SAPIENZA
UNIVERSITÀ DI ROMA

Convex Optimization of Launch Vehicle Ascent Trajectories

Facoltà di Ingegneria Civile e Industriale
Dipartimento di Ingegneria Meccanica e Aerospaziale

Dottorato di Ricerca in Ingegneria Aeronautica e Spaziale – XXXIV Ciclo

Candidate

Boris Benedikter
ID number 1580339

Thesis Advisor

Dr. Alessandro Zavoli

Co-Advisors

Dr. Enrico Cavallini
Prof. Guido De Matteis

Convex Optimization of Launch Vehicle Ascent Trajectories
Ph.D. thesis. Sapienza – University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Version: April 17, 2023

Author's email: boris.benedikter@uniroma1.it

Dedicated to
the people who always believed in me

Abstract

This thesis investigates the use of convex optimization techniques for the ascent trajectory design and guidance of a launch vehicle. An optimized mission design and the implementation of a minimum-propellant guidance scheme are key to increasing the rocket carrying capacity and cutting the costs of access to space. However, the complexity of the launch vehicle optimal control problem (OCP), due to the high sensitivity to the optimization parameters and the numerous nonlinear constraints, make the application of traditional optimization methods somewhat unappealing, as either significant computational costs or accurate initialization points are required. Instead, recent convex optimization algorithms theoretically guarantee convergence in polynomial time regardless of the initial point. The main challenge consists in converting the nonconvex ascent problem into an equivalent convex OCP. To this end, lossless and successive convexification methods are employed on the launch vehicle problem to set up a sequential convex optimization algorithm that converges to the solution of the original problem in a short time. Motivated by the computational efficiency and reliability of the devised optimization strategy, the thesis also investigates the suitability of the convex optimization approach for the computational guidance of a launch vehicle upper stage in a model predictive control (MPC) framework. Being MPC based on recursively solving onboard an OCP to determine the optimal control actions, the resulting guidance scheme is not only performance-oriented but intrinsically robust to model uncertainties and random disturbances thanks to the closed-loop architecture. The characteristics of real-world launch vehicles are taken into account by considering rocket configurations inspired to SpaceX's Falcon 9 and ESA's VEGA as case studies. Extensive numerical results prove the convergence properties and the efficiency of the approach, posing convex optimization as a promising tool for launch vehicle ascent trajectory design and guidance algorithms.

Acknowledgments

My research would have been impossible without the aid and support of many people, whom I would like to acknowledge and I will be forever grateful to.

First of all, I would like to thank my advisor, Dr. Alessandro Zavoli. He did not only supervise my work but he has been constantly guiding my research since day one of my PhD, transmitting to me his passion for research and teaching. His ingenious solutions made the difference on many occasions where I was stuck and much of the contributions of my research would have been impossible without him.

I would also like to show gratitude to Prof. Guido Colasurdo, who, together with Dr. Zavoli, was my master's thesis advisor and instilled in me the idea of applying for the PhD program. His endless experience proved right again, as the PhD career turned out to be one of the most fulfilling journeys of my life.

None of this could have happened without the support and funding of the Italian Space Agency (ASI), which awarded me with a scholarship for the entire duration of the PhD program that allowed me to attend international conferences and spend a research period abroad. In particular, I would like to acknowledge Dr. Enrico Cavallini, head of the Space Transportation Programs Office at ASI, who supervised my research with keen interest and enthusiasm, shared insightful thoughts, and always encouraged me to pursue novel research directions.

I would like to thank Prof. Zhenbo Wang, who hosted me as a visiting scholar at the Autonomous Systems Lab at the University of Tennessee in Knoxville. He approved my visit in a period when many professors would have hesitated due to the COVID-19 pandemic, which forced students and teachers to work remotely. In these difficult times, he welcomed me into his lab, punctually helped me to get through all the bureaucratic aspects, and, most importantly, closely supervised my research, contributing with excellent pieces of advice.

I would like to mention the incredible support of my mom and dad, who have been by my side during my whole PhD career (and way before that). The assurance of having a loving and comforting family behind helped me focus on my research, and the results I achieved are also due to them.

My doctoral experience was so enjoyable also thanks to the closeness of all my friends, older and newer. I would like to cite all of them individually, because every one of them contributed in his own way to make my life so fun and likable over the years, but it would require too many pages. I have to mention Lorenzo though, who started the PhD on the same day I did and, from that moment on, shared with me every moment of our PhD years. I have been very lucky to share this journey with a true friend like him and I cannot imagine how my PhD career would have been without him.

I would like to mention also my other PhD colleagues. Mimmo, being the oldest student advised by Dr. Zavoli, shared with me a lot of his expertise and I am grateful to him for all the notions and skills he transmitted me. Vincenzo, who just recently joined Dr. Zavoli's lab and watched over my workstation (which was essential to obtain the results of this thesis) while I was abroad. I have to thank Simone, who mentored me when I attended my first technical conference. That experience was a stimulus to attend many more conferences and present original papers every time, as conferences can be, as he says, "magical" places where to present the outcome of your work and share thoughts with other students and professionals.

I would like to show my gratitude to all the beautiful people that I met in Knoxville. Gage, my roommate, has been there for me since the moment I stepped foot off the plane until the day I flew back. I have rarely met a person as kind and *welcoming as Gage, we had a great time together and I hope I will have the chance to return his hospitality by welcoming him in Italy someday. I have to say thanks to Jacopo, who has been a great friend and, being the only Italian guy I met in Tennessee, together with his beautiful family, saved me from the nostalgia of my home country. I want to thank Alex for being a friend and also a great roommate the last few weeks that I stayed in Knoxville. Last, but not least, I am grateful for the unforgettable weekends around the US that I spent, during that time, with Enrico, "Maggico", and, of course, Lorenzo.*

This research was supported by the Agreement n. 2019-4–HH.0 CUP F86C170000-80005 "Technical Assistance on Launch Vehicles and Propulsion" between the Italian Space Agency and the Department of Mechanical and Aerospace Engineering of Sapienza University of Rome.

Contents

1 Introduction **1**
1.1 Background 2
1.2 Convex Optimization 3
1.3 Model Predictive Control 5
1.4 Objectives and Contributions 7
1.5 Thesis Summary 7

2 Methods for Optimal Control **9**
2.1 General Statement of an Optimal Control Problem 9
2.1.1 System Dynamics 10
2.1.2 Constraints 10
2.1.3 Objective Function 10
2.1.4 Phases 11
2.2 Overview of Numerical Methods 12
2.3 Direct Methods 13
2.3.1 Shooting Methods 14
2.3.2 Collocation Methods 14
2.4 Alternative Methods 18
2.4.1 Evolutionary Optimization 18
2.4.2 Machine Learning 19

3 Convex Optimization **21**
3.1 Preliminaries 21
3.1.1 Linear and Convex Functions 21
3.1.2 Affine and Convex Sets 21
3.1.3 Cones 22
3.1.4 Hyperplanes and Polyhedra 22
3.2 Convex Programming Problems 23
3.2.1 Linear Programming 23
3.2.2 Quadratic Programming 23
3.2.3 Quadratically-Constrained Quadratic Programming 23
3.2.4 Second-Order Cone Programming 24
3.3 Convexification Methods 24
3.3.1 Lossless Convexification Methods 24
3.3.2 Successive Convexification Methods 25
3.4 Virtual Controls and Buffer Zones 27
3.5 Trust Regions 28
3.6 Reference Solution Update 29
3.7 Convergence Criteria 30
3.8 Example: Low-Thrust Interplanetary Mission 30

3.8.1 System Dynamics 30
3.8.2 Optimal Control Problem 31
3.8.3 Convex Transcription 32
3.8.4 Numerical Results 34

4 Launch Vehicle Ascent Trajectory Optimization 39
4.1 VEGA Launch Vehicle 39
4.2 Phases 40
4.2.1 General Phase Sequence 40
4.2.2 VEGA's Phase Sequence 42
4.3 System Dynamics 43
4.3.1 Gravity 44
4.3.2 Atmosphere 44
4.3.3 Propulsion 46
4.3.4 Equations of Motion 47
4.4 Optimal Control Problem 47
4.4.1 Objective Function 48
4.4.2 Duration of the Phases 48
4.4.3 Initial Conditions 49
4.4.4 Target Orbit 49
4.4.5 Stage Separation 50
4.4.6 Liquid Rocket Engine Burn Time 50
4.4.7 Heat Flux 50
4.4.8 Maximum Dynamic Pressure 50
4.4.9 Splash-Down Constraint 51
4.5 Convex Formulation 52
4.5.1 Change of Variables 52
4.5.2 Constraint Relaxation 53
4.5.3 Successive Linearization 54
4.5.4 Trust Region on Time-Lengths 56
4.5.5 Virtual Controls and Buffers 56
4.5.6 Augmented Objective Function 57
4.6 Initialization Strategy 57
4.7 Continuation Strategy 59

5 Case Study: Ascent Toward a Low-Earth Polar Orbit 61
5.1 Problem Statement 61
5.2 Convergence Behavior 63
5.3 Sensitivity to the Initialization 66
5.3.1 Filtering Approach 66
5.3.2 Trust Region Approach 69
5.3.3 Comparison Between Filtering and Trust Regions 71
5.4 Analysis of the Discretization Grid 71
5.5 Parametric Analysis of the Splash-Down Constraint 72

6 Case Study: Ascent of a Two-Stage Vehicle to the ISS 77
6.1 Two-Stage Launch Vehicle Model 77
6.2 Phases 77
6.3 Problem Statement 78
6.4 Initialization 79
6.5 Convergence Behavior 80

7 Application to Upper Stage Guidance 85
7.1 Motivation for Upper Stage Guidance 85
7.2 Model Predictive Control Algorithms 86
7.3 Upper Stage Guidance Strategy 88
7.3.1 Single-Return Strategy 88
7.3.2 Multi-Return Strategy 89
7.4 Single-Return Optimal Control Problem 90
7.4.1 Phase Sequence 90
7.4.2 System Dynamics 90
7.4.3 Objective and Constraints 91
7.4.4 Convex Formulation 92
7.5 Multi-Return Optimal Control Problem 94
7.5.1 Phase Sequence 94
7.5.2 Objective and Constraints 95
7.5.3 Convex Formulation 96
7.6 Update of the OCP 96
7.7 Softening the Heat Flux Constraint 97
7.8 Perturbation Model 98
7.9 Numerical Results 98
7.9.1 Problem Data 99
7.9.2 Nominal Trajectories 99
7.9.3 Monte Carlo Campaigns 100

8 Conclusions 107
8.1 Future Work 108

A Linearization Matrices for the Low-Thrust Problem 110

B Proof of Lossless Relaxation 112

Bibliography 114

List of Figures

3.1 Reference frames used for the low-thrust transfer. 30
3.2 Optimal low-thrust trajectory. 36
3.3 Optimal thrust profile. 37
3.4 Spacecraft mass over time. 37
3.5 Evolution of the discretization grid due to the adaptive mesh refinement process. 37
3.6 Control constraint relaxation error. 38

4.1 Europe's launch vehicles: VEGA, Soyuz, Ariane 5-GS, and Ariane 5-ECA. 40
4.2 VEGA launch vehicle. 41
4.3 Phases of a typical launch vehicle ascent trajectory. 41
4.4 Phase sequence of a VEGA-like launch vehicle. 43
4.5 Atmospheric density. 45
4.6 Atmospheric temperature. 46
4.7 Angles used to parametrize the control direction. 58

