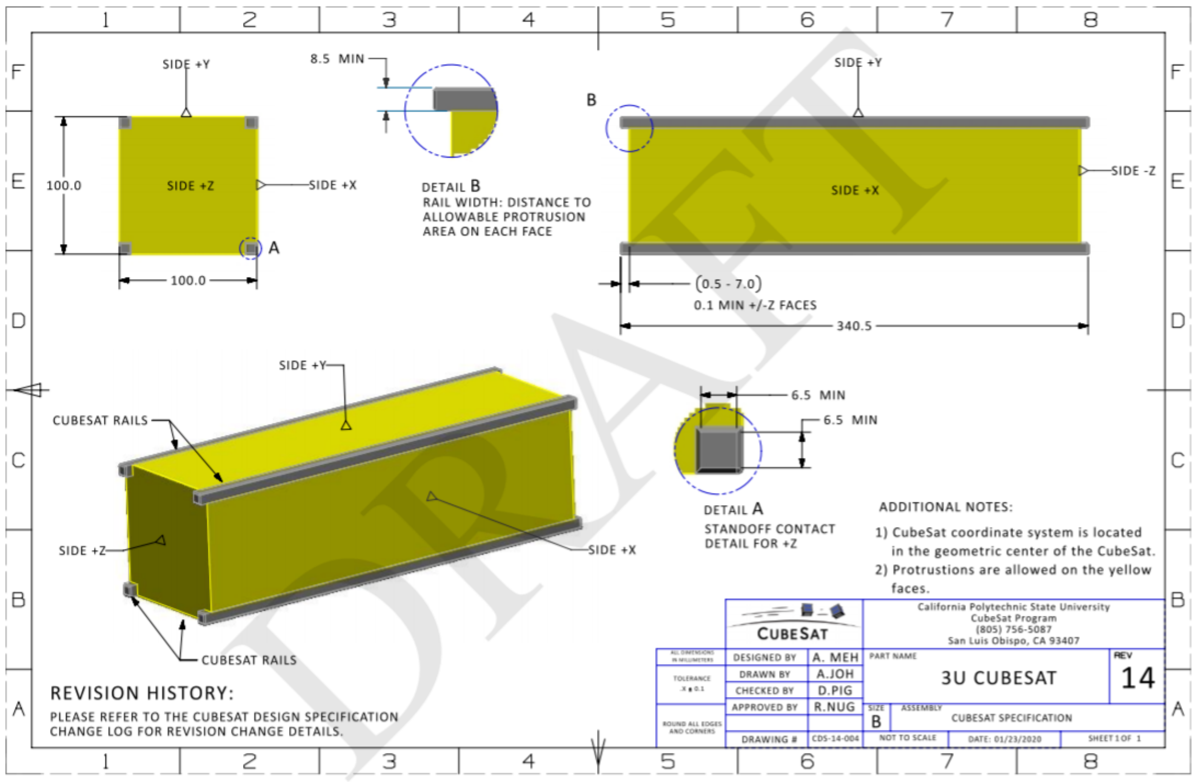


Figure 3 – Technical schematics and sizing from the 3U CubeSat, extracted from (THE CUBESAT PROGRAM,, 2020)

Thus, it can be concluded that the 3U Cubesat design is the most fit candidate for a model payload as it is the most commonly used by the market. With its dimensions presented Figure 3.

1.4 Objectives

The objective of this project is to develop an optimization algorithm for the optimization of the geometric parameters for a suborbital research rocket. The optimization parameter will be the launch weight of the rocket, considering a fixed payload and altitude to attain. The focus areas of the project will be: aerodynamics (focused in booth subsonic and supersonic drag estimation), performance (with focus on thrust and motor mass estimation) and stability (with focus on the subsonic static stability of the rocket for fin sizing), other areas (as materials and numerical methods) will be explored to facilitate a more realistic simulation. The expected result is a highly customizable algorithm that facilitates decision-making in early project stages.

A secondary objective will be defined as to use the algorithm developed to optimize a sounding rocket for a payload of a 3U CubeSat to reach an altitude of 150km with the least inital mass possible.

2 BACKGROUND

With the objective of having a better understanding of the topics within the scope of the project, the theoretical background of rocket launch was studied. The focus of this was to understand the main phenomena that affect rocket trajectory a to obtain models simple enough to be implemented and avoid excessive computational costs yet accurate enough to reflect reality. Therefore, the main pillars of this study were Modern Sounding Rocket architecture, rocket aerodynamic drag, rocket propulsion, rocket stability, rocket materials and structures and optimization and numerical solution methods.

2.1 Modern Sounding Rocket Architecture

As a first a approach to the topic, different sounding rocket designs were studied to understand key common components that to be studied in detail later. Modern sounding rockets vary significantly depending on mission and the payload carried (NASA SOUNDING ROCKETS PROGRAM OFFICE, 2015). Sounding rockets for small payloads have a "slender and long shape", having a length to diameter ratio of 5-20 (PALLONE et al., 2018), (OKNINSKI, 2017) claims this number to goes up to 30. Most of the examples cited in this Section fall within this category.

In both professional and amateur rockets, the following components can be identified, a hull and internal structural components, a nose cone, the payload, internal telemetry systems, fins, a propulsion system and a recovery system (ESA, 2013?),(BUSSE; LEFFLER, 1997), (CYCLONE ROCKETRY CLUB, IOWA STATE UNIVERSITY, 2018), (KALRA et al., 2018). Nose fineness, defined as it's length divided by it's maximum diameter can be up to 6 (OKNINSKI, 2017). Different nose formats are parabolic ogives, cones, paraboloids, ellipsoids, between others (CROWELL, 1996).

2.2 Aerodynamic Drag

The following definitions can be found in (HOUGHTON; CARPENTER, 2003).Total Drag is formally defined as the force corresponding to the rate of decrease in momentum in the direction of the undisturbed external flow around the body, this decrease being calculated between stations at infinite distances upstream and downstream of the body. Thus it is the total force or drag in the direction of the undisturbed flow. It is also the total force resisting the motion of the body through the surrounding fluid. The drag coefficient can be defined as Equation 2.1.

$$C_D = \frac{D}{\frac{1}{2}\rho_{AMB} \cdot V^2 \cdot S_{REF}} \tag{2.1}$$

The value of S_{REF} is defined by the context of the situation as a geometric area easy to correlate to the studied body. In airplanes this value usually refers to the wing-area, while in rockets the value is usually used as its maximum section area. Usually this value is defined as in Equation 2.2.

$$S_{REF} = \frac{\pi \cdot d_{MAX}^2}{4} \tag{2.2}$$

There are a number of separate contributions to total drag. As a first step it may be divided into pressure drag and skin-friction drag.

1. **Skin-Friction Drag**:
   is the drag that is generated by the resolved components of the traction due to the shear stresses acting on the surface of the body. This traction is due directly to viscosity and acts tangentially at all points on the surface of the body. At each point it has a component aligned with but opposing the undisturbed flow (i.e. opposite to the direction of flight). The total effect of these components, taken (i.e. integrated) over the whole exposed surface of the body, is the skin-friction drag. It could not exist in an invisicid flow.

2. **Pressure Drag**:
   is the drag that is generated by the resolved components of the forces due to pressure acting normal to the surface at all points. It may itself be considered as consisting of several distinct contributions:

   a) Induced Drag (also known as vortex drag))

   b) Wave Drag

   c) Form Drag (also known as boundary layer drag)

   While induced drag depends on purely on lift generation and wave drag is the drag associated with the formation of shock waves in high-speed flight, the definition of the form drag is more complex. We can define profile drag the drag due to the losses in total pressure and total temperature in the boundary layers. Using this definition, form drag is defined as the difference of profile drag and skin-friction drag.

Induced drag, is outside this project's scope as it will be shown further on in this work, the implemented model will have 1 DOF and thus no lift generation will be assumed. Thus, Equation 2.1 can be expressed as Equation 2.3 for a zero-lift body.

$$D = \frac{1}{2} \cdot \rho_{AMB} \cdot V^2 \cdot S_{REF} \cdot \left(C_D^{SUB} + C_D^{SUP}\right) \tag{2.3}$$

While different methods for aerodynamic simulations and drag estimation exist e.g. vortex lattice, finite volume methods (usually known as CFD), etc. Most of the later are based on numerical methods and require either considerable computational power, significant calculation time or the use of external software or even all of the previous.(HOUGHTON; CARPENTER, 2003). Priority will be given to the study of simplified methods that can accurately estimate drag in different conditions and can be easily implemented into the program.

2.2.1 Profile Drag

A good estimation of the zero-lift profile drag of an aircraft can be obtained by the components method presented in (RAYMER, 2004), which will be the base reference for the next section. This method uses the drag coefficient of a flat plate obtained by semi-empirical formulas and assumes it applied on the wet area of the different components of the aircraft.

$$C_D^{SUB} = \frac{\sum(C_{fc} \cdot FF_c \cdot Q_c \cdot S_{wetc})}{S_{REF}} + C_{Dmisc} + C_{DL\&P} \tag{2.4}$$

By this method, the total drag of an object is the sum of the drag of each component, given by the product of the friction coefficient, the component's form factor, the interference factor and its wetted area. As no miscellaneous external items nor leakages or protuberances are present in the rocket architectures discussed in Section 2.1, the terms outside the fraction in 2.4 will not be studied in detail.

2.2.1.1 Friction Coefficient

The friction coefficient of a surface can be modeled after a flat plate. For laminar flow, c_f can be calculated with Equation 2.5 and for turbulent Equation 2.6. The resulting c_f should be a value proportional to how much of the wetted surface is covered in either laminar or turbulent flow.

$$c_{fc} = \frac{1.328}{\sqrt{R_e}} \tag{2.5}$$

$$c_{fc} = \frac{0.455}{(log_{10} R_e)^{2.58} + (1 + 0.144 \cdot M_{ach}^2)^{0.65}} \tag{2.6}$$

$$R_e = \frac{\rho_{AMB} \cdot V \cdot \ell_c}{\mu_{AMB}} \tag{2.7}$$

It's important to note that the R_e used are the ones relevant to each component, so the value of ℓ_c for the fuselage should be its length, while for the fin it should be its MAC. Most current current aircraft have turbulent flow over virtually the entire wetted surface, so this will be assumed true for the rockets simulated. For elevated speeds, the value of R_e must be the minimum between the result of Equation 2.7 and the result of using either Equation 2.8 or Equation 2.9, depending on whether flow is subsonic or supersonic/transonic respectively.

$$R_{ecutoff} = 38.21 \cdot (\ell_c/k_c)^{1.053} \tag{2.8}$$

$$R_{ecutoff} = 44.62 \cdot (\ell_c/k_c)^{1.053} \cdot M_{ach}^{1.16} \tag{2.9}$$

2.2.1.2 Form Factor

The form factor equation for wing, tails, and consequently fins is given in Equation 2.10

$$FF_c = \left[1 + \frac{0.6}{x^r el_{th_{max}}}\left(\frac{t}{c}\right) + 100\left(\frac{t}{c}\right)^4\right]\left[1.34 M_{ach}^{0.18}(\cos\Lambda_m)^{0.28}\right] \tag{2.10}$$

The form factor equation for the fuselage is given in Equation 2.11, where f is given by Equation 2.12

$$FF = \left(1 + \frac{60}{f_c^3} + \frac{f_c}{400}\right) \tag{2.11}$$

$$f_c = \frac{\ell}{d_{MAX}} \tag{2.12}$$

For supersonic speeds, the equations presented for the form factor are no longer valid and a value of 1 should be used.

2.2.1.3 Interference factor

For components as fuselage or wings, the $Q_c = 1.0$ and for tail components $1.03 < Q_c < 1.08$. Thus, the effects of the interference factor will be unconsidered.

2.2.2 Wave Drag

Supersonic flow has been studied for a long time. The first solution for the supersonic flow over a cone was obtained by A. Busemann in 1929. While some methods for the study of supersonic flow around are given, e.g. supersonic linearized theory for the shock-expansion method and compressible CFD methods, they either aren't fit for the simulation of a rocket body or are computationally costly to implement (HOUGHTON; CARPENTER, 2003).

(JONES, 1953) provides a comprehensive technique to estimate wave drag for a wing-body systems. At subsonic speeds the pressure drag arising from the thickness of the body or wings is negligible so long as the shapes are sufficiently well streamlined to avoid flow separation. At supersonic speeds this tolerance, which was permitted the designer, disappears and the drag becomes sensitive to the shape and arrangement of the bodies.

2.2.2.1 Estimation of Wing-Body Wave Drag for Supersonic Speeds

The theory presented in (JONES, 1953) is an application of the linear supersonic flow theory presented in (HAYES, 1947). Here, it was shown that when $M_{ach} = 1$ is approached from above, the resulting drag depends only on the longitudinal area distribution over the longitudinal coordinate.

$$S'(x) = \sum_{n=1}^{\infty} A_n \cdot sin(n \cdot \phi) \tag{2.13}$$

$$D = \frac{\pi \cdot \rho_{AMB} \cdot V^2}{8} \sum_{n=1}^{\infty} n \cdot A_n^2 \tag{2.14}$$

Therefore, for a given length, drag minimization can be achieved when the $S'(x)$ is has the least possible harmonics. As A_1 and A_2 have a relation to the Base area and the volume of the body, thus, for a same length and volume, the minimum drag is achieved when the resulting series only contains these terms.

In order to extend this conclusions beyond sonic speed, we need to study the cross section of the system with oblique planes inclined at the Mach angle, as the disturbance of the system affects the whole region. Thus, parallel inclined planes "cut" the system in different cross section areas which allow the construction of an equivalent $S(x)$ curve where formulations 2.13 and 2.14 can be applied, as shown in Figure 4. In order to superimpose the effect of the infinite oblique planes that can be inclined at the Mach angle in relation to the system's longitudinal axis, the final drag value can be obtained as the average of the drag values obtained through a complete rotation of the Mach planes.

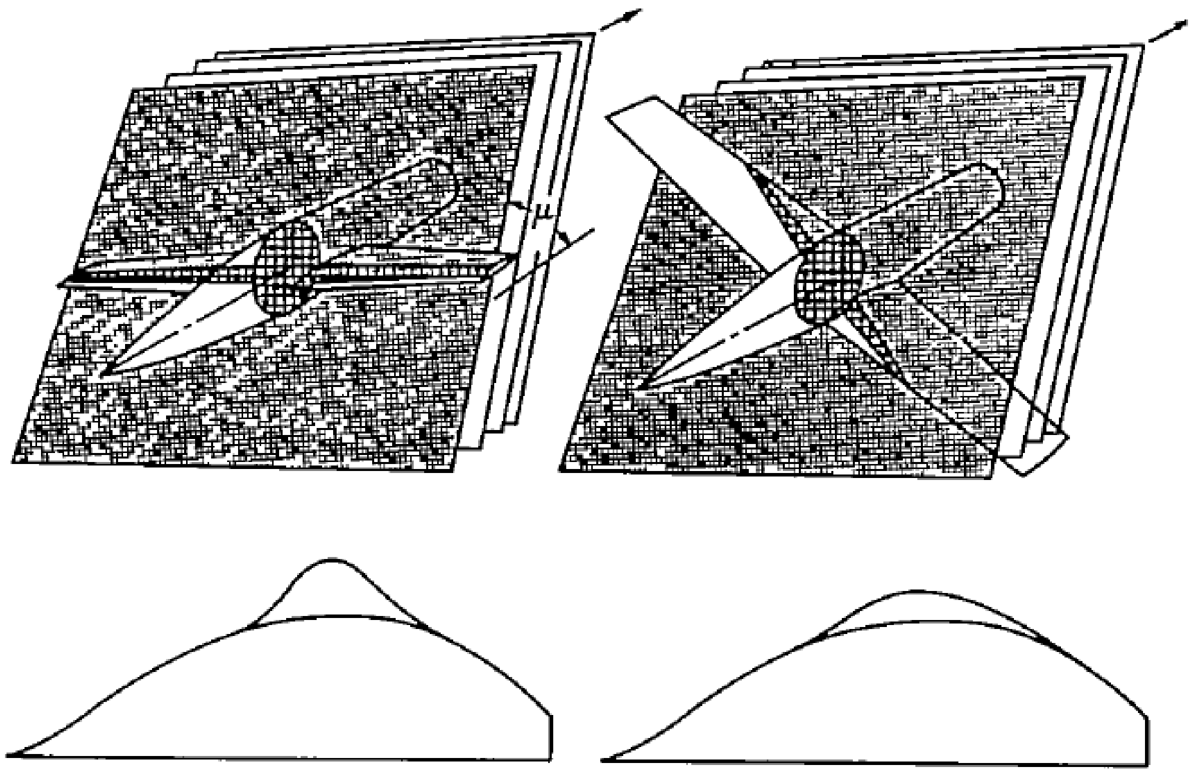


FIGURE 2.—Area distribution given by intersections of Mach planes.

Figure 4 – Example of intersection of an aircraft by M_{ach} planes and the resulting area distribution, extracted from (JONES, 1953)

Being S(x) the intercepted area of the system by a plane perpendicular to the flow and $S'(x) = \frac{dS'(x)}{dx}$. When the curve of $S'(x)$ is expanded as a sines Fourier Series (Equation 2.13) the resulting wave drag can be expressed as Equation 2.14.

$$D'(\theta) = \frac{\pi \cdot \rho \cdot V^2}{8} \sum_{n=1}^{\infty} n \cdot A_n^2 \tag{2.15}$$

$$D = \frac{1}{2\pi} \int_0^{2\pi} D'(\theta) \, d\theta \tag{2.16}$$

An example of the implementation of this theory can be found in OpenVSP, which is a comprehensive parametric design program developed by NASA for aircraft and rocket design. (OPEN VSP, 2021) This program has an implementation of the theory analyzed as an internal tool and also provides for visualization and interactions with the process. (WADDINGTON, 2015).

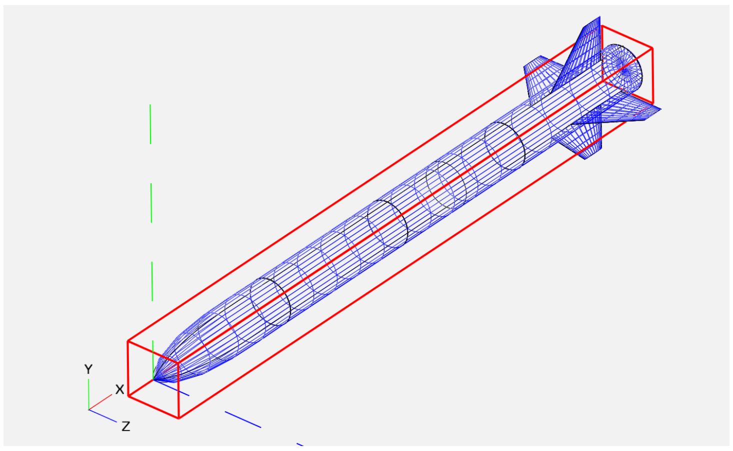
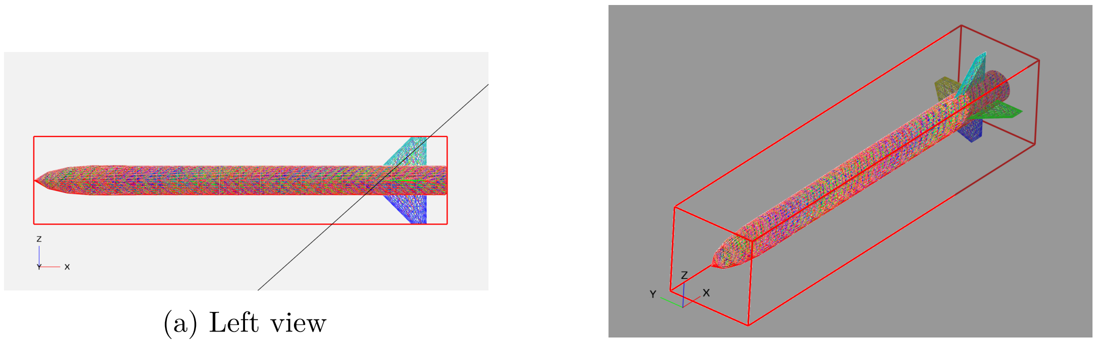


Figure 5 – The example of a random rocket implemented in (OPEN VSP, 2021)



(a) Left view

(b) Oblique view

Figure 6 – Intersection of a $M_{ach} = 1.5$ plane and the rocket implemented in (OPEN VSP, 2021)

2.2.2.2 Estimation of wave drag in the Transonic region

Based on the values of the supersonic wave drag curve (RAYMER, 2004) provides with a method to estimate the wave drag in the transonic region. The transonic region is typically defined for $0.8 \leqq M_{ach} \leqq 1.2$, the theoretical lower limit of the region is in M_{ach}^{cr} where shock-waves start to form on the body or in M_{ach}^{dd} where there's a rise in 0.002 in the objects C_D due to wave drag. (RAYMER, 2004)

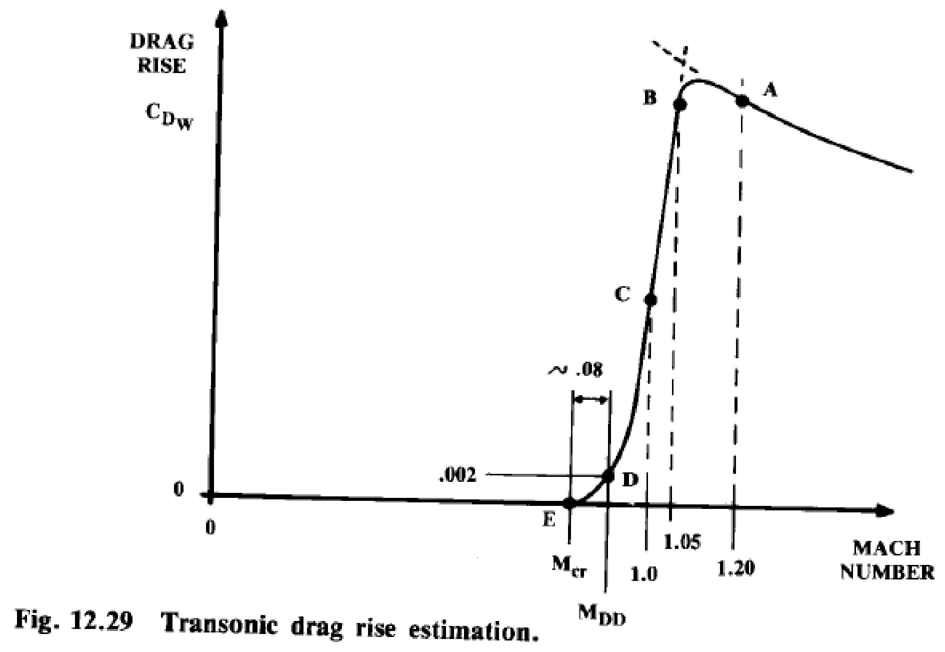


Figure 7 – Example of the method for the estimation of transonic C_D^{SUP}, extracted from (RAYMER, 2004).

The method consists in defining the drag on the points M_{ach}^{cr} and M_{ach}^{dd} as 0 and 0.002 by definition. Then, the drag at $M_{ach} = 1.05$ is set as equal to the drag at $M_{ach} = 1.2$ as point B and in $M_{ach} = 1$ as half of that value. The value of $M_{ach} = 1.2$ is determined by any of the available methods. A representation of this methodology is presented in Figure 7.

2.3 Atmospheric Conditions

As seen in sections 2.4.2 and 2.2, external conditions affect both drag and propulsion by parameters such as ρ, P_{AMB}, μ_{AMB} and c_{sound}. These parameters vary greatly with the altitude and region were the rocket is operating, and considering the wide mission altitude range proposed on Section 1.2 it is necessary to also study atmospheric models to be applied.

