Master of Science in Aerospace Engineering

Research Project S2 Report

Trajectory Module for Launcher Multidisciplinary Design, Analysis and Optimization

Jorge L. valderrama

supervised by:

Dr. Annafederica urbano
DCAS, ISAE-SUPAERO

Dr. Mathieu balesdent
DTIS, ONERA

Dr. Loïc brevault
DTIS, ONERA

Starting date of project: January 27, 2020
Submission date: September 7, 2020

Declaration of authenticity

This assignment is entirely my own work. Quotations from literature are properly indicated with appropriated references in the text. All literature used in this piece of work is indicated in the bibliography placed at the end. I confirm that no sources have been used other than those stated.

I understand that plagiarism (copy without mentioning the reference) is a serious examinations offence that may result in disciplinary action being taken.

Signature: _____ Date: _____

Jorge Luis Valderrama

Contents

List of Figures

List of Tables

Abstract

Multidisciplinary optimization is used in launch vehicle design to maximize performance and minimize cost. With these aims in mind, the LAST tool is being developed at SUPAERO's SacLab and the goal of this research project is to provide it with a 3D trajectories optimization module. In this report, the strategy to gradually develop it is presented and the results of the implementation of a direct single shooting method and partial implementation of a pseudospectral method for the optimization of 2D launcher trajectories are discussed.

Keywords: Launch Vehicle Trajectories, Multidisciplinary Design, Analysis and Optimization, Optimal Control, Pseudospectral Methods

1 Introduction

In the design of a launch vehicle it is desired to obtain the maximum performance at the lowest cost possible for a given objective mission. In the pursuit of these aims, the interactions between multiple disciplines must be taken into account, for example: trajectory optimization, structures, propulsion and aerodynamics.

In a traditional design approach, individual optimization of each discipline is performed and its outputs are transferred to the design offices in charge of the other disciplines in an iterative process. This approach does not take into account all of the complex interactions between the different disciplines thus yielding non-optimal results.

In the quest for designing more performant and lower cost launch vehicles, Multidisciplinary Design Analysis and Optimization (MDAO) approaches have been been used to better assess the complex interactions among disciplines. NASA's OpenMDAO is one of the platforms enabling this integration of disciplines through efficient gradient computations for the optimization process, allowing parallel computing in multiple processors.

Trajectory optimization is a central approach to launch vehicle MDAO. Due to the high complexity of the mathematical models describing the different disciplines, it is important to integrate an efficient trajectory optimization method into the MDAO model to keep low computing times. Packages as DYMOS integrate highly accurate and efficient direct pseudospectral methods with the OpenMDAO platform to achieve this goal.

The Launcher Analysis Sizing Tool (LAST) is being developed at ISAE-SUPAERO's SACLAB using OpenMDAO. Structure and propulsion design are being integrated as disciplines into it as well as a 2D trajectory module. The subject of this research project is going to be the integration of a 3D trajectory module for expendable and reusable launchers using pseudospectral methods with the Dymos package into the LAST tool.

2 Aims and objectives

Aim

To develop and integrate an efficient and accurate 3D trajectory optimization module for a launcher into the LAST tool.

Objectives

- To implement trajectories of several level of complexity , including 2D and 3D
- To include different command laws for expendable and reusable launchers under different stage recovery scenarios
- To provide the analytic derivatives required for the MDAO for each case

3 Literature Review

As a first topic, this literature review will asses the governing equations of motion (EoM) for launcher trajectories for 2D and 3D models in spherical coordinates and some strategies to cope with the singularities of the 3D case, followed by the guidance laws used for launcher ascent and some recovery scenarios. Then, a review of some optimal control strategies used for trajectory optimization will be presented, justifying the advantage of using direct pseudospectral methods. Some case studies demonstrating the advantages of using OpenMDAO in trajectory optimization problems will be discussed and finally the use of optimal control control strategies within the MDAO frame will be treated.

3.1 Equations of Motion (EoM)

The model to simulate launcher trajectories is commonly simplified by neglecting rotation dynamics in order to keep low computation times. This simplification is based on the assumption that the attitude control system of the launcher ensures the right attitude at every time.

Additional simplifications for the model can be done by assuming motion within the equatorial plane resulting in a 2D model represented by a set of 4 differential equations describing the changes in velocity, radial distance to the center of the planet, longitude, and flight path angle. Given that launchers are a variable mass system, an additional equation must be solved to account for the changes in mass. This 2D simplification will be used in the early stage of the project as it allows a faster implementation and verification.

To describe the motion of the Launcher in 3D space, a set of 6 differential equations is commonly used in the literature, adding 2 extra equations that are necessary to describe the changes in latitude and heading angles. In the work of Briese *et al.* [1], the following coordinate systems are defined to obtain the set of equations in spherical coordinates:

- Earth Centered Inertial (ECI)
- Earth Centered Earth Fixed (ECEF)
- North-East-Down (N)
- Kinematic (K)
- Body fixed (B)

The different transformations between the coordinates systems are presented and a transformation matrix to express the launcher's position according to the *World Geodetic System 1984* (WGS'84) is given. Also in that work, a transformation between the ECEF and N coordinate systems is used to avoid the singularity presented when the velocity is near $0m/s$ and the flight path angle near $90°$. The EoM in spherical coordinates according to the *Aerospace Standard ISO 1511* are presented in the work of Castellini [2] in the context of MDAO for expendable launch vehicles. Cartesian coordinates to express the EoM were used by Volo [3] to perform trajectory optimization using pseudospectral methods, as an important remark, the EoM in these coordinates do not present the singularity of their spherical counterpart.

All of the aforementioned models neglect the rotation dynamics to reduce the number of equations and variables and thus the computing time, in their 3D version this simplification is said to have 3 degrees of freedom (3DOF). Berend an Talbot [4] described CNES's packages ORAGE and OPTAX, the former being used to perform atmospheric reentry optimization using 3DOF models and the later to optimize Ariane's trajectories and having the possibility of interconnection with 6DOF Ariane simulators. 6DOF models account for rotational dynamics thus requiring information about the inertia tensor of the launcher generating a system of 12 differential equations. Wang and Wu [5] compared the performance of 3DOF and 6DOF models for reusable launcher optimization using pseudospectral methods, finding the later to require 46% more time to be solved.

3.2 Forces

The EoM are function of gravity, thrust and aerodynamic force terms. In this subsection some approaches found in the literature referring to these topics will be described.

A simplified model for gravity is Newton's law of universal gravitation, this will be used in the early stage of the project. A more complete model accounting for Earth's elliptical shape is used in [2], by default it includes only the main gravitational field harmonic, but there exists the possibility to include up to the J_4 term too. The aforementioned models are broadly explained by Tewari [6]. Finally, the *Earth Gravitational Model 1996* (EGM96) is used in [1].

Aerodynamic data is usually provided in the form of tables where drag an lift coefficients are given as functions of the angle of attack and Mach number. Any interpolation used to integrate this tabular data into the trajectory optimization model must be C_2 continuous if Non-Linear-Programming (NLP) solvers are to be used as expressed by Betts in [7], he suggests to use B-splines for this purpose. The use of NLP solvers will be explained in the section covering Optimal Control. Variations on atmospheric properties were modeled with the *U.S. Standard Atmosphere 1976* in [2], [8] and [9], and in [3] the *COSPAR International Reference Atmosphere 2012* was used.

In the early stage of the project a very simplified thrust model will be used based on the specific impulse of the engine, a standard gravity parameter and the mass flow rate. A MDAO tool for optimizing the thrust system has been designed in another master thesis within the SACLAB and its integration within LAST and the trajectory module to be developed in this work could be considered.

3.3 Guidance laws

The guidance laws used to steer the launcher through a certain trajectory at a desired speed will be first described for the ascent phase and later for different recovery scenarios. To steer, a launcher can perform maneuvers with its gimbaling engines or its reaction control system (RCS) composed by thrusters with thrust components perpendicular to its longitudinal axis. The effect of these steering strategies is reflected on the pitch and yaw angles of the vehicle. In their works, Castellini [2] and Pagano [9] used similar approaches for the control strategy during the ascent phase of the launcher. This strategy considers the following sequence for the control of the pitch angle:

1. Vertical lift-off: The vehicle ascends vertically at least up to the minimum height constrained by launchpad safety criteria.
2. Linear pitch-over: The vehicle modifies its attitude and generates angle of attack
3. Exponential decay of pitch: The vehicle returns to its zero angle of attack position with a pitch angle different from the vertical position
4. Gravity turn: The gravity force helps steer the vehicle towards its orbital attitude
5. Bilinear tangent law: Insertion of vehicle to its orbital attitude and motion
6. Coast phases: Engines are off and trajectory is ballistic. It happens after stage and fairings separation.
7. Stage burns: After stage separation the mass of the launcher changes

The yaw angle guidance laws are also described by the aforementioned authors as a function of vehicle latitude and orbit inclination.

There exist many different strategies for the total or partial recovery of launchers. The recovery off the Liquid Fly-back Booster stage of a Vertical Take-off Horizontal Landing (VTHL) vehicle was described in [1] and the control strategy for the Vertical Take-off Vertical Landing (VTVL) Callisto demonstrator was described by Sagliano *et al.* [10] covering two recovery scenarios for the reusable first stage, namely Downrange Landing (DR) and Return-to-Launch-site (RTLS). The use of thrust magnitude, thrust pointing, RCS and aerodynamic control surfaces was also described by these authors during the take-off, ascent, boost-back, aerodynamic descent and landing phases.

3.4 Optimal Control

Optimal control problems are sometimes called also trajectory optimization problems. Betts introduced a compendium of methods for trajectory optimization in [11] and [7]. In the other hand, Rao did the same for optimal control in [12] and defined trajectory optimization problems as a subset of Optimal Control where the input parameters are static, thus dealing with the optimization of functions, whereas in the more general case Optimal Control is related to functional optimization problems.

Three main classifications can be done for Optimal Control methods: Heuristic Optimization Methods, Indirect methods and Direct methods. Heuristic Optimization methods are global, that implies that they are good at finding solutions in search spaces with many local minimums. Usually they are easy to implement but they are based on stochastic approaches rather than in gradient information thus having as output non-optimal solutions and being considered of inferior performance for the application discussed in this work.

Indirect methods are founded on the calculus of variations and the formulation of first-order optimality conditions for the original problem, forming a Hamiltonian boundary value problem (HBVP) whose solutions are determined numerically. This strong formal background can ensure that solutions, when obtained, are indeed optimal. In [11] the application of indirect methods for low thrust orbit analysis was demonstrated. Berend and Talbot [4] described its use within CNES and Ariane programs and addressed some difficulties at implementing aerodynamic controls. In general these methods require experience to provide an initial guess due to its sensibility, making really difficult to explore changes on the launcher design variables.

On the other hand, Direct Methods are less sensitive to the quality of the initial guess, and for some specific methods the mapping between the direct and indirect formulation has been done. Herman and Conway [13] used these relations to determine the accuracy of a Direct high-order Gauss Lobatto Collocation scheme. Garg *et al.* [14] demonstrated this mapping for a Radau Pseudospectral Method. The importance of this notion is that some direct methods benefit from a low sensibility to the initial guess, and their outputs can be verified with the mapping with the indirect formulation, thus guaranteeing high accuracy. This also represents and advantage for the multidisciplinary optimization of launch vehicles as trajectories can vary drastically from iteration to iteration. For these reasons, the remaining part of this work will be centered on direct methods.

Direct methods use two classes of numerical methods as a basis, the first one dealing with Numerical Solution of differential Equations and Integration and the second one with Non-Linear Optimization. The first class is used in a transcription process to formulate a Non-Linear Optimization problem which then will look for the values of the inputs that minimize the objective function satisfying some given constraints. These classes will be described in the following sections together with the transcription methods.

3.4.1 Numerical Solution of differential Equations and Integration

Time marching methods are used to approximate numerically the solution of a differential equation at a given time step using information of current or previous information about the solution. Adams methods and Runge-Kutta methods including Euler and Hermite-Simpson formulations are covered in [7] and [12].

In collocation methods coefficients of an interpolating polynomial are calculated simultaneously to "collocate" it at the collocation points when approximating the solution of the equation. The solution itself can be discretized into intervals. Different formulations as Gauss, Radau and Lobatto methods exist and their differences are based on the inclusion or not of endpoints on the current interval. A subset of these methods is orthogonal collocation which uses orthogonal polynomials to approximate the functions, typically Chebyshev or Legendre polynomials as explained in [12], but also Jacobi polynomials were used for a High-Order Gauss Lobatto method implemented in [13].

As the objective function in the optimal control formulation includes often an integral term, numeric integration is also required. The approximation of the objective function and of the differential equations must be consistent.

3.4.2 Non-Linear Optimization

To find the values of the inputs that minimize the objective function and satisfy the given constraints a Non-Linear Programming (NLP) problem is formulated with a transcription method. NLP solvers are mostly based on Newton's method and formulate the problem in the Karush-Khun-Tucker (KKT) form. This matrix formulation contains the Jacobian and Hessian matrices of the problem that can be calculated analytically to obtain the best efficiency but also can be numerically approximated. These formulations are widely covered in [7]. One of the biggest challenges of the transcription process and the NLP solvers is to take advantage of the sparsity of the matrix formulation, specially when solutions for the equations are discretized in multiple segments, thus increasing the number of NLP variables. Some NLP solvers that are used frequently in the literature are SNOPT in [8] [15] [12] , IPOPT in [12] and NPSOL in [11].

3.4.3 Direct Transcription Methods for Solving Optimal Control Problems

The simplest direct method is called single shooting. In the case of a two point boundary value, it uses an initial guess for the initial boundary conditions and a numerical method to propagate the solution accordingly. The obtained value at the final time is then compared whit the boundary conditions to calculate an error. Then the NLP solver is used to find the set of initial conditions that take the error to zero. A more accurate method is multiple shooting, it divides the time into intervals and requires an initial guess for all of them. To guarantee continuity of the solution this method adds some constraints that increase the size of the NLP problem. Both methods are covered in [11] and [12]. Due to the low number of control parameters, multiple shooting has been used effectively for launcher ascent trajectories.

In Global Orthogonal Collocation or Pseudospectral methods the state or solution is approximated using global polynomials collocated at the collocation points. Even so, the state is still discretized into segments. In these methods the number of segments or meshes is fixed and the degree of the polynomial is varied. One remarkable advantage of these methods is the spectral convergence (exponential rate) as a function of the number of collocation points. In the survey of methods done by Rao [12] the following pseudospectral methods of interest for this project because of the possibility to integrate them with MDAO methods are covered:

- Gauss-Lobatto Pseudospectral Methods
- Pseudospectral method Using Legendre-Gauss-Radau points

As mentioned before, Herman and Conway [13] worked with High Order Gauss Lobatto methods. They showed the better convergence characteristics for these methods when compared to Low order methods of the same family thanks to the reduction of rounding errors associated to the lower number of segments required to approximate a function with high order polynomials. Garg *et al.* [14] demonstrated the spectral convergence of a Radau Pseudospectral method and developed the mapping with its indirect formulation. In a case study integrating a propulsion and a trajectory optimization module into an MDAO formulation, Hendricks *et al.* [8] showed that Legendre-Gauss-Radau have better performance over Legendre-Gauss-Lobatto when parallelization is used.