5.1 Iteration sequence starting from the initial reference solution (dashed black line) and transitioning from red (iter = 1) to green (iter = 22) until convergence. 64
5.2 Three-dimensional visualization of the optimal trajectory with the simulation of third stage ballistic return colored in red. 64
5.3 Control laws of the optimal solution. 65
5.4 Payload mass as function of the splash-down latitude. 73
5.5 Stage 3 trajectories from liftoff to splash-down. 74
5.6 Altitude profiles from fairing jettisoning to $t_f^{(10)}$. 74
5.7 Heat flux profiles from fairing jettisoning to $t_f^{(10)}$. 75
5.8 Relaxation error during Phase 10 of the trajectory constrained to $\varphi_R = 55\,^\circ$. 75

6.1 Phases of the optimal control problem for the two-stage vehicle. . . . 78
6.2 Altitude profile across the iteration sequences for the ISS mission. . . 81
6.3 Visualization of the ISS ascent trajectory. 82
6.4 Optimal control laws in the ISS mission. 83
6.5 Relaxation error across Phase 5 of the ISS solution. 84

7.1 MPC flowchart. 87
7.2 Phases of the upper stage single-return guidance strategy. 89
7.3 Phases of the upper stage multi-return guidance strategy. 89

7.4 Phases of the OCP recursively solved by the single-return MPC algorithm. 90
7.5 Phases of the OCP recursively solved by the multi-return MPC algorithm. 94
7.6 Nominal and perturbed thrust profiles. 99
7.7 Splash-down footprints for Case H. 102
7.8 Altitude profiles for Case H. 103
7.9 Control elevation profiles for Case H. 104
7.10 Heat flux profiles for Case H. 105

List of Tables

3.1 Low-thrust trajectory design problem data. 34

5.1 VEGA-like rocket data. 62
5.2 Time-lengths of time-fixed arcs. 62
5.3 Discretization segments, order, and nodes in each phase. 63
5.4 Values used for the generation of the starting reference solution. . . . 63
5.5 Optimal solution of the convex approach compared with the EOS solution. 65
5.6 Optimal solution of the convex approach compared with the EOS solution with free coasting arcs. 66
5.7 Scattering range of the seeding parameters for the Monte Carlo analysis. 67
5.8 Results of the Monte Carlo analysis for several values of K and different families of filtering weights. 68
5.9 Results of the Monte Carlo analysis on the combined effect of the continuation strategy and the filtering approach. 69
5.10 Results of the Monte Carlo analysis using common trust region algorithms. 70
5.11 Discretization grids considered in the Monte Carlo analysis. 71
5.12 Results of the Monte Carlo analysis on different grids. 72

6.1 Two-Stage Rocket Data 77
6.2 Nominal time-lengths of time-fixed arcs 79
6.3 Discretization segments, order, and nodes in each phase. 79
6.4 Values used for the generation of the first guess trajectory of the two-stage rocket. 80
6.5 Final solution compared with the DE solution. 83

7.1 Initial number of discretization segments, order, and nodes in each phase of the single-return OCP. 96
7.2 Initial number of discretization segments, order, and nodes in each phase of the multi-return OCP. 97
7.3 Standard deviations of the random-direction Gaussian in-flight disturbance in terms of T/m percentage. 100
7.4 Results of the Monte Carlo campaigns (payload mass). 101
7.5 Results of the Monte Carlo campaigns (splash-down point). 101

Chapter 1

Introduction

The design of a safe and optimal launch vehicle ascent trajectory is a problem of great interest in the aerospace industry, as present launch systems, which rely on chemical propulsion, can inject into orbit only a small fraction of their initial mass. Hence, the trajectory optimization is a crucial step in order to maximize the system carrying capacity and reduce the cost of access to space. In this respect, a systematic and computationally efficient optimization procedure is an asset during the preliminary design phases of a launch system, as a means to determine the maximum payload capability of various configuration concepts, in the advanced pre-flight analysis, to assess the feasibility of specific mission scenarios, and, eventually, in the definition of real-time guidance algorithms, where computational speed and reliability are essential.

The design of a rocket ascent trajectory usually reduces to solving an optimal control problem (OCP). The goal of the optimization is to find the control law (i.e., the thrust direction time history) and the optimal values of other mission parameters (e.g., the ignition and cut-off time instants of the stages) that maximize a performance index (e.g., the carrying capacity to a desired orbit). However, the resulting OCP is typically quite difficult to solve since it features nonlinear dynamics and other numerous mission requirements, which also are often transcribed as nonlinear constraints, and it is greatly sensitive to the optimization variables. Due to this inherent complexity, traditional optimization methods may often appear computationally expensive or too sensitive to the starting guess provided by the user. In this thesis, convex optimization methods are investigated as a means to solve a realistic and practical instance of the launch vehicle ascent problem in a computationally efficient and systematic way.

Launch vehicle dynamics are subject to significant uncertainties, due to hard-to-predict aerodynamic coefficients, scattering of propulsion system performance, and sudden variations of the local environment, to name a few. Therefore, robustness of the implemented guidance algorithms is an absolute necessity for the success of a launch vehicle ascent mission. Traditionally, launch vehicles rely on open-loop guidance algorithms in the atmospheric portion of the flight and on analytical closed-loop guidance schemes when operating in vacuum. However, such approaches are intrinsically suboptimal, as open-loop schemes do not account for actual in-flight conditions (thus requiring extensive validation and verification tasks to ensure robustness before each mission) and analytical guidance laws are formulated on the basis of assumptions that only approximate the complexity of the ascent problem (thus leading to suboptimal control policies). So, in this thesis, convex optimization is embedded into a model predictive control (MPC) framework to set up an advanced control synthesis method that recursively computes the optimal control and the associated trajectory by solving in real time the ascent OCP updated with navigation measurements. The MPC framework is particularly appealing since it maximizes the system performance thanks to the optimization-based update of the nominal trajectory and inherently provides robustness to model uncertainties and external random disturbances through the closed-loop architecture.

1.1 Background

The earliest variant of the ascent problem is due to Goddard [1], who posed the problem of maximizing the final altitude of a vertically-ascending rocket subject to gravity and atmospheric drag. Many authors have treated the problem, either proposing analytical solutions under simplifying assumptions [2, 3] or, once superior computing hardware was available, numerical approaches [4, 5].

The launch vehicle problem gained a renewed interest at the beginning of the space race, when the need to maximize the payload capacity of the new launch system concepts became crucial. The steepest descent methods represented the most straightforward approach; however, the computational limits of existing hardware restricted the practical employment of such methods only to simplified instances of the ascent problem, which required continuous corrections based on post-flight data to properly approximate the real problem [6].

In the 1960s, advancements in optimal control theory produced methods capable of providing extremely accurate solutions with a minor computational effort, namely, indirect methods. One of the first applications of an indirect approach to the ascent problem is due to Jurovics [7]. Spurlock and Teren in Refs. [8, 9] essentially outline the indirect procedure that underlies the DUKSUP computer program [6], which has been consistently employed for three decades for trajectory design of several families of NASA launch systems, such as Atlas, Titan, and the Space Shuttle, in addition to other miscellaneous aerospace optimal control problems. More recent work on the application of indirect methods to the design of ascent trajectories can be found in Refs. [10–12].

Besides the pre-flight trajectory design task, indirect approaches were also proposed in the definition of optimal guidance algorithms. Indeed, an open-loop scheme is adopted while flying in the atmosphere and a closed-loop algorithm is implemented only in the exoatmospheric fraction of the ascent mission [13]. This heterogeneity is due to the intrinsic difficulty that underlies the endoatmospheric problem, which precludes the derivation of analytical solutions and significantly hinders the convergence of numerical algorithms. Indeed, rocket vacuum optimization software is available since the earliest missions, as in the cases of the Saturn rocket Iterative Guidance Mode (IGM) algorithm [14] and of the Power Explicit Guidance (PEG) of the Space Shuttle [15]. Efforts to extend closed-loop vacuum guidance schemes to the atmospheric flight, such as linear tangent steering with atmospheric terms in the equations of motion [16], do not yield significant performance improvements, as the presence of the atmosphere makes the underlying guidance strategy not optimal (e.g., the linear tangent steering is optimal only for a rocket in vacuum over a flat Earth). Instead, optimized open-loop atmospheric trajectories, coupled with early release of closed-loop vacuum guidance (that is, while still in the atmosphere but after the high dynamic pressure region), yield good performance results and still represent the most popular strategy [13].

One of the first efforts to design a closed-loop guidance scheme for the whole ascent is due to Leung and Calise [17], who proposed a hybrid approach to speed up the numerical solution of the atmospheric problem by exploiting the analytical solution to the necessary conditions of the vacuum problem. Later work proved the real-time applicability of the method also in scenarios of increasing complexity [18–20]. It is worth mentioning also the work by Lu et al. [21, 22], who outline an indirect approach and solve the resulting boundary value problem (BVP) by means of classical finite differences, instead of collocation, to further reduce the computational time. However, indirect methods require the analytical derivation of the first-order necessary conditions, which can be a cumbersome process when solving an actual real-world engineering problem. Furthermore, their effectiveness is strongly related to the quality of the first guess (both of the solution and of the adjoint variables), which is usually quite difficult to provide for the ascent problem. In addition, if path constraints are included in the formulation, an *a priori* knowledge of the structure of the constrained arcs is necessary, which, in general, is hard to guess.

The difficulties related to the solution of complicated BVPs and the concurrent progress in digital computer technology drove the attention of the aerospace community toward direct optimization methods [23], which are based on a straightforward transcription of the optimal control problem into a general nonlinear programming (NLP) problem and are quite easy to set up. Over the years, a broad spectrum of software tools based on direct transcription have been developed for solving the ascent problem and a wide variety of other aerospace problems. One of the oldest is POST [24], firstly released in the 1970s and still used today. It is noteworthy to mention also OTIS [25], which is based on a direct optimization procedure detailed in Ref. [26], and other general-purpose direct programs commonly employed for ascent trajectory optimization, such as SOCS [27] and ASTOS [28]. Refs. [29, 30] illustrate successful applications of direct procedures to the optimization of ascent trajectories, which account for realistic mission requirements, such as the ballistic return of burned-out stages, the maximum heat flux path constraint, and the deployment of multiple satellites.

However, direct methods do not enforce first-order necessary conditions for optimality on the continuous problem directly; thus, the optimal discrete solution may differ from the continuous one, and typically exhibits control chattering [27]. Furthermore, even though they are characterized by a larger radius of convergence than indirect ones, the attained solution depends on the accuracy of the initial guess. Therefore, usually, when highly-sensitive problems are dealt with and no accurate initialization is readily available, only convergence to a suboptimal solution (i.e., featuring a merit index significantly worse than the global optimum) can be reasonably expected. On top of that, the solution of a NLP problem requires a relatively large computational effort, making general direct methods unsuitable for time-critical applications.

1.2 Convex Optimization

In the last years, convex optimization techniques became increasingly popular for solving optimal control problems in the aerospace community [31]. Convex optimization is a special class of mathematical programming that allows for the use of polynomial-time algorithms that provide a theoretically guaranteed optimal solution with a limited computational effort. However, most aerospace problems are not inherently convex, so these algorithms cannot be directly employed. Therefore, several *convexification* techniques have been developed to convert a nonconvex problem into a convex one. These methods can be grouped into *lossless* and *successive* convexification techniques. The former consist in exploiting either a convenient change of variables or a suitable constraint relaxation to reformulate the problem as convex. For example, Açıkmeşe and Blackmore [32] proved that problems with a certain class of nonconvex control constraints, such as rocket powered descent [33], can be equivalently posed as relaxed convex problems. Since lossless convexification methods introduce no approximation at all, they should always be used when possible, as better performing than other convexification strategies [34].