The International Standard Atmosphere (also referred as ISA) model is one of the most commonly used atmospheric models. Developed by the International Civil Aviation Organization (ICAO), the model approximates the value of atmospheric properties for different altitudes (up to 80km) by assuming linear temperature gradients and calculates the values of ρ_{AMB} and P_{AMB} using the hydro-static equilibrium equation (Equation 2.17) in combination with the molar form of the ideal gas equation (Equation 2.18). The value of c_{sound} is also calculated under ideal gas assumptions, while other magnitudes such as mu_{AMB} and thermal conductivity are calculated through empirical models explained on the reference (ICAO, 1993).

$$\frac{dP_{AMB}}{dh} = -\rho \cdot g \tag{2.17}$$

$$P_{AMB} = \rho_{AMB} \cdot R_{spec} \cdot T_{AMB} \tag{2.18}$$

The US Standard Atmosphere (USSA) was developed in a joint effort of several governmental agencies. While the methodology used is similar to the one of the ISA model, it perfectly match the ISA model only up to 32km because of differences in the temperature profile assumed. The USSA model reaches much farther than the ICAO model, reaching up to 1000km.(NOAA; NASA; USAF, 1976)

2.4 Rocket Propulsion

Rocket propulsion is a class of jet propulsion of jet propulsion that produces thrust by ejecting stored matter, called the propellant. In contrast to duct propulsion, that is a class of jet propulsion which includes turbojets and ramjets; these engines are also commonly called air-breathing engines (SUTTON; BIBLARZ, 2000).

The rarefied air in high altitudes and the effects of supersonic and hyper-sonic and supersonic flow, make duct jet propulsion engines a poor choice for space-related propulsion. The uniqueness of the rocket, for example, high thrust to weight, high thrust to frontal area, and thrust independence of altitude, enables extremely long flight ranges to be obtained in rarefied air and in space (SUTTON; BIBLARZ, 2000).

Even though developments have been made in the use of high-altitude rarefied air-breathing propulsion, its uses are limited to LEO satellites and other low-thrust applications (FERRATO et al., 2017). Thus, duct propulsion will not be studied in this work. The energy source most commonly used for rocket propulsion is chemical combustion. Energy can also be supplied by solar radiation and, in the past, also by nuclear reaction. (SUTTON; BIBLARZ, 2000)

2.4.1 Main Metrics for Rocket propulsion performance

In order to be able to understand a rocket propulsion performance, there's a need to define and understand the mayor metrics that govern it. The following definitions can be found in (SUTTON; BIBLARZ, 2000).

2.4.1.1 Thrust

It can be defined as the reaction experienced by a rocket's structure due to the ejection of matter at high velocity. It's equation derived, from newtons second law, is presented in Equation 2.19. Usually measured in N.

$$F = \dot{m} \cdot v_e + (P_e - P_{AMB}) \cdot A_e \tag{2.19}$$

2.4.1.2 Total Impulse

Is the total variation in momentum that will be experimented by the rocket due to Thrust. It's formulated through Equation 2.20. Usually measured in Ns.

$$I_t = \int_{t_0}^{t_{burnout}} F dt \tag{2.20}$$

2.4.1.3 Specific Impulse

It can be defined as the impulse that each small amount of weight generates as it is ejected. It is directly correlated to efficiency, as it measures how much impulse is generated by each unit of propellant carried. It's formula is shown in Equation 2.21. Usually measured in s.

$$I_{sp} = \frac{I_t}{m_p \cdot g_0} = \frac{F}{\dot{m} \cdot g_0} \tag{2.21}$$

2.4.1.4 Mass ratio

It is defined as the ratio of the final mass (the mass of the rocket after burnout) by the initial mass of the rocket (before operation). It is presented in Equation 2.22.

$$M_R = \frac{m_f}{m_0} = \frac{m_f}{m_f + m_p} \tag{2.22}$$

2.4.1.5 ToW Ratio

It is defined as the ratio between the rocket's thrust and the rocket's weight for a given moment. For a rocket to take off, it is necessary that $ToW > 1$ in the initial condition. It's formulation is present in Equation 2.23.

$$ToW = \frac{F}{m \cdot g} \tag{2.23}$$

2.4.2 Rocket Propulsion Mechanisms

While thrust in rocket propulsion can be generated through various different methods(O'CONNELL, 2016). The currently viable propulsion methods can be mainly divided in two main categories, chemical and electrical (SUTTON; BIBLARZ, 2000).

2.4.2.1 Chemical Propulsion

The principle of chemical reaction or combustion of one or more fuels with one or more oxidizing reactants is the basis of chemical rocket propulsion. The heat liberated in this reaction transforms the propellants (reactants) into hot gaseous reaction products, which in turn are thermodynamically expanded in a nozzle to produce thrust. (SUTTON; BIBLARZ, 2000). The nozzle will be explained separately in Subsection 2.4.3.

Chemical propulsion is commonly divided in Liquid and Solid propellants. Hybrid rocket propulsion exists by mixing solid rocket fuel with a liquid oxidizer. Usual values for I_{sp} for solid propellants is 220-300, while for liquid propellants it can reach 330. Nevertheless, solid propellant motors are usually lighter, require less movable parts, are more reliable and simpler to build and assemble (SUTTON; BIBLARZ, 2000) (FACULTY OF AEROSPACE ENGINEERING, TU DELFT, 2013). While hybrid rocket propulsion for sounding rockets is of great interest and may en-able conducting more flexible missions, solid rocket motors remain the most common choice for propelling suborbital payloads (OKNINSKI, 2017).

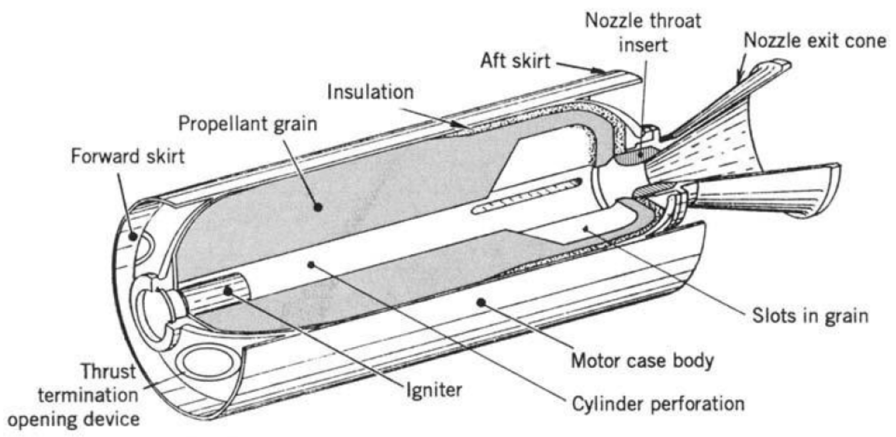


FIGURE 1–5. Simplified perspective three-quarter section of a typical solid propellant rocket motor with the propellant grain bonded to the case and the insulation layer and with a conical exhaust nozzle. The cylindrical case with its forward and aft hemispherical domes form a pressure vessel to contain the combustion chamber pressure. Adapted with permission from Reference 11–1.

Figure 8 – Simplified design of a solid rocket motor, extracted from (SUTTON; BIBLARZ, 2000)

2.4.2.2 Electric Propulsion

Electric propulsion can be defined as any type of propulsion system that uses electricity as a main energy source to generate thrust. One examples of electrical propulsion are Hall-effect engines, shown in Figure 9. Other examples of electrical propulsion are resitojet or thermoelectrical propulsion that uses an electrical resistance to heat propellant which is ejected through a nozzle, plasma ion propulsion which ionizes particles and accelerates then through an electrical field are the most common types. Even though electrical propulsion is widely used for satellites and has I_{sp} in the ranges of over 1000s, it's not of viable use for launch vehicles for its low thrust (usually under 1N) and ToW (SUTTON; BIBLARZ, 2000) (EXOTRAIL SA, 2021?).

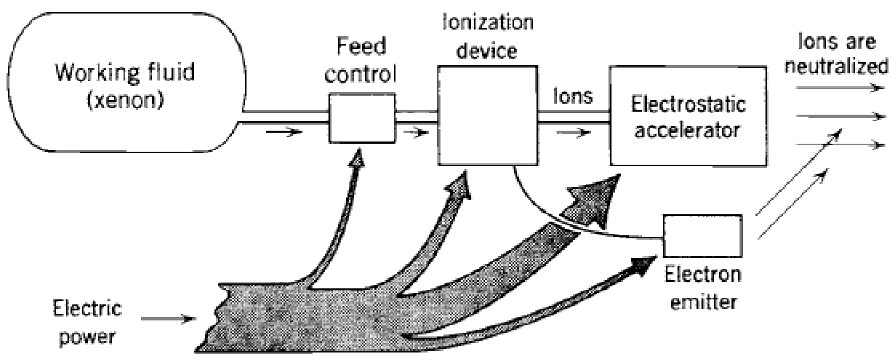


FIGURE 1–9. Simplified schematic diagram of a typical ion rocket, showing the approximate distribution of the electric power.

Figure 9 – Simplified design of an electric rocket thruster, extracted from (SUTTON; BIBLARZ, 2000)

2.4.3 Rocket Nozzles

In most thermal rocket propulsion systems, a nozzle is responsible for accelerating combustion gases in order actually generate thrust. Nozzles usually have a convergent, sub-sonic region, and then a supersonic divergent region. In order to elaborate a simplified nozzle theory, assume a quasi-one-dimensional isentropic flow of an ideal gas throughout the nozzle (SUTTON; BIBLARZ, 2000)(NAKKA, 1997). This assumptions are in accordance with the theory for isentropic flow of compressible fluids presented in (HOUGHTON; CARPENTER, 2003).

2.4.3.1 Isentropic Flow

The main relations for a one-dimensional isentropic flow in a tube are presented in Equations 2.24, 2.25 and 2.27

$$T_0 = T_i \cdot \left[1 + \frac{1}{2} \cdot (\gamma - 1) \cdot {M^i_{ach}}^2\right] \tag{2.24}$$

$$P_0 = P_i \cdot \left[1 + \frac{1}{2} \cdot (\gamma - 1) \cdot {M^i_{ach}}^2\right]^{\frac{\gamma}{\gamma-1}} \tag{2.25}$$

$$v_y = \sqrt{\frac{2\cdot\gamma}{\gamma-1} \cdot \frac{R_u \cdot T_x}{M_{gas}} \cdot \left[1 - \left(\frac{P_y}{P_x}\right)^{\frac{\gamma-1}{\gamma}}\right] + v_x^2} = \sqrt{\frac{2\cdot\gamma}{\gamma-1} \cdot \frac{R_u \cdot T_0}{M_{gas}} \cdot \left[1 - \left(\frac{P_y}{P_0}\right)^{\frac{\gamma-1}{\gamma}}\right]} \tag{2.26}$$

$$\frac{\dot{m}}{A_i} = P_0 \sqrt{\frac{2\gamma}{\gamma-1} \cdot \frac{M_{gas}}{R_u \cdot T_0} \cdot \left(\frac{P_i}{P_0}\right)^{\frac{2}{\gamma}} \cdot \left[1 - \left(\frac{P_i}{P_0}\right)^{\frac{\gamma-1}{\gamma}}\right]} \tag{2.27}$$

Equation 2.27 describes flow density, it's maximum value for a given P_0, T_0 is obtained from the derivation. As $\dot{m}$ is constant throughout the nozzle, this maximum value denotes the minimum cross-sectional area, which is given when $M^y_{ach} = 1$. This point is denominated the nozzle's throat and area value A_t. The relation between mass flow $\dot{m}$ and A_t can be then calculated using Equation 2.28. Furthermore, for $\frac{A_c}{A_t} > 4$, the flow on the combustion chamber can be considered stagnated, thus $P_0 \approx P_c$ and $T_0 \approx T_c$.

$$\dot{m} = A_t \cdot v_t \cdot \rho_t = A_t \cdot P_C \cdot \gamma \sqrt{\left(\frac{2}{\gamma+1}\right)^{\frac{\gamma+1}{\gamma-1}} \cdot \frac{M_{gas}}{R_u \cdot T_C}} \tag{2.28}$$

2.4.3.2 Nozzle's Effect on Thrust Equation

Considering the Equations 2.26 and 2.28 in Equation 2.19, Equation 2.29 is obtained.

$$F = A_t \cdot P_c \cdot \sqrt{\frac{2\gamma^2}{\gamma-1}\left(\frac{2}{\gamma+1}\right)^{\frac{\gamma+1}{\gamma-1}} \cdot \left[1 - \left(\frac{P_e}{P_C}\right)^{\frac{\gamma-1}{\gamma}}\right]} + (P_e - P_{AMB}) \cdot A_e \tag{2.29}$$

Considering that the first part of Equation 2.29 depends only on internal factor of the rocket, it can be rewritten as Equation 2.30

$$F = F^* + (P_e - P_{AMB}) \cdot A_e \tag{2.30}$$

2.4.3.3 Nozzle Geometric definition

Once determined de values of A_c, A_t and A_e, the geometry of the nozzle can be calculated using the approach presented by (SUTTON; BIBLARZ, 2000). While converging-diverging nozzles usually are designed as two intersected paraboloids with a rounded filleted junction, they are initially approximated as cones which are calculated using the process presented on Figure 10. The spacing between the combustion chamber and the throat and the throat and the exit plane is calculated using Equation 2.31. This spacing is usually reduced for de diverngent part by a factor and a parabola with a desired starting and finishing angles is designed over the cones.

$$l_{cone} = \frac{r_2 - r_t}{tan(\theta_{nozzle})} \tag{2.31}$$

The values for θ_{nozzle} are suggested as 15° for the divergent half and as 60° for the convergent half. A thrust correction factor has to be applied in Equation 2.30 using Equation 2.32

$$\lambda_{thrust} = \frac{1}{2} \cdot (1 + cos(\theta_{nozzle_{div}})) \tag{2.32}$$

2.4.4 Solid Propellant Combustion

Given the predominance of solid rocket motors explained in subsection 2.4.2, rocket motors and their technology will be further studied. A main aspect of rocket motors is

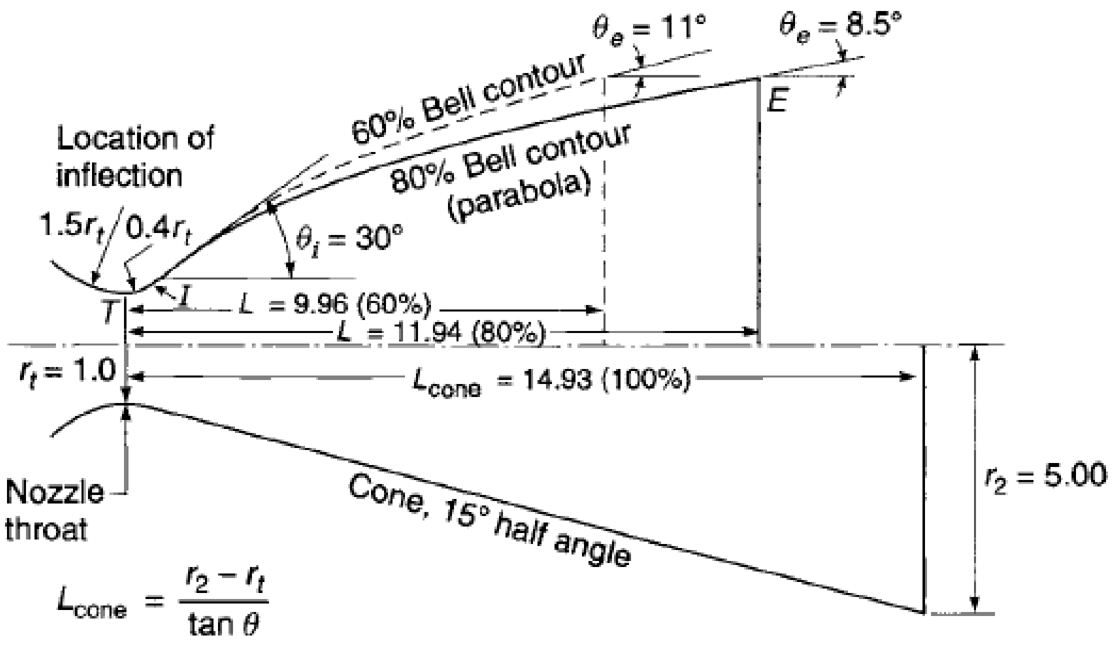


Figure 10 – Nozzle sizing process, extracted from (SUTTON; BIBLARZ, 2000)

that their combustion chambers and propellant tanks are merged into a single structure. Combustion is usually not actively controlled, being $\dot{m}$ an outcome of the combustion and not controlled through pumps as in most liquid rocket propulsion examples. Equation 2.33 is presented in (SUTTON; BIBLARZ, 2000) explaining the combustion of solid propellants.

$$\dot{m} = A_B \cdot \rho_{prop} \cdot r_B \tag{2.33}$$

The burning rate (r_B) is a function of different factors present of the combustion chamber and the values of A_B depend on the initial propellant (grain) configuration.

2.4.4.1 Propellant burning rate

While many factors can alter the value of r, special attention must be given to the effects of P_C. (SUTTON; BIBLARZ, 2000) states that for a given propellant, the burning rate can be modeled using Equation 2.34, which in comparison to Figure 11 seem to bear validity.

$$r_B = a_p \cdot P_{C_p}^{n} \tag{2.34}$$

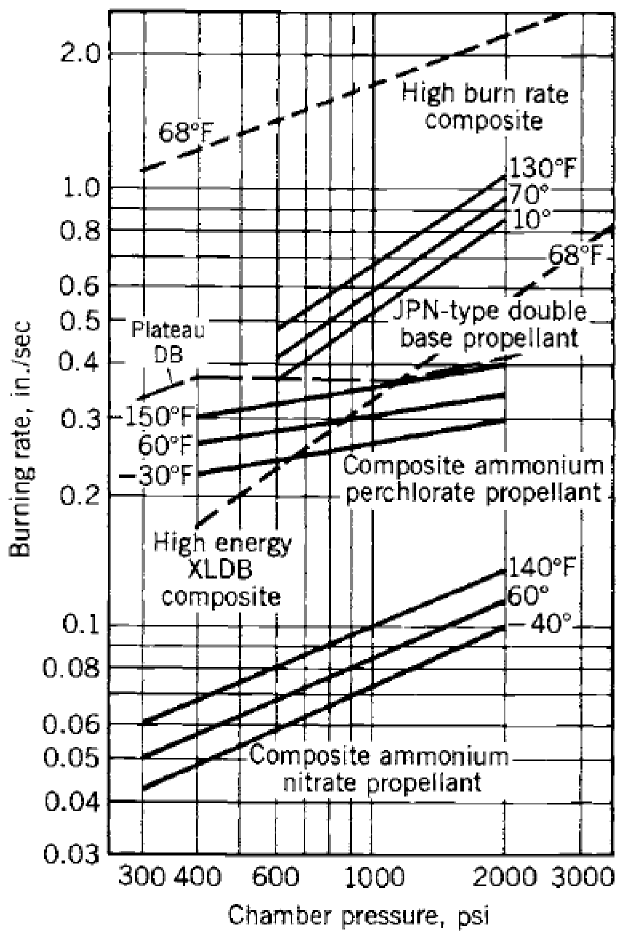


Figure 11 – Burning rate for different materials and temperatures, extracted from (SUTTON; BIBLARZ, 2000)

The value of n, presented on Equation 2.34 is of great interest, typically ranging between 0.2 and 0.6. In practice, as n approaches 1, burning rate and chamber pressure become very sensitive to one another and disastrous rises in chamber pressure can occur in a few milliseconds. When the n value is low and comes closer to zero, burning can become unstable and may even extinguish itself. Some propellants display a negative n which is important for "restartable" motors or gas generators. A propellant having a pressure exponent of zero displays essentially zero change in burning rate over a wide pressure range. Plateau propellants are those that exhibit a nearly constant burning rate over a limited pressure range (SUTTON; BIBLARZ, 2000).

Rocket rotation, differences in propellant initial temperature, propellant aging, and other factors which also affect the performance in significant ways will not be studied in this work. Nevertheless an example of the magnitude of this effects can be seen in Figures 11 and 12. Even though these effects will not be studied in the conceptual design phase, they must be accounted for in subsequent design phases.(SUTTON; BIBLARZ, 2000)

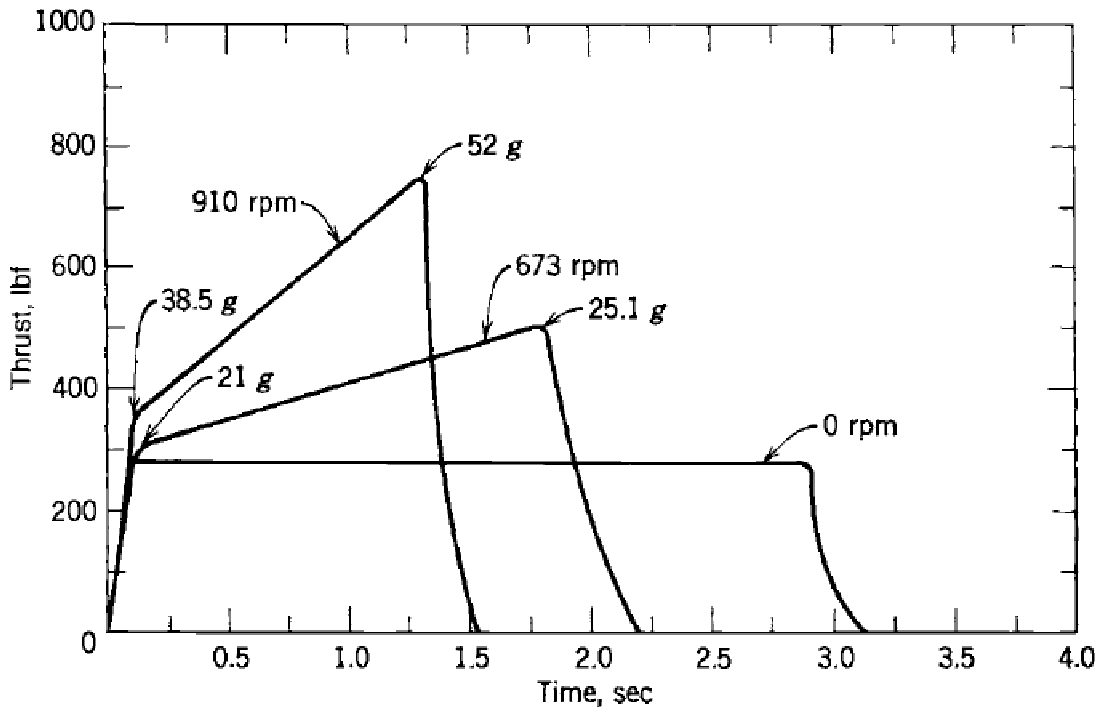


FIGURE 11–12. Effect of axial spin on the thrust–time behavior of a rocket motor with composite propellant using aluminum and PBAN (polybutadiene acrylonitrile) as fuels. (Adapted with permission from Ref. 11–7.)

Figure 12 – Effects of rocket axial spin in the thrust-time curve extracted from (SUTTON; BIBLARZ, 2000)

A useful equation, derived from Equations 2.34, 2.33 and 2.28, is presented in (NAKKA, 1997) where the regime value of P_C is calculated, a formulation for transient pressure is also given but this is outside of the scope of the project.

$$P_C = \left[\frac{A_B}{A_t} \frac{a_r \cdot \rho_p}{\sqrt{\frac{\gamma}{R \cdot T_C} \cdot \left(\frac{2}{\gamma+1}\right)^{\frac{\gamma+1}{\gamma-1}}}} \right]^{\frac{1}{1-n_r}} \tag{2.35}$$

2.4.4.2 Propellant Configuration

The configuration of the grain used for the combustion can greatly alter the generation of thrust for the otherwise same rocket configuration. The configuration can have a constant section along the length of the rocket or can change to acquire desired values of A_B for a given point in the trajectory. Figure 13 shows different configurations commonly used. (SUTTON; BIBLARZ, 2000)

The configurations can be classified by the evolution of A_B over time, this has significant effects on the thrust-time curve:

- Cylindrical: A grain in which the internal cross section is constant along the axis regardless of perforation shape.
- Progressive: Provides a steadily increasing A_B with time, which usually increases $\dot{m}$
- Regressive: Provides a steadily decreasing A_B with time, which usually increases $\dot{m}$
- Neutral: Provides an approximately constant A_B with time, with a constant $\dot{m}$
- Progressive-Regressive: Firstly, provides a steadily increasing A_B with time, which usually increases $\dot{m}$. After a designated point, this relationship is inverted.