3.5 MDAO and its integration with Optimal Control

Multidisciplinary Design, Analysis and Optimization (MDAO) is an engineering strategy to optimize complex systems involving multiple disciplines. This process is done with gradient based methods that can involve analytic and numerically approximated derivatives. OpenMDAO is a MDAO open source framework developed at NASA Glenn Research Center written in Python. In this tool, the problem can be divided into components that must provide the derivatives of its outputs with respect to its inputs, and then the software will compute the total derivative of the system using the chain rule to perform the optimization of the desired variables.

In [8] a coupled modeling approach of trajectories and propulsion systems of an aircraft was proposed using OpenMDAO. The strategy was validated using the trajectory optimization for minimum time-to-climb of an F-4 supersonic jet. Falck and Gray [15] proposed a Legendre-Gauss-Lobatto based collocation

scheme with analytic derivatives for the trajectory optimization of an aircraft using OpenMDAO. The same authors presented in [16] the Dymos tool, a package designed to solve Optimal control problems in conjunction with MDAO. Initially, it implements Legendre-Gauss-Radau (LGR) and Legendre-Gauss-Lobatto pseudospectral methods, but the possibility to integrate other transcription methods exists. The package has also built-up interfaces with NLP solvers like SNOPT.

4 Justification of the potential degree of novelty

Multiple optimal control packages using both direct and indirect methods exist and their implementation in high performance programming languages renders them highly efficient. The direct pseudospectral optimal control algorithms implemented in Dymos equate that performance, at the same time that they allow the interface with the MDAO environment of OpenMDAO. The trajectories vary significantly during the different iterations of MDAO of launch vehicles, requiring optimal control methods with low sensitivity to the initial guess. The MDAO of launch vehicles using pseudospectral methods is seldom addressed in the literature and the potential degree of novelty of this project is based on the study of its suitability.

5 Research Project Plan and Description of Tasks

Task	Description	Due date
1	Implement Single and Multiple Shooting Direct Methods for trajectory optimization in Python but outside OpenMDAO for 2D and 3D trajectories	15 / 05 / 2020
2	Implement Direct Pseudospectral Methods for trajectory optimization in Dymos and OpenMDAO for 3D trajectories, this may be Legendre-Gauss-Radau or Legendre-Gauss-Lobatto. Implement different guidance laws for these methods	15 / 11 / 2020
3	Integrate the trajectory optimization method into the LAST tool in OpenMDAO with the other disciplines (Structures, Propulsion, etc.)	15 / 02 / 2021
4	Document the process and results	15 / 03 / 2021

Table 1: Research Project Plan and Description of Tasks

6 Methods

The main goal of this research project is to develop and integrate a 3D trajectory optimization module into the LAST tool with 3DOF. With the aim of making it efficient and accurate a direct pseudospectral method is to be used, moreover, these methods are not that sensitive to the initial guess as some indirect methods are, as already described in sections 3.4.3 and 3.5. For the initial steps, it was decided to first implement the optimization of 2D trajectories as it uses 5 state equations instead of the 7 necessary for the 3D trajectories. This simplifies the learning curve at the same time that allows to work towards the same goal as many physical models are common to both problems as, for instance, the control strategy for the pitch angle. Also, the recovery scenarios and multiple staging of the launcher will be implemented once the 3D model is working correctly.

As mentioned in the literature review, the OpenMDAO framework allows the optimization of aerospace systems in an efficient way and its modular approach allows to gradually refine the complexity of models,

furthermore, multiple open source packages are available in this framework that allow a faster implementation of models. As the LAST tool is being developed in OpenMDAO and its Dymos package implements pseudospectral methods for trajectory optimization, it became an evident choice to use Dymos for the scope of this project. Both, the Legendre-Gauss-Radau and the Legendre-Gauss-Lobatto are highly efficient transcriptions using collocation of orthogonal polynomials to approximate the states and controls of the optimal control problem and transcript it into a non-linear programming problem, nevertheless, these methods carry the disadvantage that during intermediate iterations of the optimization process the trajectory does not have a sense according to the physical laws that govern it. This makes really hard to iteratively improve a random initial guess to obtain a feasible solution.

In the other hand, the trajectories in shooting methods do have a physical sense and is easier to iterate different values of parameters for the initial guess. In addition, they can be used with gradient-free optimizers when the number of parameters to optimize is low, making them ideal for the first steps of the learning process. For these reasons the first implementation of the 2D trajectory optimization was decided to use a shooting method. This allows to provide an initial guess for the pseudospectral method and to use the shooting method as a reference.

In the following part of this section, the methods used to perform the optimization of 2D trajectories for a single stage launcher using both a shooting method and a pseudospectral one will be described.

6.1 Equations of motion and guidance laws

The 2D trajectory model can be represented by the set of 4 differential equations shown below in spherical coordinates and depicted in figure 1.

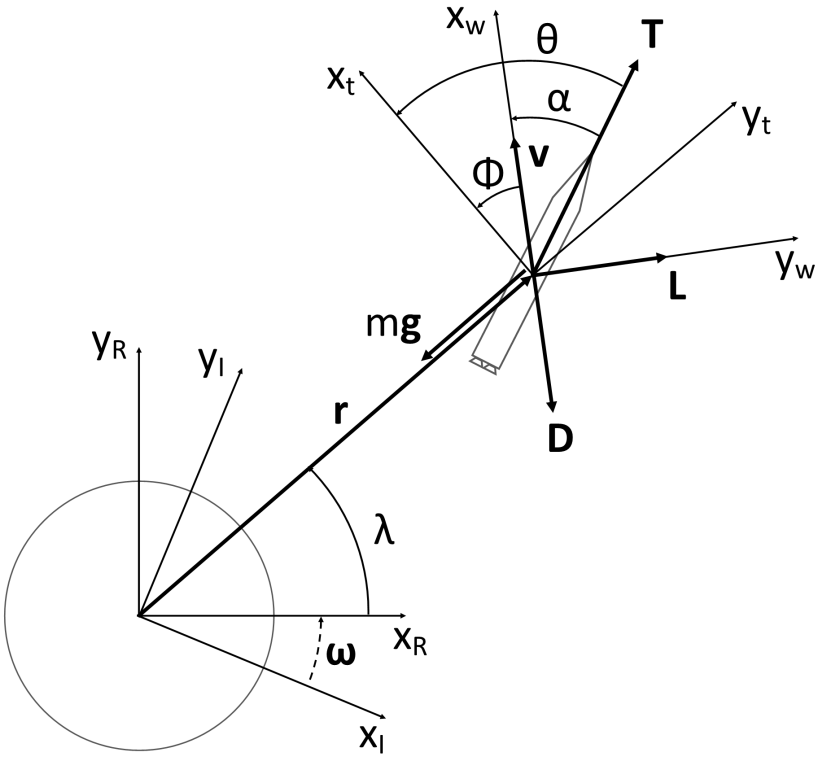

Figure 1: Reference frames and variables for 2D EoM

Where ω is Earth's angular velocity, λ is the longitude, $\mathbf{r}$ is the position vector from Earth's center to a fixed point in the launcher (Ex.: the geometric center), θ is the pitch angle, ϕ is the flight path angle, α is the angle of attack, m is the mass of the launcher at any given time, $\mathbf{g}$ is the acceleration of gravity, $\mathbf{D}$ is the drag force, $\mathbf{L}$ is the lift force, $\mathbf{T}$ is the thrust force and $\mathbf{v}$ the velocity vector.

In figure 1 the following reference frames are used:

- $\mathcal{F}_I \coloneqq \{x_I, y_I, z_I\}$, Inertial Earth Centered reference frame

- $\mathcal{F}_R \coloneqq \{x_R, y_R, z_R\}$, Rotating Earth Centered reference frame
- $\mathcal{F}_t \coloneqq \{x_t, y_t, z_t\}$, Local Vertical - Local Horizontal reference frame
- $\mathcal{F}_w \coloneqq \{x_w, y_w, z_w\}$, Wind reference frame

The demonstration of the equations of motion departing from Newton's second law and using transformations between 4 different coordinate systems is included in the appendix A.

$$\dot{r} = v \sin(\phi) \tag{6.1}$$

$$\dot{\lambda} = \frac{v \cos(\phi)}{r} \tag{6.2}$$

$$\dot{v} = \frac{-D + T\cos(\theta - \phi)}{m} + \left(-g + \omega^2 r\right)\sin(\phi) \tag{6.3}$$

$$\dot{\phi} = \frac{L}{mv} + \frac{T\sin(\theta - \phi)}{mv} + \frac{(\omega^2 r - g)\cos(\phi)}{v} + 2\omega + \frac{v\cos(\phi)}{r} \tag{6.4}$$

This set of 4 differential equations, in addition to one equation describing the change of mass of the launcher due to the consumption of propellants is solved numerically to obtain the history of the states in a given time span.

In order to control the launcher, the attitude control system changes the launcher's attitude by using thrust vectoring of the engines and transverse thrusters. Additionally, the propulsion system controls the throttle to vary the thrust magnitude. At the this stage of the research project, with the aim of simplifying the model, the throttle level is considered constant (i.e. constant mass flow rate) and the changes in attitude are represented by changes of the pitch angle θ. Consequently, the goal of the optimal control strategy will be to determine the history of the pitch angle that minimizes a given objective function.

In this work, the control strategy of the pitch angle will be done using static parameters as defined by Rao [12]. These control parameters will be the input of some guidance laws that are to be used through the trajectory and will serve to define different phases for the problem.

6.1.1 Lift-off phase

Launcher trajectories are a two-point boundary value problem, this implies that the trajectory starts at a fixed state that specifies the initial location of the launcher (corresponding to that of the launch pad), its initial mass, its initial flight path angle of 90° corresponding to a vertical direction, and its initial speed. The speed as described by the equations of motion corresponds to the relative speed with respect to a fix point on the rotating planet, representatively, as initially the launcher is not moving with respect to the launch pad, the initial speed should be zero. However, this value will cause a singularity in the equations of motion and is thus chosen to be slightly more than zero. The final boundary conditions of the two point boundary value problem correspond to those of the desired orbit and will be defined in the last phase.

As for the guidance law for the pitch angle, it just ensures that vehicle remains vertical following the expression

$$\theta = 90° \tag{6.5}$$

The end of the lift-off phase happens when the launcher clears the launchpad and can start changing its attitude safely. In some launchers as Ariane 5 and Falcon 9 this happens when the altitude is near $200m$. In this work such parameter is called h_{lo} and can be optimized in the vicinity of $200m$.

6.1.2 Linear pitch over phase

For defining this phase it's first necessary to define the concept of angle of attack. In figure 1 the angle of attack is represented by the Greek letter, α, and it's defined as the difference between the pitch angle, θ, and the flight path angle ϕ. During this phase the angle of attack increases linearly by a quantity $\Delta\theta_{lpo}$. This quantity, along with the duration of the phase, Δt_{lpo}, is part of the optimization variables or as they will be called later, design parameters. Consequently, the guidance law reads

$$\theta = \phi + \frac{\Delta\theta_{lpo}}{\Delta t_{lpo}} t_{lpo} \tag{6.6}$$

Where t_{lpo} is the local time of the phase, beginning at 0 and finishing at Δt_{lpo}. It is important to notice that at the early stage of the flight the air density is high, thus high angles of attack result in high aerodynamic loads. As launchers are build up in a lightweight fashion those loads are kept low by bounding $\Delta\theta_{lpo}$ to low values.

6.1.3 Exponential decay of pitch phase

After the linear pitch over phase the vehicle returns to a state of zero angle of attack to reduce aerodynamic loads. This process is done by following an exponential law that is defined as follows

$$\theta = \phi - \Delta\theta_{lpo} \cdot exp\left(\frac{-3 \cdot t_{edp}}{\Delta t_{edp}}\right) \tag{6.7}$$

The linear pitch over and exponential decay of pitch phases are shown in figure 2 for different values of $\Delta\theta_{lpo}$.

6.1.4 Gravity turn phase

During the previous phases the flight path angle changes allowing the vehicle to have a downrange velocity component. At this phase, the vehicle continues to increment its downrange velocity component thanks to the action of gravity that "bends" the trajectory. The law that governs this phase simply reads as

$$\theta = \phi \tag{6.8}$$

Making zero the angle of attack. The end of the phase is reached once the vehicle surpasses a threshold height at which the dynamic pressure is low and an aggressive change in attitude is possible for the following phase, this height is usually defined as $50km$ and it's a fixed variable or as will be called later, an input parameter.

6.1.5 Bilinear tangent law phase

The last phase is the bilinear tangent law, it starts by an aggressive change in pitch angle by an angle $\Delta\theta_{btl}$. Such maneuver is only possible because at the height where it starts the air density is really low, hence the aerodynamic forces experienced by the launcher are low despite the high angle of attack that can be created. In the bilinear tangent law equation, the initial angle θ_{btl_0} is calculated as the sum of the pitch angle at the end of gravity turn phase, θ_{gt}, and $\Delta\theta_{btl}$. This phase finishes at an angle of attack θ_{btl_f} that can take values different from zero even tough the speed v, the flight path angle ϕ and the radius r at the end of the phase correspond to a circular orbit state, this is because once the desired orbital state is reached, the vehicle can change its attitude easily with the attitude control system. The shape of the trajectory during this phase is controlled by the parameter ξ that can take values in the interval $[-1, 1]$. The parameter a controls the amplitude that θ can reach during the phase and it's usually fixed at 100. In figure 3 are shown some plots of θ for different values of ξ, and fixed values of θ_{btl_f} and θ_{btl_0} at -5 and 45 degrees correspondingly.

$$\theta = \arctan\left(\frac{a^{\xi}\tan\theta_{btl_0} + \left(\tan\theta_{btl_f} - a^{\xi}\tan\theta_{btl_0}\right)\frac{t_{btl}}{\Delta t_{btl}}}{a^{\xi} + \left(1 - a^{\xi}\right)\frac{t_{btl}}{\Delta t_{btl}}}\right) \tag{6.9}$$

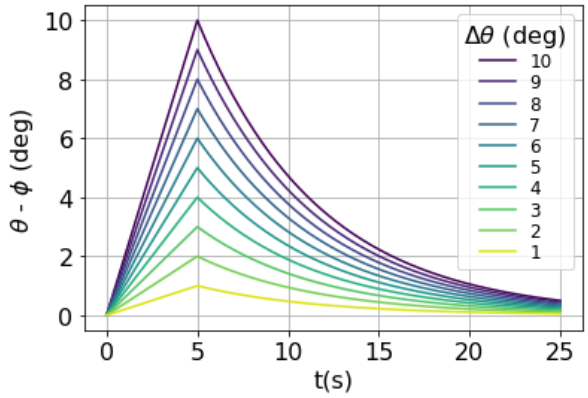

Figure 2: Pitch over (linear and exponential)

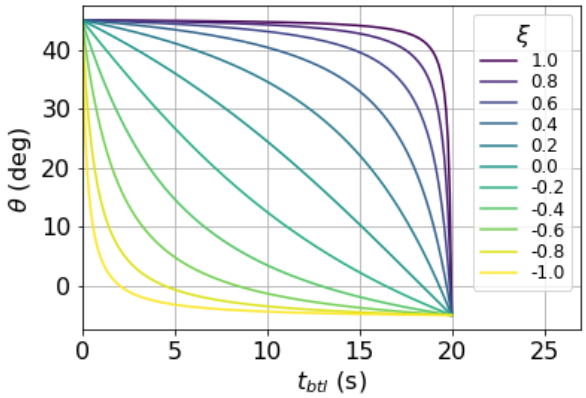

Figure 3: Bilinear tangent law phase

In the following sections the models used to calculate variables related to the forces exerted by the launcher will be discussed. Comprising gravity, atmosphere, aerodynamics, propulsion and mass.