When no lossless convexification can be performed, successive convexification must be employed. Indeed, successive convexification offers a way to tackle the nonconvexities that cannot be handled by lossless convexification. A common successive convexification approach consists in linearizing the nonconvex expressions around a reference solution, which is recursively updated until convergence. Differently from lossless convexification, the successive linearization generates a sequence of approximated problems. The theoretical proof that successive convexification leads to a (locally) optimal solution of the originally intended problem is available only under appropriate assumptions [35–37]. Nevertheless, current research offers wide numerical evidence of the effectiveness of successive convexification over a broad spectrum of applications, including spacecraft rendezvous [38], proximity operations [39], formation flying [40], low-thrust transfers [41, 42], UAV path planning [43, 44], rocket powered landing [45–48], atmospheric entry [49–52], and asteroid landing [53].

Convex optimization has been proposed also for solving the launch vehicle ascent trajectory design problem. However, successful applications are limited to simplified scenarios, where a flat Earth is assumed [54], atmospheric forces are neglected [55, 56], or only the upper stage trajectory is optimized [35, 57]. Instead, a much more complex instance of the ascent problem, featuring a realistic dynamical model and practical mission requirements, is necessary to accurately predict the system performance and assess its criticalities.

For instance, when designing the ascent trajectory of an expendable multi-stage vehicle, the ballistic reentry of the stages after separation from the rocket must be planned. While for many launch systems the return of the spent stages does not represent a concern, as the burned-out stages fall in open water or in uninhabited regions not too distant from the launch site, posing few safety concerns and not requiring to enforce additional constraints in the optimization process, some rockets, among which VEGA [58], require to carefully predict and actively constrain the splash-down points of the spent stages to safe locations. This requires the inclusion of a complete simulation of the ballistic return of the spent stages in the OCP formulation, increasing the complexity of the problem, since return phases feature nonlinear dynamics, making the OCP further nonconvex. Also, capturing the dynamics of a high-velocity reentry in Earth's atmosphere requires the use of a sufficiently dense discretization mesh, which significantly increases the problem dimension and the overall computational burden of the solution procedure.

The aforementioned limits of traditional optimization methods, such as the sensitivity to the initial guess or the large computational effort of the solution process, motivate the study of convex optimization strategies for the design of launch vehicle ascent trajectories. Indeed, the inclusion of practical requirements and an accurate description of the system dynamics are essential to guarantee the validity of the attained solutions and evaluate the performance and criticalities of a launch system, but imply a significant computational complexity due to the nonconvexity of the related constraints. Therefore, this thesis investigates mindful convexification techniques combined with state-of-the-art discretization methods to solve a realistic instance of the ascent problem and yet set up a computationally efficient and systematic optimization procedure to design the ascent trajectory of a launch vehicle.

1.3 Model Predictive Control

Model predictive control is an advanced method to control dynamical systems subject to uncertain operating conditions and random disturbances. MPC is one of the few control synthesis strategies that can optimize the system performance while systematically accommodating mission constraints [59]. Specifically, MPC consists in solving repeatedly an optimal control problem, with initial condition updated using onboard system measurements, and implementing the computed optimal control law in the time frames between the optimization procedures. The closed-loop architecture inherently provides robustness to model uncertainties and in-flight disturbances, as the continuous update of the optimal path compensates for deviations from the nominal one, while the solution of an OCP guarantees the optimality of the performance.

MPC was originally developed in the late 1970s for the control of petroleum refineries [60, 61], where constraint satisfaction is an extremely important requirement since operating points are often located on the boundaries of the admissible state and control sets due to economic considerations [62]. MPC quickly gained popularity in a wide variety of industries including chemical [63], pulp and paper [64], and automotive [65]. Among the aerospace community, the interest toward MPC has grown constantly over the years [66], featuring successful employment over diverse problems, including rocket landing [67–69], spacecraft landing [70, 71], rendezvous [72, 73], and formation flying [40, 74], to name a few. Indeed, aerospace systems are among the most challenging applications for controller design due to tight mission requirements, limited onboard resources, often unpredictable environmental conditions, and operating points at the limits of the attainable performance. Thus, MPC, being one of the few control techniques that can systematically address such a wide spectrum of demands, represents an uniquely promising tool to control such systems.

The earliest MPC controllers were limited to relatively simple problems, mostly concerning linear systems, but the advances in technology and control theory enabled the application of MPC to problems of increasing complexity. In this respect, the interest toward nonlinear MPC has grown tremendously in the last decades [75], motivated by the desire to extend the appealing properties of MPC controllers to nonlinear systems. Still, the system nonlinearities pose serious challenges to the design of fast and robust nonlinear controllers, but expectations are high and the ongoing research continuously provides both significant theoretical results [76] and numerical evidence of numerous successful applications [77].

The effectiveness of an MPC framework depends on two key aspects: the *update frequency* and the *recursive feasibility*. The former dictates how often an updated OCP is solved. The higher the frequency, the more the algorithm is capable of rejecting the disturbance associated with the mismatch between the predicted optimal states and the measured ones, thus becoming more robust to model uncertainties and dispersions. Its value is bounded by the limited onboard computational capacity. Instead, the recursive feasibility concerns the reliability of the optimization algorithm, which must be able to solve the OCP over the entire mission duration. These two requirements generally make traditional indirect and direct optimization methods unsuitable for real-time applications. Indeed, although indirect methods provide an extremely accurate solution with a limited computational cost, they exhibit high sensitivity to the initialization and come with few convergence guarantees. Instead, direct methods, which discretize the OCP over a mesh to obtain a general nonlinear programming problem, are usually too expensive to be used onboard with acceptable update frequencies. On the other hand, convex optimization techniques are natural candidates for real-time applications, as convex OCPs can be solved by means of highly efficient interior point algorithms, which converge to the optimal solution in polynomial time regardless of the initialization point [78].

This thesis investigates the use of MPC for the computational guidance of the upper stage of a launch vehicle. In particular, VEGA's third stage is taken as case study. As already mentioned, some rockets require to actively constrain the splash-down point of the upper stages to safe locations. Moreover, uncertainties on the propulsive performance (especially on the cut-off time of a solid rocket motor) do not allow to easily pinpoint a splash-down location but rather define a finite-dimension footprint of possible impact points. Thus, a further system requirement consists in bounding the extent of this region. To this aim, robust optimization and robust MPC can be used to endow the guidance and control with some robustness guarantees. However, these methods are usually overly conservative since they are primarily based on min-max OCP formulations [79] or on constraint tightening [80]. On the other hand, more recent tube-based MPC [81] or stochastic optimal control methods, such as chance-constrained optimization [82–84] or covariance control [85–87], could also be considered as viable options, but their application to the case of non-additive disturbances, such as those that arise from dynamics under uncertain time-lengths, can be quite challenging.

For all these reasons, VEGA's third stage currently relies on a (pre-scheduled) neutral axis maneuver to robustly minimize the size of the splash-down footprint and make the return as insensitive as possible to the solid rocket motor (SRM) performance dispersions [88]. This maneuver is based on the *null miss condition* developed for ballistic missiles [89]: over the last few seconds of operation, the stage is constrained to hold on to an attitude such that the splash-down point is retained regardless of any additional velocity increments. The maneuver reduces the carrying capacity of the launch vehicle since part of the propulsive energy is spent in a non-optimal direction, but it robustly guarantees that the actual splash-down location is sufficiently close to the predicted one, despite uncertainties on the cut-off time of the SRM. However, the neutral axis maneuver can be quite difficult to design with traditional methods and usually extensive trajectory validation and verification tasks are required before each launch. As an alternative to the neutral axis maneuver, the use of retro-rockets has been proposed to control the third stage after the separation [90]. However, such a solution would need the integration of additional hardware and mass into a consolidated architecture, hence it is scarcely appealing.

This thesis investigates a novel guidance algorithm, based on embedding convex optimization into a MPC framework. The convex OCP solved onboard explicitly incorporates and optimizes the neutral axis maneuver. In this way, the model predictive controller updates in real time the direction of the neutral axis attitude, which must be maintained in the last seconds of operation in an open-loop fashion, enhancing the system performance and reducing the size of the splash-down footprint. This online computation of the neutral axis direction overcomes traditional issues in the design of the neutral axis maneuver, which is often based on cumbersome and time-consuming pre-flight procedures, and allows the integration of such a maneuver in a modern computational guidance scheme, ensuring the system robustness to extra burn seconds of the SRM.

1.4 Objectives and Contributions

This thesis features several technical contributions to the state of the art. First, the seminal lossless convexification strategy originally proposed for powered descent problems is extended and modified to suit the ascent problem. Leveraging optimal control theory, rigorous proof of the validity and lossless property of the said convexification strategy is provided under mild assumptions for this problem. Second, the algorithmic robustness provided to a successive convexification framework by a novel recursive update of the reference solution, named *filtering* is investigated. This strategy successfully filters out oscillations in the search space and other common undesired phenomena due to the successive linearization, such as artificial unboundedness, thanks to reduced weight assigned to the newly found solutions. Moreover, conversely to other expedients commonly used to prevent artificial unboundedness, such as trust regions, the implementation of the filtering technique does not alter the OCP, as it does not imply the inclusion of additional constraints or penalty terms in the formulation. This approach is of general validity and could be successfully applied to other problems.

The thesis also investigates strategies to include safety-related mission requirements, often neglected in preliminary analyses, such as the prediction and the active constraining of the splash-down points of the burned-out stages, which significantly increase the computational complexity of the problem. Nevertheless, combinations of original and consolidated convexification methods are devised and compared throughout the dissertation and, in the end, an efficient and reliable strategy for the optimization of a realistic launch vehicle ascent trajectory is retrieved.

Motivated by the algorithmic robustness and the short computational times of the convex approach, the thesis also aims at investigating the real-time implementation of convex optimization algorithms in a closed-loop guidance architecture such as model predictive control. Indeed, minor modifications to the ascent trajectory design algorithm are devised to cast the guidance problem of a launch vehicle upper stage as a convex problem with satisfactory convergence properties. The application is of great interest, as upper stages generally implement simple vacuum rocket guidance schemes that cannot account for complex mission requirements. Instead, the convexification process allows to consider realistic scenarios in the optimization and thus compute online an optimal guidance law that meets all mission requirements.

Eventually, a significant contribution of the thesis is the definition of a novel guidance strategy that robustly ensures the splash-down constraint of the spent stage even in presence of uncertainties on the engine cut-off time, which is not negligible for most solid rocket motors and is usually dealt with a neutral axis maneuver. The MPC guidance poses itself as a systematic and innovative method to update online the optimal ascent trajectory of the rocket on the basis of the encountered conditions (hence, enhancing the system performance and providing robustness to in-flight disturbances) and ensure the safety of the spent stage return without the extensive validation and verification tasks that traditional design of the neutral axis maneuver require before each mission.

1.5 Thesis Summary

The thesis is organized as follows.

Chapter 2 formulates a general optimal control problem and introduces some essential definitions and notation convention that are adopted through all the dissertation. The chapter surveys the traditional methods to solve an optimal control problem, with particular focus on the direct methods, which are the ones generally used to cast a convex optimal control problem into a discrete convex program. Alternative methods, based on evolutionary optimization and machine learning, for solving an OCP are also discussed and compared with the traditional deterministic approaches.