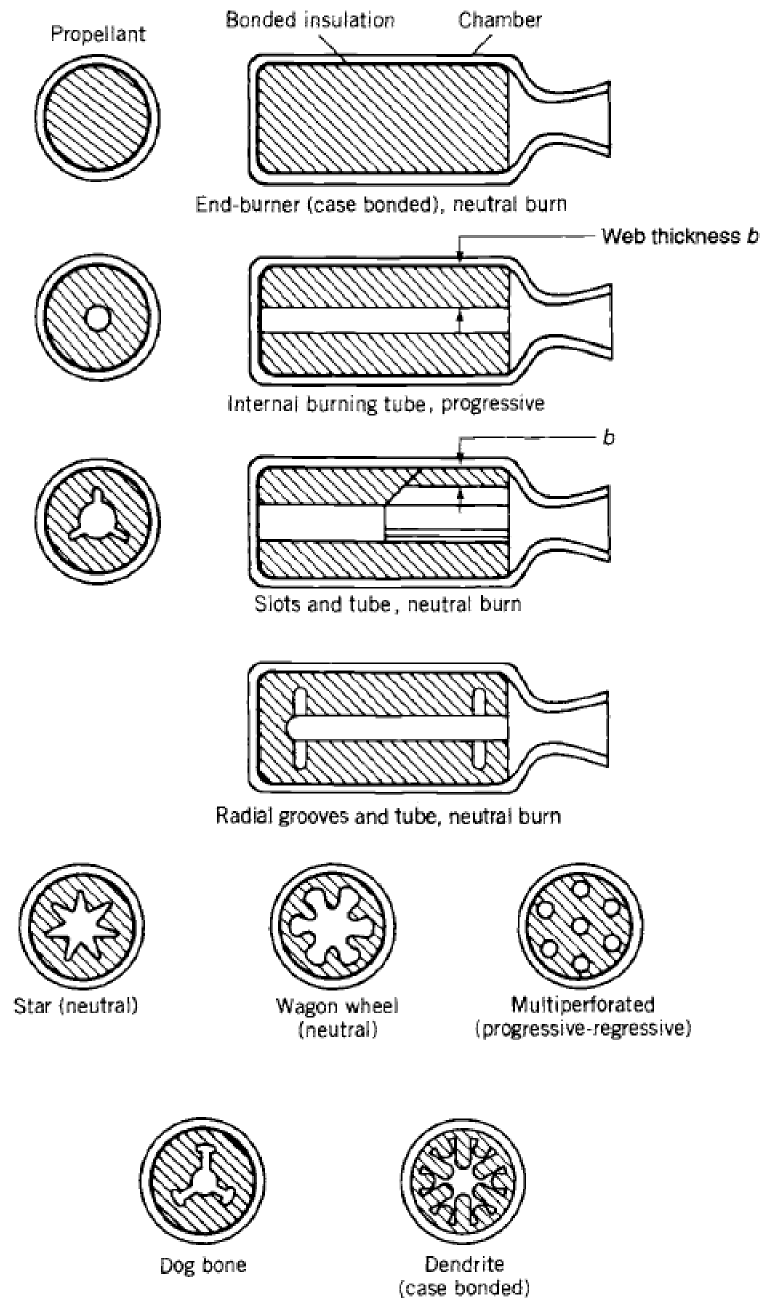

Propellant
Bonded insulation
Chamber
End-burner (case bonded), neutral burn
Web thickness b
Internal burning tube, progressive
b
Slots and tube, neutral burn
Radial grooves and tube, neutral burn
Star (neutral)
Wagon wheel (neutral)
Multiperforated (progressive-regressive)
Dog bone
Dendrite (case bonded)

FIGURE 11–16. Simplified diagrams of several grain configurations.

Figure 13 – Examples of Grain Configuration extracted from (SUTTON; BIBLARZ, 2000)

2.5 Stability and Control

2.5.1 Stability systems

Passive stability systems are those which do not rely on situational feedback to keep the rocket in a desired trajectory. In classical rockets, the main source of passive control are the fixed fins attached to the rocket.

While aircraft stability is well discussed in the methods presented in (ETKIN; REID, 1996) and (RAYMER, 2004), rocket stability analysis is significantly different. The static margin is defined as the distance between the rocket's center of pressure (not the aerodynamic center) and its center of gravity, divided by the rockets maximum diameter(UTAH STATE UNIVERSITY, 2010). The method presented in (BARROWMAN, 1967) allows for a simplified calculation of the center of pressure for a slender finned body. The method is based on the calculation of normal coefficients for different components such as nose-cone, conical transitions and fins, with the position of each component and its normal coefficient, the center of pressure is calculated.

The minimal static margin recommended is 1 and a static margin of 2 is considered excessive(UTAH STATE UNIVERSITY, 2010). While the extension of the method for different angles of attacks presented in (LABUDDE, 1999) allows for a more detailed study of rocket stability, it is outside of the scope of this project. No efforts were made to study a rocket's dynamic stability.

2.5.2 Active control systems

While modern larger rockets use different technologies for active control, such as flapped fins, thrust vectoring, nozzle gimbals and even some missiles use rocket thrusters as stability measures (SUTTON; BIBLARZ, 2000). These technologies are considered outside of the scope of the project.

2.6 Materials and Structures

While these topics are not a main focus of the project, materials and structures in launch vehicles were studied to be able to realistically estimate the necessary structural weight of the hull, the combustion chamber and other components.

2.6.1 Skin and Fins

Common materials for sounding rocket's skin and fins are composites as carbon, glass or Kevlar fiber and metallic as aluminum. Isolation layers in the skin of high density foam or metallic or composite honeycomb are common for high flying rockets with significant reentry heating, but will not be studied on this work. In (VANDAS et al., 2018), aluminum and fiberglass were used for the rocket's body and carbon fiber for the fins.

2.6.2 Motor

For the estimation of the wall thickness of a rocket motor, the expression of the radial pressure for a pressurized cylinder is commonly used with a factor of safety f_s is added. A common value for the factor of safety is 2 (OKNINSKI, 2017) (SUTTON; BIBLARZ, 2000). This is presented in Equation 2.36.

$$t_{wall} = f_s \cdot \frac{P_C \cdot d_C}{2 \cdot \sigma_y} \tag{2.36}$$

As far as materials go, the motor case is usually made either from high-resistance steels or high-strength aluminum alloys (MORGADO, 2019). As the motor case reaches high temperatures (above 700K), usually steel or another metal with a high melting point is used (SUTTON; BIBLARZ, 2000).

2.7 Optimization and Numerical Solution Methods

Beyond technical rocket-related knowledge, there is a need to further study methods to accurately estimate derivatives, numerically integrate functions and to optimize complex functions to achieve global maximums and minimums with a given set of constrains.

2.7.1 Optimization Methods

A more detailed analysis on MDO applied to space LVs can be found in (MORGADO, 2019), a quick overview will be given here based on what is presented there.

Multidisciplinary Design Optimization (MDO) is a field of engineering that deals with finding the global optimal design solution, increasing the complexity of the problem while decreasing computational cost. Typically, the design problem is decomposed into different disciplines, each one subjected to specific constraints. The disciplines may be, for instance, aerodynamics, propulsion, structure, weight and sizing, costs and trajectory.

The most popular MDO method for general design optimization is MDF. It solves the interdisciplinary coupling equations at each iteration using one optimizer at system-level and a Multidisciplinary Design Analysis (MDA). Using this approach, the optimization variables are the disciplines design variables. Due to the coupling between disciplines, they must be analyzed sequentially.

The optimizer analyses the design performance to verify if the design complies with the given constraints at the end of each iteration. It is a simple solution, and at the end of every iteration a feasible solution is given. The drawbacks are the computational cost, which is high, and the lack of parallelization between disciplines. In addition, the method does not necessarily converge.

In order to apply MDO methods, it is necessary to use optimization algorithms. These algorithms search for the best solutions under given constraints, depending on the optimization objective.

The classical optimization algorithms are divided into gradient-based and gradient-free. Gradient-based algorithms, although can present a quick convergence when near a minimum, have several drawbacks. They can only optimize continuous parameters, requires the functions to be differentiable, and can only find the local optima. Among diverse methods to compute the gradient, the simplest is finite differences. It is necessary to make an initial-guess in order to initialize a gradient-based algorithm, and at every iteration the next parameters set is calculated. The most commonly used gradient based methods are the Sequential Quadratic Programming and Interior-Point.

The gradient-free algorithms, however, require only the function values. They are able to solve highly discontinuous and noisy functions, and allow global search. The biggest drawback is the slow convergence when near a local optimum. There are two types of them: deterministic and stochastic. A classical example of local-search deterministic algorithm that is capable of solving non-linear, unconstrained optimization problems is the Nelder-Mead simplex algorithm.

Stochastic algorithms have as a characteristic some inherited randomness. Therefore, the solution may differ every time the algorithm is executed, considering the same initial conditions. The heuristic approaches are inspired by natural phenomena.

2.7.2 Genetic Algorithm

Some of the most popular stochastic algorithms are Genetic Algorithms (GA), based on the concept of Darwin's theory of evolution. They feature selection, crossover and mutation techniques.

An initial generation is created and filled with individuals with a set of properties. The best approach to create the initial population is choosing randomly to improve diversity, since a heuristic initialization technique could cause little diversity and lead to failure when finding the optimum. GA algorithms then update the set of individuals throughout the generations. Each individual is evaluated and ranked based on their fitness values, considering the objective function.

Then, some individuals are chosen, randomly or based on fitness, to breed the new generation. In a process known as crossover, the parents have their characteristics mixed, as it is expected that the children share the properties of the best individuals. Also, the individuals can be mutated, randomly changing their properties, in order to maintain diversity in the population. If not, the individuals may become too similar to each other, stopping evolution.

The process continues once the mutation operation is concluded, and the algorithm repeats until convergence.A GA was developed to optimize the rocket design, allowing an easy implementation and parallel optimization.

2.7.3 Numerical Solution Methods

Given the importance of the calculation of the area derivative for the theory presented in Section 2.2.2, we need a formula to estimate these values with good accuracy. Furthermore, the application of a Fourier Series in a discrete function will require a numerical approach to solve numerical integrals.

2.7.3.1 Finite Differences

A method for approximating a function's nth-order derivative in one point as a linear combination of the the value of the function for equally spaced points in the vicinity of the first one is presented in (FRANCO, 2006). Through such methods, the Equations 2.37, 2.38 and 2.39 were obtained in order to estimate first order derivative of a function u respective to x with a third order error, which means that the error of the approximation decreases with h^3. The first is a centered finite differences formula and the others are lateral differences formulas.

$$u_x(x,t) = \frac{u(x+h,t) - u(x-h,t)}{2h} \tag{2.37}$$

$$u_x(x,t) = \frac{-3 \cdot u(x,t) + 4 \cdot u(x+h,t) - u(x+2h,t)}{2h} \tag{2.38}$$

$$u_x(x,t) = \frac{3 \cdot u(x,t) - 4 \cdot u(x-h,t) + u(x-2h,t)}{2h} \tag{2.39}$$

2.7.3.2 Trapezoid Integration

In order to accurately calculate integrals of functions, different methods are proposed in (CHENEY; KINCAID, 2008), the trapezoid method was chosen because of it's simplicity, independence on the number of points used and easy implementation, which allow a high flexibility. It's formula is described in Equation 2.40 and it has a second order error, this means, it is is proportional to h^2.

$$\int_a^b f(x)\,dx \approx \frac{1}{2}\sum_{i=0}^{n-1}(x_{i+1} - x_i) \cdot [f(x_i) + f(x_{i+1})] \tag{2.40}$$

2.7.3.3 Runge-Kutta Methods

The Runge-Kutta Methods are a family of methods for the resolution of DEs. Named after Carl Runge and Wilhelm Kutta, they are designed achieve a high order precision without requiring analytic differentiation of the original differential equation. The methods consist in evaluating the function f n times in which each variable is incremented by partial steps, previous calculated values are used as slopes in these steps. The error of a Runge-Kutta method of order n is proportional to h^{n+1}. The 4-th order Runge-Kutta method is of common use for scientific calculations(CHENEY; KINCAID, 2008) and is shown in Equations 2.42 and 2.43 for a system described in the form of Equation 2.41, it's important to note that x can be a vector and consequently K_1, K_2, K_3 and K_4. The 4th order Runge-Kutta method is also implemented in the software OpenRocket, widely used for amateur rocketry (NISKANEN, 2013).

$$\dot{x} = f(t, x) \tag{2.41}$$

$$x(t+h) = x(t) + \frac{1}{6}(K_1 + 2K_2 + 2K_3 + K_4) \tag{2.42}$$

$$\begin{cases} K_1 = h \cdot f(t, x) \\ K_2 = h \cdot f(t + \frac{1}{2}h, x + \frac{1}{2}K_1) \\ K_3 = h \cdot f(t + \frac{1}{2}h, x + \frac{1}{2}K_2) \\ K_4 = h \cdot f(t + h, x + K_3) \end{cases} \tag{2.43}$$

3 METHODS

In order to implement all the theory studied in Chapter 2, several considerations and simplifications had to be made both to simplify the implementation and also to optimize the use of computational resources, as explained in Section 3.1. Also, in order to be able to create a truly open and customizable tool, the choice of the programming language had to be studied. Finally, the algorithms for the simulation were created and tested.

3.1 Conceptual Considerations

Based on the studies developed on Chapter 2, some first considerations had to be made in order to simplify the model and reduce the number of variables, which also augments model convergence as explained in Section 2.7. Considerations will be divided on Aerodynamic, Propulsion, Stability and Mass considerations.

The rocket's internal telemetry and other components were considered as having the same mass and volume than a 6U CubeSat. Differing in the form, which is assumed as an hexagon limited by the inner diameter of the rocket section an dimensions are variable according to d_{MAX}.

Furthermore, for simplicity, only single stage rockets were considered and implemented in the analysis.

3.1.1 Aerodynamic Considerations

Drag estimations will be considered as following: Subsonic drag estimations were made using the methods described in (RAYMER, 2004) for the profile drag of an airplane. As it considers body and aerodynamic surfaces on an empirical model, its easy implementation allows for a quick yet accurate estimation of subsonic drag.

As supersonic drag is a mayor component of the overall drag at supersonic speeds (HOUGHTON; CARPENTER, 2003), a more detailed model is necessary. The model proposed by (JONES, 1953) allows for an analytical approach and accurate results.

Effects on drag due to the transonic region were estimated using the methodology described in Section 2.2.2.2, the values of M_{ach}^{cr} and M_{ach}^{dd} were considered as the ones suggested on (RAYMER, 2004).

3.1.2 Numerical Simulation Considerations

The model implemented to determine the rockets maximum altitude was a 1 DOF dynamic model (in which the only DOF considered was altitude) due to the complexity of higher DOF simulations. The forces considered are explained on image 14. The governing equation of the rocket's uni-dimensional flight path is given in Equation 3.1.

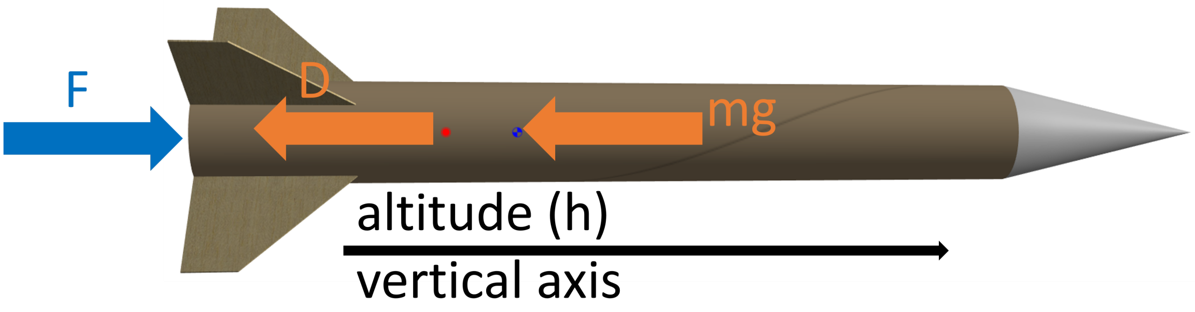


Figure 14 – Rocket's free body diagram, elaborated by the author, rocket image generated using OpenRocket(OPEN ROCKET, 2021?)

$$m(t) \cdot \ddot{h} = F(t) - D(h, \dot{h}) - m(t) \cdot g(h) \tag{3.1}$$

Equation can be manipulated to obtain Equation 3.2 considering $v = \dot{h}$.

$$\begin{gathered} \dot{V} = \frac{F(t)-D(h,v)}{m(t)} - g(h) \\ \dot{h} = V \end{gathered} \tag{3.2}$$

In this form, Equation 3.2 can be used as a vectored form from the f function in Equation 2.41. In order to integrate this equation over time, a RK4 method was implemented, as explained in Section 2.7.3.3

3.1.3 Propulsion Considerations

3.1.3.1 Solid Rocket Propulsion Selection

As stated in section 2.4.4 solid rocket propulsion is the most common in sounding rocket in the market entry segment for these main reasons:

1. Comparative High Thrust-to-Weight ratio
2. Simple composition with few complex components
3. High reliability

Thus, only solid rocket motors were considered for the mission and are the only ones implemented in the program.

The propellant was considered as Ammonia Perchlorate, as it is readily available and of widespread use (SUTTON; BIBLARZ, 2000). While it has been shown that the compound can be toxic, it's main hazards are listed as explosive and incendiary and only prolonged exposure is deemed dangerous, being perfectly suitable for this task (NATIONAL CENTER FOR BIOTECHNOLOGY INFORMATION, 2021). Values of the propellant utilized for the simulations are available in Table 2.

Variable	Magnitude	Unit
T_C	2816	K
ρ_p	1.69	g/cm^3
M_{gas}	25	g/mol
γ	1.21	-

Table 2 – Typical values for the combustion of Ammonia Perchlorate used for the simulations, extracted from (SUTTON; BIBLARZ, 2000).

The values of $a_p = 0.123$ and $n_p = 0.287$ for Equation 2.34 were obtained from (NAVARRETE-MARTIN; KRUSB, 2017) for a composition of in AP 80%, Al 2%, HTPB 18%; typical proportion in high-power rocketry. This composition is within the ranges specified by (SUTTON; BIBLARZ, 2000) for the values provided in Table 2. Note that the output of the equation using these values will be given in mm

3.1.3.2 Motor Geometric Constrains

The motor case will be assumed as a cylinder, for an irregular body fuselage, the outer radius of the motor casing will be considered as equal to the minimal internal radius of sections of the fuselage where the motor is lodged. In the specific case of a fuselage with a mostly constant diameter, this definition does not change significantly.

3.1.3.3 Grain Composition and Disposition Selection

As the focus of this work is geometric optimization of the rocket. Generic grain composition will be assumed, leaving the specifics of improved configurations for the preliminary stage of the rocket's design, which is outside the scope of this project. According to Sutton (SUTTON; BIBLARZ, 2000), the grain "star" configuration, described in Figure 15, yields an approximately constant burning area, which in regime conditions translates to a constant thrust. Another neutral grain disposition is the end-burner disposition presented in Figure 13. For the first A_B can be estimated as the internal area of the combustion chamber, thus Equation 3.3, for the second (SUTTON; BIBLARZ, 2000) states that for $d_C < 0.5m$ the burning surface can be approximated as the cross-section of the cylindrical tank, thus Equation 3.4.

$$A_B = \pi \cdot \frac{d_C}{2} \cdot l_C \tag{3.3}$$

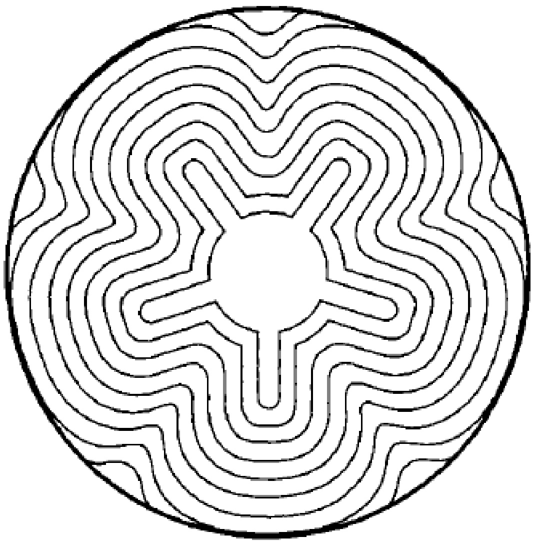


Figure 15 – Propellant star disposition that allows and approximately constant A_B, extracted from (SUTTON; BIBLARZ, 2000)

$$A_B = \pi \cdot \left(\frac{d_C}{2}\right)^2 \tag{3.4}$$

Both dispositions will be assumed as a cylindrical, as it provides almost constant thrust and has a simple formulation. Furthermore, they will be implemented separately, as the optimization parameters can differ from one to the other. The presence of l_C in Equation 3.3 leads to large values of A_B in elongated rockets, this will yield high $\dot{m}$ and F^*, but will also diminish $t_{burnout}$ which means all the propellant can be burned in the lower atmosphere where higher D is expected. For the end-burner configuration, a comparatively small value of A_B means per Equation 2.35 that much lower values of A_t will be needed to achieve a similar P_C. This can also be advantageous, as it means high $t_{burnout}$ comparative to the "star" configuration, consequently, a higher amount of propellant might burn higher in the atmosphere, where the aerodynamic effects are less prevalent.

3.1.4 Stability Considerations

For stability, only the theory presented in (BARROWMAN, 1967) was considered. No efforts were made in estimating the effects of supersonic aerodynamics into stability.

3.1.5 Material Considerations

The rocket's hull was assumed as composed by a 1/16" sheet of aluminum, while not a perfect assumption, as usually the nose is machined and not based on a sheet of metal, its judged as a good starting point. The rocket's fins were considered as made from carbon fiber as cited in (VANDAS et al., 2018) for easier casting. The motor casing was considered as grade S355 Steel, as recommended by (SUTTON; BIBLARZ, 2000). Relevant material properties were extracted from (DEPARTMENT OF DEFENSE, UNITED STATES OF AMERICA, 1998)

3.1.6 Body Geometric Considerations

While initial intention of the project was to include non-linear rocket bodies and study their effect on total drag, the need to reduce variables and the examples commented on Section 2.1 lead to the consideration of mostly straight rocket bodies with a constant diameter outside the nose and boat-tail.

3.1.6.1 Rocket Diameter and Length

As the payload is defined as a 3U CubeSat, which according to Figure 3 has a diagonal of $100 \cdot \sqrt{2} = 141mm$. Thus, rockets with section diameters lower than $150mm$ were not be considered. Limitations on the total length were considered as a product of the diameter by a ratio that varies according to what presented in Section 2.1.

As the focus of the project is weight minimization, the upper diameter constrains was not studied in much detail. Defining the maximum diameter at $0.5m$. Results showed no tendency of rockets towards this value as it's shown in Chapter 4 later in this work, so the upper constrain in diameter can be assumed to have no significant effect in the output of the optimization.

3.1.6.2 Fin geometry

In an effort to avoid an elevated number of optimization variables, the fin's number and geometry were predefined to an specific format, based on the one presented on (PALLONE et al., 2018), leaving only scaling and position as optimization variables. Four fins where assumed, though the program is fully capable of handling other numbers. The dimensional relations are shown in Figure 16. As visible in the Figure, an effort was made to define the fin without making assumptions about the rocket's body, this helps in possible future non-constant diameter studies.