6.2 Gravity model

Newton's law of universal gravitation is used to model the gravitational field of a spherical planet. A more precise model accounting for earth's spherical harmonics will be implemented in the future. The magnitude of the gravitational acceleration according to Newton's model reads

$$g = \frac{GM}{r^2} \tag{6.10}$$

Where GM is the product of the universal gravitational constant an the celestial body's mass and r is the radius of the point where the magnitude of the gravitational acceleration is to be calculated. For the case of planet Earth, the constant GM equals $3.986004 \times 10^{14} \quad m^3/s^2$.

In the case of the model of the planet Earth as an spherical body the radius at sea level is $6378.135 \times 10^3 \quad m$ and the gravity at this radius according to Newton's law of universal gravitation is $g_0 = 9.8 \quad m/s^2$.

Another important variable for the scope of this project is the orbital speed. The inertial orbital speed at a given radius is linked to the the constant GM by

$$v_{I_{orb}} = \sqrt{\frac{GM}{r}} \tag{6.11}$$

6.3 Atmospheric models

One of the purposes of the atmospheric model is to provide the value of air density as a function of altitude that is used to calculate the aerodynamic forces experienced by the vehicle. Another outputs of the atmospheric model are the local Mach number that is used to determine the vehicle's drag coefficient and the local atmospheric pressure that is used to model the variation of thrust due to the expansion of the combustion products at the exit of the nozzle. Two atmospheric models will be described, the first one is a simple exponential model for air's density and the second one is a model for air density, mach number and atmospheric pressure based on the 1976 and 1962 U.S. Standard Atmospheres as described in [6]. It is worth mentioning that both models ignore the variation of atmospheric parameters with time, latitude, longitude and winds.

6.3.1 Exponential model for air density

The advantage of such a simple model is that its derivative is continuous and easy to calculate. These are desired characteristics when using pseudospectral methods that use Non-linear programming solvers as will be explained later on. A drawback of this model is the loss of precision in a given range based on its parameters. The equation modeling the air density, ρ, can be written as

$$\rho = \rho_{ref} \cdot exp\left(\frac{-h}{h_{scale}}\right) \tag{6.12}$$

Where ρ_{ref} is the reference air density, not necessarily at sea level, h is the height and h_{scale} is the reference scale height. Both ρ_{ref} and h_{scale} can be modified to better adapt the model in a specific zone. In figures 4 and 5, two exponential atmospheric models can be observed. Namely, the exponential atmosphere model a, with $h_{scale} = 6.7 \quad km$ and $\rho_{ref} = 1.725 \quad kg/m^3$ and the exponential atmosphere model b, with $h_{scale} = 8.4 \quad km$ and $\rho_{ref} = 1.225 \quad kg/m^3$.

6.3.2 Atmospheric model based on the 1976 and 1962 U.S. Standard Atmospheres

This atmospheric model outputs atmospheric parameters by modeling the variation in temperature as a function of height. Height is divided into 21 segments called layers and the rate of change of temperature within a layer is constant but varies from layer to layer. The corresponding tabulated data is shown in table 2.

i	h_i (km)	T_i (K)	R (J/kg · K)	a (K/km)
1	0	288.15	287.0	-6.5
2	11.0191	216.65	287.0	0.0
3	20.0631	216.65	287.0	1.0
4	32.1619	228.65	287.0	2.8
5	47.3501	270.65	287.0	0.0
6	51.4125	270.65	287.0	-2.8
7	71.8020	214.65	287.02	-2.0
8	86	186.946	287.02	1.693
9	100	210.02	287.84	5.0
10	110	257.0	291.06	10.0
11	120	349.49	308.79	20.0
12	150	892.79	311.80	15.0
13	160	1022.2	313.69	10.0
14	170	1103.4	321.57	7.0
15	190	1205.4	336.68	5.0
16	230	1322.3	366.84	4.0
17	300	1432.1	416.88	3.3
18	400	1487.4	463.36	2.6
19	500	1506.1	493.63	1.7
20	600	1506.1	514.08	1.1
21	700	1507.6	514.08	0.0

Table 2: Atmospheric model based on the 1976 and 1962 U.S. Standard Atmospheres

Where h_i is the height above the sea level, T_i is the standard temperature, R is the gas constant and a is the slope that describes the variation of temperature with height across layers, better called thermal lapse rate.

The values of temperature between layers can be found according to the relationship

$$T = T_i + a\,(h - h_i) \tag{6.13}$$

It is worth noticing that in the first layer of this atmospheric model,between 0 and 11 km height, the temperature decreases with a thermal lapse rate of $-6.5 \quad K/km$ but in the second layer, between 11 km and 20 km the temperature does not vary with height, the former is called an isothermal layer.

For those layers whose thermal lapse rate, a, is different from zero, the pressure reads

$$p = p_i \left[1 + \frac{a\,(h - h_i)}{T_i}\right]^{\frac{g_0}{aR}\left[1+\beta\left(\frac{T_i}{a}-h_i\right)\right]} e^{\frac{\beta g_0}{aR}(h-h_i)} \tag{6.14}$$

And for the isothermal layers, the following expression is used

$$p = p_i e^{-\left[\frac{g_0(h-h_i)}{RT_i}\right]\left[1-\frac{\beta(h-h_i)}{2}\right]} \tag{6.15}$$

Where

$$\beta = \frac{2}{r_e} \tag{6.16}$$

Once the temperature and pressure are known for a given point, the air density can be calculated as

$$\rho = \frac{p}{RT} \tag{6.17}$$

The speed of sound at the given height is calculated according to the equation

$$a_\infty = \sqrt{\gamma RT} \tag{6.18}$$

Where γ is the adiabatic index assumed as 1.405.
Finally the Mach number, M, is defined as

$$M = \frac{v}{a_\infty} \tag{6.19}$$

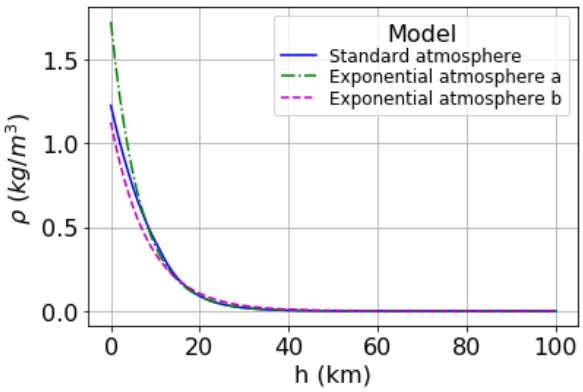

Figure 4: Air density in linear scale

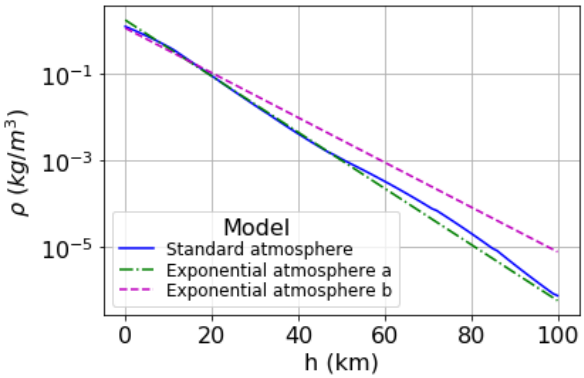

Figure 5: Air density in logarithmic scale

In figures 4 and 5 the result of air density obtained with the model based on the standard atmosphere is compared to those obtained with the exponential atmospheres. It can be noticed that the exponential atmosphere 'b' fits better the model based on the standard atmospheres at low altitudes where the air density is still high, for this reason it wil be preferred over the exponential atmosphere 'a' for the case of launcher's trajectories when the model based on the standard atmospheres cannot be used.

6.4 Aerodynamic models

Typically, aerodynamic forces experienced by a launcher are decomposed into the drag force, D, the lift force , L, and lateral force. As launchers usually have one plane of symmetry and the side slip angle is close to zero during the trajectory the lateral forces can be ignored. In a launcher as the space shuttle that haves big lifting surfaces, the lift force plays an important role. In the other hand, for cylindrical-shape launchers, lift forces are really low due to the low lift characteristics of cylindrical shapes and the fact that during a significant part of the early phases of the trajectory (that is when the dynamic pressure is significant) the angle of attack is zero. For the scope of this work, lift forces will also be ignored leaving only drag forces.

Drag forces are modeled according to the expression

$$D = qC_dS \tag{6.20}$$

Where C_d is the drag coefficient based on surface area , S the surface area and where the dynamic pressure, q, follows the equation

$$q = \frac{1}{2}\rho v^2 \tag{6.21}$$

The drag coefficient changes as a function of the angle of attack and the Mach number for a given launcher. To model this changes it is necessary to perform a wind tunnel testing campaign or multiple computational fluid dynamics (CFD) simulations. Usually the angle of attack is kept low when the dynamic pressure is high, therefore the variation of drag coefficient with angle of attack can be ignored.

The Cd curve of an Ariane 5 launcher is reported in [17]. In order to use it, sample points were taken and a cubic spline was used to transform the discrete data into a continuous equation as shown in figure 6.

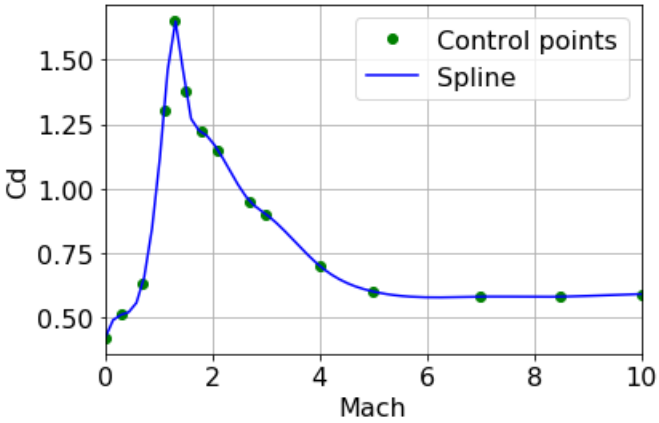

Figure 6: M Vs. C_d curve for an Ariane 5 launcher

As can be seen, the maximum drag coefficient is reached when the vehicle is trans-sonic. The data reported only extends up to $M = 10$, but as in some simulations the launcher exceeds this Mach number it was decided to extend the M Vs. C_d curve up to $M = 30$ with a constant C_d value equal to that of $M = 10$. It is important to notice that such big M values are reached at high altitudes where the atmospheric pressure is low, thus the drag force is low too. This proved to be useful in the shooting method that will be described later.

6.5 Propulsion model

The thrust, T, is modeled as function of the thrust at vacuum,T_{vac} the nozzle expansion area, A_e, and the local atmospheric pressure, p_a, as expressed by the expression

$$T = T_{vac} - A_e p_a \tag{6.22}$$

At altitudes near the sea level the atmospheric pressure is maximum and therefore the losses in thrust are maximum. Once the launcher is at vacuum the losses are null and the thrust T is maximum.

The thrust at vacuum itself is modeled as

$$T_{vac} = g_0 I_{sp} \mu \tag{6.23}$$

Where I_{sp} is the specific impulse and μ is the mass flow rate. As at this stage of the project the trust magnitude is not to be controlled the mass flow rate is assumed constant.

6.6 Mass model

In order to define the different masses comprising the rocket, the following terminology is used.

- Propellant mass (m_p): Mass of fuel and oxidizer.
- Payload mass (m_d): Mass of payload to be placed into orbit
- Structural mass (m_s): Mass of the rocket subtracting m_p and m_d.
- Empty mass (m_e): $m_s + m_d$
- Full mass (m_f): $m_s + m_d + m_p$

Additionally, to have some dimensionless values to compare different launchers, the relationships find below are used

- Structural coefficient (ϵ): $\frac{m_s}{m_p+m_s}$
- Propellant mass ratio (m_R): $\frac{m_f}{m_e}$

The mass at a given instant is represented as m. At the initial time, t_0, it takes the value $m(t_0) = m_f$ and at the final, t_f, time it is usually $m(t_f) = m_e$. The change of mass in time, which is one of the ODEs to be solved, is

$$\dot{m} = -\mu \tag{6.24}$$

6.7 Optimal control problem

So far all the physic models that are to be used in the optimization problem have been described. The following step is to describe the optimization techniques that are going to be used to find feasible and optimal trajectories.

The 4 equations of motion plus the equation of mass describe the evolution of the state of the launcher. Arranging them into the state vector $\bar{x}$ it reads

$$\bar{x} = \begin{bmatrix} r \\ \lambda \\ v \\ \phi \\ m \end{bmatrix} \tag{6.25}$$

The ordinary differential equations 6.1, 6.2, 6.3, 6.4 and 6.24 describe the evolution of the state vector according to

$$\dot{\bar{x}} = f_{ode}(\bar{x}, t, \bar{u}, \bar{d}) \tag{6.26}$$

Where t is the time variable, $\bar{u}$ are the dynamic controls and $\bar{d}$ are the design parameters. The dynamic controls are variable in time, for instance, if the mass flow rate or the pitch angle of the launcher could take different values in time they would be a dynamic control. So far in this project, no dynamic controls are considered. In the other hand, the design parameters or static controls are fixed in time. An example is the height at the end of lift-off, h_{lo}, as used in the shooting method that will be described later on.

Constraints can be used in order to limit the search space for the optimal solution. They can be chosen using physical information of the problem. The following constraints can be applied as bounds on the independent variables $(\bar{x}, t, \bar{u}, \bar{d})$

$$\begin{aligned} \text{Time}: \quad & t_{lb} \leq t \leq t_{ub} \\ \text{State Variables}: \quad & \bar{x}_{lb} \leq \bar{x} \leq \bar{x}_{ub} \\ \text{Dynamic Controls}: \quad & \bar{u}_{lb} \leq \bar{u} \leq \bar{u}_{ub} \\ \text{Design Parameters}: \quad & \bar{d}_{lb} \leq \bar{d} \leq \bar{d}_{ub} \end{aligned} \tag{6.27}$$

Where the subscript lb indicates a lower bound and subscript ub indicates an upper bound.

Other possible constraints that can be used are functions of the independent variables that are restrained on the boundary or along the trajectory, this are also called non-linear constraints.

$$\begin{aligned} \text{Initial Boundary Constraints :} \quad & \bar{g}_{0,lb} \leq g_0\left(\bar{x}_0, t_0, \bar{u}_0, \bar{d}\right) \leq \bar{g}_{0,ub} \\ \text{Final Boundary Constraints :} \quad & \bar{g}_{f,lb} \leq g_f\left(\bar{x}_f, t_f, \bar{u}_f, \bar{d}\right) \leq \bar{g}_{f,ub} \\ \text{Path Constraints :} \quad & \bar{p}_{f,lb} \leq p_f(\bar{x}, t, \bar{u}, \bar{d}) \leq \bar{p}_{f,ub} \end{aligned} \tag{6.28}$$

For example, to constrain the aerodynamic loads experienced by the launcher, one could use a path constraint to fix an upper bound on the product of the angle of attack and the dynamic pressure.