In Chapter 3, after introducing some preliminary concepts of convex analysis, several classes of convex optimization problems are defined. Then, convexification methods to convert general problems into convex ones are described. The chapter also discusses undesired phenomena due to the convexification of a nonlinear problem and details state-of-the-art safeguarding modifications that can be implemented to prevent such phenomena. A low-thrust interplanetary trajectory design problem is taken as an example to illustrate the use of convexification techniques on a practical problem.

Chapter 4 introduces the launch vehicle ascent trajectory optimization problem. The chapter illustrates the phases, the dynamical model, and requirements that characterize the ascent of a general launch vehicle, focusing on specific aspects of the VEGA rocket. The ascent trajectory design problem is formulated as an optimal control problem; thus, the chapter outlines the objective function, the optimization variables, and the mission constraints that must be considered. Since the resulting problem is nonconvex, a convexification strategy to convert the problem into a convex one is presented along with some practical solution strategies, including a systematic way to design a suitable initial guess and a numerical continuation procedure to reduce the sensitivity of the algorithm to inaccurate initializations.

In Chapter 5, an ascent of a VEGA-like vehicle toward a low-Earth orbit is taken as case study to investigate the effectiveness of the convex approach. The convergence of the successive convexification algorithm is discussed on the basis of a typical sequence of iterations. Then, focus is posed on the sensitivity of the algorithm to the initialization, which is discussed on the basis of the results of several Monte Carlo campaigns. The accuracy of the discretization is also investigated by means of a Monte Carlo analysis that compares grids of different order. Finally, a sensitivity analysis of the launch vehicle carrying capacity to different splash-down locations is presented.

Chapter 6 reformulates the optimal control problem for a two-stage rocket (inspired to SpaceX's Falcon 9 rocket) to show the generality of the convex approach and its effectiveness in diverse scenarios. The modifications to the phase sequence, the mission constraints, and the initialization strategy compared to the VEGA-like case are thoroughly discussed. Numerical results for an ascent from Cape Canaveral to the International Space Station's (ISS) orbit are presented and compared with an unrelated optimization program.

In Chapter 7, the convex optimization approach is embedded into a model predictive control framework to define a closed-loop guidance algorithm for the upper stage of a launch vehicle. Two different guidance strategies, which differ by the optimal control problem to recursively solve onboard, are proposed. In the first one, the OCP comes directly from the complete ascent problem, while the second one includes an additional return phase to robustly ensure the splash-down constraint in the case of uncertain cut-off time of the stage. The results of a series of Monte Carlo campaigns are presented to assess the performance and robustness of the MPC framework in presence of in-flight disturbances, off-nominal conditions, and uncertain motor performance.

Chapter 8 summarizes the key contributions of the thesis and suggests potential future research directions.

Chapter 2

Methods for Optimal Control

Optimization problems date back at least to the Greeks. One of the oldest was *Dido's problem*: the problem of finding the figure bounded by a curve which has the maximum area for a given perimeter. Such figure is a circle, and the Greeks already knew it. However, it took until the 19th century to prove it rigorously. In 1696, Johann Bernoulli posed his famous *brachistochrone* problem [91]: given two points A and B in a vertical plane, find the curve traced out by a point acted on only by gravity, which starts at A and reaches B in the shortest time. This is considered to be the first optimal control problem since it explicitly deals with controlling the path or behavior of a dynamical system.

The brachistochrone problem served as a stimulus to formulate and solve a number of other more general problems and thereby to establish optimal control as a mathematical field [92]. Indeed, since then, many significant contributions have been made in this topic. In 1766 Euler, after working several years with Lagrange, published the *Elementa Calculi Variationum* [93], from which the *calculus of variations* got its name. In the 1950's, the appearance of practical, high-speed digital computers radically changed the field by enabling the use of efficient numerical methods. The aim of this chapter is to illustrate the different present-day methods that can be used to solve an optimal control problem.

2.1 General Statement of an Optimal Control Problem

An optimal control problem consists in finding a control signal (e.g., rocket thrust direction time-law) and other design parameters (e.g., the time-length of burn and coasting arcs of a spacecraft) that drive a dynamical system from an initial state to a final one while meeting all system requirements, expressed as a series of constraints, and maximizing its performance (in terms of a convenient merit index).

The following notation conventions will be used through all the thesis to provide, as much as possible, a consistent and clear formulation of the equations. Lowercase Roman letters (e.g., a) are used to denote scalar quantities and functions, while lowercase bold letters (e.g., $\boldsymbol{v}$) denote vector variables and vector functions. Unless stated otherwise, the Euclidean norm of a vector $\boldsymbol{v}$ is denoted with the same symbol but non-bold, thus $v = \|\boldsymbol{v}\|$. Lastly, uppercase Roman letters (e.g., M) denote matrices.

2.1.1 System Dynamics

The state of the dynamical system at time t is defined by a vector $\boldsymbol{x}(t)$, which includes all the variables necessary to characterize, in a complete and unequivocal way, the system under examination (e.g., position, velocity, and mass of a spacecraft). The time-evolution of the state vector defines the trajectory of the system over a considered integration interval $[t_0, t_f]$ and is obtained as the solution of a set of first-order ordinary differential equations (ODEs),

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{p}, t) \tag{2.1}$$

The vector function $\boldsymbol{f}$ on the right-hand side of Eq. (2.1) is usually referred to as the *system dynamics*. The vector $\boldsymbol{u}(t)$ includes all the control variables and $\boldsymbol{p}$ is the vector of time-independent optimization parameters. The thrust vector generated by a rocket engine or the aileron deflection of an aircraft are examples of control variables, while the ignition time of a landing rocket or the duration of a coasting arc during an orbit transfer are examples of optimization parameters.

2.1.2 Constraints

The solution of an optimal control problem must usually satisfy several system requirements, which can be formulated as equality or inequality constraints. Constraints that are functions of only the state and control variables at initial and final time are referred to as a *boundary constraint* and can be collected in a vector of homogeneous algebraic equations

$$\boldsymbol{\chi}(\boldsymbol{x}(t_0), \boldsymbol{x}(t_f), \boldsymbol{u}(t_0), \boldsymbol{u}(t_f), \boldsymbol{p}, t_0, t_f) = \boldsymbol{0} \tag{2.2}$$

Instead, general constraints that must be enforced through all the considered time interval are called *path constraints*

$$\boldsymbol{\psi}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{p}, t) \leq \boldsymbol{0} \tag{2.3}$$

Note that Eq. (2.3) is posed as an inequality constraint, but path constraints can be formulated also as equality constraints. Indeed, Eq. (2.3) is a general formulation that includes equality constraints as special cases, as an equality $\psi(x) = 0$ can be replaced by two inequalities, $\psi(x) \leq 0$ and $-\psi(x) \geq 0$.

2.1.3 Objective Function

The goal of the optimal control problem is to determine the continuous-time functions $\boldsymbol{x}(t)$ and $\boldsymbol{u}(t)$ and the parameters $\boldsymbol{p}$ that extremize an objective function J, representative of the system performance. This implies either maximizing a merit index (e.g., the payload mass of a launch vehicle) or minimizing a cost function (e.g., the time required to reach a target spacecraft).

Without loss of generality, in this thesis minimization problems are always considered, since any maximization problem can be cast as an equivalent minimization problem by changing the sign of the objective function.

The general expression of the objective function of an OCP is

$$J = \varphi(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f, \boldsymbol{p}) + \int_{t_0}^{t_f} \Phi(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{p}, t) dt \tag{2.4}$$

The first contribution, φ, is a function only of the boundary values of state and time. Instead, the second one, Φ, depends on the time-evolution of the state and control variables. If the function φ is null, then the problem is known as the *problem of Lagrange*. Instead, if there are no integral terms Φ, the OCP is referred to as *problem of Mayer*. The general case that features both Mayer and Lagrange terms is called *problem of Bolza*.

It is worth noting that the same problem can be posed in any of the three forms. For example, if the problem is in the Lagrange form,

$$J = \int_{t_0}^{t_f} \Phi(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{p}, t) dt \tag{2.5}$$

then, by including the additional state variable x_{n+1} with its corresponding state equation,

$$\frac{dx_{n+1}}{dt} = \Phi(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{p}, t) \tag{2.6}$$

and the initial condition $x_{n+1}(t_0) = 0$, it is possible to replace the original objective function of Eq. (2.5) with

$$J = \varphi(\boldsymbol{x}(t_f)) = x_{n+1}(t_f) \tag{2.7}$$

Note that, while Mayer's form is usually preferred for notational simplicity, it increases the size of the OCP.

2.1.4 Phases

Often, it may be necessary (or just convenient) to split the time horizon of an OCP into a collection of *phases* (or *arcs*). A phase is a portion of a trajectory in which the system dynamics, Eq. (2.1), remain unchanged and the state and control variables are not discontinuous. However, this does not imply that the ODEs *must* change across phase boundaries, as the need to split the OCP may arise from other necessities. For instance, many problems feature constraints at some interior point $t_1 \in (t_0, t_f)$ and the only way to enforce a constraint at an internal point t_1 is dividing the OCP in correspondence of said point. Indeed, introducing a separation may be useful in regions where the system state or the controls change rapidly or even instantaneously, as quick variations may be difficult to incorporate in a global time-continuous solution.

Phases usually occur sequentially in time and are delimited by the *internal boundaries*. Specifically, internal boundaries are the extremal points of the phases of an OCP, while *external boundaries* are the extremal points of the whole problem. A superscript (i) indicates that a variable belongs to phase i. Thus, the boundaries of each phase are denoted as $t_0^{(i)}$ and $t_f^{(i)}$. When the superscript is omitted, then t_0 and t_f denote the external boundaries of the OCP.

The initial and final boundaries of all phases can be grouped in the following sets

$$\mathcal{T}_0 = \left\{ t_0^{(i)} \,\middle|\, i = 1, \ldots, N \right\} \tag{2.8}$$

$$\mathcal{T}_f = \left\{ t_f^{(i)} \,\middle|\, i = 1, \ldots, N \right\} \tag{2.9}$$

Likewise, the state and controls at the phase boundaries can be grouped in

$$\mathcal{X}_0 = \left\{ \boldsymbol{x}(t_0^{(i)}) \,\middle|\, i = 1, \ldots, N \right\} \tag{2.10}$$

$$\mathcal{X}_f = \left\{ \boldsymbol{x}(t_f^{(i)}) \,\middle|\, i = 1, \dots, N \right\} \tag{2.11}$$

$$\mathcal{U}_0 = \left\{ \boldsymbol{u}(t_0^{(i)}) \,\middle|\, i = 1, \dots, N \right\} \tag{2.12}$$

$$\mathcal{U}_f = \left\{ \boldsymbol{u}(t_f^{(i)}) \,\middle|\, i = 1, \dots, N \right\} \tag{2.13}$$

The generalization of the boundary constraint of Eq. (2.2) to a problem of N phases is

$$\boldsymbol{\chi}(\mathcal{X}_0, \mathcal{U}_0, \mathcal{T}_0, \mathcal{X}_f, \mathcal{U}_f, \mathcal{T}_f, \boldsymbol{p}) = \boldsymbol{0} \tag{2.14}$$

If a constraint includes variables from different phases is said to be a *linkage* constraint.