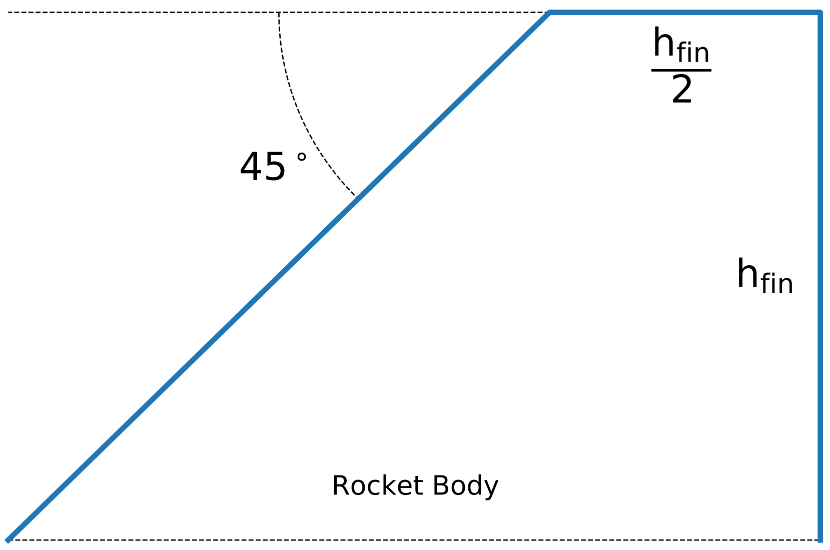


Figure 16 – Format of the fins chosen for the optimization, based on the one presented in (PALLONE et al., 2018)

Ideally, the fin's airfoil, would be chosen as a double wedge with 2% thickness as presented in (PALLONE et al., 2018). But the implementation of variable thickness was deemed too complex for the initial purpose of the program. Thus a constant thickness equal to $2\% \cdot MAC_{fin}$ was assumed in a similar approach as the one presented in (OKNINSKI, 2017).

3.2 Definition of optimization variables and their studied range

For the implementation of the a genetic algorithm, a set of variables (genes) must be defined to generate each individual rocket. The choice of this set of variables is of great importance as an elevated number of variables hinders the method's convergence, but a low number brings little insight to the studied domain and can be even misleading if they aren't enough to correctly detail the problem to be optimized.

3.2.1 Diameter

Rocket diameter was chosen as one of the main variables to describe the problem, due to the geometric direct geometric constrain that the rocket body must be able to fit the payload inside.

3.2.2 Length to Diameter Ratio

Rocket length is a mayor variable affecting almost all other component in one way or the other. A length to diameter ratio was defined, as seen in Section 2.1, this value usually varies from 10-20 as described in 2.1.

3.2.3 Fin Size Ratio

As explained in Section 3.1, fin geometry was limited to avoid having a considerable amount of variables. The main variable considered, fin semi-span h_{fin} was defined as the proportionality to rocket diameter, defining the fin size ratio as $\frac{h_{fin}}{D_{max}}$.

3.2.4 Fin Position

The fin position was defined as the distance between the fin's chord trailing edge and the rocket's tail. The values studied were within 0-20% of the length of the rocket.

3.2.5 Nose Length Ratio

The length of the nose was defined as a proportion to the total length of the rocket. Thus, nose length ratios of 0.05 to 0.3 were studied.

3.2.6 Nose Type

The nose format has relevant influence on the drag generated by the rocket. A set of different nose formats were studied following examples seen in the bibliography, these are a parabolic configuration with the vertex on the rockets nose, an elliptic configuration with the vertex on the nose cone and co-vertex on the junction of the nose and fuselage, a conic configuration, a parabolic ogive configuration with the vertex on junction of the nose and fuselage and a LD-Haack (Von Kármán) series ogive based on formulations presented on (CROWELL, 1996). Such examples are presented on Figure 17.

3.2.7 Throat area ratio

In order to allow some variation in F, P_C and $\dot{m}$, throat area ratio, defined as $\frac{A_C}{A_t}$ was introduced. Based on (SUTTON; BIBLARZ, 2000), this ratio cannot be smaller than 4 because this can invalidate the hypothesis that flow in the chamber is stagnated. The upper limit was set at 10 arbitrarily to avoid excessive pressure on the chamber. As shown in Equation 2.35, its relation to A_B is of high importance to determine the value of P_C. For the star configuration, the range was defined as from 4 to 10 and for the end-burner, from 50 to 1500.

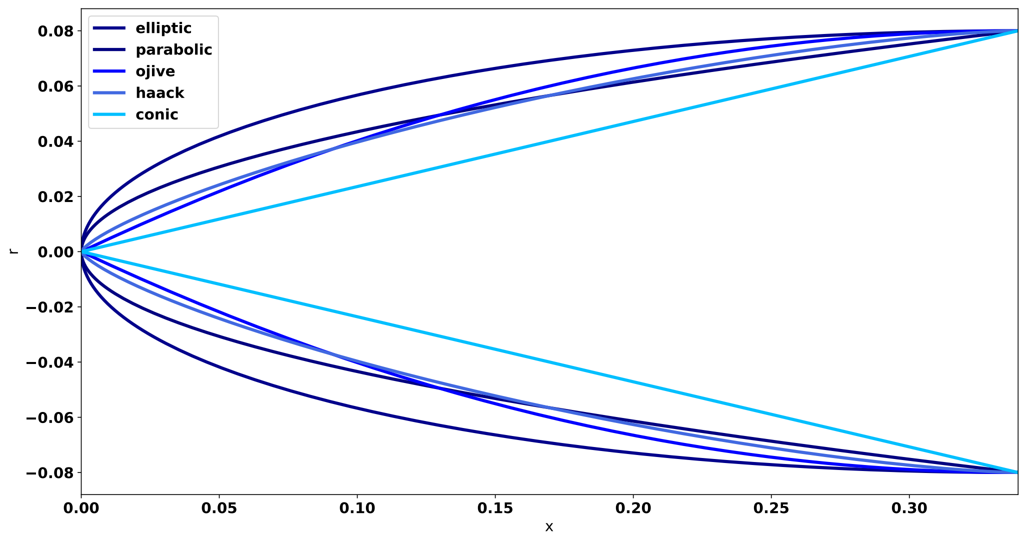


Figure 17 – Different nose configurations studied for each rocket based on the definitions of (CROWELL, 1996)

3.3 Algorithms and Implementation

The algorithmic implementation was modular, taking advantage of smaller code blocks and object oriented programming, but adding further computational costs on their interactions. The developed modules follow the interactions described in Figure 18. An arrow coming from one block and pointing to another means that the pointed block is called and used on the first one.

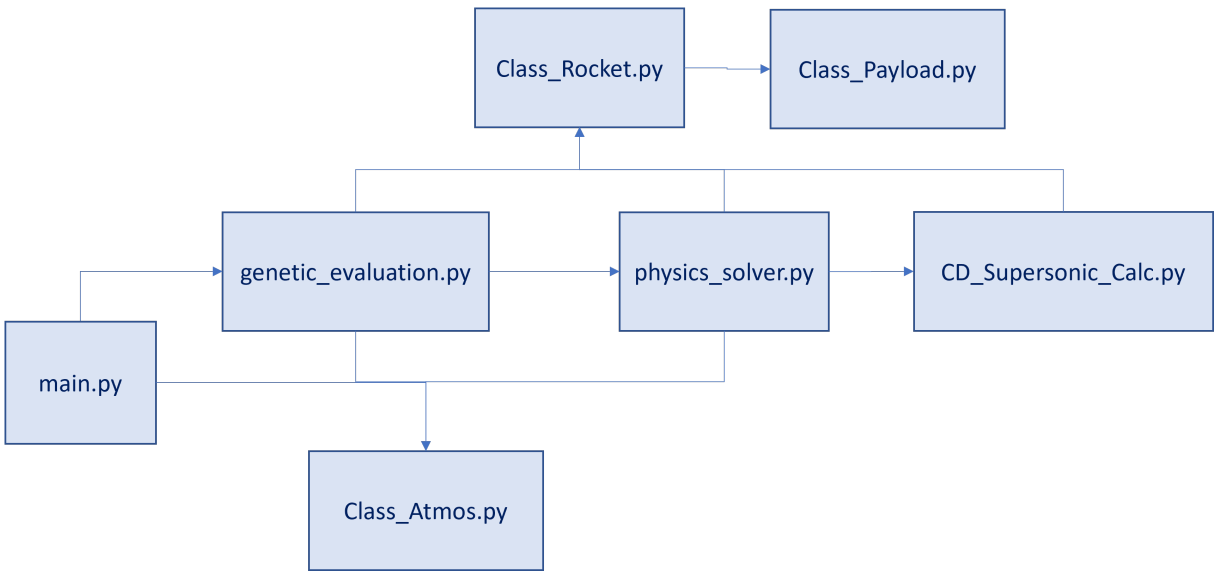


Figure 18 – Interactions between different modules that compose the program

3.3.1 Programming Language Choice

The chosen programming language for the project was *Python* in its 3.9 version. This choice came considering three main factors:

- familiarity with the programming language
- language flexibility and availability of libraries
- widespread use

Based on the author's previous experiences, the languages studied were shortlisted to the following: Python, Matlab, Excel-VBA and C/C++. While most languages from the past list are highly flexible, with OOC implementations and readily available libraries, Excel-VBA, while higly powerful, is constrained to spreadsheets and libraries developed by the software provider.

According to the PyPL (Popularity of Programing Languages) ranking the most used programming languages as in August 2021 is Python, this ranking is based on Google searches after the languages tutorials, help forums, etc. (CARBONNELLE, 2021), as can be seen in Table 3. A second ranking (Tiobe), based on the number of existing pages in official help and online forums, was consulted and determined C as the most popular programming language, closely followed by Python as can be seen in Table 4. Widespread use allows for a wider intelligibility of the written code for future reference, which is highly desired, but also allows for more sources of help and reference in the implementation. Languages linked to specific paid-for software as Matlab and Excel-VBA were less favored based on this factor.

Rank	Language	Share
1	Python	29.93%
2	Java	17.78%
3	JavaScript	8.79%
4	C#	6.73%
5	C/C++	6.45%

Table 3 – Data extracted from PyPL (CARBONNELLE, 2021)

Position (Aug 2021)	Language	Ratings
1	C	12.57%
2	Python	11.86%
3	Java	10.43%
4	C++	7.36%
5	C#	5.14%

Table 4 – Data extracted from TIOBE (TIOBE, 2021)

On a poll presented in (SILVA, 2021), in a group where over 95% had some formation in the field of engineering and over 70% in aeronautics/mechanical engineering, Python was shown to be the preferred computational tool of almost 53% of the interviewees. This reinforces the argument that a tool developed in Python is most likely to be highly intelligible for other people in the field.

Based on the information above, the choice of a programming language can be resumed in Table 5.

Language	Author's skill	Flexibility and Libraries	Widespread Use
Python	High	High	High
C/C++	Medium	High	High
Matlab	High	High	Limited
Excel VBA	High	Limited	-

Table 5 – Comparison of programming languages

3.3.2 Main Module

The main module recieves it name because it is the one that when run activates the other modules an it is the main workflow of the programm. This module actually only calls two other modules, the Atmosphere Class to initiate the atmospheric model and the Genetic Evaluation module, which manages the simulation, classification and hybridization of the generations. The main workflow can be seen on Figure 19.

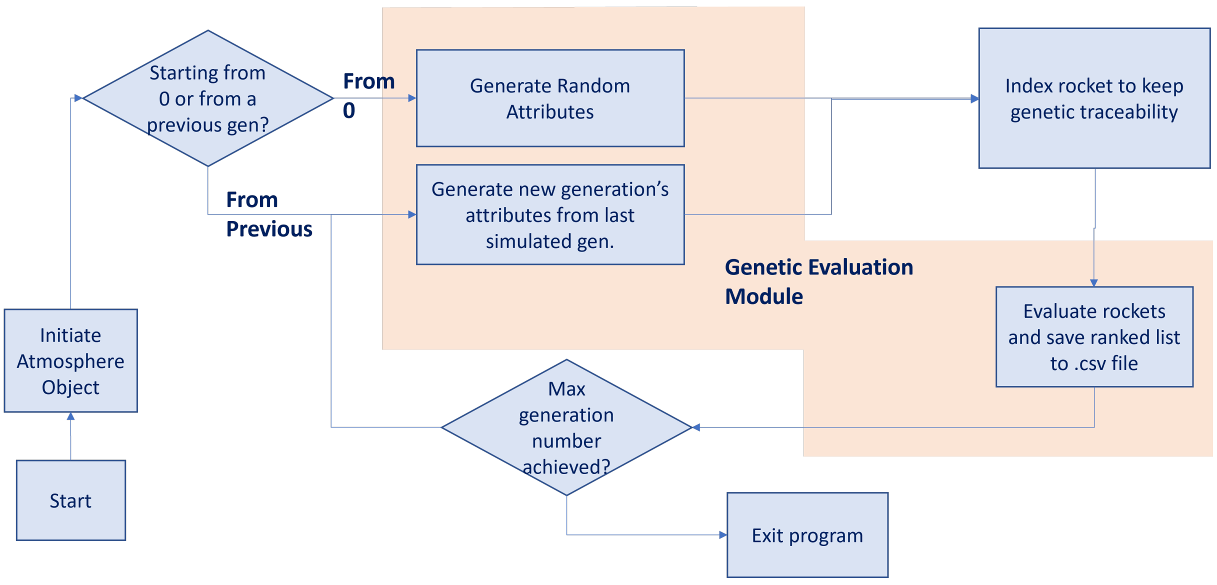

Start
Initiate Atmosphere Object
Starting from 0 or from a previous gen?
From 0
From Previous
Generate Random Attributes
Generate new generation's attributes from last simulated gen.
Index rocket to keep genetic traceability
Evaluate rockets and save ranked list to .csv file
Genetic Evaluation Module
Max generation number achieved?
Exit program

Figure 19 – Flowchart of the main module

3.3.3 Atmosphere Class - Module

Even though a python implementation of an atmospheric model can be found in (WAAL, 2019), an object class 'Atmosphere' was created to facilitate its integration to the code. Counter-intuitively, the code in the reference actually uses the model described in (NOAA; NASA; USAF, 1976). A problem raised from the direct implementation of the code into de Physics module (Section 3.3.5), was that the need for it to be run on each iteration, which eventually led to a higher computational cost by running the program over 1 million times. The implemented class allows for the program to be run in the beginning of the execution, creating arrays of values at each 5m with corresponding interpolation functions.

A limitation of the isacalc (WAAL, 2019) module implemented into the class, is that it only calculates the values up to a 110km altitude. Nevertheless, The required values of ρ_p and P_{AMB} can be assumed as negligible at this altitude and higher for the considerations of suborbital space vehicle, therefore after the 110km threshold these were interpolate as 0. Values of μ_{AMB} and c_{sound} remain constant with the values of $h =$110km just to have finite quotients on the calculations of Re and M_{ach}, but their overall effect will be unconsidered as drag will be a numerical 0.

3.3.4 Rocket Class - Module

In order to link the different variables and functions that describe a rocket and its behavior, a Rocket class was created. The objective of this class was to include all variables and methods which could be determined using only data of the rocket. The class requires as an input a dictionary of the values of the optimization variables and the Payload object that chosen for the rocket. The initialization of the rocket class generates the external points of the rocket's body, determines fin geometry, calculates the rocket's propulsion system's size, weight and capabilities, the weight and size of the recovery system and determines the rocket's center of gravity while empty and on launch configuration. A stability validation is made to see if the rocket meets static stability criteria defined in Section 3.1.

3.3.4.1 Propulsion Calculations

As the value of F^* is solely determined by variables of the rocket, the calculation of this value was included on the Rocket Class module. Based on the assumptions made on 3.1 regarding the motors geometry and grain disposition, the values of P_C is calculated iteratively. The total mass and disposition of propellant and the structure supporting the propulsion system are also calculated. For the calculations the value of P_e was assumed constant for all rockets as $0.8P_{AMB}(h = 0)$. The logic implemented is shown in Figure 20.

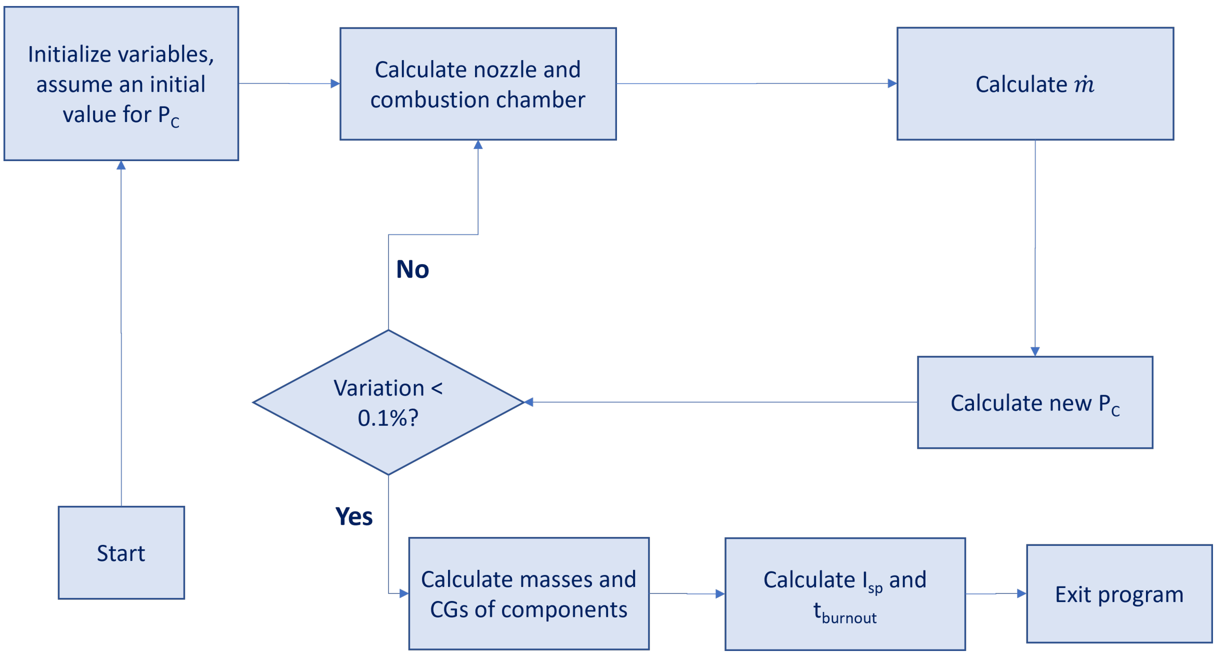

Initialize variables, assume an initial value for P_C
Calculate nozzle and combustion chamber
Calculate $\dot{m}$
No
Variation < 0.1%?
Calculate new P_C
Yes
Start
Calculate masses and CGs of components
Calculate I_{sp} and $t_{burnout}$
Exit program

Figure 20 – Flowchart of the estimations regarding the propulsion system

In the case of the star configuration, to calculate the rocket's propulsion system, first an iterative process takes place to . First, a value of P_C is given, which allows calculate A_e for a given A_t. With these values it is possible to calculate the size of the nozzle and then A_B. With this, P_C can be obtained from Equation 2.35. Trough iterations, a value of P_C that satisfies the conditions is found.

Two options are possible if through the iterations a value of A_e is found that is greater than the the area of the tail section of the rocket. The first is to consider this value as the new rocket section, increasing the rocket's end section area and drag, a second option is to enforce the geometric restriction of the rocket upon A_e and a new value of P_e is calculated, this means, the nozzle is cut and flow will be under-expanded.

Once the error value of P_C within iterations is under 0.1%, the final sizes of the nozzle and combustion chamber are recorded. Mass and CG of the combustion chamber, the propellant and the nozzle are calculated. The value of $t_{burnout}$ is calculated. As the value of A_e can be different than the section of the rocket, a boat-tail curved transition is added to reduce drag (TRAN et al., 2013?) at the end of the rocket.

3.3.4.2 Recovery system Calculations

For calculations of the recovery system, first, a desired terminal velocity of $V_{ter} = 9m/s$ was defined based on (PEPERMANS et al., 2018) and (VANDAS et al., 2018). The parachute weight and volume estimation was implemented acording to the simplified approach presented on Chapter 8 of (USAF, 1978), a $C_D^{PAR} = 0.6$ value has selected according to values presented in the book. An iterative process begins assuming no weight on the recovery system, with the empty weight of the rocket, a first estimation of the parachutes size and mass can be made. This new weight is now considered within the vehicles empty mass and the process is repeated.

Once the mass difference between two iterations is less than 0.1% of the previous mass, a calculation of the volume required is made based on the average packaging density presented in (USAF, 1978). After this, the volume is fitted in the nose con and if needed it can exceed this limits. In case of a high volume parachute that requires moving the other components further back on the rocket's main axis, the propulsion system is recalculated. With the diminution in size of the propulsion system, mass will necessarily not grow and the calculated parachute will remain viable.

3.3.4.3 Static Stability

The rocket's center of pressure is calculated with the methods presented in (UTAH STATE UNIVERSITY, 2010)(CULP, 2008). The criterion of minimal static margin is applied for CG_{empty} and CG_{full}. In the case of the rocket failing to meet the criterion in any of the points, a boolean variable used in the scoring is set as False.

3.3.4.4 Scoring

As the objective is to minimize rocket launch weight, the score is based in this metric. For a rocket with no penalties, the score is set as exactly equal to the launch weight. Two penalties were introduced as shown in Equation 3.5, for rockets that fail to meet the stability standard, the score is multiplied by a factor described in Equation 3.7; for rockets that fail to meet the altitude h_{ref} the score is multiplied by a factor of defined in Equation 3.6. For rockets that did meet the desired values the penalties were set as 1. Unfeasible rockets are defined as those in which the payload, nose and HUB occupy over 85% of the body's length, leaving no space for a combustion chamber and nozzle. These rockets are assigned as score an arbitrarily big number ($score = 10^{10}$). The exponent on the altitude factor is to minimize the chance of small light rockets achieving a good position by just being light and missing the objective by big margins. The penalty factors are purposely discontinued at the limit of the tolerances to maximize the reward of complying with the conditions.

$$score = m_0 \cdot f_{stability} \cdot f_{altitude} \tag{3.5}$$

$$f_{altitude} = 1.1 \cdot \left(\frac{h_{max}}{h_{ref}}\right)^2 \tag{3.6}$$

$$score = min\left(1.1 + \cdot (1 - MS_{min}) \cdot 4; 5\right) \tag{3.7}$$

3.3.5 Payload Class - Module

This class was introduced to simplify the use of different CubeSat configurations in case needed. A payload object allows for a 'configuration' input, which can be any of the U configurations listed in Section 1.3. Furthermore, in case the keyword 'custom' is given, custom values for mass, length, width1 and width2 will be used, theoretically accommodating any payload in the form of a rectangular prism. A further variable can be included to specify the payload's CG in it's most lengthy dimension, which will be assumed in the middle of it if not specified.

3.3.6 Physics Module

The main objective of the physics module is to determine the maximum attainable height for each rocket. It's main function was developed following the flowchart specified in Figure 21.

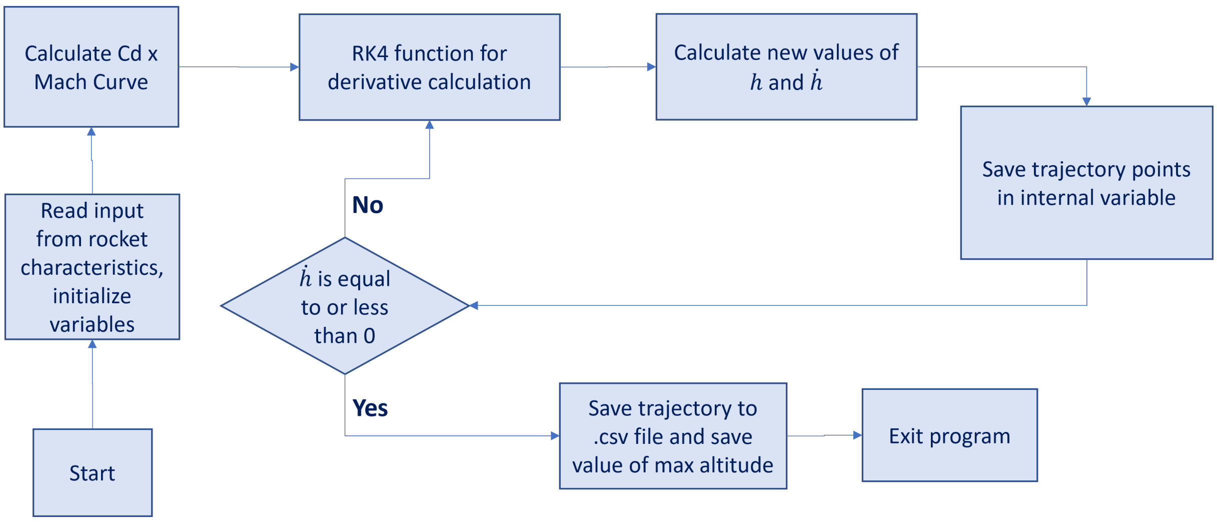

Calculate Cd x Mach Curve
RK4 function for derivative calculation
Calculate new values of h and $\dot{h}$
Save trajectory points in internal variable
Read input from rocket characteristics, initialize variables
No
$\dot{h}$ is equal to or less than 0
Yes
Start
Save trajectory to .csv file and save value of max altitude
Exit program

Figure 21 – Flowchart of the simulation of the trajectory of a rocket

As presented in Section 3.1 the system can be considered a vectorized system with variables V and h. Derivatives are calculated using Equation 2.41 and implemented inside a Runge-Kutta 4 method loosely based on the algorithm presented in (FRIEDMAN, 2018). Steps are subsequently calculated until the value of $V \approx 0$ when the function exits writing output '.csv' files if requested and returns the values of the maximum launch height achieved. An optional "verbose" argument was added to print regular status of the simulation on the Python Console.