The goal of solving an optimal control problem is then to find the variables t , $\bar{u}$ and $\bar{d}$ that cause the trajectory described by the history of the state vector, $\bar{x}$, to minimize an objective function, J, while respecting the bound and the non-linear constraints. The objective function can be defined as

$$\mathbf{J} = f_{obj}(\bar{x}, t, \bar{u}, \bar{d}) \tag{6.29}$$

6.8 Shooting method

The method described in this section is a single shooting method. This strategy to solve optimal control problems is probably the simplest one and has the advantage that the intermediate steps of the optimization process represent a trajectory that has a physical sense, hence it is easy to iterate different initial guess by modifying values by intuition to obtain a feasible and optimal solution.

In this case the optimal control problem is expressed as a two-point boundary value problem where the initial boundary conditions correspond to the values of the state vector at the initial time, $\bar{x}(t_0)$ and the final boundary conditions correspond to the value of the state vector at the final time, $\bar{x}(t_f)$.

The initial boundary conditions vector is then

$$\bar{x}(t_0) = \begin{bmatrix} r(t_0) \\ \lambda(t_0) \\ v(t_0) \\ \phi(t_0) \\ m(t_0) \end{bmatrix} \tag{6.30}$$

In this case, the initial conditions $r(t_0)$, $\lambda(t_0)$, $v(t_0)$, $\phi(t_0)$ are fixed and chosen by the user. For the optimization they will be considered as input parameters, in other words, they are not optimized and they don't need to be bounded. In the other hand, the initial boundary condition $m(t_0)$ will be changed by the optimizer as the mass of propellants m_p is a design parameter and bounds will be needed to avoid, for instance, a negative initial mass value.

The final boundary conditions vector can be written as

$$\bar{x}(t_f) = \begin{bmatrix} r(t_f) \\ \lambda(t_f) \\ v(t_f) \\ \phi(t_f) \\ m(t_f) \end{bmatrix} \tag{6.31}$$

Where the final boundary conditions $r(t_f)$, $v(t_f)$ and $\phi(t_f)$ will be input parameters, i.e. fixed values chosen by the user. And their values will correspond for instance to an orbital state at a desired height. The value of the final longitude angle $\lambda(t_f)$ will be free and the value of the final mass $m(t_f)$ will probably be taken by the optimizer to the value of the empty mass, m_e, when the objective function is to minimize mass.

The bounds on the independent variables and the non-linear constraints can be limited by one side by infinity to provide a one-sided constraint and also can be bounded by both sides by the same value to provide an equality constraint.

As mentioned before, no dynamic controls are used at this stage of the research project, only design parameters. The vector of design parameters used is

$$\bar{d} = \begin{bmatrix} h_{lo} \\ \Delta\theta_{lpo} \\ \Delta t_{lpo} \\ \Delta t_{edp} \\ \xi \\ \Delta\theta_{btl} \\ \theta_{btl_f} \\ m_p \end{bmatrix} \tag{6.32}$$

It is worth noticing that all but the last term of $\bar{d}$ are design parameters related to the guidance laws, but it was also necessary to add the design parameter m_p to have an extra degree of freedom.

The objective function defined corresponds to the mass of consumed propellants at the final time of the trajectory

$$J = m(t_f) - m_e \tag{6.33}$$

In this case the single shooting method works by propagating the equations of motion as in a one-point boundary value problem and trying to find the design parameter vector ,$\bar{d}$, that minimizes the error between the final state and the final boundary conditions and that minimizes the cost function.

Briefly, the single shooting works as presented in figure 7

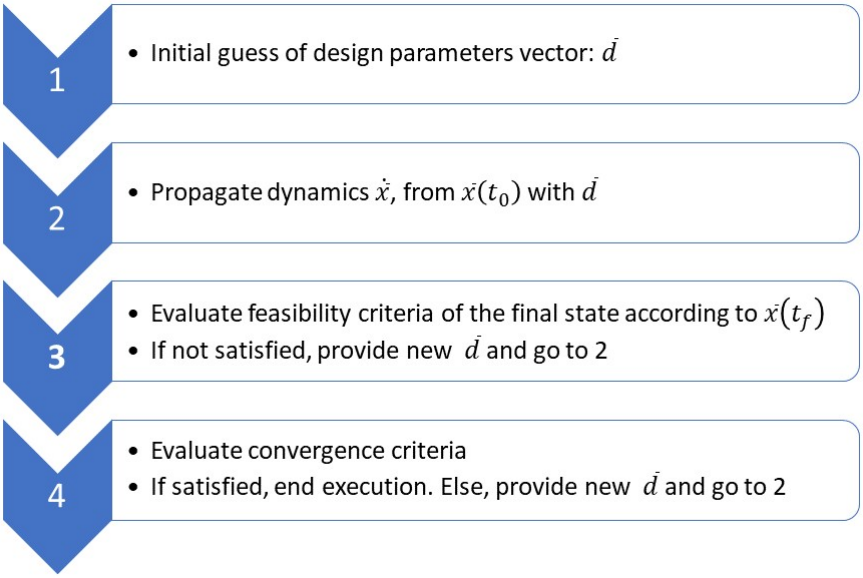

Figure 7: Schema of direct single-shooting method

To propagate the equations of motion as in step 2, a numerical integration method is used. Python's package SciPy allows the implementation of multiple numerical integration methods with its function Solve_IVP. In this work, the Runge-Kutta 5(4) method has been used. For steps 3 and 4, the updates of the design parameter vector, $\bar{d}$, were obtained with the COBYLA optimizer from the NLopt package.

6.8.1 Numerical integration

Numerical integration is fundamental in the solution of optimal control problems by direct methods. In the frame of this work, it is used to propagate the equations of motion from the initial boundary conditions $\bar{x}(t_0)$ by following the dynamics $\dot{\bar{x}}$. Briefly, the objective is to solve the initial boundary value problem

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}(t), t), \quad \mathbf{x}(t_0) = \mathbf{x}_0 \tag{6.34}$$

With the objective of obtaining a solution in the time interval $[t_i, t_{i+1}]$. Or in a equivalent manner, to obtain the solution at $\mathbf{x}(\mathbf{t_{i+1}}) = \mathbf{x_{i+1}}$ departing from $\mathbf{x}(\mathbf{t_i}) = \mathbf{x_i}$. We can integrate to obtain

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \int_{t_i}^{t_{i+1}} \dot{\mathbf{x}}(s)ds = \mathbf{x}_i + \int_{t_i}^{t_{i+1}} \mathbf{f}(\mathbf{x}(s), s)ds \tag{6.35}$$

The Runge-Kutta methods are multiple stage methods, they divide the time interval $[t_i, t_{i+1}]$ into K sub-intervals $[\tau_j, \tau_{j+1}]$, following

$$\tau_j = t_i + h_i\alpha_j, \quad (j = 1, \ldots, K), \quad h_i = t_{i+1} - t_i, \quad 0 \leq \alpha_j \leq 1 \tag{6.36}$$

Where the different τ_j are the stages. By numerically approximating the integral using a quadrature and using the notation $\mathbf{x}(\tau_j) = \mathbf{x}_j$ we can write

$$\int_{t_i}^{t_{i+1}} \mathbf{f}(\mathbf{x}(s), s)ds \approx h_i \sum_{j=1}^{K} \beta_j \mathbf{f}\left(\mathbf{x}_j, \tau_j\right) \tag{6.37}$$

As can be noticed, the values of the state at the time τ_j are needed. The following relationship is used to find them

$$\mathbf{x}\left(\tau_j\right) - \mathbf{x}\left(t_i\right) = \int_{t_i}^{\tau_j} \mathbf{f}(\mathbf{x}(s), s)ds \approx h_i \sum_{l=1}^{K} \gamma_{jl} \mathbf{f}\left(\mathbf{x}_l, \tau_l\right) \tag{6.38}$$

Finally, by combining the last two equations the K-stage Runge-Kutta method is obtained.

$$\begin{aligned} \int_{t_i}^{t_{i+1}} \mathbf{f}(\mathbf{x}(s), s)ds &\approx h_i \sum_{j=1}^{K} \beta_j \mathbf{f}_{ij} \\ \mathbf{f}_{ij} &= \mathbf{f}\left(\mathbf{x}_i + h_i \sum_{l=1}^{K} \gamma_{jl}\mathbf{f}_{il}, t_i + h_i\alpha_j\right) \end{aligned} \tag{6.39}$$

For the Runge-Kutta 5(4) method, the error is controlled by using the method with $K = 4$, but the steps are taking using $K = 5$. In the Solve_IVP function a parameter controlling the maximum step size is used to control the accuracy of the solution.

In the shooting method implemented in this work the algorithm used to minimize the objective function, as described in step 3 of the single shooting description, is called COBYLA. Its name stands for Constrained Optimization By Linear Approximation and is a gradient-free optimizer.

6.8.2 Constrained Optimization By Linear Approximation

The Constrained Optimization By Linear Approximation algorithm, COBYLA, is an optimization algorithm that does not use derivative information to minimize the objective function of a constrained problem. This algorithm was invented by Powell and presented in [18].

Its aim is to solve problems of the form

$$\left.\begin{array}{ll} \text{minimize} & F(\underline{x}), \quad \underline{x} \in \mathcal{R}^n \\ \text{subject to} & c_i(\underline{x}) \geq 0, \quad i = 1, 2, \ldots, m \end{array}\right\} \tag{6.40}$$

Each one of the double-sided bound and non-linear constraints presented in section 6.7 can be converted into two one-sided inequality constraints. When equality constraints are desired it is necessary to provide a tolerance so that it can be expressed by two single-sided inequality constraints.

The algorithm works as explained by Powell

"Each iteration forms linear approximations to the objective and constraint functions by interpolation at the vertices of a simplex and a trust region bound restricts each change to the variables. Thus a new vector of variables is calculated, which may replace one of the current vertices, either to improve the shape of the simplex or because it is the best vector that has been found so far, according to a merit function that gives attention to the greatest constraint violation. The trust region radius p is never increased, and

it is reduced when the approximations of a well-conditioned simplex fail to yield an improvement to the variables, until p reaches a prescribed value that controls the final accuracy"

6.9 Pseudospectral method

As previously mentioned, the pseudospectral methods to be used in this research project are those implemented in Dymos. The objective is to implement an optimization process for the 2D launcher trajectories using the same models as in the shooting method, so that they can be compared. However, as the programming structure in Dymos is complex, the implementations of such models is going to be done gradually.

Dymos is built up into the OpenMDAO environment, this allows to take advantage of its modular programming structure to increase step by step the complexity of the different models that are constructed as Dymos subsystems.

The ODE governing the dynamics of the optimal control problem can then be decomposed into multiple subsystems, and here is where DYMOS excels. As Pseudospectral methods transcribe the optimal control problem into a Non-linear programming one, they need derivative information to find feasible solutions and minimize the cost function. In Dymos, this derivative information is defined for each individual subsystem as the Jacobian of its inputs with respect to its outputs and the software then calculates the total derivatives using the chain rule. In this work the Jacobians were calculated analytically.

The two transcriptions used in Dymos are: Legendre-Gauss-Lobatto (LGL) and Legendre-Gauss-Radau (LGR).

As some similar examples for launcher trajectory optimization are available in Dymos using LGL transcription, it was decided to use this method at the beginning. Although it is important to say that LGR transcription has been found to have better parallelization characteristics in the literature.

The models implemented for the pseudospectral methods were chosen to be simple. The gravity is modeled with a constant magnitude. The drag coefficient was also modeled constant to avoid the calculation of the derivatives of the interpolating polynomial. The atmospheric model corresponds to the exponential atmosphere b described in section 6.3.2.

The ODE is described by 3 subsystems. One calculating the atmospheric parameters, another one the guidance law and the final one the equations of motion. The flux of information between them will be described in a following section with the aid of N2 diagrams. The state vector, $\bar{x}$, initial boundary conditions, $\bar{x}(t_0)$, final boundary conditions, $\bar{x}(t_f)$, and design parameter vector,$\bar{d}$, are the same as the ones used for the shooting method. Dymos allows to fix the final value of mass, $m(t_f)$, thus the objective functions is re-defined to minimize the initial mass of propellants

$$J = m(t_0) \tag{6.41}$$

The steps of the pseudospectral method are illustrated in figure 8.

For step 2 the chosen transcription was Legendre-Gauss-Lobatto. To provide the updates of the optimization variable of step 3 and 4 the non-linear programming solver SLSQP was used.

6.9.1 Legendre-Gauss-Lobatto transcription

The LGL transcription as described in Dymos documentation [19] works as follows:

1. Phase segmentation: The time of each phase is divided into a number of segments chose by the user

2. Discretization: Each segment is discretized by using the LGL nodes that depend on the order chosen for the method.

3. Input: An initial guess for the initialization time and duration of each phase is given by the user. With this information, the value of time at each node is calculated. For the states, the user inputs an initial the guess of the state value at each state discretization node (Even index LGL nodes). For the dynamic controls, the user provides an initial guess at every node. All of these are design parameters.

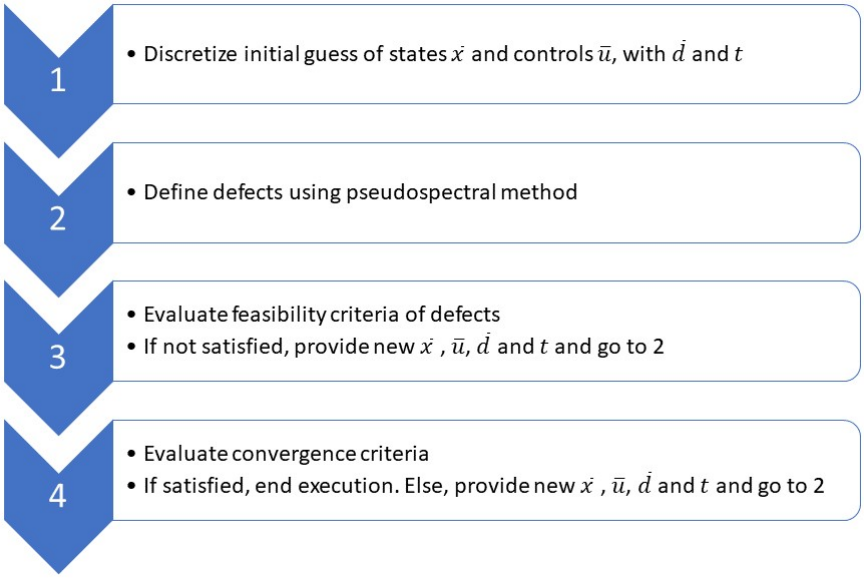

Figure 8: Schema of pseudospectral method

4. Control rate interpolation: As the values of the controls are known at all the nodes, a Lagrange interpolating polynomial can be fitted. This allows to provide derivative values of the dynamic controls as inputs for the ODE. If the segment has 3 LGL nodes, the interpolating polynomial will be of order 2.

5. Evaluation of the ODE at the state discretization nodes: If the segment has 3 LGL nodes, the state discretization nodes will correspond to the nodes at the endpoints. As the values of the states and controls are available at these points, the ODE can be evaluated to calculate the derivatives there.

6. State interpolation: Using the values of the states and their derivatives at the state discretization nodes, an Hermite interpolating polynomial is fitted. This allows to estimate the value of the state and its derivative at the collocation nodes.

7. Evaluation of the ODE at the collocation nodes: The ODE is evaluated this time at the odd-index LGL nodes, i.e. the collocation nodes.