Finally, the objective function of a multi-phase problem is

$$J = \varphi(\mathcal{X}_0, \mathcal{T}_0, \mathcal{X}_f, \mathcal{T}_f, \boldsymbol{p}) + \sum_{i=1}^{N} \int_{t_0^{(i)}}^{t_f^{(i)}} \Phi^{(i)}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{p}, t) dt \tag{2.15}$$

2.2 Overview of Numerical Methods

In general, it is not always possible to solve analytically an OCP, thus numerical methods must be employed. Optimal control problems are infinite-dimensional problems due to the fact that the state and control signal are continuous-time functions. However, numerical methods for solving OCPs require a *finite* set of variables and constraints. Therefore, the optimal control problem must be converted somehow into a finite-dimensional problem. This is carried out by means of a so-called *transcription method.*

Existing transcription methods can be grouped into two categories: *indirect methods* and *direct methods.* Indirect methods are based on calculus of variations for deriving the first-order optimality conditions. These necessary conditions form a Hamiltonian boundary value problem (HBVP) that can be solved with consolidated numerical schemes. From a practical standpoint, the differential constraints are enforced through the introduction of continuous-time Lagrange multipliers, called *adjoint* or *costate* variables, in the optimization problem. Indirect methods are characterized by a high accuracy in the solution and require a small computational cost. Furthermore, they guarantee the *local* optimality of the attained solution. However, they also have several disadvantages. First, in practical problems, formulating the HBVP is usually nontrivial or even impossible. Secondly, the radius of convergence is small, hence a very accurate initialization point is required. This task can be quite complicated since the costate variables do not have a clear physical meaning and guessing suitable initial values is often challenging. Lastly, the handling of path constraints in the optimization process is generally troublesome, as an a priori knowledge of the constrained and unconstrained arc sequence is required.

On the other hand, direct methods discretize the optimal control problem over a *mesh* (or *grid*) and turn it into a general nonlinear programming (NLP) problem. Direct methods do not require the derivation of optimality conditions and are characterized by a larger radius of convergence than the indirect methods, so the initialization process is much easier. Also, since the system dynamics are converted into a set of algebraic constraints through the discretization process, there is no need to introduce (and initialize) the costate variables. Finally, the discretization of path constraints is such that the solution process autonomously determines the intervals where the constraints are active or not. However, the discretization generally defines a great number of variables, resulting into a large, albeit sparse, NLP problem, which is much more expensive to solve than a HBVP. Another downside of direct methods concerns the optimality of the attained solution, which is generally unknown or difficult to assess.

Recently, convex programming emerged as a popular means to solve optimal control problems thanks to the theoretical guarantees on the solution optimality and the availability of highly efficient numerical algorithms [31]. Even though most real-world problems cannot be readily solved as convex optimization problems, several ideas have been proposed to convert a given nonconvex problem into a convex one, through a process referred to as *convexification.* Once an optimal control problem is posed in a convex form, it can be discretized by means of a direct collocation method and solved as a convex program. In the following section, direct methods are described, while a survey of state-of-the-art convexification techniques is provided in Chapter 3.

2.3 Direct Methods

A direct method basically consists of discretizing the continuous-time functions, such as state and control, to obtain a finite set of NLP variables and constraints. The time discretization is carried out by dividing the time domain $[t_0^{(i)}, t_f^{(i)}]$ of each phase i into $M^{(i)}$ segments. So, omitting from now on the superscript (i), one obtains $M+1$ points

$$t_0 \leq t_1 \leq \cdots \leq t_{M+1} \leq t_f \tag{2.16}$$

Each point is referred to as *node* of the mesh.

Note that, in order to discretize time, the initial and final instants must be known. However, in general, t_0 and t_f may be optimization variables (hence, unknown). To account for free-time phases in the optimization procedure, a common strategy [94] consists in replacing time t with a new independent variable τ, defined for each phase over a fixed domain $[0, 1]$. Being the domain unitary, the time dilation σ between t and τ corresponds to the arc time-length

$$\sigma = \frac{dt}{d\tau} = t_f - t_0 \tag{2.17}$$

Then, σ is included as an additional optimization parameter (i.e., in the $\boldsymbol{p}$ vector) for each phase and the system dynamics are expressed in terms of τ

$$\frac{d\boldsymbol{x}}{d\tau} = \sigma \frac{d\boldsymbol{x}}{dt} = \sigma \boldsymbol{f}\left(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p}, \tau\right) \tag{2.18}$$

At this point, a first classification of direct methods comes up, according to which continuous-time variables should become NLP variables. Direct methods that discretize only the control signal $\boldsymbol{u}(t)$, as $\boldsymbol{u}_j = \boldsymbol{u}(t_j)$ for $j = 1, \ldots, M+1$, are said to perform a *semi-discrete parameterization.* In such an approach, the differential constraints are accounted for in the optimization through a forward propagation of the system dynamics; thus, this strategy is called *shooting.* Instead, methods that discretize both state $\boldsymbol{x}(t)$ and control $\boldsymbol{u}(t)$, as $\boldsymbol{x}_j = \boldsymbol{x}(t_j)$ and $\boldsymbol{u}_j = \boldsymbol{u}(t_j)$ for $j = 1, \ldots, M+1$, perform a *fully-discrete parameterization.* These methods are based on *collocation* to replace the differential equations with a set of algebraic constraints, called *defect* constraints.

2.3.1 Shooting Methods

When only the control is discretized, the state equations are solved via numerical integration of the corresponding boundary value problem (BVP). From a practical standpoint, starting from a first guess of the initial state, the controls, and the parameters, one propagates (i.e., “shoots”) the differential equations from the initial time t_0 to the final time t_f and evaluates the error on the boundary conditions. Shooting methods are widely used because the resulting finite-dimensional problem has a small number of variables, thus it can be solved with a limited computational effort. However, a small change in the initial conditions can produce a very large change in the final conditions.

To reduce the sensitivity of the shooting methods, a *multiple shooting* method can be adopted. This approach consists in breaking the time domain into shorter steps. State vectors at the internal points are additional variables and continuity constraints are imposed and evaluated at the internal boundaries. The resulting problem, albeit larger, features a sparse Jacobian matrix, as constraints of each interval j are function, at most, of the variables in interval j and the adjacent ones. The downside of such an approach is the increased number of variables and constraints, which somehow contrast with the main idea of shooting methods.

It is worth mentioning that shooting methods are the only choice when using a stochastic or heuristic algorithm rather than a deterministic (e.g., Newton-based) one. Indeed, since heuristic algorithms do not follow a deterministic strategy to solve the NLP problem, they feature a slower rate of convergence. Thus, to obtain a solution in a reasonable time, the finite-dimensional problem must be characterized by a limited number of variables. These algorithms can be very useful when classical methods fail: for example, when a deterministic algorithm converges to a (manifestly) local optimum instead of the global one. In fact, without the need for a good initialization, heuristic algorithms are able to locate the global optimum region very efficiently. That is why they are also referred to as *global* optimizers. However, the lack of theoretical convergence guarantees, the greater computational effort, and the reduced precision of the attained solution, make deterministic approaches the preferred option whenever their use is possible.

2.3.2 Collocation Methods

If a fully-discrete parameterization is carried out, the differential equations are collocated at each node and collected into a finite set of algebraic constraints, called defect constraints. The expression of the constraints depends on the specific collocation scheme.

In this thesis, two collocation methods are considered: *trapezoidal* and *pseudospectral*. The former leverages the trapezoidal rule to formulate the defect constraints. The ease of implementation and the sparsity of the resulting problem make it a very common approach. However, the low order of the integration scheme requires the use of a quite dense discretization grid, thus leading to a large and onerous NLP problem to solve. Instead, pseudospectral methods approximate the state and control with global interpolating polynomials and collocate the dynamics at a set of nodes that corresponds to the roots of an orthogonal polynomial. In this way, *spectral* convergence of the quadrature approximation error is ensured and extremely accurate solutions are obtained even with a few number of nodes.

Trapezoidal Discretization

In the trapezoidal method, the defect constraint linking node j to node $j+1$ is formulated as

$$\boldsymbol{x}_{j+1} - \left(\boldsymbol{x}_j + \frac{h_j}{2}(\boldsymbol{f}_j + \boldsymbol{f}_{j+1})\right) = \boldsymbol{0} \tag{2.19}$$

where $h_j = \tau_{j+1} - \tau_j$ is the time-step and $\boldsymbol{f}_j = \boldsymbol{f}(\boldsymbol{x}_j, \boldsymbol{u}_j, \boldsymbol{p}, \tau_j)$.

It is worth noting that the trapezoidal collocation produces sparse Jacobian and Hessian matrices. Thus, even though the problem size may be large, the sparsity of the coefficient matrices allow for the use of sparse linear algebra algorithms, which are much more efficient than their dense counterpart. Also, the sparsity implies a reduced sensitivity of the problem to the initialization. To understand this, consider that, with the exception of the boundary conditions, the majority of the Jacobian matrix elements will be defined as

$$G_{ij} = \frac{\text{Change in defect constraint on segment } i}{\text{Change in optimization variable on segment } j} \tag{2.20}$$

So, changing a variable at a certain mesh node will affect only nearby constraints, providing the solution algorithm with great capability of finding convenient search directions and escaping local optima.

Since the discrete problem is only an approximation of the continuous-time OCP, the accuracy of the solution of the discrete problem must be assessed. Clearly, a denser mesh provides a more accurate discrete solution, but it is also associated with a greater computational effort. Also, as documented in the literature [95], coarse grids are less sensitive to inaccurate initializations. Thus, the choice of the grid to use must result from a mindful tradeoff between the need for accuracy and the computational efficiency and sensitivity of the optimization problem.

A possible strategy consists in using first a moderate number of (evenly-spaced) nodes to obtain a solution of satisfactory accuracy, and then increasing the grid size through an iterative *mesh refinement* process until the discretization error falls below a desired threshold. To prevent the problem dimension from increasing too rapidly during the mesh refinement process, additional nodes should be added only in the segments where the local error is greater. This approach is called an *adaptive* mesh refinement process. For instance, the approach due to Betts and Huffman [96] consists of evaluating the accuracy of the attained solution at every refinement step and solving an integer programming problem that minimizes (an estimate of) the maximum relative error. Thus, the precise number and location of the extra nodes are the output of said programming problem. Also, the maximum number of new nodes that can be added in each interval and in the whole grid is limited to a small value, as adding many nodes in a single step may cause convergence issues. Once the new mesh is set up, the initialization of the OCP is obtained by interpolating the previous solution at the new nodes.

From a practical standpoint, to evaluate the discretization error, the discrete solution must be converted into a continuous-time solution $(\tilde{\boldsymbol{x}}(\tau), \tilde{\boldsymbol{u}}(\tau))$ that approximates the real (unknown) solution $(\hat{\boldsymbol{x}}(\tau), \hat{\boldsymbol{u}}(\tau))$. The state $\boldsymbol{x}(\tau)$ is approximated as a vector of cubic splines, with the conditions

$$\tilde{\boldsymbol{x}}(\tau_j) = \boldsymbol{x}(\tau_j) \tag{2.21}$$

$$\frac{d}{d\tau}\tilde{\boldsymbol{x}}(\tau_j) = \boldsymbol{f}\left(\boldsymbol{x}(\tau_j), \boldsymbol{u}(\tau_j), \tau_j\right) \tag{2.22}$$

Instead, the control is represented as a linear interpolation of the node values. Whereas the control is assumed to be correct and optimal, the error between the state $\tilde{\boldsymbol{x}}(\tau)$ and the true solution is

$$\boldsymbol{\varepsilon}_j = \int_{\tau_j}^{\tau_{j+1}} |\tilde{\boldsymbol{x}}(\tau) - \hat{\boldsymbol{x}}(\tau)| d\tau \tag{2.23}$$

The integral in Eq. (2.23) can be either computed using a very accurate quadrature method, with tolerance close to machine precision, albeit requiring a significant computational effort, or it can be estimated using a step size smaller than the one of the original grid. As for the latter, two trapezoidal (half) steps can be used to estimate $\boldsymbol{\varepsilon}_j$ as

$$\boldsymbol{\varepsilon}_j \approx \frac{1}{2} \left| \tilde{\boldsymbol{x}}(\tau_j + h_j) - \tilde{\boldsymbol{x}}(\tau_j) - \frac{h_j}{4} \left(\tilde{\boldsymbol{f}}_3 + 2\tilde{\boldsymbol{f}}_2 + \tilde{\boldsymbol{f}}_1 \right) \right| \tag{2.24}$$

where

$$\tilde{\boldsymbol{f}}_k = \boldsymbol{f}[\tilde{\boldsymbol{x}}(s_k), \tilde{\boldsymbol{u}}(s_k)] \tag{2.25}$$

$$s_k = \tau_j + \frac{1}{2}(k-1)h_j \tag{2.26}$$

Only when the error falls below a given tolerance, the mesh refinement process terminates.