3.3.6.1 Auxiliary Functions

Some auxiliary functions were created to keep the main code as simple as possible. One examples of this is the the profile drag estimation function. This was calculated using the same approach as described on Section 2.2.1, for each point in time evaluated, as the formulation.

Other auxiliary function implemented was the one that calls the supersonic drag module, this was implemented in this way to allow the possibility of other drag implementations in an simple way. Perhaps the simplest function implemented is the calculation of the local g value, using Newton's Universal Gravitation Law and the fact that on the surface of the earth $g_0 = 9.81m/s$. Another auxiliary function was the one that calculates the full value of F taking the value F^* and P_e determined by the rocket object and the value of P_{AMB} using Equation 2.30.

3.3.7 Genetic Evaluation Module

The objective of this module is to manage the genetical diversity of the populations, it's main processes are the generation of random attributes, the "mating" of a pair of rockets, the execution of the simulations and the posterior ranking of the rockets according to their scores and finally, the creation of a new generation based on the results of the last iteration.

3.3.7.1 Random Rocket Generation

The random rocket generation module creates a data structure with the attributes chosen as genes with random values. Limits are set according to the topics discussed on Sections 3.1 and 3.2. Within the established limits for all continuous variables, random uniform distributions are used to generate values in each interval, due to the relatively big interval in throat area ratio for the end-burner configuration, the distribution was considered uniform within the logarithm values. Discrete variables (nosecone type) are determined using a random choice algorithm between the different possibilities. In this way, a data structure is created that has random values for each desired variable and has the potential to analyze any value within the defined optimization space.

3.3.7.2 Mating method

Mating is achieved by mixing the genes of two different rockets. This function uses a random choice function to choose between the rockets chromosomes of each of the specified rockets to be mated for each of the genes in the optimization. While theoretically this process can be applied to mate any natural number of rockets, an implementation for the mating of only to rockets was opted for.

A random mutation possibility was added. This method generates an array filled with boolean values according to a possibility p_{mut} of a given element of the array being set as True. The possibility of none of the n_{genes} genes in a g being True, is given by $(1 - p_{mut})^{n_{genes}} \approx 1 - p_{mut} \cdot n_{genes}$. This approximation, true for small values of p, gives us valuable insight about the probability necessary for mutations to occur while generating a new rocket by the mating process.

3.3.7.3 Selection method

The selection method can be defined as one of the main pillars of the optimization, this method receives as input, between others, a list containing data structures containing genes as specified in the Random Rocket Generation method, this list can be defined as a generation. The module initializes the rocket object for each element in the generation and then gives it as input to the physics simulator to obtain the maximum altitude reached and overall score. Once all elements were simulated, the method creates a list using the Rocket Class' export method that contains the score and other relevant information about each rocket, the list is then sorted sorts the elements according to the score obtained by the rockets and sent as an output to the main module. The method saves the sorted list as a '.csv' file for future reference.

3.3.7.4 Creation of New Generations

The creation of new generations is also an important part of the GA, as it has to balance the spreading genes of well-performing elements, but at the same time has to allow enough variability to achieve local minimums. To much variability will hinder or even impossibilitate convergence, but a low variability will have a higher chance to converge to a local minimum.

The logic adopted in the generation creation algorithm was the following:

- each generation will have a number of fresh randomly generated individuals
- mating between the best performing individuals will be prioritized
- while mating between the best performing individuals will be prioritized, mating with low performing individuals will be possible
- the best performing randomized individual of a generation will forcibly mate with the overall best performing individuals
- the best performing individuals of one generation will be "cloned" to the next generation
- random mutations will be possible

Thus, each new generation, except for the first one (which is totally randomized), is composed of:

- **10%** of individuals that are copies of the best performing 10% of previous generation
- **10%** of individuals that are the result of mating random individuals of the best performing 30% of previous generation within themselves with $p_{mut} = 0$
- **20%** of individuals that are the result of mating random individuals of the best performing 50% of previous generation within themselves with $p_{mut} = 0.1$

- **20%** of individuals that are the result of mating random individuals of the best performing 50% of previous generation with random individuals overall with $p_{mut} = 0.05$
- **10%** of individuals that are the result of mating the best randomized individual with of the best performing 10% of previous generation with $p_{mut} = 0$
- **10%** of individuals that are clones of the best performing 10% of previous generation with $p_{mut} = 0.2$
- the remaining **20%** of individuals are randomized in each generation

3.3.8 Wave Drag Module

The complexity of the calculations and the amount of auxiliary functions necessary for the calculations of the supersonic drag coefficient justified a separate module. The program generates the curve $C_D^{SUP} x M_{ach}$ for a given rocket to be used on the module described in Section 3.3.6. This output is given in the form of an interpolating function of M_{ach}.

The approach given to simplify the theory presented in Section 2.2.2 was instead of trying to find the areas of intersection of an oblique plane passing on a given point X, the program determines the x coordinate of the plane passing through each of a given set of points on the body's outer layer or fin. This value can be easily calculated through the Equation 3.8. The $\pm$ refers to the fact that the plane can have a positive or negative inclination in respect to a point.

$$x_{plane} = x_p \pm r_p \cdot cotan(\mu^M) = x_p \pm r_p \cdot \frac{\sqrt{M_{ach}^2 - 1}}{M_{ach}} \tag{3.8}$$

The following process is repeated for a distribution of M_{ach} between 1.2 and 5, based on a linear distribution of the sine of each mach angle. This last was done in order to better approximate for low values of M_{ach}, where the curve has greater variation.

3.3.8.1 Revolution Body area distribution

In order to simplify calculations, as the main rocket body is a body of revolution by itself. The area intersected by the plane is the same for any value of θ. Thus, the areas of intersection were calculated outside of the θ loop and added to the areas calculated for the fins on each value of θ.

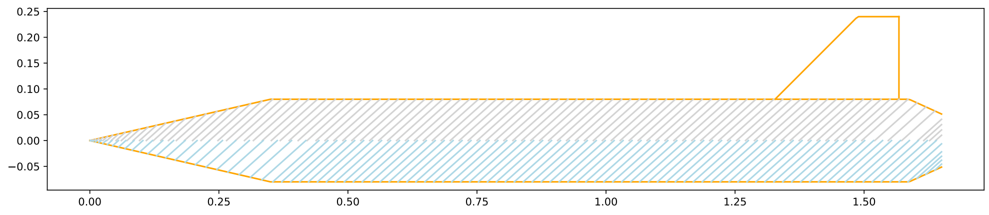


Figure 22 – Intersections of $M_{ach} = 1.5$ planes and the rocket's main body, separated in upper (orange) and lower halves (blue)

For each of the segments drawn in Figure 22, equally spaced points were taken over them with x,y coordinates. Having the radius interpolation function in $r(x)$, the maximum z coordinates to each side are found through Equation 3.9. Thus, the perpendicular projection to the flow of this cross-section is given by Equation 3.10.

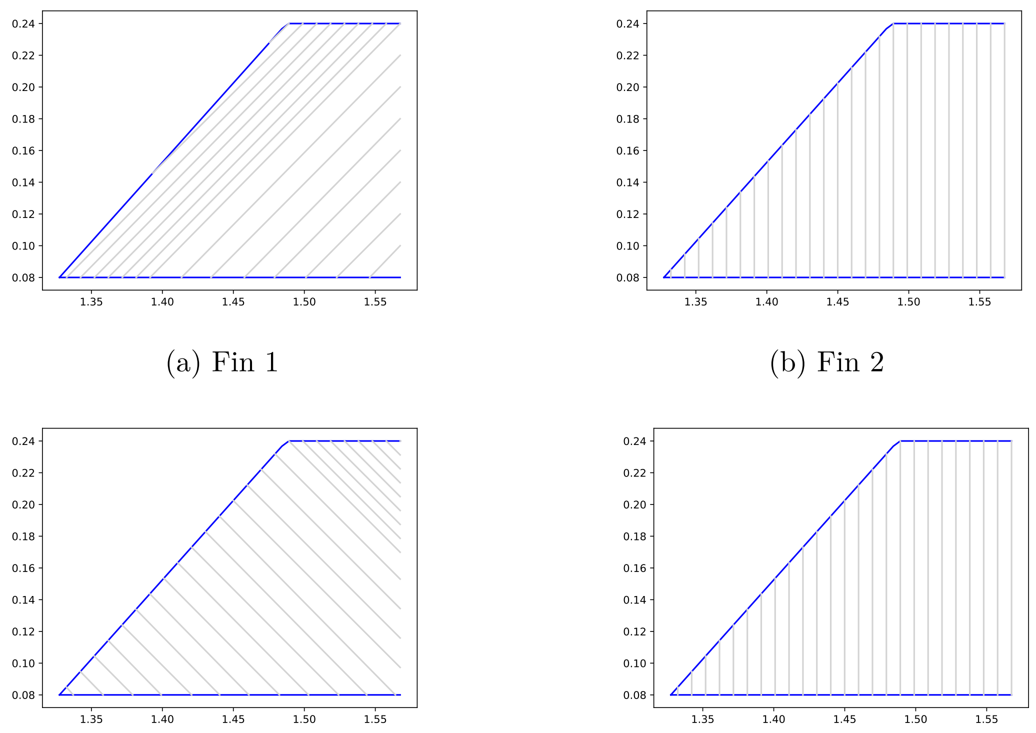
$$z_{hull} = \pm\sqrt{r(x) - (x - x_p)^2 - y^2} \tag{3.9}$$

$$S(x_{plane}) = \int_0^y 2 \cdot abs(z_{hull})\, dy \approx \sum_{i=0}^{n} \frac{z_{i+1} + z_i}{2} \cdot (y_{i+1} - y_i) \tag{3.10}$$

3.3.8.2 Fins area distribution

Considering an arbitrary fin aligned with the plane with $\theta = 0$ position as 0 and considering N fins evenly and radially distributed on the rocket body. In each fin, a plane determined by the fin will be considered as a local system of reference with the x position as one coordinate and the radial position as the other. For the $i < N$-th fin, it can be demonstrated that the effective inclination of the reference plane and the one determined by the fin is given by the Equation 3.11

$$\mu_{ef}^{M}(i) = acos(cos(\mu^{M}) \cdot cos(\theta + 2 \cdot \pi \cdot \frac{i-1}{N})) \tag{3.11}$$



(a) Fin 1 (b) Fin 2

(c) Fin 3 (d) Fin 4

Figure 23 – Intersection of the $M_{ach} = 1.5$ planes and the fins for $\theta = 0°$

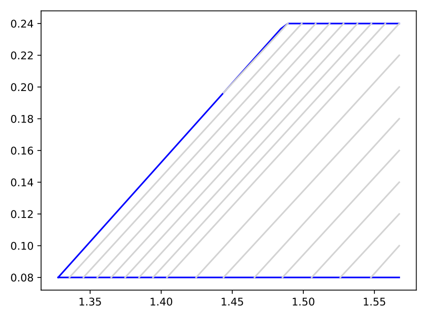
For angles in with a positive derivative, extra points were added on the trailing edge to better approximate the curve. An example of the intersections in the fins can be found in Figures 23 and 24

For each line segment presented in Figures 23 and 24, the projected area is given by Equation 3.12. Being A and B the extremes of the segments.

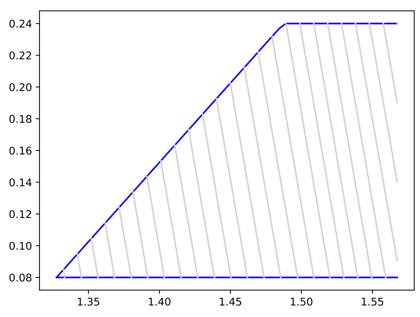
$$S(x_p) = \mid r_A - r_B \mid \cdot th_{fin} \tag{3.12}$$

3.3.8.3 Drag Coefficient calculation

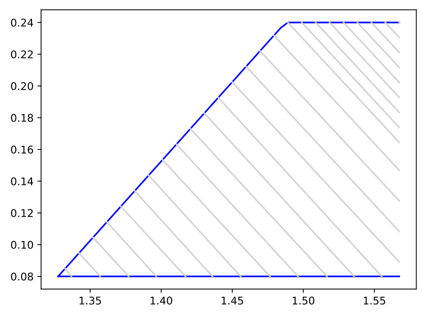
Having the area distribution of both parts of the rocket's body and having the area distribution of the fins, the maximum and minimum points in the x axis are determined, the curves are linearly interpolated, each interpolation function is evaluated for an equally spaced point distribution and summed. Through this process, a new curve that approximates the area of all intercepted objects is obtained. An example of this process is given in Figure 25.



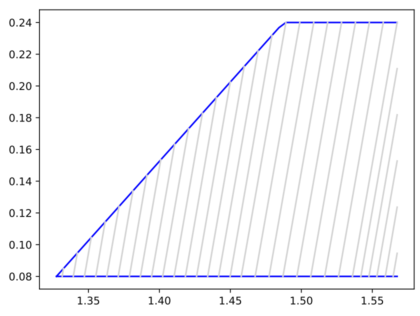
(a) Fin 1



(b) Fin 2



(c) Fin 3



(d) Fin 4

Figure 24 – Intersection of the $M_{ach} = 1.5$ planes and the fins for $\theta = 15°$

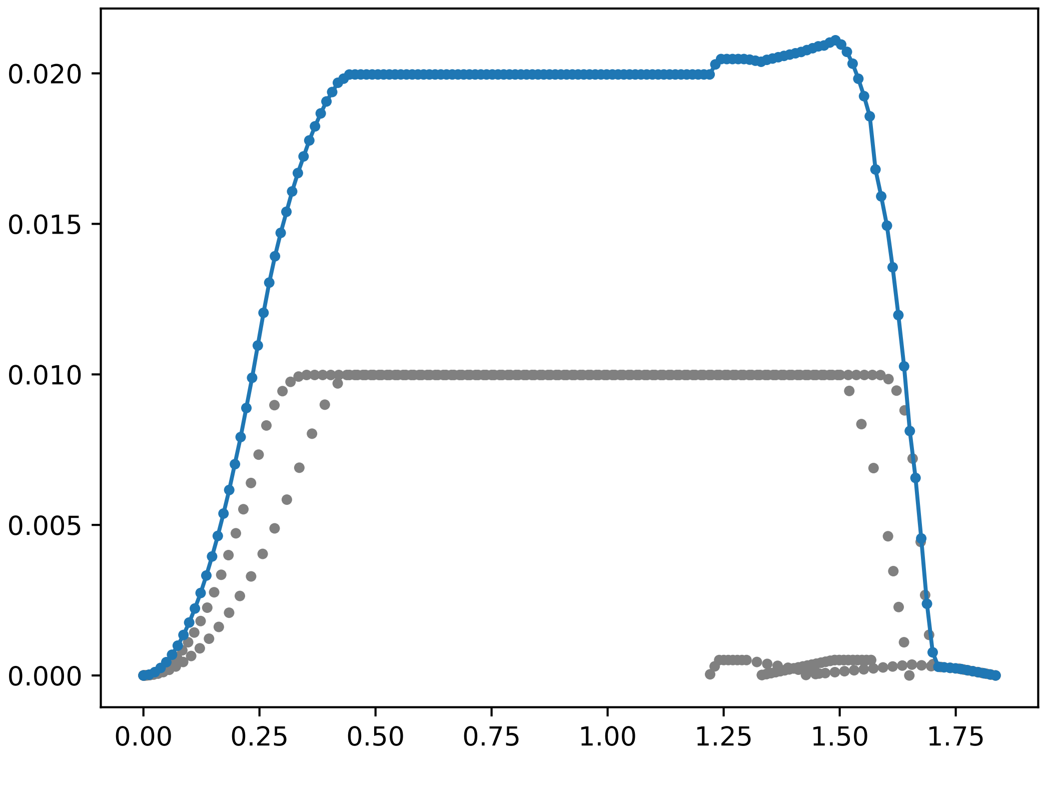


Figure 25 – Distribution of the total intercepted area for each $M_{ach} = 1.5$ and $\theta = 0°$ plane in blue, area distribution of each the separate components (upper/lower body and fins) plotted in gray.

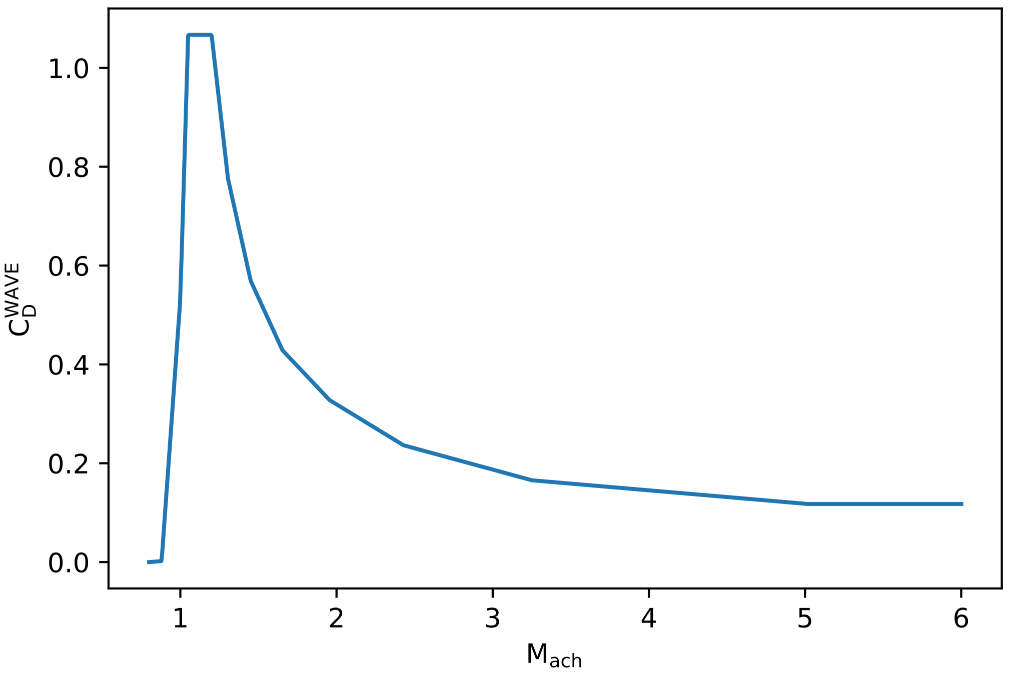


Figure 26 – Example of a C_D^{SUP} x M_{ach} obtained using the Wave Drag Module

After this step, the curve is differentiated in x to determine it's derivative using Equation 2.37 when possible and Equations 2.38 and 2.39 for the limits of the distribution. This derivative curve is then approximated with a Fourier series with Equation 3.13. Finally, the drag contribution relevant to the plane is given by Equation 2.15.

$$A_n = \int_{-x_0}^{x_0} S'(x) \cdot sin(n\frac{x-\bar{x}}{x_0} \cdot \pi)\, dx \approx \sum_{i=0}^{j} \frac{y_{i+1} \cdot sin(n\frac{x_{i+1}}{x_0} \cdot \pi) + y_i \cdot sin(n\frac{x_i}{x_0} \cdot \pi)}{2} \cdot (x_{i+1} - x_i) \tag{3.13}$$

As the distribution of fins is equally space, $C_D^{SUP}(\theta) = C_D^{SUP}(\theta + 2 \cdot \pi \cdot 1/N)$. Thus, the range of values of θ studied can be reduced from $[0, 2 \cdot \pi]$ to $[0, 2 \cdot \pi \cdot 1/N]$. The values are numerically integrated using the theory presented in 2.7.3.2.

After calculating C_D^{SUP} for each desired value of M_{ach}. The theory presented in Section 2.2.2.2 is applied to obtain the values for the transonic region. Even thought the theory presented in (RAYMER, 2004) recommends the use of smooth curves, the values are linearly interpolated for simplicity and to avoid possible negative values between M_{ach}^{cr} and M_{ach}^{dd}, which can happen with cubic or quadratic splines. Thus, a curve similar to Figure 26 is obtained.

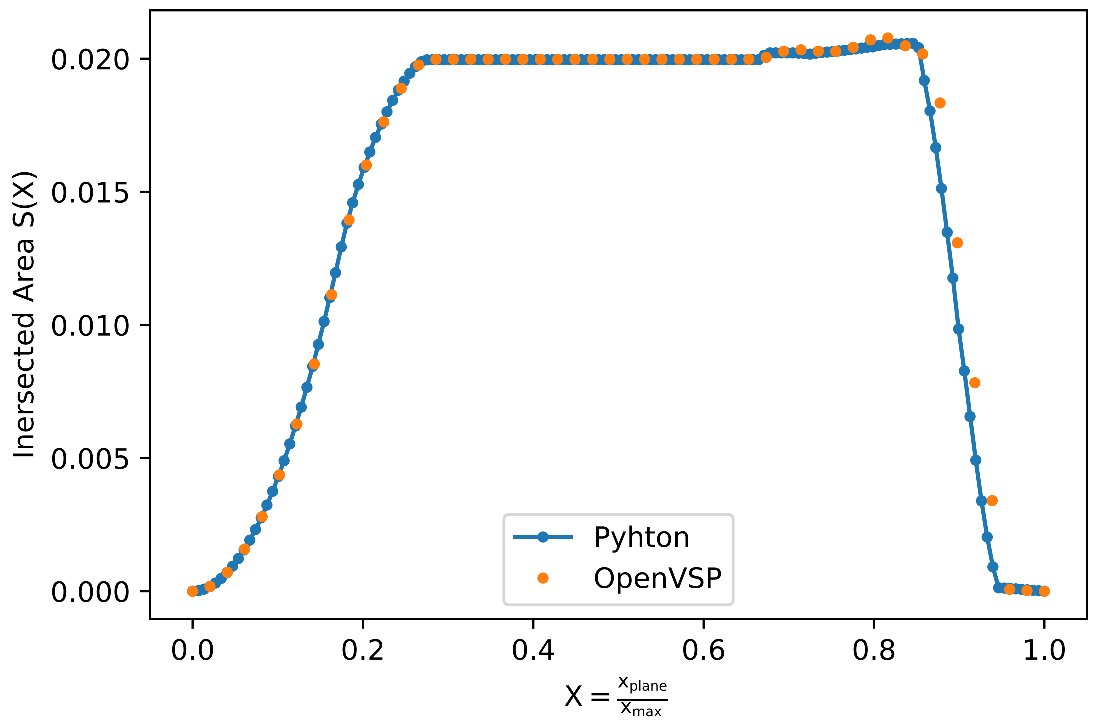


Figure 27 – Comparison between two area distributions, one obtained by OpenVSP and another calculated, for $M_{ach} = 1.5$ for the same rocket configuration

3.4 Program validation and testing

In order to guarantee the reliability of the modules developed on the previous sections, a test campaign was conducted before full program implementation.

3.4.1 Testing the Supersonic Drag module

The results given by the process described on Section 3.3.8 were compared to those obtained by the readily available software OpenVSP. For some rocket configurations the area distributions for a given M_{ach} obtained for each method were compared, one of these examples is shown in Figure 27. Rockets were approximated within reasonable similarity considering the differences between the software.