8. Evaluation of the collocation defects: The evaluation of the ODE made in step 7 is compared to the estimated value obtained in step 6 to determine the value of the defect. This value is the compared to threshold to determine if its accurate enough.

9. Iterate: Steps 3 to 8 are repeated until the solution has the desired accuracy and reaches optimality criteria. This is done by the non-linear programming solver who updates the values of the design parameters. When the threshold defined in step 7 is satisfied the solution is considered as feasible. Still the solver iterates to find the optimal feasible solution.

6.9.2 Non-linear programming

Non-linear programming (NLP) groups a compendium of optimization methods using derivative information to minimize a cost function under a given set of constraints. The set of constraints defines a feasible set where the optimal solution is to be looked and it can be comprised of equality and inequality constraints. For the later case, a family of methods called Sequential Quadratic Programming (SQP) is used to define an active set of constraints at each iteration.

According to Betts [11], in the general case for equality constraints, the objective is to minimize

$$F(\mathbf{x}) \tag{6.42}$$

by choosing the variables $\mathbf{x} \in \mathbb{R}^n$, subject to $m \leq n$ equality constraints

$$\mathbf{c}(\mathbf{x}) = \mathbf{0} \tag{6.43}$$

For this purpose the definition of the Lagrangian is required

$$L(\mathbf{x}, \boldsymbol{\lambda}) = F(\mathbf{x}) - \boldsymbol{\lambda}^\top \mathbf{c}(\mathbf{x}) \tag{6.44}$$

$L(\mathbf{x}, \boldsymbol{\lambda})$ is a scalar function. The optimal point belonging to feasible set, (x^*, λ^*), requires finding the stationary points of the Lagrangian by taking its partial derivatives equal to zero

$$\nabla_x L(\mathbf{x}, \boldsymbol{\lambda}) = \mathbf{g}(\mathbf{x}) - \mathbf{G}^\top(\mathbf{x})\boldsymbol{\lambda} = \mathbf{0} \tag{6.45}$$

$$\nabla_\lambda L(\mathbf{x}, \boldsymbol{\lambda}) = -\mathbf{c}(\mathbf{x}) = \mathbf{0} \tag{6.46}$$

Using newton's method to define the variables $\mathbf{x}$ and λ, the following system can be constructed

$$\begin{bmatrix} \mathbf{H}_L & \mathbf{G}^\top \\ \mathbf{G} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p} \\ -\bar{\lambda} \end{bmatrix} = \begin{bmatrix} -\mathbf{g} \\ -\mathbf{c} \end{bmatrix} \tag{6.47}$$

The former is referred to as the Karush-Kuhn-Tucker (KKT) system. Where $\mathbf{p}$ is a search direction, $\mathbf{g}$ is the gradient, $\mathbf{G}$ it the Jacobian matrix, $\mathbf{H_L}$ is the Hessian of the Lagrangian, defined as

$$\mathbf{H}_L = \nabla_x^2 F - \sum_{i=1}^{m} \lambda_i \nabla_x^2 c_i \tag{6.48}$$

A sequential quadratic sub-problem can be defined to find the search direction $\mathbf{p}$ by minimizing

$$\frac{1}{2}\mathbf{p}^\top \mathbf{H}_L \mathbf{p} + \mathbf{g}^\top \mathbf{p} \tag{6.49}$$

subjected to linear constraints

$$\mathbf{G}\mathbf{p} = -\mathbf{c} \tag{6.50}$$

For the inequality constrained case, the former definitions are also used as the strategy to solve the problem defines an active set of constraints to be satisfied at a given iteration.

As an important remark, the Hessian can be approximated numerically using for instance a technique called BFGS, in the frame of quasi-Newton methods.

6.9.3 N2 diagrams

The interactions of the different subsystems used in each phase of the trajectory are described using N2 diagrams.

The inputs enter vertically into the subsystems while their outputs are displayed horizontally. It can be noticed how the EOM subsystem requires the Atmos and Guidance subsystems to provide the values of air density and pitch angle respectively for all phases.

In figure 9, for the lift-off phase the input and output of the guidance subsystem is just $\theta = 90°$

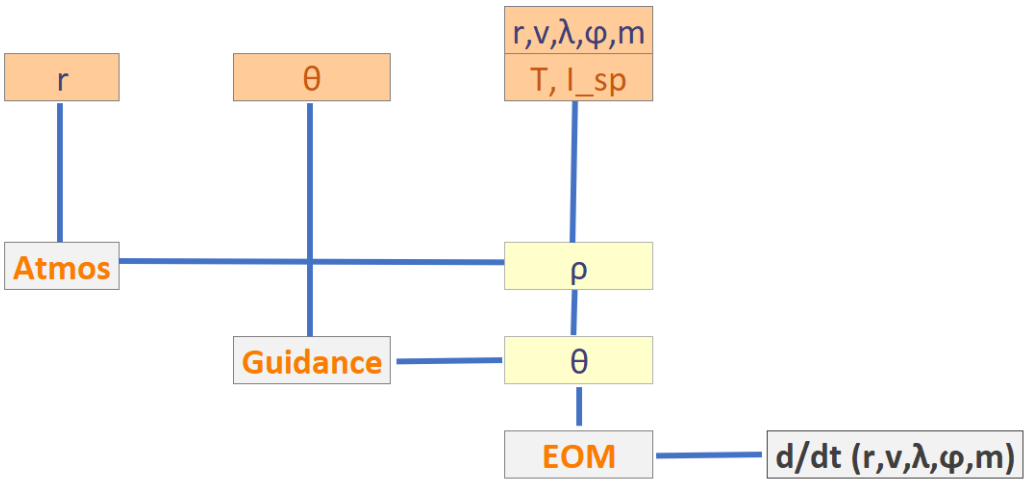

Figure 9: N2 Diagram for lift-off phase

The phases pitch over linear and exponential decay of pitch share one design parameter, namely $\Delta\theta_{lpo}$. They are represented in figures 10 and 11 respectively.

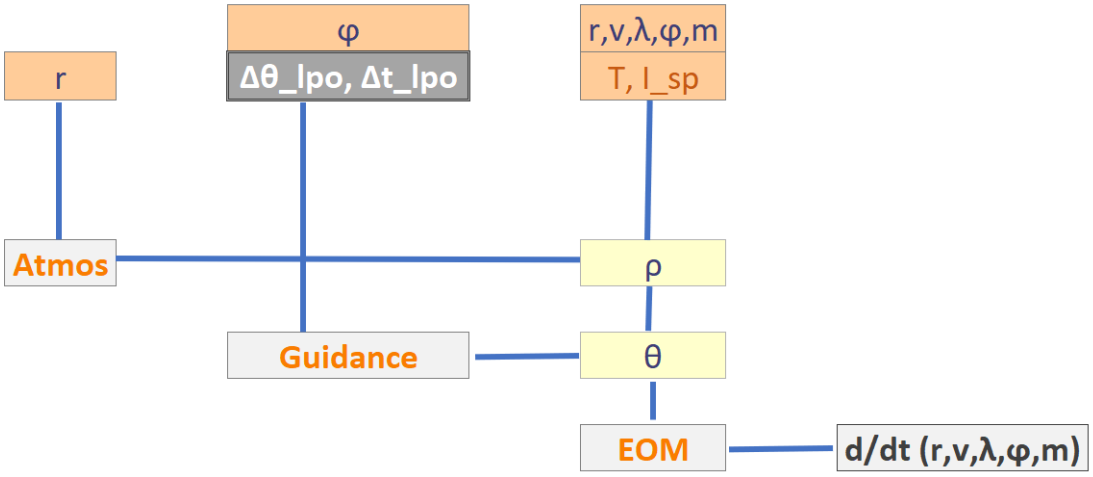

Figure 10: N2 Diagram for linear pitch over phase phase

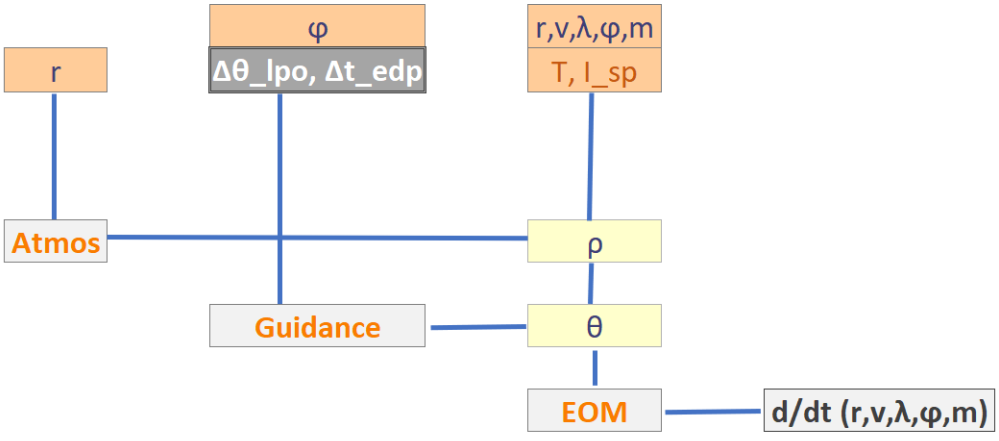

Figure 11: N2 Diagram for exponential decay of pitch phase

As the guidance law for the gravity turn phase is only dependent on the flight path angle, this is the only input for the guidance component of the N2 diagram shown in figure 12. The end of this phase is again at a 50 km height.

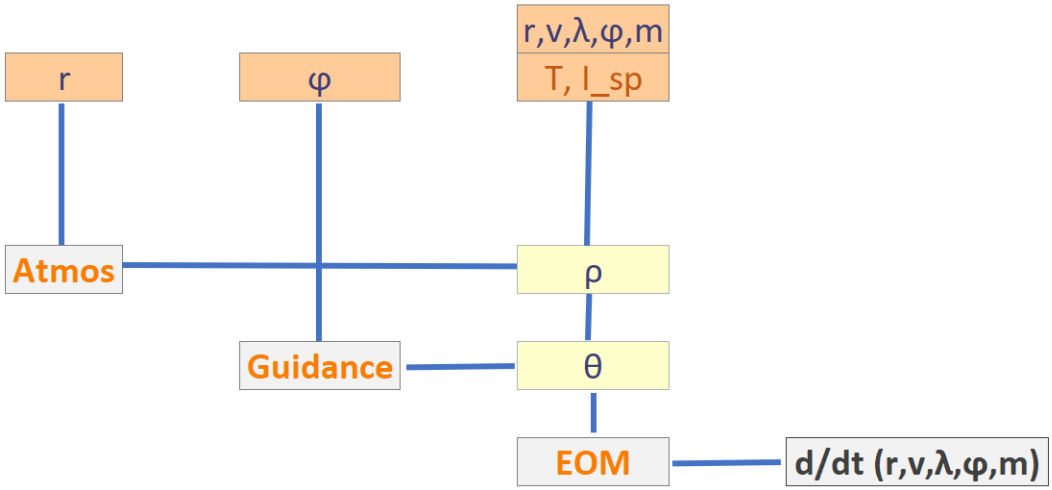

Figure 12: N2 Diagram for gravity turn phase

For the final phase, the atmospheric subsystem is no longer used, as the exponential model losses precision after 20km height and the air density is already very low. This is evidenced in figure 13.

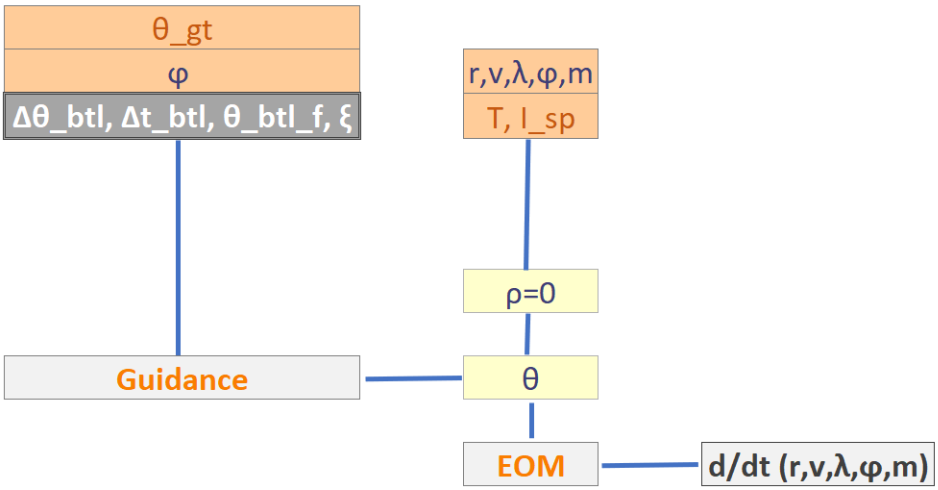

Figure 13: N2 Diagram for bilinear tangent law phase

7 Results and Discussion

7.1 Shooting method

The optimization code was implemented in Python and ran multiple times by iterating different initial guesses for the design parameter vector, $\bar{d}$.

The case study was performed for a single-stage to orbit launcher with the goal of reaching orbit at 200 km. The relative orbital speed for such height was calculated and the path angle of zero degrees was set for a circular orbit. The final boundary conditions thus read

Final boundary condition	**Value**	**Tolerance**	**Units**
$r(t_f)$	6378.135 + 200	2	km
$v(t_f)$	7319.16	500	m/s
$\phi(t_f)$	0	15	deg

Table 3: Final boundary conditions for shooting method

Originally, the COBYLA algorithm does not accept equality bound constraints as the ones that are being imposed, but the algorithm found in the package NLopt does implement equality bound constraints by using a tolerance to formulate them as inequality constraints.

The vehicle used has the following characteristics

Input parameter	**Value**	**Units**	**Description**
m_e	35	ton	Empty mass
μ	1990	kg / s	Mass flow rate
I_{sp}	400	s	Specific impulse
S	37.5	m^2	Reference area
A_e	10	m^2	Nozzle expansion area

Table 4: Vehicle parameters for shooting method

The initial guess for the design parameter vector and its bounds are

Design parameter	**Value**	**lb**	**up**	**Units**	**Description**
h_{lo}	222.45	150	300	m	Height end of lift-off
$\Delta\theta_{lpo}$	1.47	0.1	8	deg	Change in pitch angle
Δt_{lpo}	20.32	1	40	s	Phase duration
Δt_{edp}	18.4	1	25	s	Phase duration
ξ	-0.01	-1	1		Shape coefficient btl
$\Delta\theta_{btl}$	-1.66	-60	60	deg	Change in pitch angle
θ_{btl_f}	-33.56	-50	60	deg	Final pitch angle
m_p	472.6	370	800	ton	Mass of propellants

Table 5: Initial guess - Design parameters for shooting method

The initial boundary conditions for the initial guess are shown below. Only the mass at the initial time $m(t_0)$ will vary from iteration to iteration as the mass of propellants is a design parameter.

Initial boundary condition	**Value**	**Units**
$r(t_0)$	6378.135	km
$\lambda(t_0)$	0	deg
$v(t_0)$	1	m/s
$\phi(t_0)$	90	deg
$m(t_0)$	$m_e + m_p = 507.6$	ton

Table 6: Initial boundary conditions of the initial guess for shooting method

The amplitude parameter of the bilinear tangent law phase was set to $a = 100$, the maximum step size for the Runge Kutta 5(4) method was set to $1s$. The initial step size for the COBYLA algorithm was set to 1 for all the design parameters that also were normalized before being input into the optimizer. The stop criteria used for the algorithm was the relative tolerance on the design parameter vector and was set to 10^{-4}.