Pseudospectral Methods

In a pseudospectral method, a finite basis of global interpolating polynomials is used to approximate the state and control at a set of discretization points. The time derivative of the state is approximated by the derivative of the interpolating polynomial and is then constrained to be equal to the right-hand side of the equations of motion at a set of collocation points. While any set of collocation points can be used, the roots of an orthogonal polynomial are the preferred set, due to the computational efficiency of Gaussian quadrature.

Although global pseudospectral methods can be extremely powerful to efficiently capture the continuous-time dynamics with few nodes, their application is limited to problems with smooth solutions, as polynomials can approximate rapidly changing dynamics and discontinuous controls only with limited accuracy. Also, the quadrature of complex dynamics over long time horizons may require a large number of nodes, but high-order interpolating polynomials should not be used since their use may lead to summing up of truncation errors and, thus, numerical issues. Last, but not least, the defect constraints of a global pseudospectral method are associated with a dense Jacobian matrix, hindering the robustness of the solution procedure to the initialization point.

In this respect, a *hp* pseudospectral method is a more general strategy that solves many of the issues mentioned above. Indeed, while a global pseudospectral method can be classified as a *p* method, since grid convergence is attained exclusively by increasing the order of the discretization, a *hp* discretization combines the advantages of *h* and *p* schemes, as it splits the time domain into multiple subintervals and imposes the differential constraints in each segment via *local* orthogonal collocation. In this way, mesh nodes can be introduced near potential discontinuities and the exponential convergence rate of pseudospectral methods is exploited in the whole domain, as it is composed of local regions where the solution is smooth [97]. Also, the division of the time horizon into shorter intervals allows for the use of multiple lower-order polynomials instead of one high-order global polynomial, setting up a numerically more stable approach. Compared to *p* methods, the *hp* transcription generates sparser problem instances, enabling the use of efficient numerical routines and guaranteeing less sensitivity to the initialization point.

Pseudospectral methods are also classified according to which set of orthogonal collocation points is used. For example, Radau Pseudospectral Method (RPM) uses Legendre-Gauss-Radau (LGR) points [98]. The Lobatto Pseudospectral Method (LPM) uses Lagrange polynomials for the approximations and Legendre-Gauss-Lobatto (LGL) points for the orthogonal collocation [99]. In the Gauss Pseudospectral Method (GPM), the state is approximated using a basis of Lagrange polynomials similar to the LPM and the optimal control problem is orthogonally collocated at the interior Legendre-Gauss (LG) points [100]. In this thesis, only the Radau pseudospectral method (RPM) is considered since it is one of the most accurate and performing pseudospectral methods [101]. The RPM is also a particularly convenient scheme to embed in a *hp* discretization, as, differently from the LPM and GPM, it avoids redundant control variables at the segment interfaces and provides the optimal control at each mesh point (except for the final node of the final subinterval). Indeed, being based on the Legendre-Gauss-Radau (LGR) abscissas, which include the initial boundary but not the final one, locally, the RPM does not provide the terminal control in each segment, but, globally, the ambiguity drops since the final node of a segment corresponds to the initial boundary of the next one, for which, instead, the control is available.

A detailed discussion on the implementation of a *hp* Radau pseudospectral method can be found in the Refs. [102–104], thus only the major steps of the discretization scheme are outlined in the following.

First, the *hp* method splits the independent variable domain $\tau \in [0, 1]$ of each phase into h segments by defining a grid $\mathcal{H}$ of $h + 1$ nodes

$$\mathcal{H} = \{\tau_s \,|\, s = 1, \dots h + 1\} \tag{2.27}$$

Thus,

$$0 = \tau_1 < \cdots < \tau_{h+1} = 1 \tag{2.28}$$

Then, each segment $[\tau_s, \tau_{s+1}]$ for $s = 1, \dots, h$ is discretized as a grid $\mathcal{N}_s$ of $p_s + 1$ nodes

$$\mathcal{N}_s = \{\eta_j \,|\, j = 1, \dots p_s + 1\} \tag{2.29}$$

where p_s is the discretization order of the s-th segment and η is a new independent variable defined in the interval $[-1, 1]$. Thus,

$$-1 = \eta_1 < \cdots < \eta_{p_s+1} = 1 \tag{2.30}$$

The new variable η can be mapped to the original domain by the following transformation:

$$\tau = \frac{\tau_{s+1} - \tau_s}{2}\eta + \frac{\tau_{s+1} + \tau_s}{2} \tag{2.31}$$

Since we employ the RPM, the first p_s nodes of each segment correspond to the set of p_s LGR roots and constitute the collocation points $\mathcal{K}_s$. Note that $\mathcal{K}_s$ is a subset of $\mathcal{N}_s$ since it does not incorporate the terminal boundary $\eta = 1$.

Once the grid is set up, the state and control are discretized over it, and a finite set of variables $(\boldsymbol{x}_j^s, \boldsymbol{u}_j^s)$ is obtained. The superscript s denotes the s-th segment, while the subscript j refers to the j-th node of the segment. In particular, in each segment, the state is discretized over the set $\mathcal{N}_s$ and approximated using a basis of Lagrange polynomials. Note that since the state is continuous among the segments of a phase, in the algorithm implementation the same variable is used for both $\boldsymbol{x}^s_{p_s+1}$ and $\boldsymbol{x}^{s+1}_1$. Instead, the control is discretized only at the collocation points $\mathcal{K}_s$, so Lagrange polynomials of degree $p_s - 1$ are used for the approximation. The final control of the final segment is not included in the discrete problem and it is simply extrapolated from the polynomial approximation of the control signal.

Path constraints are converted into a finite set of algebraic constraints by imposing them at every node, while boundary conditions are imposed only at the initial or final point of $\mathcal{H}$. To take into account the system dynamics, the time derivative of the state interpolating polynomial is constrained to be equal to the equations of motion at the collocation points of each segment $s = 1, \dots, h$:

$$\sum_{j=1}^{p_s+1} D^s_{ij} \boldsymbol{x}^s_j = \frac{\tau_{s+1} - \tau_s}{2} \boldsymbol{f}^s_i \qquad i = 1, \dots, p_s \tag{2.32}$$

where the same notation used for discrete-time variables was used for the dynamics, $\boldsymbol{f}^s_j = \boldsymbol{f}(\boldsymbol{x}^s_j, \boldsymbol{u}^s_j, \boldsymbol{p}, \tau(\eta^s_j))$. In Eq. (2.32), D^s denotes the LGR differentiation matrix [101], which can be efficiently computed via barycentric Lagrange interpolation [105].

Similarly to local discretization methods, such as the trapezoidal collocation, adaptive mesh refinement algorithms have been proposed also for *hp* pseudospectral methods. Due to their complexity, these methods are not reported in the present thesis, but the interested reader can refer to Ref. [97]

2.4 Alternative Methods

Recently, artificial intelligence and machine learning approaches became popular methods to solve optimal control problems. While they are not the primary focus of this thesis, it is worth to survey other possible approaches and highlight their advantages and disadvantages compared to traditional deterministic optimization methods.

2.4.1 Evolutionary Optimization

Evolutionary algorithms are global optimization techniques that are based on heuristic rules, often inspired but not limited to natural paradigms, to find the solution of an optimization problem [106]. For example, genetic algorithms (GAs) generate a population of solutions, which then undergoes mutation, crossover, and selection processes, inspired by the Darwinian evolution theory, to search for optimal solutions over discrete spaces [107]. Instead, differential evolution (DE) is a variant of a GA to solve problems defined over continuous spaces [108]. Particle Swarm Optimization (PSO) [109] and Ant Colony Optimization (ACO) [110] are other popular meta-heuristic methods that are based on imitating the foraging behavior of flocks of birds and ant colonies, respectively.

Meta-heuristic approaches can be extremely appealing, as they allow finding (nearly) optimal solutions to greatly challenging problems that traditional optimization methods often fail solving because of nonlinear and non-differentiable objectives or highly irregular feasibility domains. Indeed, local search algorithms require an initialization point that lies in the basin of convergence of the global optimum to converge to the latter, and, for some problems, it may be quite arduous to identify this region in the search space; thus, only convergence to local optima with objectives significantly worse than the global solution can be reasonably expected if adopting a deterministic approach.

In the aerospace field, where mission design often requires solving extremely challenging optimal control problems, non-deterministic approaches became quite popular in the last decades. For instance, genetic algorithms are commonly employed for solving high-dimensional combinatorial problems, such as active debris removal missions [111–113]. Differential evolution has been successfully used to solve complex continuous problems, such as multiple gravity-assist capture trajectory [114] and rocket ascent trajectory design [115].

However, meta-heuristic algorithms are characterized by much slower convergence rates, which greatly increase the computational cost compared to deterministic algorithms. Also, most of these algorithms have few or no convergence guarantees, meaning that the number of iterations necessary to find the global optimum or a nearly-optimal solution within a desired tolerance is theoretically infinite or unknown. These aspects make them unsuitable for time-critical applications, where computational efficiency and convergence guarantees are crucial.

2.4.2 Machine Learning

Differently from evolutionary algorithms, which are commonly employed to aid the design of an optimal spacecraft trajectory before flight, machine learning concepts like neural networks (NNs) have been investigated by the aerospace community with the aim of defining guidance and control policies for real-time onboard implementation.

Neural networks are universal function approximators, and can be used to approximate the optimal closed-loop control law, that is, the map between the observed spacecraft state (input vector) and the thrust magnitude and direction (output vector). The evaluation of the policy is extremely fast, quicker than any optimization process, thus highly appealing for time-critical applications.

The main challenge consists in training the neural network, that is, finding the values of the network's parameters that produce a reasonable approximation of the optimal control law. Behavioral cloning (BC) is a popular training strategy that, given a dataset of sample optimal trajectories from an "expert" (e.g., a set of solutions provided by a deterministic solver), trains the network to replicate the expert behavior by minimizing the difference between the network output and the corresponding expert data. BC was successfully employed in diverse aerospace applications, including interplanetary transfers [116], powered descent landing [117, 118], and hypersonic reentry [119, 120].

Reinforcement learning (RL) is an alternative method to train a NN to solve an optimal control problem. Differently from BC, in RL the network is trained by repeatedly interacting with a large number of realizations of the environment. In practice, the training consists in simulating the problem dynamics and maximizing a so-called *reward* function, which accounts for the OCP's merit index and a measure of the performance of the NN in approximating the problem's model and constraints, to turn the NN into an optimal control policy. Although RL is quite a novel research trend among the aerospace community, several articles have been published on the subject in the last few years, with application to the powered descent landing problem [121, 122], low-thrust transfers [123–125], spacecraft rendezvous [126, 127], proximity operations [128], and terminal guidance [129].