Furthermore, the resulting C_D^{SUP} x M_{ach} curved obtained were compared, as shown in Figure 28. While some differences are visible, they were deemed negligible and the program was considered successful in estimating the wave drag of the rocket.

3.4.2 Testing the Physics Integrator Module

The first test was a Simple Harmonic Oscilator in order to study the modules reaction to constantly changing acceleration values. Results of the test can be seen in Figure 29 for different time differentials used for over 1000 seconds worth of simulation.

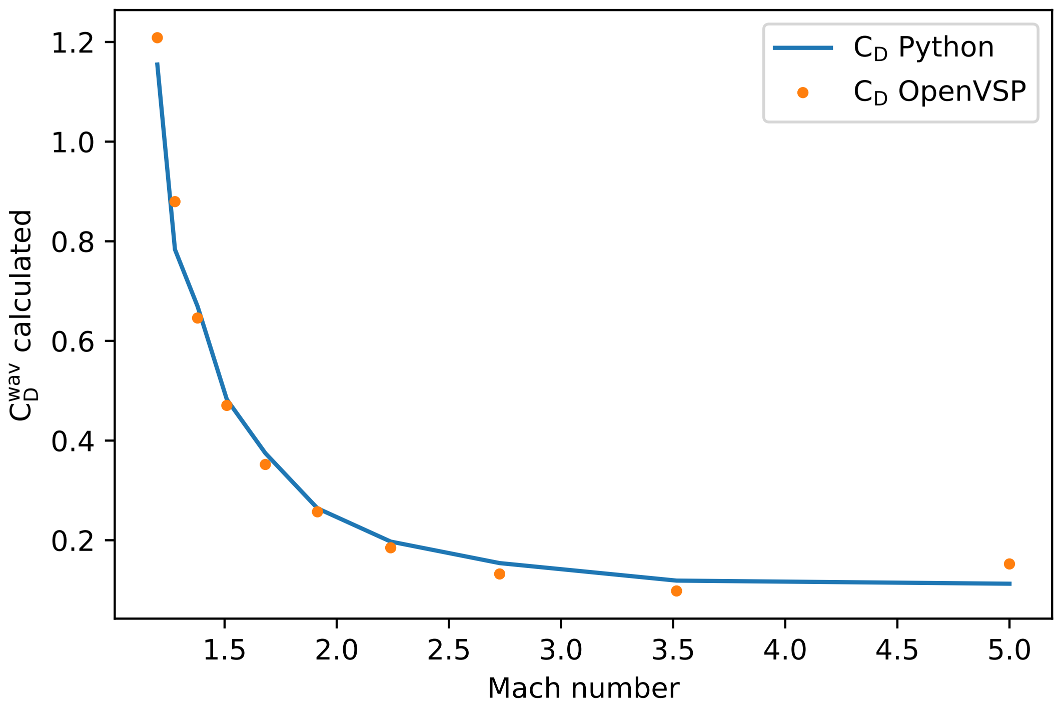
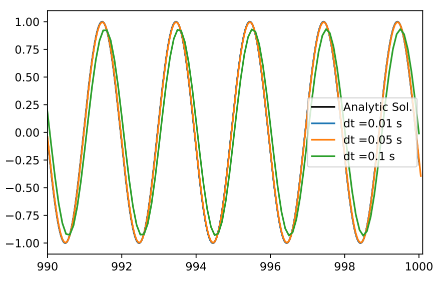
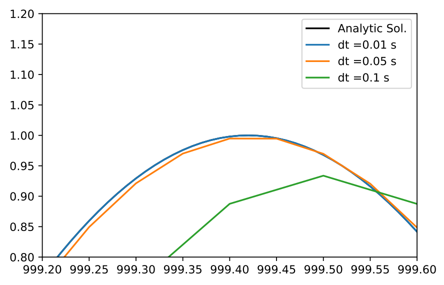


Figure 28 – Comparison between two area distributions, one obtained by OpenVSP and another calculated, for different M_{ach} for the same rocket configuration

Convergence towards the analytical solution can be seen for all time intervals studied. There was no visible difference between the calculations for $dt = 0.01$ and the analytical solution. Figure 30 shows the numeric energy dissipation of the system, it can be seem that the effect is almost negligible for $dt = 0.05s$ and $dt = 0.01s$.



(a) General view of seconds 990-1000



(b) Detailed view

Figure 29 – Comparison between analytic and numeric solution for a Simple Harmonic Oscilator

A second test was conducted simulating an ideal rocket in a zero-drag environment. The rocket was simulated using the same approach as for the optimization and compared with the solution provided by Tsiolkovsky's rocket equation (WIKIPEDIA, 2021). Constant values of g and $\dot{m}$ were assumed. The results of the simulation are exposed in Figure 31.

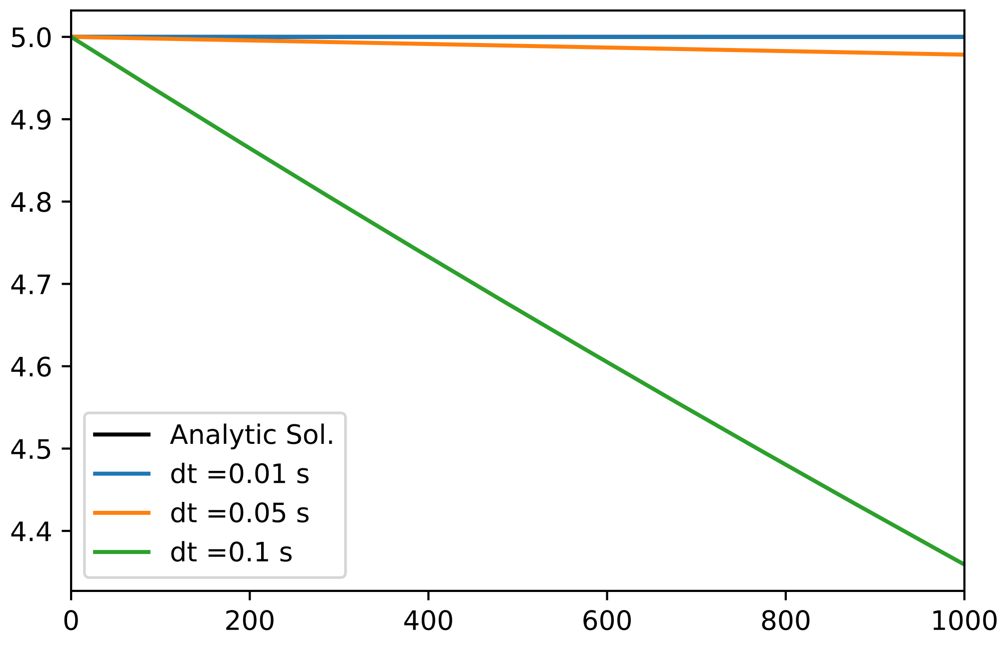


Figure 30 – Comparison of the numeric energy dissipation of the system for different values of dt

No visible differences are perceived.

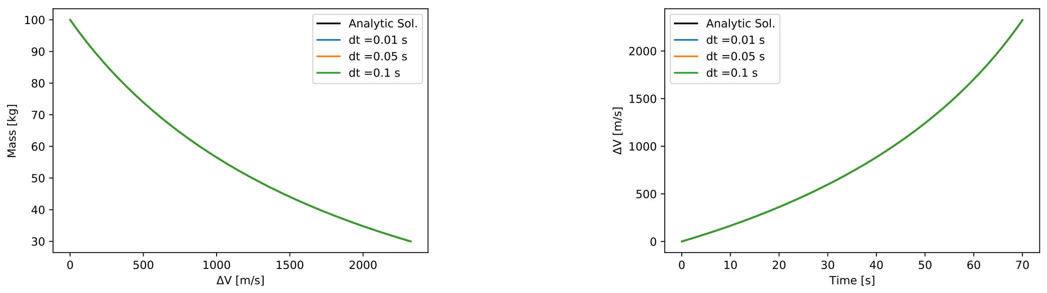
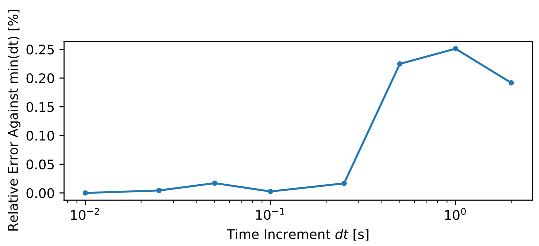


Figure 31 – Comparison of Rocket ΔV over mass and time variation between analytic and numeric solution for an ideal rocket in a drag-less environment

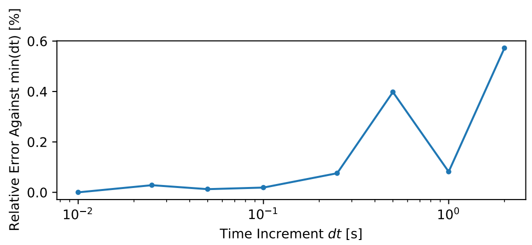
Also, numerical convergence was studied, the value of h_{max} was compared for different time intervals used on the simulation, no significant differences in maximum altitude reached ($\Delta h_{max} > 0.1\%$) were found for $dt < 0.25$ as can be seen in Figure 32. A default value of $dt = 0.05s$ was applied for the simulations

3.4.3 Tests of Propulsion Calculations

The value convergence of the iterative determination of P_C was tested for different starting values ranging from $0.001 \cdot P_C$ to $100 \cdot P_C$. Convergence to a value within the



(a) Simulation error in rocket 1



(b) Simulation error in rocket 2

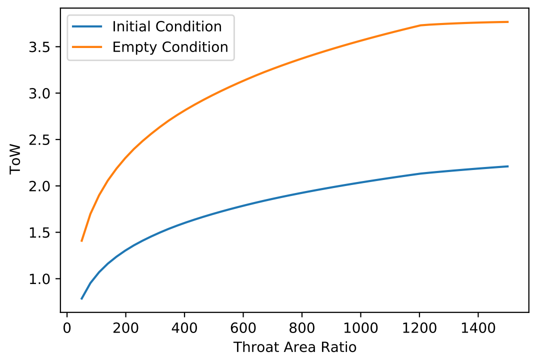
Figure 32 – Relative Variation of the maximum altitude reached for different time intervals

defined 0.1% tolerance was confirmed in all cases. For values of P_C, T_C and γ, output values as $\dot{m}$ and A_e were validated with tools available at (NASA-GRC, 2021).

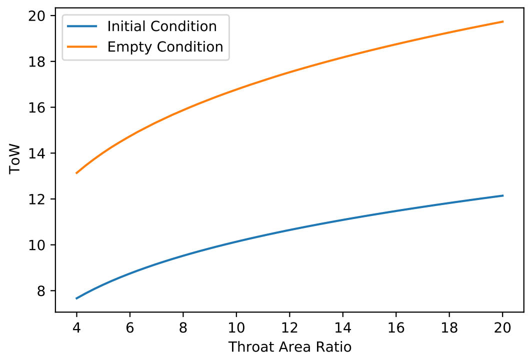
Furthermore, the values of A_B were calculated for different cases and compared to the programs output. A separate point of study was to compare the ToW of rockets for different throat ratios, and example of this can be found in Figure 33. While a low ToW in the end-burner configuration can cause an inadequate rocket, a too high ToW can cause mayor inconvenience too, as over-acceleration can damage the payload and other components. Nevertheless, it is expected that over-acceleration will be avoided by a high D that comes with a high velocity. So the optimal solution is expected to balance high acceleration and D.

3.4.4 Test of the Stability Calculations

The calculations of CP were compared to those exposed on (CULP, 2008) and (UTAH STATE UNIVERSITY, 2010). The results obtained by the program were the same as those exposed.



(a) End-burner rocket configuration



(b) Star rocket configuration

Figure 33 – Variations on ToW for different throat area ratios and both grain configurations studied for a given rocket

4 RESULTS AND DISCUSSION

As stated in Section 2.7, convergence is not guaranteed for GA and multiple runs can be needed to find optimal solutions. Thus multiple evolutionary simulations had to be made in order to achieve a solution. Furthermore, comparing the best rockets for different generations provides an insight of the effectiveness of convergence of the algorithms.

4.1 Optimization Executions for the End-Burner Configuration

Five parallel independent simulations where made spanning 50 Generations and the others (Sim 1-5). The results for the best performer of each generation are shown in Figure 34. We can see that while overall launch mass is usually reduced, Sim 1 and Sim 4 did not present convergence towards a viable solution, this could leave some space of improvement for the optimization routine. It's also important to note that the majority of the rockets presented on Figure 34 are not viable solutions to the problem. A mayor factor to this is the fact that the definition of A_B for this case is proportional to the square of the radius so rockets with higher TOW will have a lower length ratio. This lower length ratio causes mayor difficulties for the stability of the rocket.

A distribution of maximum altitude reached by total launch mass including all viable rockets simulations is displayed in Figure 35. For this, only rockets that took no penalties other than altitude-related were considered. Due to a high number of non-viable rockets on the simulation no direct frontiers are easily seen on the Figure.

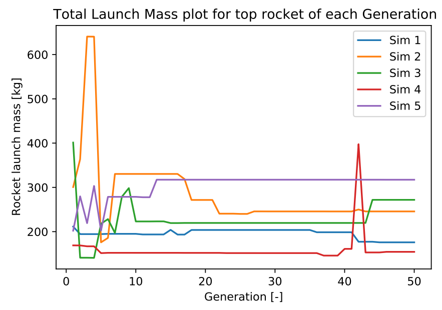
4.1.1 Best performing rocket obtained through optimization

The best characteristics of the best performing rockets of Sim 2,3 and 5 are shown on Table 6

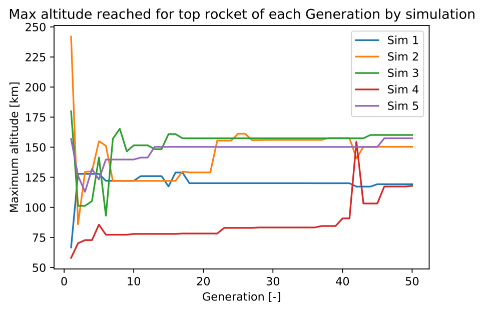
Variable	Sim 3	Sim 2	Sim 5	Unit
d	0.315182	0.307826	0.336031	m
Rocket length ratio	9.906129	8.740689	7.719809	-
Nose type	conic	conic	ojive	-
Nose length ratio	0.251942	0.161944	0.058765	-
Fin height	1.089	1.018	0.866	-
Fin position	0.0086	0.0101	0.0158	-
Throat area ratio	921.1197	818.8286	746.0239	-
Total Initial Mass	271.9	245.7	317.3	kg

Table 6 – Rockets obtained through optimization of the end-burner configuration

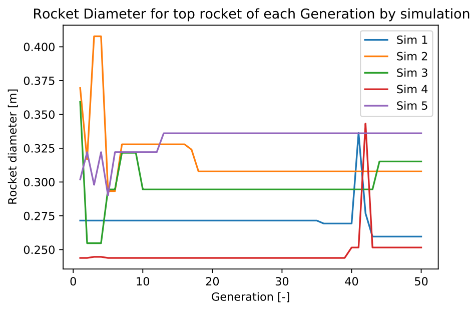
Convergence towards a $d_{max} \approx 0.3m$ can be seen. No definitive nose configuration can be concluded, but elongated conic noses seem to be the most convenient. The fin's



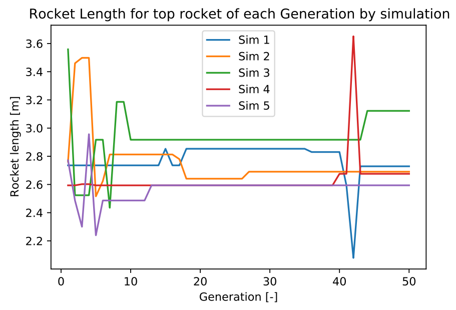
(a) Launch weight by generation



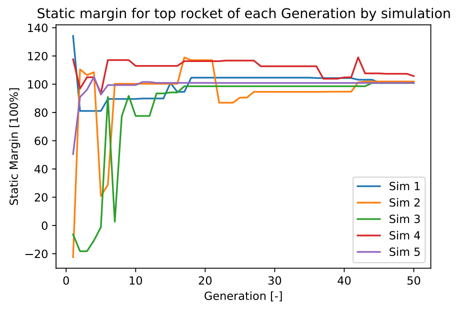
(b) Max Altitude by generation



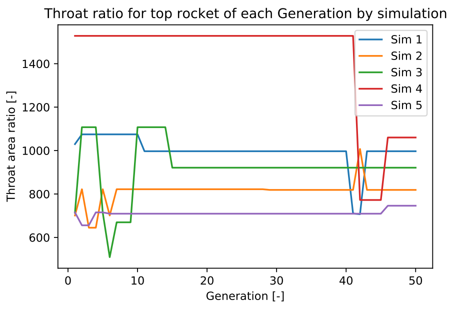
(c) Diameter by generation



(d) Length by generation



(e) Static Margin [%] by generation



(f) Throat ratio by generation

Figure 34 – Values of different variables for the best performing rocket of each generation for each of the simulations Sim 1-5 for the end-burner configuration

height should be approximately equal to d_{max} and positioned further back in the rocket. All rockets present $I_{sp} \approx 260$ and $M_R \approx 20\%$ as shown in Table 7, all these values are within the references presented on (SUTTON; BIBLARZ, 2000). While P_C values are in the upper end of the recommended values, the value of t_{wall} is calculated to provide safety on each case. Convergence towards an optimum value of MS_{min} can be seen.

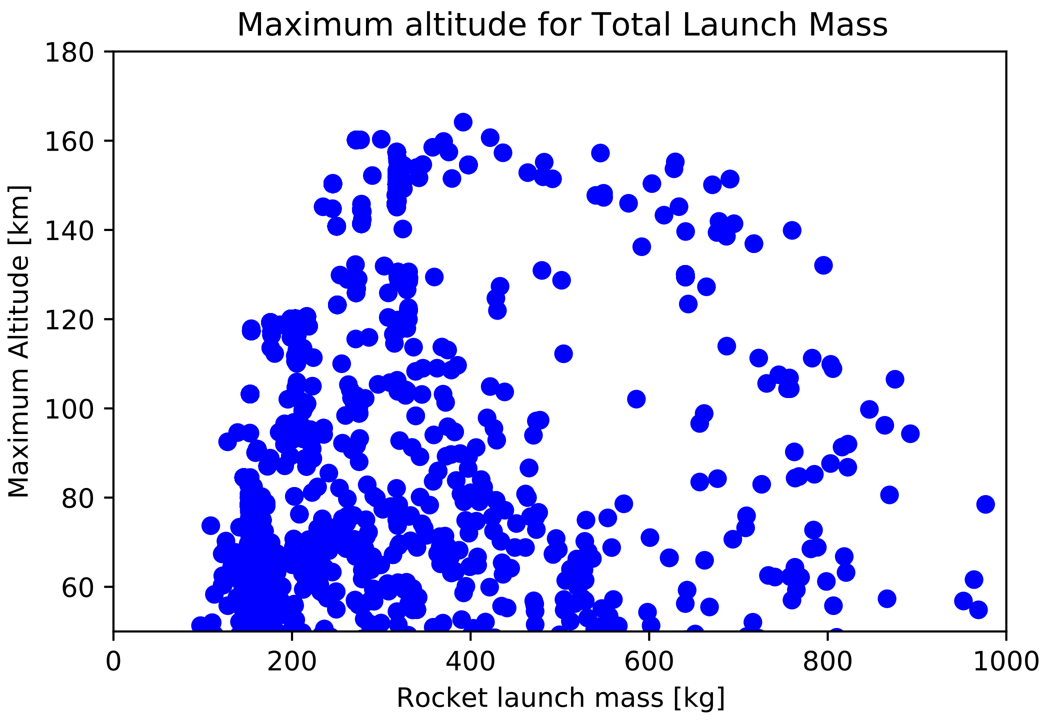


Figure 35 – Distribution of maximum altitude reached by total launch mass for the end-burner configuration

Variable	Sim 3	Sim 2	Sim 5	Unit
MS_{min}	1.01	1.02	1.01	-
P_C	15.73	13.26	11.55	Mpa
M_{ach}^{ex}	3.87	3.79	3.73	-
T^e	1095	1122	1144	K
$\dot{m}$	1.02	0.95	1.12	Kg/s
h_{max}	160.16	150.34	157.45	Km
M_R	22%	21%	18%	-
I_{sp}	261.9	259.8	258.1	s
Total Initial Mass	271.9	245.7	317.3	Kg

Table 7 – Main simulation results for the end-burner configuration

4.1.2 Simulation of the best performing rocket

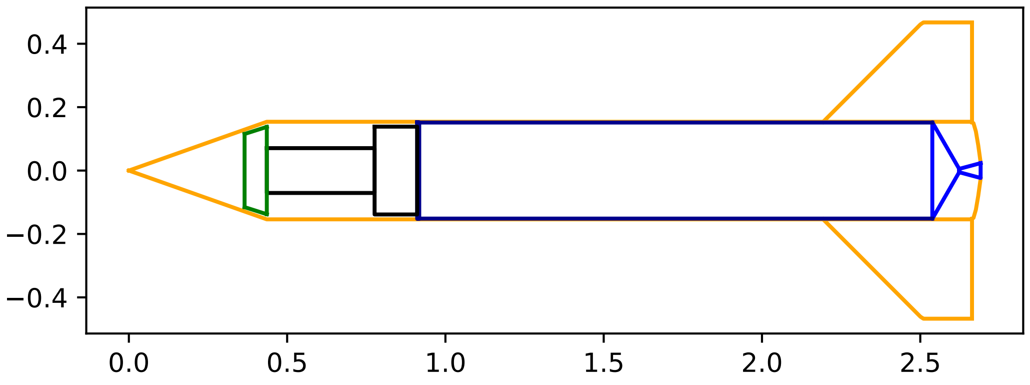
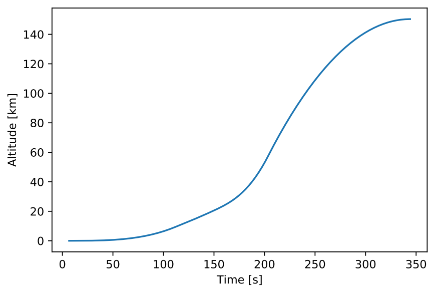


Figure 36 – Rocket obtained through optimization of the end-burner configuration

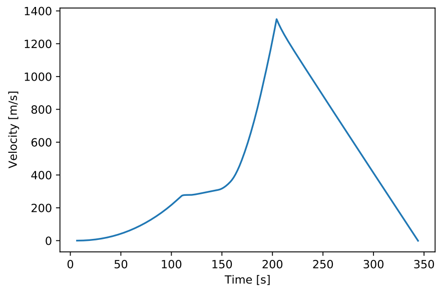
The rocket from Sim 2, presented in Figure 36, was the best performer of the whole set of simulations, it's trajectory is shown in Figure 37. It's possible to see the effects of the wave drag in the speed and acceleration graphs. It's also possible to see the increase in thrust due to the drop of P_{AMB}.

4.2 Optimization Executions for the Star Configuration

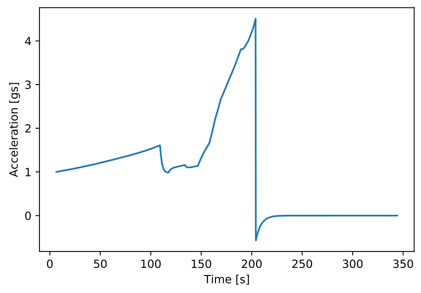
Five parallel independent simulations where made spanning 40 Generations each (Sim 1-5). The results for the best performer of each generation are shown in Figure 38. While launch weight clearly decreases by iteration, the same does not apply for maximum altitude attained which for Sim 2 and 1 seems to constantly be above the required. Nevertheless, while Sim 2 presents convergence towards a higher launch mass value while achieving higher altitudes than required, the same does not apply to Sim 1. This means, a similar rocket to the one that Sim 1 convergence shows can be achieved with less mass (removing propellant might be enough in this case) that still reaches h_{REF}.