The first iteration of the algorithm, corresponding to the initial guess is shown in figure 14. It is important to mention that such initial guess is not randomly chosen, but the result of a fine tuning trough multiple iterations. It is also worth noticing that the initial guess is not feasible, as for instance the height is not within the 2 km tolerance around the target of 200 km that is required in the final boundary conditions.

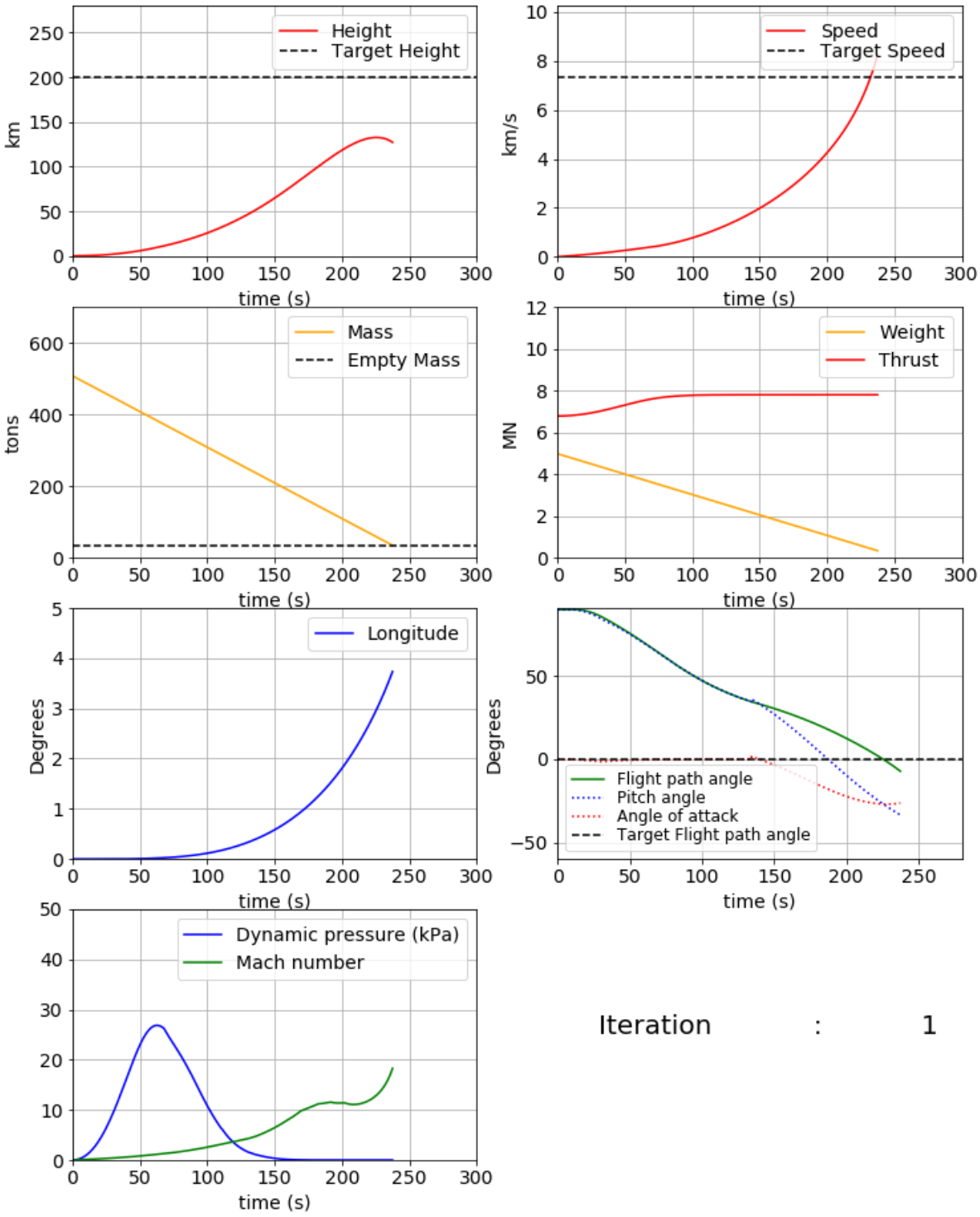

Figure 14: Initial guess for shooting method

After 275 iterations the algorithm converged to a feasible and optimal solution whose design parameter vector is shown in table 7. The reader can notice that all of the design parameters lie within the specified bounds.

Design parameter	Value	lb	up	Units	Description
h_{lo}	212.8	150	300	m	Height end of lift-off
$\Delta\theta_{lpo}$	1.77	0.1	8	deg	Change in pitch angle
Δt_{lpo}	25.33	1	40	s	Phase duration
Δt_{edp}	20.0	1	25	s	Phase duration
ξ	-0.01	-1	1		Shape coefficient btl
$\Delta\theta_{btl}$	0.9	-60	60	deg	Change in pitch angle
θ_{btl_f}	-36.4	-50	60	deg	Final pitch angle
m_p	438.4	370	800	ton	Mass of propellants

Table 7: Optimization result - Design parameters for shooting method

The final state of the vehicle is shown in table 8. It can be verified that the values of radius, speed and flight path angle correspond to those of the final boundary conditions that were specified or lie within the tolerance range, thus they correspond to a feasible solution. It is also worth noticing that the mass at the final time $m(t_f)$ corresponds to the empty mass, this means that there are not propellants left in the launcher at the final time as it can be expected from a solution that minimizes the consumed mass of propellants.

Final state	Value	Units
$r(t_f)$	6378.135 + 200	km
$\lambda(t_f)$	2.85	deg
$v(t_f)$	7320	m/s
$\phi(t_f)$	0	deg
$m(t_f)$	35	ton

Table 8: Final state - results of shooting method

The final value of the cost function corresponds to that of the mass of propellants as there is no propellants left in the thank at the final time. This optimal solution corresponds to a local minimum and running with stricter stop criteria only increments the number of iterations but no significant improvement of the cost function was evidenced. A different initial guess could lead to another feasible an optimal value.

$$J = m(t_f) - m_e = 438.4 \quad ton \tag{7.1}$$

In figure 15 the state history is shown. The height plot shows that the target of 200 km is reached and that the rate of change of the height tends to zero as the time approaches the final time, this is due to the radial deceleration of the rocket in the last portion of the flight that can also be evidenced in the pitch angle plot. During the last 30 s of the trajectory the pitch angle is negative, meaning that the thrust vector has a downward component if seeing in the Local vertical - Local horizontal reference frame. A similar behavior was evidenced in the literature with trajectories presenting "dives" to convert potential energy into kinetic energy as in [20] and [8]. Also in figure 15 it can noticed that the speed and flight path angle reach the values specified in the final boundary conditions. The plot of thrust shows how the high atmospheric pressure at low altitudes at the beginning of the trajectory decreases the magnitude of the thrust. The longitude angle increases as expected for an easterly launch in the equatorial plane. With the plot of dynamic pressure one can corroborate that at $50km$ height the aerodynamic forces are very low.

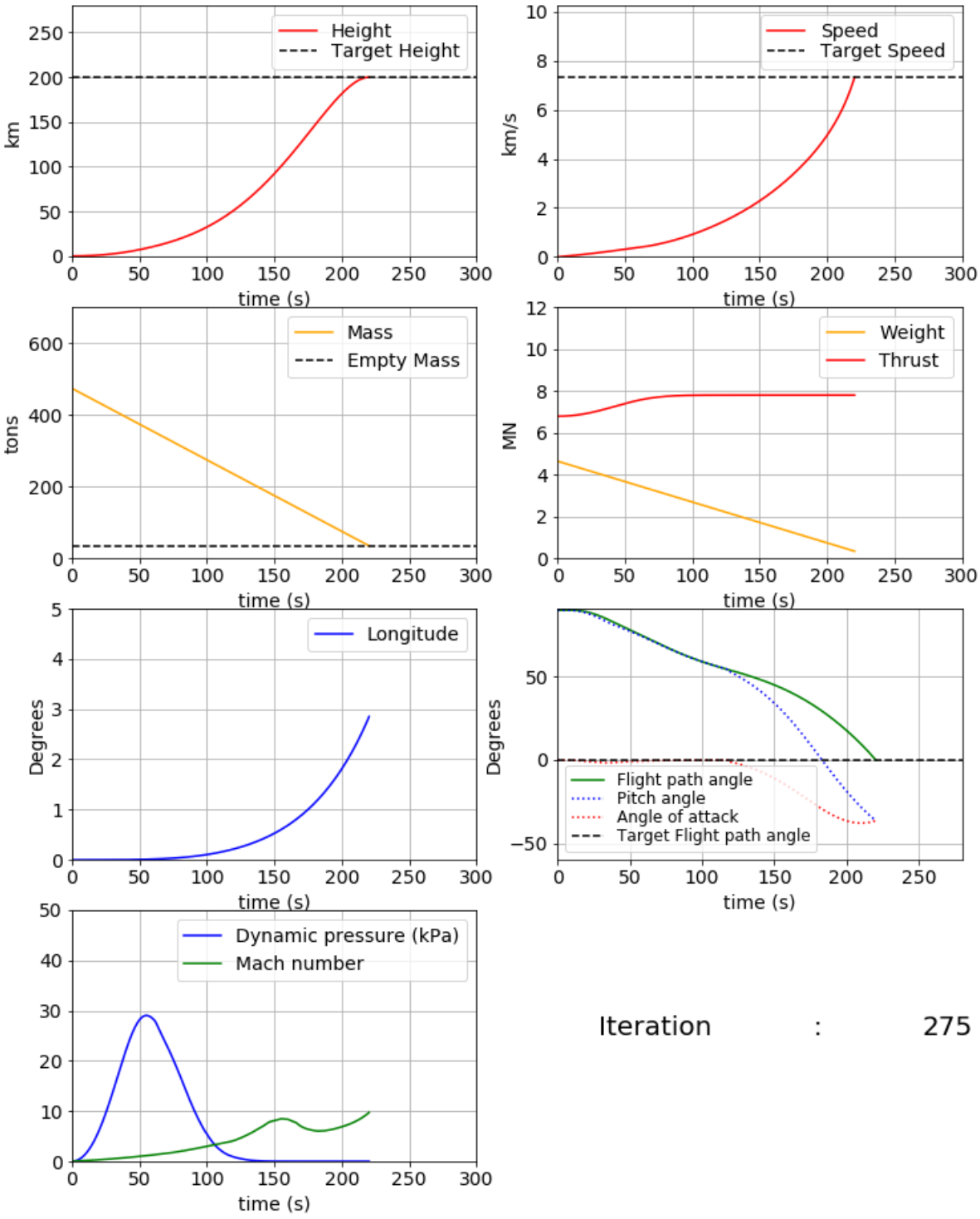

Figure 15: Final state history for shooting method

7.2 Pseudo spectral method

The optimization was ran in Dymos using the LGL pseudospectral method with polynomials of order 3. 7 segments were used for all the phases but the bilinear tangent law phase that used 20.

The same orbit as with the shooting method was targeted. The final boundary conditions were

Final boundary condition	Value	Tolerance	Units
$r(t_f)$	6378.135 + 200	2	km
$v(t_f)$	7319.16	73	m/s
$\phi(t_f)$	0	1	deg

Table 9: Final boundary conditions for pseudospectral method

The vehicle parameters were the same as in table 4. The initial guess that was used for the design parameters vector corresponded to the optimization results of the shooting method.

Design parameter	Value	lb	up	Units	Description
h_{lo}	212.8	150	300	m	Height end of lift-off
$\Delta\theta_{lpo}$	1.77	0.1	8	deg	Change in pitch angle
Δt_{lpo}	25.33	1	40	s	Phase duration
Δt_{edp}	20.0	1	80	s	Phase duration
ξ	-0.01	-1	1		Shape coefficient btl
$\Delta\theta_{btl}$	0.9	-60	60	deg	Change in pitch angle
θ_{btl_f}	-36.4	-50	60	deg	Final pitch angle
m_p	438.4	$-\infty$	∞	ton	Mass of propellants

Table 10: Initial guess - Design parameters for pseudospectral method

The initial boundary conditions vector only varied in the initial mass

Initial boundary condition	Value	Units
$r(t_0)$	6378.135	km
$\lambda(t_0)$	0	deg
$v(t_0)$	1	m/s
$\phi(t_0)$	90	deg
$m(t_0)$	$m_e + m_p = 473$	ton

Table 11: Initial boundary conditions of the initial guess for pseudospectral method

The values of the initial guess for the states in Dymos were input by defining values at the two boundaries of each phase and linearly interpolating them to obtain the values at the nodes. The values at the boundaries were chosen similar to those of the result of the optimization performed with the shooting method. In the future, a method to input more precise data from the result of the shooting method to the initial guess of the pseudospectral method could be considered.

In figure 16 the initial guess for the pseudospectral method is shown. It can be seen that the values of the pitch angle during the bilinear tangent law phase present some noise. This is probably link to a bad management of the time variables for the calculation of the Jacobian of the guidance component at such phase. The problem has not been solved yet, but the steps to work in a possible solution have been discussed. It will require the use of the "duration_targets" method of the "set_time_options" function of Dymos for this phase.

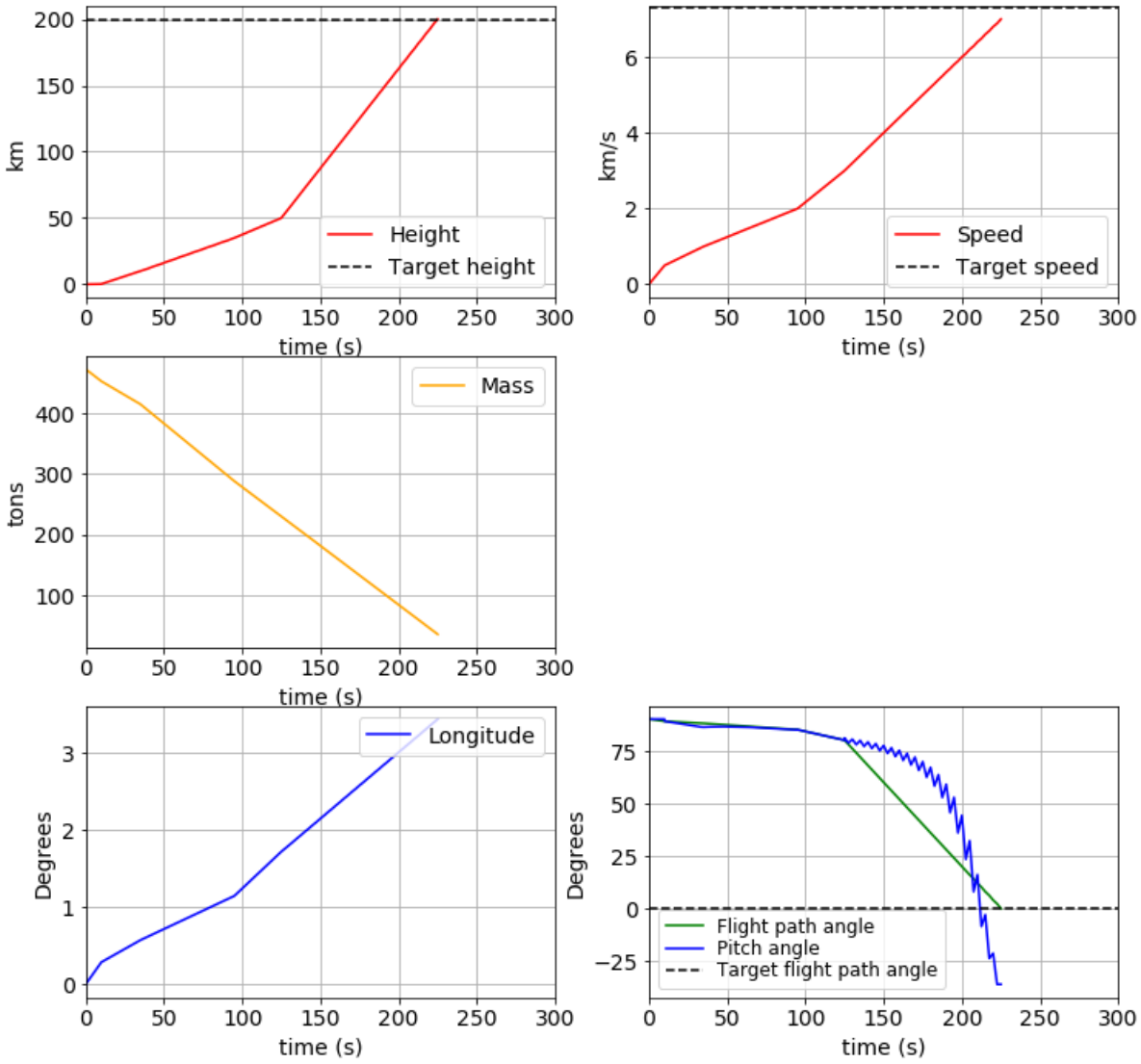

Figure 16: Initial guess pseudospectral method

Although no control of the tolerance for convergence in Dymos has been implemented yet, the results with the default values give as result after 57 iterations the state history shown in figure 17.