Machine learning techniques as those previously mentioned are highly appealing for real-time guidance applications, as the low evaluation times and high accuracy in function approximation of NNs provide a computationally light, closed-form control law, specifically tailored to the considered problem. Also, the training of the NNs with datasets that cover large portions of the problem's domain, as in BC, or in perturbed simulated environments, as in RL, provides an intrinsic layer of robustness to the resulting policies, which may improve the effectiveness of these approaches when considering off-nominal conditions. However, the extensive computational effort needed for the training process and the absence of theoretical guarantees on the effectiveness and actual robustness of such policies still limit their practical application to relatively simple problems.

Chapter 3

Convex Optimization

Convex optimization is a class of mathematical programming with polynomial complexity for which highly efficient numerical algorithms that converge in a limited, short, time exist. Thanks to the theoretical guarantees on the optimality of the attained solution and the computational efficiency, convex optimization is a highly appealing tool to solve optimal control problems. In this chapter, the general formulation of several classes of convex programming problems of increasing complexity, starting from linear programming up to second-order cone programming, is presented. Also, convexification strategies to convert a nonconvex problem into a convex one or a sequence of convex problems are detailed along with proper safeguarding modifications to the original problem, such as virtual variables or trust regions. An example nonconvex problem is solved with convex optimization techniques to demonstrate the effectiveness of the presented strategies.

3.1 Preliminaries

This section introduces a few fundamental definitions and concepts that will be frequently used throughout this thesis.

3.1.1 Linear and Convex Functions

A *linear* function is a function $f : \mathcal{X} \to \mathbb{R}$ that satisfies the equality

$$f(\alpha x + \beta y) = \alpha f(x) + \beta f(y) \tag{3.1}$$

for all $x, y \in \mathcal{X}$ and for all $\alpha, \beta \in \mathbb{R}$.

Instead, a *convex* function is a function $g : \mathcal{X} \to \mathbb{R}$ that satisfies the inequality

$$g(\alpha x + \beta y) \leq \alpha g(x) + \beta g(y) \tag{3.2}$$

for all $x, y \in \mathcal{X}$ and for all $\alpha, \beta \in [0, 1]$ such that $\alpha + \beta = 1$. So, convexity is more general than linearity, as the inequality sign is less restrictive than the equality sign and Eq. (3.2) must hold only for certain values of α and β. Thus, any linear function is also a convex function.

3.1.2 Affine and Convex Sets

A set A is *affine* if the line through any two distinct points in A lies in A, that is, if for any $x_1, x_2 \in A$

$$\theta x_1 + (1 - \theta) x_2 \in A, \qquad \theta \in \mathbb{R} \tag{3.3}$$

So, A contains the linear combination of any two points in A, provided that the coefficients of the linear combination sum to one.

Instead, a set C is *convex* if the line segment between any two points in C lies in C, that is, that is, if for any $x_1, x_2 \in C$

$$\theta x_1 + (1-\theta)x_2 \in C, \qquad \theta \in [0,1] \tag{3.4}$$

The definition of a convex set is more general than that of an affine set, as Eq. (3.4) must be satisfied only for some values of θ. Therefore, every affine set is also convex, since it contains the entire line between any two distinct points in it, and therefore also the line segment between the points.

Equation (3.4) can be extended to multiple points. Indeed, if $x_j \in C$ for $j = 1, \dots, m$ and C is convex, then

$$\theta_1 x_1 + \cdots + \theta_m x_m \in C, \qquad \theta_j \geq 0, \quad \sum_j \theta_j = 1 \tag{3.5}$$

The sum $\theta_1 x_1 + \cdots + \theta_m x_m$ is called a *convex combination* of the points x_j. Analogously to affine sets, a set is convex if and only if it contains every convex combination of its points.

3.1.3 Cones

A set K is called a *cone* if for every $x \in K$

$$\theta x \in K, \qquad \theta \geq 0 \tag{3.6}$$

A cone K is a *convex cone* if it is a convex set. In this case, according to Eq. (3.5), the following property holds

$$\theta_1 x_1 + \cdots + \theta_m x_m \in K, \qquad \theta_j \geq 0 \tag{3.7}$$

with $x_j \in K$ for $j = 1, \dots, m$. So, if x_j are in a convex cone K, then every nonnegative linear combination (also called a *conic combination*) of x_j is in K.

3.1.4 Hyperplanes and Polyhedra

A *hyperplane* is a set of the form

$$\left\{x \,\middle|\, a^T x = b\right\} \tag{3.8}$$

where $a \in \mathbb{R}^n$, $a \neq 0$, and $b \in \mathbb{R}$. Thus, an hyperplane is the solution set of a nontrivial linear equation and, hence, an affine set.

A hyperplane divides $\mathbb{R}^n$ into two *halfspaces.* A (closed) halfspace is a set of the form

$$\left\{x \,\middle|\, a^T x \leq b\right\} \tag{3.9}$$

where $a \neq 0$. Thus, a hyperplane is the solution set of a nontrivial linear inequality, and is thus a convex, but not affine, set.

A *polyhedron* is defined as the solution set of a finite number of linear equalities and inequalities

$$\left\{x \,\middle|\, a_j^T x \leq b_j, j = 1, \dots, m, c_j^T x \leq d_j, j = 1, \dots, l, \right\} \tag{3.10}$$

Thus, a polyhedron is the intersection of a finite number of halfspaces and hyperplanes. Every polyhedron is a convex set. Also, all affine sets (e.g., hyperplanes and lines) and halfspaces are polyhedra. A bounded polyhedron is sometimes called a *polytope.*

3.2 Convex Programming Problems

3.2.1 Linear Programming

A linear programming (LP) problem is the simplest class of optimization problems. The canonical form of such a problem is

$$\min_{\boldsymbol{x}} \quad \boldsymbol{c}^T \boldsymbol{x} \tag{3.11}$$
$$\text{s.t.} \quad A\boldsymbol{x} = \boldsymbol{b} \tag{3.12}$$
$$\boldsymbol{x} \geq 0 \tag{3.13}$$

where $\boldsymbol{x}$ is the vector of decision variables and A, $\boldsymbol{b}$, and $\boldsymbol{c}$ are constant coefficient matrices and vectors. The objective function and the constraints are all affine functions of the optimization variables. Thus, solving a LP problem consists in finding the non-negative values $\boldsymbol{x}$ that minimize the linear objective function over a polyhedron.

3.2.2 Quadratic Programming

If the objective function in Eq. (3.11) is replaced by a quadratic function of the decision variables, then the problem is a quadratic programming (QP) problem and can be expressed as

$$\min_{\boldsymbol{x}} \quad \boldsymbol{x}^T Q \boldsymbol{x} + \boldsymbol{c}^T \boldsymbol{x} \tag{3.14}$$
$$\text{s.t.} \quad A\boldsymbol{x} = \boldsymbol{b} \tag{3.15}$$
$$\boldsymbol{x} \geq 0 \tag{3.16}$$

Thus, the constraints are, at most, linear, but the objective is quadratic, hence featuring a greater complexity than the LP problem. It is worth mentioning that if Q is a positive semi-definite (PSD) matrix (i.e., $\boldsymbol{x}^T Q \boldsymbol{x} \geq 0 \ \forall \boldsymbol{x}$), the QP problem is convex and it can be solved in polynomial time. Otherwise, if Q is not positive semi-definite, then the problem is NP-hard and it can feature multiple local minima and stationary points.

3.2.3 Quadratically-Constrained Quadratic Programming

If not only the objective function but also the constraints are quadratic, then the resulting problem is referred to as a quadratically-constrained quadratic programming (QCQP) problem. The canonical form of a QCQP is

$$\min_{\boldsymbol{x}} \quad \boldsymbol{x}^T Q_0 \boldsymbol{x} + \boldsymbol{c}_0^T \boldsymbol{x} \tag{3.17}$$
$$\text{s.t.} \quad \boldsymbol{x}^T Q_1 \boldsymbol{x} + \boldsymbol{c}_1^T \boldsymbol{x} \leq \boldsymbol{r} \tag{3.18}$$
$$A\boldsymbol{x} = \boldsymbol{b} \tag{3.19}$$
$$\boldsymbol{x} \geq 0 \tag{3.20}$$

As for the QP class, a QCQP problem is convex if Q_0 and Q_1 are positive semi-definite matrices. Otherwise, the QCQP problem is NP-hard.

3.2.4 Second-Order Cone Programming

A class of problems that is closely related to the convex QP and QCQP is second-order cone programming (SOCP). The canonical form of a SOCP is

$$\min_{\boldsymbol{x}} \quad \boldsymbol{c}^T \boldsymbol{x} \tag{3.21}$$

$$\text{s.t.} \quad A\boldsymbol{x} = \boldsymbol{b} \tag{3.22}$$

$$\|F\boldsymbol{x} + \boldsymbol{g}\|_2 \leq \boldsymbol{h}^T \boldsymbol{x} + \boldsymbol{r} \tag{3.23}$$

Thus, a SOCP consists of a linear objective, linear equality constraints, and second-order cone inequality constraints.

Note that all previous classes of (convex) problems can be cast as a SOCP. For instance, the quadratic constraint in Eq. (3.18) is equivalent to the squared second-order cone constraint in Eq. (3.23) and the quadratic term $\boldsymbol{x}^T Q_0 \boldsymbol{x}$ of the objective in Eq. (3.17) can be added as an auxiliary variable ξ to the linear objective function in Eq. (3.21) if ξ is constrained via a second-order cone constraint as

$$\|R_0 \boldsymbol{x}\|_2 \leq \xi \tag{3.24}$$

with R_0 such that $Q_0 = R_0^T R_0$. So, a SOCP is a more general form than LP, convex QP, and convex QCQP. Nevertheless, a SOCP problem can be solved in polynomial time with interior-point algorithms.

3.3 Convexification Methods

Convex optimization can be used to solve complex real-world problem even though these are not inherently convex. Indeed, by means of proper convexification techniques, nonconvex constraints can be converted into convex expressions. Convexification methods can be grouped into two main categories: *lossless* and *successive* methods.

3.3.1 Lossless Convexification Methods

Lossless convexification techniques are extremely powerful tools in optimization, as they can transform a nonconvex problem into a convex problem that shares the same solution as the original one, without introducing any approximations. Therefore, it is apparent that such expedients should be used whenever it is possible, as they dramatically reduce the optimization problem complexity and do not alter its solution. Lossless convexification methods consist of either carrying out a convenient change of variables or relaxing some nonconvex constraint into a convex form.

Change of Variables

Changes of variables in optimal control problems generally aim at replacing nonlinear terms with linear ones. According to the literature, nonlinear terms including both state and control variables in the system dynamics should be avoided whenever possible, as their presence leads to numerical issues, such as high-frequency jitters in the solution [130]. This is typically the case of variable-mass systems, such as chemically-propelled spacecraft, since the dynamics contain the term $\boldsymbol{T}/m$, with $\boldsymbol{T}$ denoting the thrust vector and m the mass. In this case, it is convenient to define a new control variable [33]

$$\tilde{\boldsymbol{u}} = \frac{\boldsymbol{T}}{m} \tag{3.25}$$

in lieu of $\boldsymbol{u} = \boldsymbol{T}$.