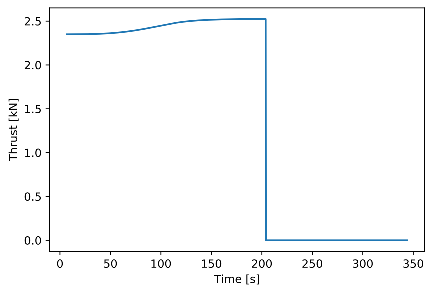
(a) Altitude over time



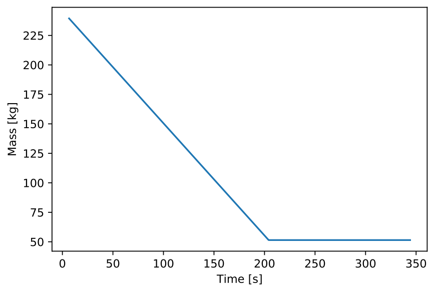
(b) Velocity over time



(c) Acceleration over time



(d) Thrust over time

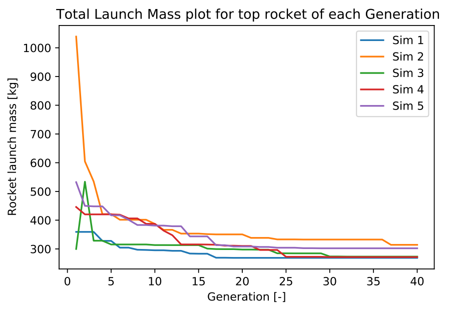


(e) Rocket mass over time

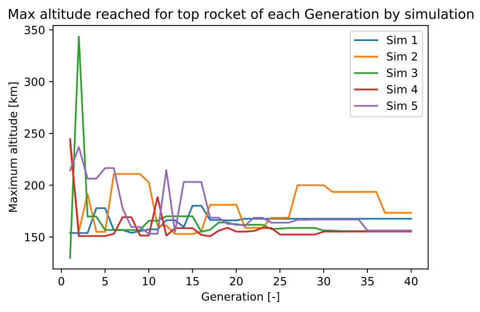
Figure 37 – Data obtained by the simulation of the best scoring rocket obtained in Sim 1-5

Some expected tendencies in all Simulations were the tendency a high nose ratio, all tending towards the maximum value of 0.3, and a high length ratio, all tending towards the maximum value of 20. An unexpected tendency is the programs tendency towards conic nose types, which was consistently repeated in all Simulations. A mayor factor to this is the fact that the definition of A_B for this case is proportional to the product of the radius and the rocket's length so rockets with higher TOW will have a higher length ratio.

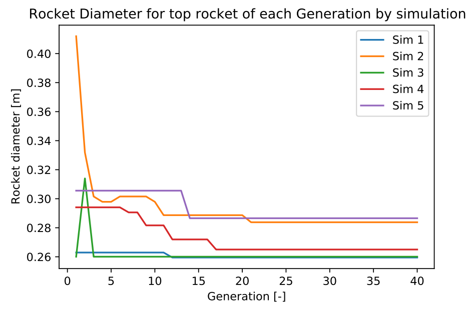
Another interesting graph obtained by the simulations is displayed in Figure 39. This represents the distribution of maximum altitude reached by total launch mass including all simulations. For this, only rockets that took no penalties other than altitude-related were considered. There seems to be a visible Pareto frontier on the



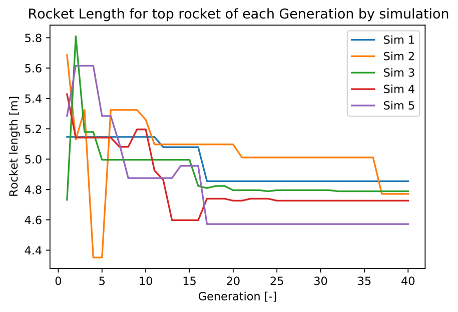
(a) Launch weight by generation



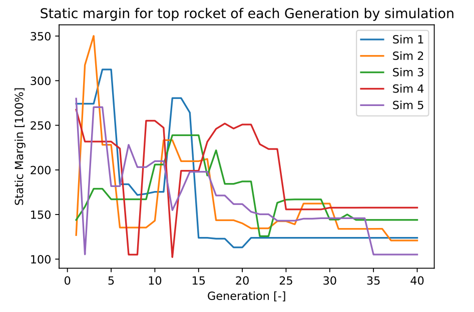
(b) Max Altitude by generation



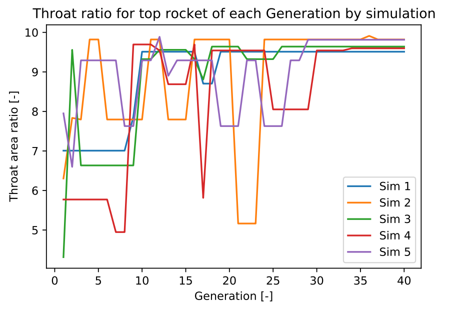
(c) Diameter by generation



(d) Length by generation



(e) Minimum static margin by generation



(f) Throat ratio by generation

Figure 38 – Values of different variables for the best performing rocket of each generation for each of the simulations Sim 1-5 in the star configuration

4.2.1 Best performing rocket obtained through optimization

The rocket with the best performance was the one obtained in generation 50 in Sim 1. It's characteristics are presented on Table 8.

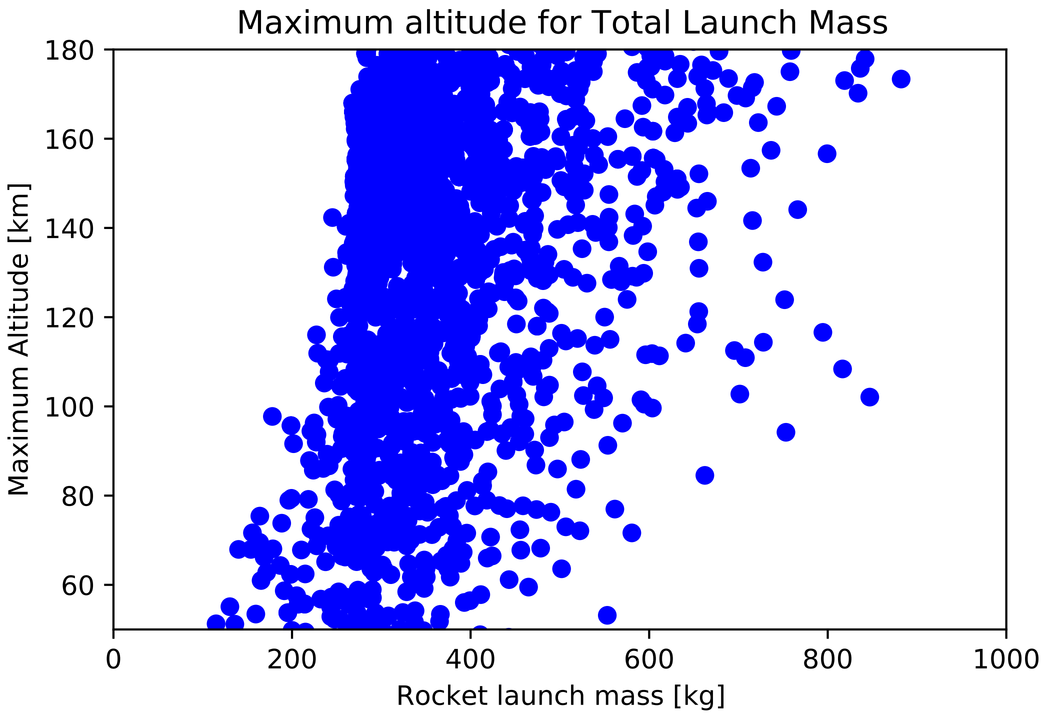


Figure 39 – Distribution of maximum altitude reached by total launch mass

Variable	Magnitude	Unit
d	0.259493	m
Rocket length ratio	18.66505	-
Nose type	conic	-
Nose length ratio	0.297733	-
Fin height	0.808597	-
Fin position	0.007239	-
Throat area ratio	9.955541	-
Total Initial Mass	267.9	kg

Table 8 – Rockets obtained through optimization of the star configuration

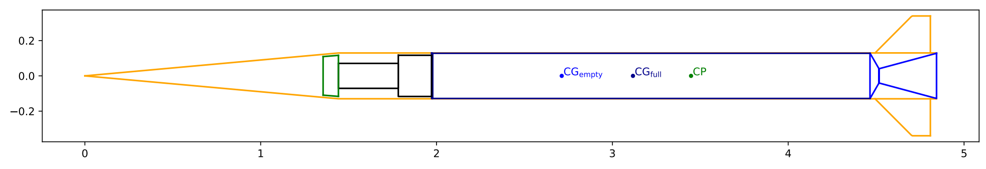


Figure 40 – Rocket obtained through optimization of the star configuration

As can be seen in Table 9 is over-performing in the requisites of h_{max} and MS_{min}, from which it can be concluded that there's space for further optimization. Other values as P_C, M_{ach}^{ex}, M_R and I_{sp} are within the common values cited in the references (SUTTON; BIBLARZ, 2000).

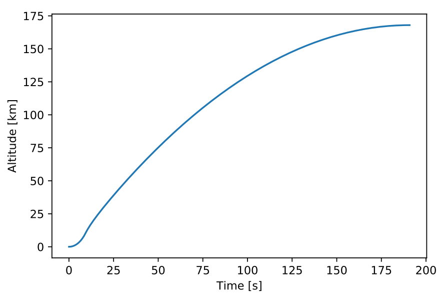
Variable	Magnitude	Unit
MS_{min}	1.27	-
P_C	6.70	Mpa
M_{ach}^{ex}	3.32	-
T^e	1307	K
$\dot{m}$	23.06	Kg/s
h_{max}	167.98	Km
M_R	21%	-
I_{sp}	245.1	s
Total Initial Mass	267.9	Kg

Table 9 – Main simulation results for the star configuration

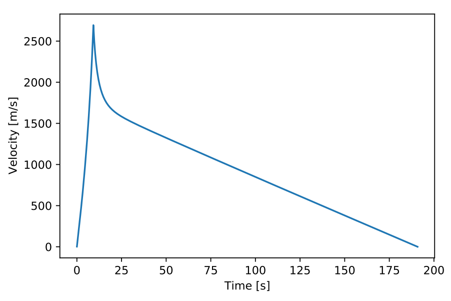
4.2.2 Possible drawbacks in the Simulation

An unsuspected problem was found in the simulation, while it was expected that rockets would naturally tend towards lower accelerations to minimize drag in the lower layers of the atmosphere, the opposite was detected. The graphics presented on Figure 41, are obtained by plotting the output for the best rocket in generation 50 in Sim 1.

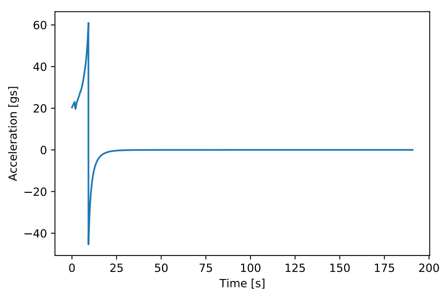
We can see accelerations peaks of over 50g, which is well above the values presented in (ESA, 2013?). Furthermore, Figure 40 shows a much longer combustion chamber



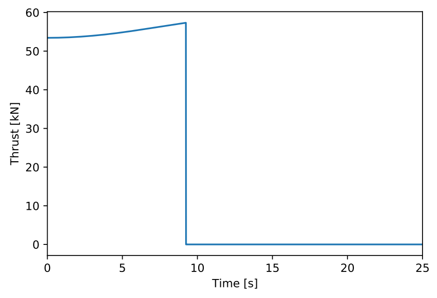
(a) Altitude over time



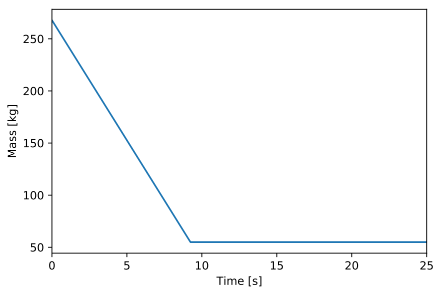
(b) Velocity over time



(c) Acceleration over time



(d) Thrust over time



(e) Rocket mass over time

Figure 41 – Data obtained by the simulation of the best scoring rocket obtained in Sim 1-5 in the star configuration

relative to rocket length than what can be seen in examples in (ESA, 2013?), (CYCLONE ROCKETRY CLUB, IOWA STATE UNIVERSITY, 2018), (KALRA et al., 2018), and others. This unusually long combustion chamber, due to the assumptions made on Section 3.1, signifies an unusually high A_B, which generates the ToW values. Other attempts to find viable solutions were made, as adding a penalty for over-acceleration and studying an end-burner propellant configuration as proposed in Figure 13. Still, no viable solutions was found for this configuration.

Nevertheless, further studies with more complex grain configurations could fix this problems, regressive and non-cylindrical configurations could be of great use for this, but they would require an implementation of variable thrust into the program. Another approach would be to reduce the l_C factor in Equation 3.3 by separating the rocket in different stages. Both of the presented solutions are outside the scope of this work.

4.3 Overall Views on the Performance of the Algorithm

While the optimization can be mostly considered a success, the rocket hybridization function could need some reviewing as the evolutionary convergence stopped prematurely in different simulations. This means that the genetic pool was no longer rich enough. Also, as suggested in 2.7 a gradient method could be added to the program at the end of the iterations to find local-minimums.

Also, considering the high number of penalized rockets with higher scores than viable solutions in some simulations, a review on score penalties is recommended. An increased severity on the penalties applied can be useful.It is important to note, that while penalties must have a degree of severity, they must also be "smooth" enough so that progress is rewarded.

In the dimension of elapsed time of the simulations, while highly dependent on the computer used, each simulation lasted 4-5h in a 3 year old computer i7, which was deemed as acceptable. The possibility of running multiple optimizations simultaneously allows for quick decision making

4.4 Possible Future Steps

This program was conceived as a steeping stone and first iteration for rocket optimization, nevertheless, some simplifications assumed to limit complexity also hinder possible optimal solutions. This section aims to show some points that could be expanded for future implementations of the program. Some ideas on which assumptions can be loosened and which changes these would required are detailed below.

4.4.1 Changing the rocket's body format

While apart from the nose, the rocket's body format is a cylinder of constant diameter, most of the program was initially designed to deal with rockets with non-constant diameter. The theory developed by (JONES, 1953) shows that this format does not minimize drag for a given volume. Furthermore, Whitcomb's area rule determines that the area generated by the fins should be reduced on the body to minimize drag.

While changes in the point generation function and the boat-tail generation function would need to be made. All programs are prepared to work with a rocket with a variable section diameter. Nevertheless, the fitting and form of the motor in this case should be studied.

4.4.2 Allowing for non-constant thrust-time curves

As the mass of the rocket decreases its mass over time, a constant thrust means a higher acceleration as time increases, which carries its own implications that where not studied on this work (payload, structures, increased drag, etc.). It would make sense therefore to study grain configurations with non-constant thrust-time ratios.

This can be achieved if a given thrust-time curve format is achieved, for example a linearly regressive configuration with a given slope. The propulsion module calculates the thrust for the last time interval when A_B is equal to the chamber's internal area, this value can be used to create a time-dependent thrust function. A special consideration must be made in relation to the pressure which will be achieved in the chamber over time, as it will also not be constant.

4.4.3 Allowing variations on the fin's format

The most obvious way to variate the fin's format is to give each of the dimensions specified in 16 as a variable each. Other fin formats, in specific a quadrilateral form with an inclined trailing edge, can be obtained by adapting the fin definition function in the Class Rocket file. One study with a more detailed approach for a similar optimization is (BARBOSA; GUIMARãES, 2012).

The fin is defined as the area between two curves, one representing the tip chord and leading edge and the second and the other representing the root chord and trailing edge. An adjustment will be needed on the supersonic drag estimation as well, as in the cases of a positive slope intersection, points are generated on the trailing edge assuming it is perpendicular to the x axis.

5 CONCLUSION

This project was able to generate software that can simulate different aspects of a rocket and apply a genetic optimization algorithm to the variables that govern its main characteristics. Special care was taking to allow the easy customization of each module to achieve a wide arrange of goals. The software allows for reliable and speedy simulations of dynamic systems, estimations of supersonic wave drag and the rocket's propulsion system.

Different aspects that affect the project of a suborbital rocket were studied in order to obtain a simple and reliable implementation. Different examples were tested to guarantee the simulation's faithfulness to reality. The program allowed to determine an initial optimal configuration was obtained for a suborbital rocket to lift a payload of a 3U CubeSat to a 150km altitude by running 10 simulations of 50 generations with 50 individuals each. The simulations were divided into two main configurations based on the propellant disposition. One of the studied configurations was found to be not viable due to constrains not initially studied (maximum acceleration) but that will affect the rocket's project. Further ideas on how to take advantage of the developed program were studied.

BIBLIOGRAPHY

ADAMI, A.; MORTAZAVI, M.; NOSRATOLLAHI, M. A new approach to multidisciplinary design optimization of solid propulsion system including heat transfer and ablative cooling. 2016.

BARBOSA, A. N.; GUIMARãES, L. N. F. Multidisciplinary design optimization of sounding rocket fins shape using a tool called mdo-sonda. 2012.

BARROWMAN, J. S. **The Practical Calculation Of The Aerodynamic Characteristics Of Slender Finned Vehicles**. 1967.

BUSSE, J. R.; LEFFLER, M. T. **The Aerobee 100, 150, and 300 Series Sounding Rockets**. High Power Rockfry, 1997. Disponível em: <http://www.rasaero.com/dl_sounding_rockets.htm>.

CARBONNELLE, P. **PYPL PopularitY of Programming Language**. 2021. Disponível em: <https://pypl.github.io/PYPL.html>.

CHENEY, W.; KINCAID, D. **Numerical Mathematics and Computing, Sixth edition**. [S.l.]: Thomson Brooks/Cole, 2008.

CROWELL, G. A. The descriptive geometry of nose cones. 1996. Disponível em: <https://www.nakka-rocketry.net/articles/Descriptive_Geometry_Nosecones_Crowell_1996.pdf>.

CULP, R. **Barrowman Equations**. 2008. Disponível em: <http://www.rocketmime.com/rockets/Barrowman.html>.

CYCLONE ROCKETRY CLUB, IOWA STATE UNIVERSITY. **Team 20 Technical Project Report for the 2018 IREC**. 2018. Disponível em: <https://www.soundingrocket.org/uploads/9/0/6/4/9064598/20_project_report.pdf>.

DEPARTMENT OF DEFENSE, UNITED STATES OF AMERICA. **MILITARY HANDBOOK: METALLIC MATERIALS AND ELEMENTS FOR AEROSPACE VEHICLE STRUCTURES**. [S.l.], 1998.

ESA. **Sounding Rockets**. 2013? Disponível em: <http://wsn.spaceflight.esa.int/docs/EUG2LGPr3/EUG2LGPr3-6-SoundingRockets.pdf>.

_____. Low earth orbit. 2020. Disponível em: <https://www.esa.int/ESA_Multimedia/Images/2020/03/Low_Earth_orbit>.

ETKIN, B.; REID, L. D. **Dynamics of Flight - Stability and Control**. [S.l.]: John Wiley & Sons, Inc., 1996.

EXOTRAIL SA. **Exotrail Products**. 2021? Disponível em: <https://exotrail.com/product/>.

FACULTY OF AEROSPACE ENGINEERING, TU DELFT. **SOME TYPICAL SOLID PROPELLANT ROCKET MOTORS**. 2013.

FERRATO, E. et al. Development roadmap of sitael's ram-ep system. 2017.

FRANCO, N. M. B. **Cálculo Numérico**. [S.l.]: Pearson Universidades, 2006.

FRIEDMAN, A. **SDOF-Simulator**. 2018. Retrieved Jul. 25, 2021. Disponível em: <https://github.com/apf99/SDOF-Simulator>.

HAYES, W. D. **Linearized Supersonic Flow**. 1947.

HOUGHTON, E.; CARPENTER, P. **Aerodynamics for Engineering Students**. [S.l.]: Butterworth-Heinemann, 2003.

ICAO. **Manual of the ICAO Standard Atmosphere**. [S.l.], 1993.

JONES, R. T. Naca report 1284: Theory of wing-body drag at supersonic speeds. 1953.

KALRA, A. et al. **Design and Construction of a Solid Experimental Sounding Rocket, Amy**. 2018. Disponível em: <https://www.soundingrocket.org/uploads/9/0/6/4/9064598/40_project_report.pdf>.

KULU, E. **Nanosatellite & Cubesat Database**. 2021. Retrieved August 22, 2021. Disponível em: <https://www.nanosats.eu/database>.

LABUDDE, E. V. **Extending the Barrowman Method for Large Angles of attack**. 1999. Disponível em: <http://ftp.demec.ufpr.br/foguete/bibliografia/Labudde_1999_CP.pdf>.

MORGADO, F. M. P. **Coupled Preliminary Design and Trajectory Optimization of Rockets using a Multidisciplinary Approach**. 2019.

MORGAN-STANLEY. **Space: Investing in the Final Frontier**. 2020. Retrieved August 15, 2021. Disponível em: <https://www.morganstanley.com/ideas/investing-in-space>.

NAGEL, G. W.; NOVO, E. M. L. de M.; KAMPEL, M. Nanosatellites applied to optical earth observation: a review. 2020.

NAKKA, R. **Richard Nakka's Experimental Rocketry Web Site**. 1997. Disponível em: <http://www.nakka-rocketry.net>.

NASA-GRC. **Interactive Nozzle**. 2021. Disponível em: <https://www.grc.nasa.gov/www/k-12/airplane/ienzl.html>.

NASA SOUNDING ROCKETS PROGRAM OFFICE. **NASA Sounding Rockets User Handbook**. Wallops Island, Virginia, United States of America, 2015.

NATIONAL CENTER FOR BIOTECHNOLOGY INFORMATION. **PubChem Compound Summary for CID 24639, Ammonium perchlorate**. 2021. Retrieved Aug. 13, 2021. Disponível em: <https://pubchem.ncbi.nlm.nih.gov/compound/Ammonium-perchlorate>.

NAVARRETE-MARTIN, L.; KRUSB, P. Sounding rockets: analysis, simulation and optimization of a solid propellant motor using hopsan. 2017.

NISKANEN, S. **OpenRocket technical documentation**. 2013.

NOAA; NASA; USAF. **U.S. Standard Atmosphere 1976**. Washington, DC, United States of America, 1976. Disponível em: <https://www.ngdc.noaa.gov/stp/space-weather/online-publications/miscellaneous/us-standard-atmosphere-1976/us-standard-atmosphere_st76-1562_noaa.pdf>.

OKNINSKI, A. Multidisciplinary optimisation of single-stage sounding rockets using solid propulsion. 2017.

OPEN ROCKET. **Open Rocket Simulator**. 2021? Disponível em: <https://openrocket.info>.

OPEN VSP. **Open VSP**. 2021. Retrieved Aug. 12, 2021. Disponível em: <http://openvsp.org/>.

O'CONNELL, C. **NASA plans for deep space propulsion**. 2016. Retrieved Aug. 11, 2021. Disponível em: <https://cosmosmagazine.com/technology/antimatter-ion-drives-nasas-plans-deep-space-propulsion/>.

PALLONE, M. et al. Design methodology and performance evaluation of new generation sounding rockets. International Journal of Aerospace Engineering, 2018.

PEPERMANS, L. et al. Systematic design of a parachute recovery system for the stratos iii student built sounding rocket. 2018.

RAYMER, D. P. **Aircraft Design: A Conceptual Approach**. Sylmar, California, United States of America: American Institute of Aeronautics and Astronautics, 2004.

ROETGEN, R.; DESCH, M. **#04 Matt Desch, Iridium - Satellite Communications**. 2020. Spotify. (Space Business Podcast). Disponível em: <https://open.spotify.com/episode/1N8CAKmf9RJGhyXFzlR25h>.

ROGERS, M. J. B.; VOGT, G. L.; WARGO, M. J. **Microgravity: A Teacher's Guide With Activities in Science, Mathematics, and Technology**. 1997. Disponível em: <https://www.nasa.gov/pdf/62474main_Microgravity_Teachers_Guide.pdf>.