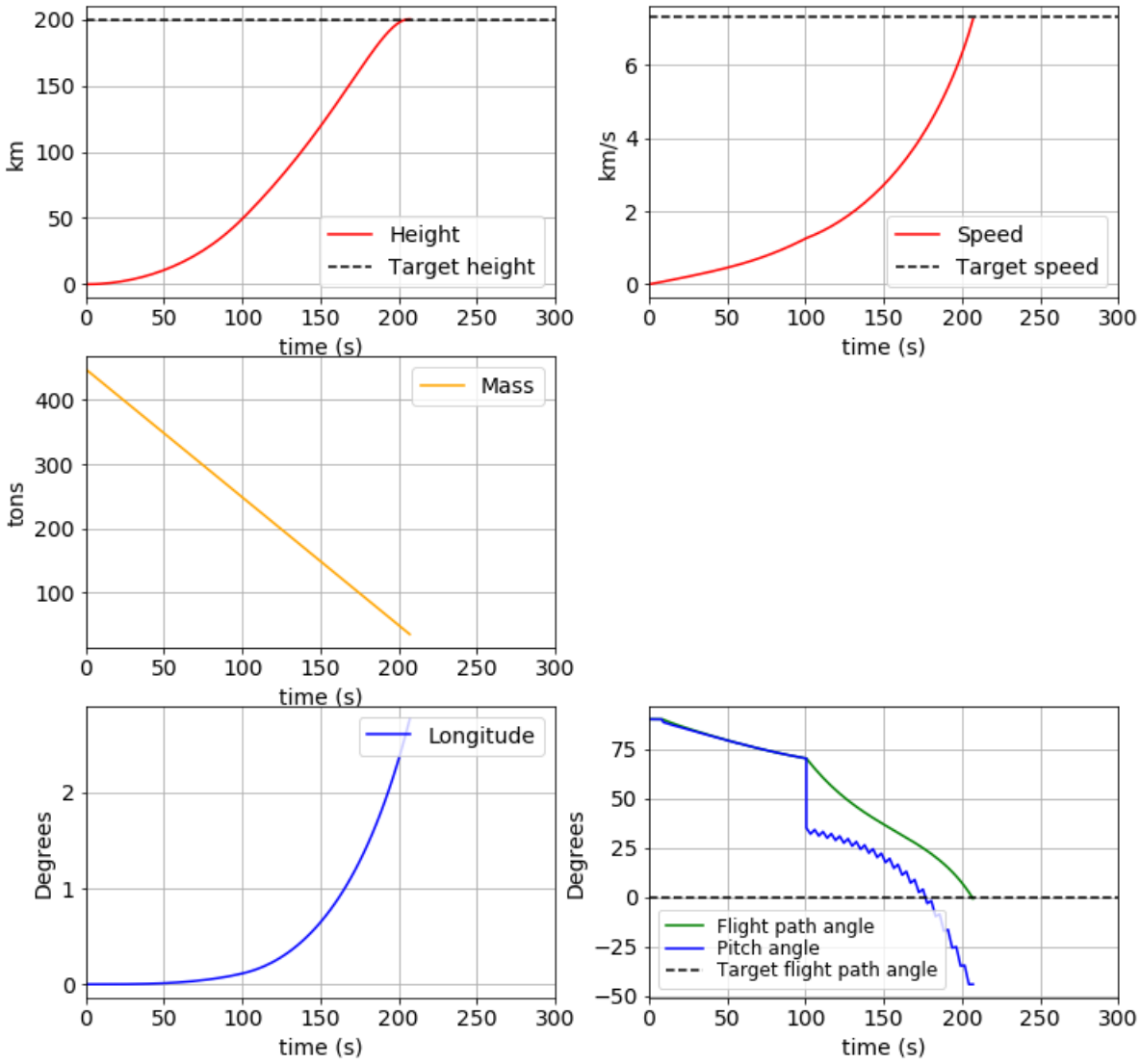

Figure 17: Final state history for pseudospectral method

As can be noticed the optimization results also contain the noise for the pitch angle during the bilinear tangent phase and the focus of immediate work will be to fix this issue before continuing to upgrade the gravity, drag coefficient and atmospheric models. Although, the other states seem to be behaving as expected. The "simulate" method of Dymos allows to propagate numerically the equations of motion using the vector of design parameters obtained during the optimization process. To corroborate that the pseudospectral method is yielding a feasible and physically meaningful solution, both states history can be superposed to verify that they match. So far the "simulate method" has not been succesfully implemented due to a singularity caused by a time parameter of the pitch over linear phase that is causing a division by zero.

8 Conclusions and perspective

The optimization of 2D launcher trajectories by using a single shooting method with the gradient-free optimizer COBYLA has been demonstrated. The method was used successfully to find a feasible trajectory for an orbital insertion at 200 km height of a single stage to orbit launcher.

The progress on the implementation of 2D trajectory optimization for launchers using pseudospectral methods in Dymos has also been discussed and two key issues to be solved were pointed: The noise on the pitch angle during the bilinear tangent law phase probably due to a bad management of a time variable and a singularity during the execution of the validation method "simulate", also caused by a time variable.

The immediate actions will of course consist on the solution of the two issues that were pointed out in the implementation of the pseudospectral method, followed by the refinement of its gravity, atmospheric and drag coefficient models. Once the pseudospectral method is validated, the following step will be to implement the optimization multiple variables from different disciplines to study the feasibility of using pseudospectral methods for the MDAO of launch vehicles and the extension to 3D trajectories with the study of different staging and recovery scenarios.

9 Corrections and progress 2020/09/07

The issues related to the definition of the jacobian of the guidance component for the bilinear tangent law phase for the pseudospectral method were solved by using the option "duration_targets" from Dymos. The "simulate" method is now working correctly, so it can be concluded that the results obtained correspond to the physics of the optimal control problem.

Furthermore, the atmospheric model for the pseudospectral method was updated to use the atmospheric model based on the 1976 and 1962 U.S. Standard Atmospheres. To do so, data points for air pressure, density and temperature were calculated every kilometer height up to 600 km and then an Akima interpolator was used to transform the discrete data into continuous data whose derivative can be provided. The Akima interpolator is used for this purpose by the Dymos team. The Cd. vs Mach curve was updated to use a Pchip interpolator as it was found to have the smaller overshoot for the second derivative between the different interpolators shown in figure 18

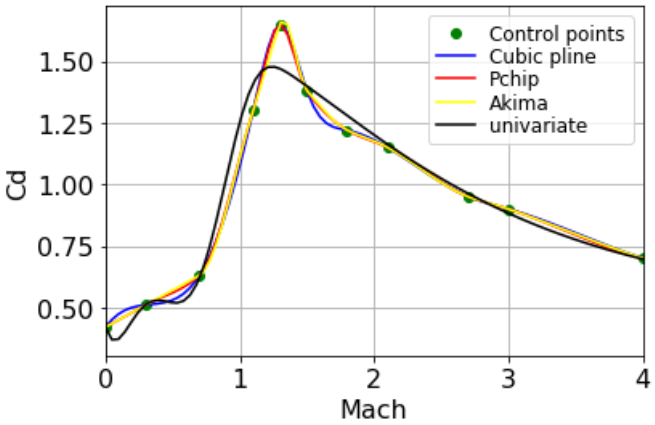

Figure 18: Comparison of different interpolators

The results for the updated pseudospectral method are shown in figure 19. It can be seen that that the noise in the pitch angle, θ, disappeared and that the results match those of the simulate method, labeled with "sim" in the figure.

Also, the optimization was performed using the throttle as a dynamic control. the throttle command controls the mass flow rate and can take values between 0 and 1. The throttle history can be seen in figure 20. It can be noticed that during some portions of the trajectory this dynamic control varies really fast, in reality this behavior would not be possible for a real engine that is subjected to different physical constraints. In the future, the modeling of such constraints could be considered.

As the pseudospectral method is working fine for the SSTO, the following step will be to implement a Two-State To Orbit vehicle (TSTO).

[0]This section was created to show the progress and corrections on the report as of 2020/09/07 as required by the dean of studies

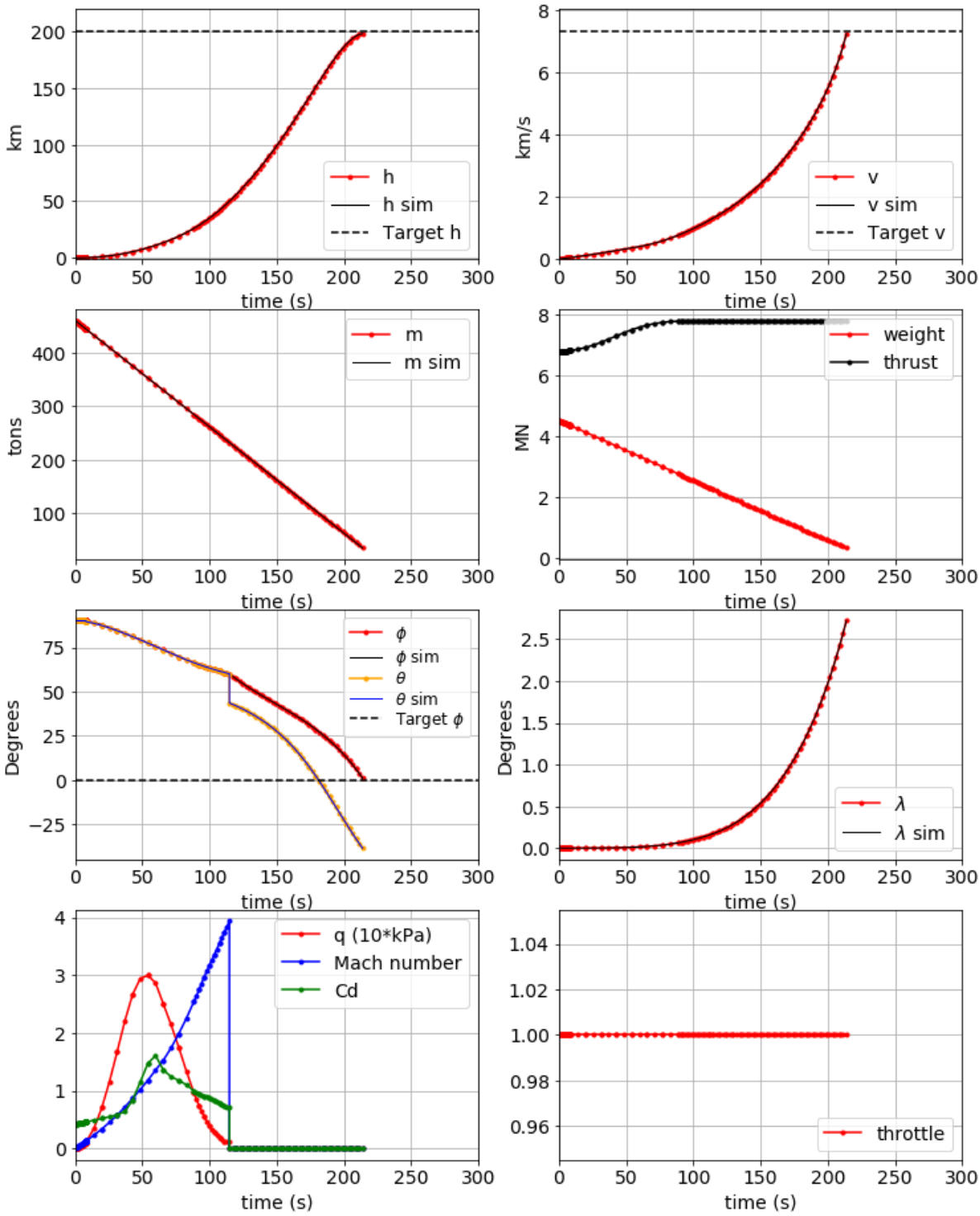

Figure 19: Updated final state history for pseudospectral method

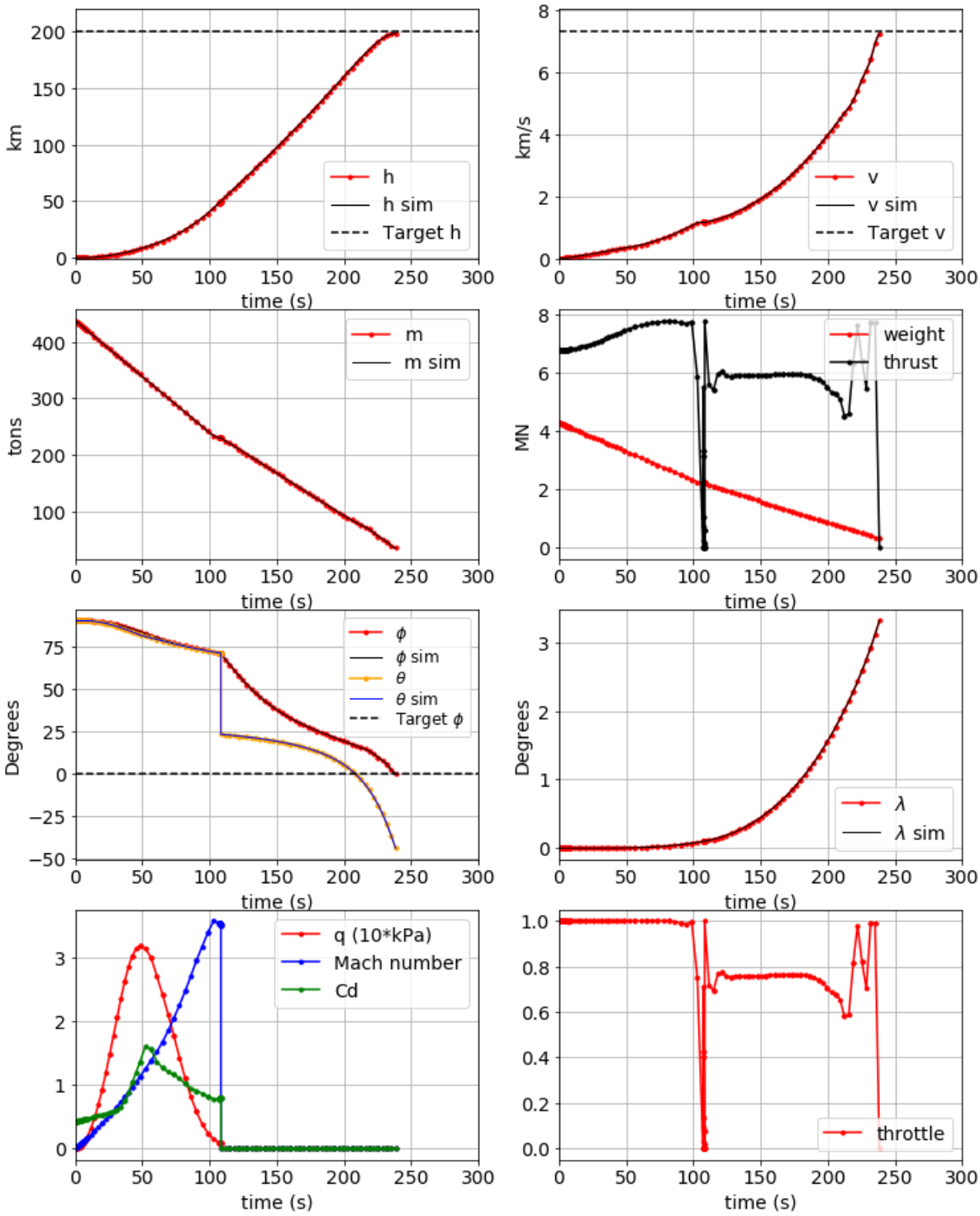

Figure 20: Updated final state history for pseudospectral method. Dynamic control