SILVA, G. Claudino e. **Estudo de ferramentas computacionais para simulação e visualização de aplicações de drones**. 2021.

SUTTON, G. P.; BIBLARZ, O. **Rocket Propulsion Elements**. [S.l.]: John Wiley & Sons, Inc., 2000.

THE CUBESAT PROGRAM,. **CubeSat Design Specification (1U – 12U) REV 14**. San Luis Obispo, California, United States of America, 2020. Retrieved Jun. 02, 2021. Disponível em: <https://www.cubesat.org/cds-announcement>.

______. **The Cubesat Standard**. 2021? Retrieved Jun. 02, 2021. Disponível em: <https://www.cubesat.org/about>.

TIOBE. **TIOBE Index for August 2021**. 2021. Disponível em: <https://www.tiobe.com/tiobe-index//>.

TRAN, P. H. N. et al. Development and test of an experimental hybrid sounding rocket. 2013?

USAF. **Recovery Systems Design Guide**. [S.l.], 1978.

UTAH STATE UNIVERSITY. **Longitudinal Vehicle (Pitch) Dynamics, Static and Dynamic Stability**. 2010. Disponível em: <http://mae-nas.eng.usu.edu/MAE_5900_Web/5900/USLI_2010/Flight_Mechanics/Section5.2.pdf>.

VANDAS, M. et al. Team 117 project technical report for the 2018 irec. 2018.

WAAL, L. R. de. **ISA Calculator**. 2019. Retrieved May. 25, 2021. Disponível em: <https://github.com/LukeDeWaal/ISA_Calculator>.

WADDINGTON, M. J. **Development of an interactive wave drag capability for the OpenVSP parametric geometry tool**. 2015.

WELLS, H. T.; WHITELEY, S. H.; KAREGEANNES, C. E. **SP-4402 Origins of NASA Names: SECTION V SOUNDING ROCKETS**. 1975. Retrieved July 17, 2021. Disponível em: <https://history.nasa.gov/SP-4402/ch5.htm>.

WIKIPEDIA. **Tsiolkovsky rocket equation**. 2021. Accessed: Aug. 29, 2021. Disponível em: <https://en.wikipedia.org/wiki/Tsiolkovsky_rocket_equation>.

Appendices

APPENDIX A – SUPERSONIC DRAG ESTIMATION PROGRAM CODE

The code of the supersonic drag module is presented below. For the code of the full project, please access *here*.

```python
'''
The following program is an implementation of the theory preseneted by Jones in the NACA Report 1284
applied to a rocket. Drag is studied on the win-body system composed of the rockets main body and the
fins.
'''
import numpy as np
from scipy.interpolate import interp1d
from Class_Payload import Payload
from Class_Rocket import Rocket
from matplotlib import pyplot as plt


def intersect_hull(x_hull, d_plane, f_interp, mach_number, direct=True):
    '''
    The following function finds second the point of intersection for a mach plane given
    the first point of intersection and the curve that determines the radius of the revolution body.
    Parameters
    ----------
    x_hull : x coordinate of the intersection of the plane and the hull
    d_plane : distance messured over the plane of intersection from the point to the intersection of
            the plane and the x axis
    f_interp : funtion that interpolates the section's radius of the rocket based on the x coordinate
    mach_number : mach number studied
    direct : Direction of the intersection, if the point of intersection has x<x_hull then use False,
            else, use True

    Returns
    -------
    (x,y) : coordinates of the intersection point, in of an error it returns
```

```
        the planes intersection with
            the x axis

    '''
    mach_cotg = np.sqrt(mach_number ** 2 - 1)
    mach_sin = 1 / mach_number
    mach_cos = mach_cotg * mach_sin
    direction = 1 if direct else -1
    y_hull = f_interp(x_hull)
    xs = x_hull + direction * np.linspace(0, 1, 200)[1:] * d_plane * mach_cos
    ys_line = y_hull - np.linspace(0, 1, 200)[1:] * d_plane * mach_sin
    ys_hull = [f_interp(x) if f_interp.x.min() < x < f_interp.x.max() else 100
        ** 3 for x in xs]
    n = np.argwhere(np.diff(np.sign(ys_line - ys_hull))).flatten()
    try:
        return xs[n[0] - 1], ys_line[n[0] - 1]
    except IndexError:
        return 0, y_hull - x_hull / mach_cotg


def trapezoid_integration(ys, xs):
    '''
    The following function gives the trapezoid integration of y values of a
        function
    for x values given. Values are assumed to be ordered and corresponding
        between
    arrays.
    Parameters
    ----------
    ys : Array of values to be integrated.
    xs : Can be either array or float. In the case of float, a constant
        spacing is assumed in
        between the y points to be integrated (x is recognized as "delta x").
        In case of an array it is matched with y values. Errors will rise if
            dimensions are
        not the same.


    Returns
    -------
    ans : Integrated value of the function
    '''
```

```python
    if type(xs) == np.float64:
        ans = sum((ys[1:] + ys[:-1]) / 2 * xs)
    else:
        ans = sum((ys[1:] + ys[:-1]) / 2 * (xs[1:] - xs[:-1]))
    return ans


def fin_segments(fin_xs, f_top, f_base, cos_eff, show_graph=False, interval=3):
    '''
    The following function calculates the intersections of the oblique mach
        plane with the
    fins of a rocket
    Parameters
    ----------
    fin_xs : x position in the fin
    f_top : function that interpolates the radially outer part of the fins for
        x
    f_base : function that interpolates the radially inner part of the fins
        for x
    cos_eff : effective cosine of the inclination of the mach plane towards
        the fin
    show_graph : boolean, determines the presentation of graphs of the
        intersection at the end
            of the excecution
    interval: picks only one of each n points to evaluate, used for faster runs

    Returns
    -------
    x_exports : coordinate of the intersection of each parallel mach plane and
        the x axis
    lengths : length of the desired segments

    '''
    sin_eff = np.sqrt(1 - cos_eff ** 2)
    r_top = f_top(fin_xs)
    lengths = []
    x_exports = []

    if show_graph:
        plt.figure()
        plt.plot(fin_xs, r_top, color='blue')
        plt.plot(fin_xs, f_base(fin_xs), color='blue')

    if cos_eff > 0.001:
        trailing_edge_points = np.linspace(r_top[-1], f_base(fin_xs[-1]),
            round(10 * (1 + cos_eff)))[1:-1]
        r_top = np.append(r_top, trailing_edge_points)
        fin_xs = np.append(fin_xs, np.ones(trailing_edge_points.shape) *
            fin_xs[-1])

    x_plane = fin_xs - r_top * cos_eff / sin_eff
    x_start_fins = np.argmin(x_plane)
    d_plane = np.sqrt(r_top ** 2 + (fin_xs - x_plane) ** 2)
    for (n_temp, x_top) in enumerate(fin_xs[x_start_fins + 1::interval]):
        error_intersect = False
        n = n_temp * interval + 1 + x_start_fins
        xs = x_top - np.linspace(0, 1, 100)[1:-1] * d_plane[n] * cos_eff
        ys_line = r_top[n] - np.linspace(0, 1, 100)[1:-1] * d_plane[n] *
            sin_eff

        if x_plane[n] < x_plane[0]:
            func = f_top
            ys_inter = [func(x) if func.x.min() <= x <= func.x.max() else 0 for
                x in xs]
            n_inters = np.argwhere(np.diff(np.sign(ys_line -
                ys_inter))).flatten()
            try:
                x_base = xs[n_inters[0]]
                y_base = func(x_base)
            except IndexError:
                error_intersect = True
        elif x_plane[n] <= fin_xs[-1] - f_base(fin_xs[-1]) * cos_eff / sin_eff:
            func = f_base
            ys_inter = [func(x) if func.x.min() <= x <= func.x.max() else 0 for
                x in xs]
            n_inters = np.argwhere(np.diff(np.sign(ys_line -
                ys_inter))).flatten()

            try:
                x_base = xs[n_inters[0]]
                y_base = func(x_base)
            except IndexError:
                error_intersect = True
        else:
            x_base = fin_xs[-1]
            y_base = r_top[n] - (x_top - x_base) * sin_eff / cos_eff

        if not error_intersect:
            lengths.append(np.sqrt((r_top[n] - y_base) ** 2))
            x_exports.append(x_plane[n])
            if show_graph:
                plt.plot([x_base, x_top], [y_base, r_top[n]], color='lightgray')

    return x_exports, lengths


def merge_areas(areas_merge, plot_graphs=False, point_number=200):
    '''
    This funtion is made to sum the values of different curves that do not have the same
    x points. It does so by finding the minimum and maximum values, and then interpolating
    each curve on the interval and summing. Extrapolations are considered as 0
    Parameters
    ----------
    areas_merge : Array. It has n elements, each of which is an array of two elements, each
            of them an array. The first an array of the "x" points and the second is the
            curve values for those points
    plot_graphs : Boolean. Optional. Determines wether or not a plot of the sum of the
            curves is generated at the end of the execution
    point_number : Optional. Numbers of points to be evaluated for the output. Default 200

    Returns
    -------
    x_areas_merge : x positions of the sums of the curves. A total of [point_number] elements
    total_area : the sum of the inputted curves for each point in [x_areas_merge]

    '''
    xs_lims = []
    for comb in areas_merge:
        xs_lims.extend([max(comb[0]), min(comb[0])])
    x_planes_max = max(xs_lims)
    x_planes_min = min(xs_lims)

    x_areas_merge = np.linspace(x_planes_min, x_planes_max, point_number)
    total_area = np.zeros(x_areas_merge.shape)

    for array in areas_merge:
        f_interp = interp1d(array[0], array[1], kind='linear',
            bounds_error=False, fill_value=0)
        total_area += f_interp(x_areas_merge)

    if plot_graphs:
        plt.figure()
        for array in areas_merge:
            plt.plot(array[0], array[1], '.', color='gray')
        plt.plot(x_areas_merge, total_area, '.-')

    return x_areas_merge, total_area


def fourier_series_sine_terms(xs, ys, n_max=31, x_min=None, x_max=None,
                              full=False, plot_comparison=False):
    '''
    This function calculates the terms of the Fourier Series that aproximate a
        given set
    of points.
    Parameters
    ----------
    xs : Array. x poisitions of the points to be approximated. Assumed to be
        in crescent
            order
    ys : Array. Values of the function at the [xs] position
    n_max : Int. Optional. Determines the maximum degree used for the Fourier
        Series Used
    x_min : Float. Optional. Determines the lower limit of the studied
        interval. Default is None
    x_max : Float. Optional. Determines the upper limit of the studied
        interval. Default
            is None
    full : Boolean. Optional. Controls wether if the full series is calculated
        or only the
```

```
            sine terms. Default False
    plot_comparison : Boolean. Optional. Determines if a comparisson of the
        original points
            and the Fourier Series is ploted.

    Returns
    -------
    fourier_terms : Dict. Contains either only the b coefficients of the
        Fourier Series or
            the full a0, a coeffients and b coeff. Keys: 'a_0', 'a_array',
                'b_array'

    '''
    x_min = min(xs) if x_min is None else x_min
    x_max = max(xs) if x_max is None else x_max
    a_array = []
    b_array = []
    L = (x_max - x_min) / 2
    x_med = (x_min + x_max) / 2
    for n in range(1, n_max + 1):
        product = ys * np.sin(n * np.pi * (xs - x_med) / L)
        b_n = 1 / L * trapezoid_integration(product, xs)
        b_array.append(b_n)

    if full:
        for n in range(1, n_max + 1):
            product_cos = ys * np.cos(n * np.pi * (xs - x_med) / L)
            a_n = 1 / L * trapezoid_integration(product_cos, xs)
            a_array.append(a_n)

    a_0 = trapezoid_integration(ys, (xs - x_med)) / (2 * L)
    if plot_comparison:
        plt.figure()
        plt.plot((xs - x_med) / L, ys, '.')
        x_test = np.linspace(-np.pi, np.pi, 100)
        y = np.ones(x_test.shape) * a_0
        for (n, b_n) in enumerate(b_array):
            a_n = a_array[n] if full == True else 0
            y += b_n * np.sin((n + 1) * x_test) + a_n * np.cos((n + 1) * x_test)
        plt.plot(x_test / np.pi, y, color='orange')

    fourier_terms = {'a_0': a_0, 'a_array': a_array, 'b_array': b_array}
    return fourier_terms


def calculate_supersonic_drag(random_rocket: Rocket, mach_number, plot_rocket=False, plot_fins=False,
                              plot_areas_merge=False, plot_fourier_comparison=False, export_areas=False):
    '''
    This funtion calculates the Wave Drag coefficient based on the method described above.
    Parameters
    ----------
    random_rocket : Object of the rocket class. See documentation.
    mach_number : Mach number to be determined
    plot_rocket : Boolean. Optional. Determines wether a plot of the rocket intesected by
        the mach planes is drawn. Default is False.
    plot_fins : Boolean. Optional. Determines wether plots of the fins intesected by
        the mach planes are drawn. Default is False.
    plot_areas_merge : TBoolean. Optional. Determines wether a plot of the sum of the areas
        are drawn. Default is False.
    plot_fourier_comparison : Boolean. Optional. Determines wether a plot of the Fourier
        Series is drawn. Default is False.
    export_areas : Boolean. Optional. Determines wether intersected area values are exported
        as output. Default is False.

    Returns
    -------
    (cd, opt:areas_export): The first is the value of c_d determined. The second is an array
        which contains arrays with the x coordinates and the areas intersected for each
        plane coordinate

    '''
    print(f'Supersonic Drag Estimation for M={format(mach_number, "0.2f")}')
    # Calculating Drag for a given M >= 1
    # Rocket Body Geometric Data
    f_interp = random_rocket.body_interp
    xs_max = f_interp.x[-1]
    xs_array = f_interp.x
    rs_array = f_interp.y

    # Fin geometric data
    fin_xs = random_rocket.fin_points[0]
    fin_top = random_rocket.fin_points[3]
    fin_base = random_rocket.fin_points[2]
    fin_th = random_rocket.fin_th

    # Using interpolating functions for the area calculation function
    f_int_top = interp1d(fin_xs, fin_top, kind='linear', bounds_error=False,
        fill_value=0)
    f_int_base = interp1d(fin_xs, fin_base, kind='linear', bounds_error=False,
        fill_value=0)

    mach_cotg = np.sqrt(mach_number ** 2 - 1) # Mach Cone Inclination
        Component [-]
    mach_sin = 1 / mach_number
    mach_cos = mach_cotg * mach_sin

    # Projecting Hull points into the r=0 axis by lines parallel to mach cone
        (pos and negative)
    xs_start_array = xs_array - rs_array * mach_cotg
    xs_end_array = xs_array + rs_array * mach_cotg

    n_start_l = np.argmin(xs_start_array)
    n_end_l = np.argmax(xs_end_array)

    interp_points = 15
    points_vector = np.linspace(0, 1, interp_points)
    interval =4

    if plot_rocket:
        # plt.figure()
        random_rocket.draw_rocket()
    # Calculating the area intersection of the superior half of the rocket
    area_sup = []
    x_planes = []
    for (i, x_plane) in enumerate(xs_start_array[n_start_l:-1:interval]):
```

```python
        r_hull = rs_array[n_start_l + i * interval]
        x_hull = xs_array[n_start_l + i * interval]
        d_plane = np.sqrt(r_hull ** 2 + (x_hull - x_plane) ** 2)
        if x_plane < 0:
            if i == 0:
                area_sect = 0
                x_points = x_hull
                y_points = r_hull
            else:
                x_inter, y_inter = intersect_hull(x_hull, d_plane, f_interp, mach_number, False)
                d_plane = np.sqrt((r_hull - y_inter) ** 2 + (x_hull - x_inter) ** 2)
                x_points = x_inter + points_vector * d_plane * mach_cos
                y_points = y_inter + points_vector * d_plane * mach_sin
                inter_rad = np.sqrt(abs(f_interp(x_points) ** 2 - y_points ** 2))
                area_sect = trapezoid_integration(2 * inter_rad, y_points)
        else:
            x_points = x_plane + points_vector * d_plane * mach_cos
            y_points = points_vector * d_plane * mach_sin
            inter_rad = np.sqrt(abs(f_interp(x_points) ** 2 - y_points ** 2))
            area_sect = trapezoid_integration(2 * inter_rad, y_points)

        area_sup.append(area_sect)
        x_planes.append(x_plane)
        if plot_rocket:
            plt.plot(x_points, y_points, color='lightgray')

    # Getting the area of the last bit that ends on the rocket final length
    if mach_number > 1:
        for r_hull in np.linspace(f_interp(xs_max), 0, round(6 / mach_sin))[:-1]:
            x_plane = xs_max - r_hull * mach_cotg
            x_hull = xs_max
            d_plane = np.sqrt(r_hull ** 2 + (x_hull - x_plane) ** 2)
            x_points = x_plane + points_vector * d_plane * mach_cos
            y_points = points_vector * d_plane * mach_sin
            inter_rad = np.sqrt(abs(f_interp(x_points) ** 2 - y_points ** 2))
            area_sect = trapezoid_integration(2 * inter_rad, y_points)
            area_sup.append(area_sect)
            x_planes.append(x_plane)
            if plot_rocket:
                plt.plot(x_points, y_points, color='lightgray')
    else:
        area_sup.extend([0, 0, 0])
        x_planes.extend([x_planes[-1] + 0.01, x_planes[-1] + 0.02, x_planes[-1] + 0.03])

    # Calculating the area intersection of the inferior half of the rocket
    area_inf = []
    x_planes_inf = []
    areas_export = {}
    for (i, x_plane) in enumerate(xs_end_array[:n_end_l:interval]):
        r_hull = rs_array[i * interval]
        x_hull = xs_array[i * interval]
        d_plane = np.sqrt(r_hull ** 2 + (x_hull - x_plane) ** 2)

        if xs_end_array[-1] < x_plane < xs_end_array[n_end_l]:
            # for points which intersect the hull and have an x_plane greater than the exhaust
            x_inter, y_inter = intersect_hull(x_hull, d_plane, f_interp, mach_number, True)
            d_plane = np.sqrt((r_hull - y_inter) ** 2 + (x_hull - x_inter) ** 2)
            x_points = x_hull + points_vector * d_plane * mach_cos
            y_points = r_hull - points_vector * d_plane * mach_sin
            inter_rad = np.sqrt(abs(f_interp(x_points) ** 2 - y_points ** 2))
            area_sect = trapezoid_integration(2 * inter_rad[::-1], y_points[::-1])
        elif x_plane > xs_max:
            # for points where the line would cross the last section, ellipse is only partial
            r_max = (x_plane - xs_max) / mach_cotg
            d_plane = np.sqrt((r_hull - r_max) ** 2 + (x_hull - xs_max) ** 2)
            x_points = xs_max - points_vector * d_plane * mach_cos
            y_points = r_max + points_vector * d_plane * mach_sin
            inter_rad = np.sqrt(abs(f_interp(x_points) ** 2 - y_points ** 2))
            area_sect = trapezoid_integration(2 * inter_rad, y_points)
        else:
            x_points = x_plane - points_vector * d_plane * mach_cos
            y_points = points_vector * d_plane * mach_sin
            inter_rad = np.sqrt(abs(f_interp(x_points) ** 2 - y_points ** 2))
            area_sect = trapezoid_integration(2 * inter_rad, y_points)

        area_inf.append(area_sect)
        x_planes_inf.append(x_plane)
        if plot_rocket:
            plt.plot(x_points, -y_points, color='lightblue')

    # Defining the rotation of the plane for non-revolution components
    # Only the angle between fins has to be studied as it's cyclical
    thetas = np.linspace(0, 2 * np.pi / random_rocket.fin_numb, 7)
    cds = np.ndarray([0])
    # print(thetas)
    fourier_points = 150
    for theta in thetas:

        # Calculating the inclination for each fin
        ang_phis = theta + np.linspace(0, 2 * np.pi, random_rocket.fin_numb + 1)[:-1]
        areas_fin = []
        for angle in ang_phis:
            cos_eff = mach_cos * np.cos(angle)
            x_planes_fin, lengths = fin_segments(fin_xs, f_int_top, f_int_base, cos_eff, plot_fins)
            fin_eff_thickness = fin_th
            areas_fin.append([x_planes_fin, [length * fin_eff_thickness for length in lengths]])

        # Summing different areas calculated in different ranges
        areas_merge = [[x_planes, area_sup], [x_planes_inf, area_inf]]
        areas_merge.extend(areas_fin)
        x_merge, total_area = merge_areas(areas_merge, plot_graphs=plot_areas_merge, point_number=fourier_points,)

        if export_areas:
            #print(theta)
            areas_export[theta] = ([x_merge, total_area])

        # Total Area first derivative
        diff_total_area = []
        delta_x = x_merge[1] - x_merge[0]
        for (n, area) in enumerate(total_area):
            if n == 0:
                derivative = (-3 * total_area[n] + 4 * total_area[n+1] - 1 * total_area[n+2])/(2 * delta_x)
            elif area == total_area[-1] or n == fourier_points - 1:
                derivative = (3 * total_area[n] - 4 * total_area[n - 1] + 1 * total_area[n-2])/(2 * delta_x)
            else:
                derivative = (- total_area[n-1] + total_area[n+1]) / delta_x
            diff_total_area.append(derivative)

        diff_total_area = np.array(diff_total_area)

        # Applying a fourier series to approximate the function
        fourier_series = fourier_series_sine_terms(x_merge, diff_total_area, n_max=20, full=True,
                                                   plot_comparison=plot_fourier_comparison)

        # Calculating Cd as a series based on the Fourier Sine Coefficients
        c_d_sup = 0
        for (n, a_n) in enumerate(fourier_series['b_array']):
            c_d_sup += (n + 1) * a_n ** 2

        c_d_sup = np.pi / 4 * c_d_sup / random_rocket.s_ref
        cds = np.append(cds, c_d_sup)

    cd_sup = trapezoid_integration(cds, thetas) / (thetas[-1] - thetas[0])
    if export_areas:
        return cd_sup, areas_export
    else:
        return cd_sup


def supersonic_drag_curve(random_rocket: Rocket, mach_max, points_numb=8, mach_start=1.2, plot_curve=False):
    '''
    This programs generates the interpolation curve of wave drag coefficient by Mach number
    The transonic region is estimated based on the theory presented in Aircraft Project: A
    Conceptual Approach, by Raymer.
    Parameters
    ----------
    random_rocket : Object of the rocket class. See documentation.
    mach_max : maximum Mach number for the calculations
    points_numb : Int. Optional. Number of mach values to calculate drag
```

```
        coefficient. The
            default is 8.
    mach_start : Optional. Minimum mach number for the calculation of the drag
        coefficient.
            Default is 1.2
    plot_curve : Boolean. Optional. Determines wether a plot of the MachxC_d
        curve is drawn.
            Default is False.

    Returns
    -------
    cd_mach_curve : Function. Interpolates mach values for each mach number,
        below the limit it
            is set as 0, above it is set as de value of the point in mach_max

    '''
    mach_number_list = 1 / np.cos(np.linspace(np.arccos(1 / mach_start),
        np.arccos(1 / mach_max), points_numb))
    cd_curve = []

    for mach_number in mach_number_list:
        c_d = calculate_supersonic_drag(random_rocket, mach_number,
            plot_rocket=False)
        cd_curve.append(c_d)

    # Raymer calculations
    m_cr = 0.8
    m_dd = m_cr + 0.08
    m_c = 1
    m_b = 1.05
    machs_raymer = [m_cr, m_dd, m_c, m_b]
    cds_raymer = [0, 0.002, cd_curve[0] / 2, cd_curve[0]]

    mach_number_list = np.append(machs_raymer, mach_number_list)
    cd_curve = np.append(cds_raymer, cd_curve)
    cd_mach_curve = interp1d(mach_number_list, cd_curve, kind='linear',
        bounds_error=False,
                             fill_value=(0, cd_curve[-1]))

    if plot_curve:
        plt.figure()
        xs = np.linspace(m_cr, mach_max, 1000)
        cds = cd_mach_curve(xs)
        plt.plot(xs, cds, color='orange')
        plt.plot(mach_number_list, np.array(cd_curve), '.', color='blue')
        plt.show()

    return cd_mach_curve
```