References

[1] L. E. Briese, K. Schnepper, and P. Acquatella B., "Advanced modeling and trajectory optimization framework for reusable launch vehicles," Mar. 2018.

[2] F. Castellini, "MULTIDISCIPLINARY DESIGN OPTIMIZATION FOR EXPENDABLE LAUNCH VEHICLES," 2012.

[3] G. D. C. B. de Volo, "Vega Launchers' Trajectory Optimization Using a Pseudospectral Transcription," 2017.

[4] N. Berend and C. Talbot, "Overview of some optimal control methods adapted to expendable and reusable launch vehicle trajectories," *Aerospace Science and Technology*, Apr. 2006.

[5] Z. Wang and Z. Wu, "Six-DOF trajectory optimization for reusable launch vehicles via Gauss pseudospectral method," *Journal of Systems Engineering and Electronics*, Apr. 2016.

[6] A. Tewari, *Atmospheric and Space Flight Dynamics: Modeling and Simulation with MATLAB and Simulink*. Boston, Mass.: Birkhäuser, 2007. OCLC: 180887853.

[7] J. T. Betts, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*. Advances in Design and Control, Society for Industrial and Applied Mathematics, Jan. 2010.

[8] E. S. Hendricks, R. D. Falck, and J. S. Gray, "Simultaneous Propulsion System and Trajectory Optimization," AIAA AVIATION Forum, American Institute of Aeronautics and Astronautics, June 2017.

[9] A. Pagano and E. Mooij, "Global Launcher Trajectory Optimization for Lunar Base Settlement," in *AIAA/AAS Astrodynamics Specialist Conference*, (Toronto, Ontario, Canada), Aug. 2010.

[10] M. Sagliano, T. Tsukamoto, J. A. Macés-Hernández, D. Seelbinder, S. Ishimoto, and E. Dumont, "Guidance and Control Strategy for the CALLISTO Flight Experiment," July 2019.

[11] J. T. Betts, "Survey of Numerical Methods for Trajectory Optimization," *Journal of Guidance, Control, and Dynamics*, vol. 21, no. 2, 1998.

[12] A. V. Rao, "A SURVEY OF NUMERICAL METHODS FOR OPTIMAL CONTROL," 2010.

[13] A. L. Herman and B. A. Conway, "Direct optimization using collocation based on high-order Gauss-Lobatto quadrature rules," *Journal of Guidance, Control, and Dynamics*, 1996.

[14] D. Garg, M. Patterson, C. Darby, C. Francolin, G. Huntington, W. Hager, and A. Rao, "Direct Trajectory Optimization and Costate Estimation of General Optimal Control Problems Using a Radau Pseudospectral Method," June 2012.

[15] R. D. Falck, J. S. Gray, and B. Naylor, "Parallel Aircraft Trajectory Optimization with Analytic Derivatives," in *17th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, American Institute of Aeronautics and Astronautics, June 2016.

[16] R. D. Falck and J. S. Gray, "Optimal Control within the Context of Multidisciplinary Design, Analysis, and Optimization," in *AIAA Scitech 2019 Forum*, American Institute of Aeronautics and Astronautics, 2019.

[17] A. Pagano, *Global Launcher Trajectory Optimization for Lunar Base Settlement*. PhD thesis, TU DELFT, May 2010.

[18] M. J. D. Powell, "A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation," in *Advances in Optimization and Numerical Analysis* (S. Gomez and J.-P. Hennart, eds.), Mathematics and Its Applications, pp. 51–67, Dordrecht: Springer Netherlands, 1994.

[19] "Dymos: Open-source Optimal Control for Multidisciplinary Systems — dymos 0.15.0 documentation." https://openmdao.github.io/dymos/index.html.

[20] M. Balesdent, *Multidisciplinary Design Optimization of Launch Vehicles*. PhD thesis, Ecole Centrale de Nantes, Jan. 2012.

A Derivation of 2D EoM for a rotating planet

The launcher is modeled as a particle moving in the equatorial plane of a rotating planet according to Newton's Second Law of dynamics. An Inertial Earth Centered reference frame, $\mathcal{F}_I \coloneqq \{x_I, y_I\}$, is defined as inertial reference frame. In this reference frame it can be written for a variable mass system

$$\mathbf{f} = m\mathbf{a_I} \tag{A.1}$$

Where m is the mass of the launcher at any given time, $\mathbf{a_I}$ is the inertial acceleration and $\mathbf{f}$ is the vector comprised of aerodynamic forces (drag ($\mathbf{D}$) and lift ($\mathbf{L}$)), thrust force ($\mathbf{T}$) and gravity force ($m\mathbf{g}$).

The position of the launcher is better expressed with respect to a fixed point on Earth, this is achieved with the Rotating Earth Centered reference frame, $\mathcal{F}_R \coloneqq \{x_R, y_R, z_R\}$, this frame is the one used by the reader to know its own position in Earth and has unitary vectors $\hat{\mathbf{I}}$, $\hat{\mathbf{J}}$ and $\hat{\mathbf{K}}$. To give a notion of the attitude of the vehicle the Local Vertical - Local Horizontal reference frame, $\mathcal{F}_t \coloneqq \{x_t, y_t, z_t\}$, is defined with its center lying at a fix point on the launcher (Ex.: the geometric center) and with unitary vectors $\hat{\mathbf{i}}$, $\hat{\mathbf{j}}$ and $\hat{\mathbf{k}}$. Finally, a fourth reference frame orientated with respect to the launcher velocity component is defined to better express the aerodynamic forces and is called the Wind reference frame $\mathcal{F}_w \coloneqq \{x_w, y_w\}$. All of the aforementioned reference frames are shown in figure 1.

For a vector $\mathbf{p}$ expressed in terms of the unitary vectors of the reference frame $\mathcal{F}_t$, the transformation matrix expressed in A.2 can be used to transform it and express it in terms of the unitary vectors of reference frame $\mathcal{F}_w$.

$$\mathbf{p_w} = \mathbf{R}'_\phi \mathbf{p_t} \quad , \quad \mathbf{R}_\phi = \begin{bmatrix} \cos(\phi) & -\sin(\phi) \\ \sin(\phi) & \cos(\phi) \end{bmatrix} \tag{A.2}$$

To deduce the equations of motion some expressions for the change in time of the position $\dot{r}$ and the longitude $\dot{\lambda}$ will be find by using the velocity $\mathbf{v}$ as a firs step. Then the inertial velocity $\mathbf{v_I}$ will be used to find the inertial acceleration $\mathbf{a_I}$ and the necessary transformations will be done to use equation A.1 and finally find expressions for the change of velocity $\dot{v}$ and flight path angle $\dot{\phi}$.

From figure 1 a first expression for the velocity of the launcher relative to $\mathcal{F}_R$ but expressed in terms of the unitary vectors of $\mathcal{F}_t$ can be deduced.

$$\mathbf{v} = v(\cos(\phi)\mathbf{i} + \sin(\phi)\mathbf{j}) \tag{A.3}$$

A second expression for $\mathbf{v}$ is obtained by the relation with the linear change of the position vector $\dot{\mathbf{r}}$ and its angular rate $\mathbf{\Omega}$. Given that $\mathbf{r}$ can be expressed in terms of $\mathcal{F}_t$ unitary vectors as $r\mathbf{j}$, it can be written

$$\mathbf{v} = \dot{r}\mathbf{j} + \mathbf{\Omega} \times (r\mathbf{j}) \tag{A.4}$$

$\mathbf{\Omega}$ represents the angular velocity of the reference frame $\mathcal{F}_t$ with respect to $\mathcal{F}_r$ and can be written as $\Omega_x\mathbf{i} + \Omega_y\mathbf{j} + \Omega_z\mathbf{k}$. Solving the cross product it reads

$$\mathbf{v} = \dot{r}\mathbf{j} + \Omega_x r\mathbf{k} - \Omega_z r\mathbf{i} \tag{A.5}$$

By comparison of equations A.3 and A.5, it is obtained

$$\dot{r} = v\sin(\phi) \tag{A.6}$$

$$\Omega_z r = -v\cos(\phi) \tag{A.7}$$

As can be noticed in figure 1 the angular velocity of the reference frame $\mathcal{F}_t$ with respect to $\mathcal{F}_r$ is equal to the rate of change of λ, hence it can be expressed $\mathbf{\Omega} = \dot{\lambda}\mathbf{K}$ or whats is equivalent $\mathbf{\Omega} = -\dot{\lambda}\mathbf{k}$. Using this result and rewriting equation A.4 as

$$\mathbf{v} = \dot{r}\mathbf{j} + \dot{\lambda}r\mathbf{i} \tag{A.8}$$

and equating with A.7

$$\dot{\lambda} = \frac{v}{r}\cos(\phi) \tag{A.9}$$

the equations A.6 and A.9 are obtained and represent the first two EoM.

The inertial velocity $\mathbf{v_i}$ is comprised by the velocity $\mathbf{v}$ and the tangential component due to Earth's angular velocity ω. As the later can be expressed as $\omega\mathbf{K}$ or the equivalent $-\omega\mathbf{k}$, it reads

$$\mathbf{v_I} = \mathbf{v} + \omega r\mathbf{i} \tag{A.10}$$

The inertial acceleration can be find by derivating $\mathbf{v_I}$. This time the angular rate of change is $\omega + \mathbf{\Omega}$ or the equivalent $-(\omega + \dot{\lambda})\mathbf{k}$. Hence it can be written

$$\mathbf{a_I} = \frac{d\mathbf{v_I}}{dt} = \dot{\mathbf{v}} + \omega\dot{r}\mathbf{i} - (\omega + \dot{\lambda})\omega r\mathbf{j} \tag{A.11}$$

taking the derivative of $\mathbf{v}$ as expressed in A.8 and again using $-(\omega + \dot{\lambda})\mathbf{k}$ as the angular rate of change, it reads

$$\dot{\mathbf{v}} = \ddot{r}\mathbf{j} + \ddot{\lambda}r\mathbf{i} + \dot{\lambda}\dot{r}\mathbf{i} - (\omega + \dot{\lambda})\mathbf{k} \times (\dot{r}\mathbf{j} + \dot{\lambda}r\mathbf{i}) \tag{A.12}$$

By using A.12 and simplifying, equation A.11 can thus be rewritten as

$$\mathbf{a_I} = [\ddot{\lambda}r + 2\dot{r}(\dot{\lambda} + \omega)]\mathbf{i} + [\ddot{r} - r(\omega + \dot{\lambda})^2]\mathbf{j} \tag{A.13}$$

By finding the second derivatives of $\mathbf{r}$ and λ

$$\ddot{r} = v\cos(\phi)\dot{\phi} + \sin(\phi)\dot{v} \tag{A.14}$$

$$\ddot{\lambda} = -\frac{v\sin(\phi)\dot{\phi}}{r} + \frac{\cos(\phi)\dot{v}}{r} - \frac{v^2\sin(\phi)\cos(\phi)}{r^2} \tag{A.15}$$

and by replacing in equation A.13, we can rewrite $\mathbf{a_I}$ once again but in matrix form

$$\mathbf{a_I} = \begin{bmatrix} 2\omega v\sin(\phi) - v\sin(\phi)\dot{\phi} + \cos(\phi)\dot{v} + \frac{v^2\sin(2\phi)}{2r} \\ -\omega^2 r - 2\omega v\cos(\phi) + v\cos(\phi)\dot{\phi} + \sin(\phi)\dot{v} - \frac{v^2\cos^2(\phi)}{r} \end{bmatrix} \tag{A.16}$$

So far $\mathbf{a_I}$ is expressed in terms of the unitary vectors $\mathbf{i}$ and $\mathbf{i}$ that are the basis of $\mathcal{F}_t$. A final transformation to the Wind reference frame $\mathcal{F}_w$ must be done using the matrix transformation expressed in equation A.2. The new unitary vectors were not explicitly defined but $\mathbf{a_I}$ as a function of them reads

$$\mathbf{a_I} = \begin{bmatrix} -\omega^2 r\sin(\phi) + \dot{v} \\ -\omega^2 r\cos(\phi) - 2\omega v + v\dot{\phi} - \frac{v^2\cos(\phi)}{r} \end{bmatrix} \tag{A.17}$$

Also in the $\mathcal{F}_w$ reference frame, the forces exerted by the launcher as can be seen in figure 1 can be expressed in matrix form as

$$\mathbf{f} = \begin{bmatrix} -D + T\cos(\theta - \phi) - gm\sin(\phi) \\ L + T\sin(\theta - \phi) - gm\cos(\phi) \end{bmatrix} \tag{A.18}$$

Finally, by applying Newton's second law as expressed in equation A.1, using A.17 and A.18 and solving for $\dot{v}$ and $\dot{\phi}$ we can find the two remaining equations of motion

$$\dot{v} = \frac{-D + T\cos(\theta - \phi)}{m} + \left(-g + \omega^2 r\right)\sin(\phi) \tag{A.19}$$

$$\dot{\phi} = \frac{L}{mv} + \frac{T\sin(\theta - \phi)}{mv} + \frac{(\omega^2 r - g)\cos(\phi)}{v} + 2\omega + \frac{v\cos(\phi)}{r} \tag{A.20}$$