In general, new control variables are introduced with the aim of formulating an OCP with control-affine dynamics

$$\frac{d\boldsymbol{x}}{d\tau} = \hat{\boldsymbol{f}}(\boldsymbol{x}(\tau), \boldsymbol{p}, \tau) + B(\tau)\tilde{\boldsymbol{u}}(\tau) \tag{3.26}$$

where $\tilde{\boldsymbol{u}}(\tau)$ is the transformed control vector. The result of replacing $\boldsymbol{u}(\tau)$ with $\tilde{\boldsymbol{u}}(\tau)$ is that the dynamics are linear in the control, as the input matrix B depends, at most, on the independent variable τ. It is worth noting that $\hat{\boldsymbol{f}}$ may still be a nonlinear function of the other optimization variables, so further changes of variables or other convexification techniques must be employed to obtain a convex problem instance. Indeed, usually the convexification process is not based on just one transformation but, rather, consists of a combination of several convexification techniques that must be mindfully tailored to the specific problem in exam.

Constraint Relaxation

The relaxation of a nonconvex constraint is another common convexification strategy. The relaxation usually concerns substituting a quadratic equality constraint, such as

$$\boldsymbol{x}^T Q \boldsymbol{x} = \boldsymbol{q}^T \boldsymbol{x} + \boldsymbol{r} \tag{3.27}$$

with a quadratic inequality constraint,

$$\boldsymbol{x}^T Q \boldsymbol{x} \leq \boldsymbol{q}^T \boldsymbol{x} + \boldsymbol{r} \tag{3.28}$$

Note that Eq. (3.27) is a nonconvex constraint, while Eq. (3.28) is convex if the coefficient matrix Q is positive semi-definite.

A relaxation is said to be *exact* (or lossless) if the optimal solution of the relaxed problem is the same as the original. Therefore, even though Eq. (3.28) defines a larger feasible domain for $\boldsymbol{x}$, the optimal solution of the OCP satisfies Eq. (3.28) with the equality sign. In practice, this occurs because the relaxed domain of $\boldsymbol{x}$ contains only solutions with values of the objective function equal to or worse than those that lie in the original domain, thus the solutions that strictly satisfy Eq. (3.28) are naturally discarded.

The exactness of a specific constraint relaxation is often rigorously provable by using elements of optimal control theory. More detailed discussions will be presented in the next sections and chapters, with the help of practical examples.

3.3.2 Successive Convexification Methods

When no lossless convexification method can be applied to convert a nonconvex constraint into a convex one, successive convexification must be employed. Successive convexification consists in replacing a nonlinear, nonconvex, constraint with a linear expression. For instance, a general nonlinear constraint

$$g(\boldsymbol{x}) \leq 0 \tag{3.29}$$

can be replaced with

$$g_0(\bar{\boldsymbol{x}}) + G_x(\bar{\boldsymbol{x}})\boldsymbol{x} \leq 0 \tag{3.30}$$

where $\bar{\boldsymbol{x}}$ denotes a reference solution that is used to define the linear expression. The resulting constraint in Eq. (3.30) is a linear function of the optimization variables $\boldsymbol{x}$

and a general function of the reference values. Indeed, g_0 and the matrix G_x are (possibly nonlinear) functions only of the reference solution.

There are multiple ways of defining the linearized constraint, but, in general, it must be ensured that Eq. (3.30) is equivalent to Eq. (3.29) whenever $\boldsymbol{x} = \bar{\boldsymbol{x}}$. Indeed, Eq. (3.30) is just an approximation of the original constraint of Eq. (3.29), but successive convexification is based on the fact that if the difference between the optimal solution and the reference values is small enough, then the linearized constraint approximates with great accuracy the original constraint.

To achieve *convergence* of the optimal and reference solutions, a sequential convex optimization procedure must be implemented, where, at every iteration, a convex problem is solved and the obtained solution is used to update the reference values. Due to the fact that a *sequence* of convex problems is solved, these methods are called *successive* convexification methods. The algorithm is said to *converge* when the difference between the optimal and reference solution goes below a desired threshold, and the converged solution satisfies Eq. (3.29). Convergence of the successive convexification algorithm can be theoretically guaranteed only under appropriate assumptions, but numerical evidence proves that convergence can be achieved also in more general cases.

A general way of obtaining the linearized expression of Eq. (3.30) starting from Eq. (3.29). is *successive linearization.* In practice, successive linearization consists in replacing a nonlinear expression with its first-order Taylor series expansion around a reference solution. Taylor series expansions provide quite accurate approximations, as long as the solution does not deviate excessively from the reference values. For instance, the general nonlinear constraint in Eq. (3.29) is linearized as

$$g(\bar{\boldsymbol{x}}) + \frac{\partial \bar{g}}{\partial \boldsymbol{x}}(\boldsymbol{x} - \bar{\boldsymbol{x}}) \leq 0 \tag{3.31}$$

Note that the Jacobian matrix $\partial \bar{g}/\partial \boldsymbol{x}$ is evaluated on the reference solution.

Despite the need to analytically derive the Jacobian matrix, which is the main downside of this strategy, successive linearization is a very powerful tool since it can be applied to any nonlinear constraint. This is usually the preferred way to handle nonlinear system dynamics, which are rarely suitable for any lossless convexification technique.

The linearized dynamics can be expressed as

$$\frac{d\boldsymbol{x}}{d\tau} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p}, \tau) \approx A\boldsymbol{x} + B\boldsymbol{u} + P\boldsymbol{p} + \boldsymbol{c} \tag{3.32}$$

where the matrices A, B, and P, and the vector $\boldsymbol{c}$ are evaluated on the reference solution and are defined as

$$A = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}}, \bar{\boldsymbol{p}}, \tau) \tag{3.33}$$

$$B = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}}, \bar{\boldsymbol{p}}, \tau) \tag{3.34}$$

$$P = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{p}}(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}}, \bar{\boldsymbol{p}}, \tau) \tag{3.35}$$

$$\boldsymbol{c} = \bar{\boldsymbol{f}} - A\bar{\boldsymbol{x}} - B\bar{\boldsymbol{u}} - P\bar{\boldsymbol{p}} \tag{3.36}$$

In the literature [31], any method that is not based on Taylor series expansions for deriving the form in Eq. (3.30) starting from the nonlinear constraint in Eq. (3.29) is referred to as *successive approximation.* Successive approximations are generally easier to implement than Taylor series, as no analytical derivation of the Jacobian matrix is needed; however, a suitable *ad hoc* approximation of the nonlinear constraint must be devised.

3.4 Virtual Controls and Buffer Zones

The linearization of a nonconvex problem may generate an infeasible convex problem, even though the original formulation is feasible. This phenomenon is known as *artificial infeasibility* [131–133]. This is a very undesirable phenomenon, as successive convexification is based on solving a sequence of problems, and, if one intermediate problem cannot be solved, then the entire process may end prematurely.

To prevent the occurrence of an infeasible problem, additional unconstrained variables $\boldsymbol{q}$, referred to as *virtual controls*, are added to the linearized dynamics

$$\frac{d\boldsymbol{x}}{d\tau} = A\boldsymbol{x} + B\boldsymbol{u} + P\boldsymbol{p} + \boldsymbol{c} + \boldsymbol{q} \tag{3.37}$$

The addition of the virtual control signal makes any state $\boldsymbol{x}$ reachable in a finite time, thus removing any infeasibility that might be related to the linearization of the system dynamics.

Analogously, since the artificial infeasibility may arise also from the linearization of other constraints, additional free variables $\boldsymbol{w}$ should be included to relax such constraints. The introduction of these variables solves the infeasibility issue by defining a virtual buffer zone, and $\boldsymbol{w}$ are thus called *virtual buffers* [52, 131]. For instance, let us consider a linearized constraint such as in Eq. (3.31). If the linearized expression may be a source of artificial infeasibility, then it must be replaced with the *buffered* constraint

$$g(\bar{\boldsymbol{x}}) + \frac{\partial \bar{g}}{\partial \boldsymbol{x}}(\boldsymbol{x} - \bar{\boldsymbol{x}}) + w \leq 0 \tag{3.38}$$

Both virtual controls and virtual buffers should be used only when necessary by the SOCP solver, as they are nonphysical variables. To ensure a correct behavior, penalty terms must be added to the objective function of the convex problem , such as

$$J_q = \lambda_q P_q(\boldsymbol{q}) \tag{3.39}$$

$$J_w = \lambda_w P_w(\boldsymbol{w}) \tag{3.40}$$

where $P_q(\cdot)$ and $P_w(\cdot)$ are suitable penalty functions and λ_q and λ_w denote positive penalty weights. The penalty terms must highly penalize the use of the virtual variables. However, excessively high coefficients in the objective function should be avoided, as they may introduce numerical issues and hinder convergence; on the other hand, too small penalties may lead to a nonphysical solution that actively exploits virtual controls and buffers. Nevertheless, the process of selecting the value of the penalty weights is generally straightforward, as finding the right order of magnitude is usually enough to ensure convergence of $\boldsymbol{q}$ and $\boldsymbol{w}$ below a desired tolerance. For instance, suitable values of λ_q and λ_w can be found through a trial-and-error process, starting with very low weights and increasing the weights until sufficiently large values that prevent the unnecessary use of the virtual variables are found.

3.5 Trust Regions

Another undesired effect due to the linearization is *artificial unboundedness* [36, 131, 132], that is, the convexified problem may appear unbounded (i.e., the objective function can be improved indefinitely without violating the constraints) even though the original problem is not. This phenomenon generally occurs when the optimization variables deviate excessively from the reference values, hence the first-order Taylor series expansions do not represent anymore valid approximations of the original expressions. To mitigate the risk of artificial unboundedness, a common approach consists in limiting the search space to a so-called *trust region.*

To enforce a trust region, the deviation from the reference values must be constrained somehow. For instance, a radius δ_i can be chosen for each optimization variable x_i and the following constraint can be imposed

$$|x_i - \bar{x}_i| \leq \delta_i \tag{3.41}$$

If the value of the trust radius δ_i is prescribed, then the trust region is said to be *hard*, meaning that the maximum deviation is fixed and cannot be violated by the solution. On the other hand, if δ_i is an optimization variable itself and it is included in the objective function through a proper penalty term, then the trust region is *soft*, as the algorithm must autonomously determine its optimal value.

Hard trust regions allow to solve a problem with full optimality if the optimal solution lies in the (now-limited) search space. Thus, the choice of the trust radius is crucial for the success of a hard trust region approach. *Adaptive* hard trust regions have been proposed as an improvement of standard hard trust regions. When using an adaptive trust region, the value of the trust radius is not fixed through all the solution process, but updated at every iteration of the successive convexification algorithm according to a set of rules. In general, the radius update criteria are aimed at improving the convergence rate or facilitating rigorous proofs of guaranteed convergence [37]. Examples of these rules vary from simple exponentially shrinking sequences [44] to sophisticated algorithms that evaluate the linearization error at each iteration and decide whether to expand or shrink the search space [131, 134].

Soft trust regions, instead, autonomously determine the trust radius size, without any initialization. However, these may lead to a suboptimal solution if the penalty weight is too high. Conversely, artificial unboundedness may still arise if a low weight is picked. Thus, the user must be able to select a suitable penalty weight, which must not be exaggeratedly high nor low. Whether it is easier to select a suitable penalty weight rather than a starting trust radius depends on the specific problem instance. In general, if an accurate reference solution is provided, then a soft trust region is very effective; this is the case of real-time guidance, where only small deviations from the nominal path are expected [135]. On the other hand, when designing a trajectory from scratch, large deviations from the reference values are expected and an adaptive hard trust region with a large initial radius is generally more effective than a soft trust region [134].

Trust regions can be also classified according to which variables are constrained. Indeed, constraining all variables may be unnecessary, as only some variables may be responsible for artificial unboundedness. For instance, free-time problems are usually more sensitive to changes in time-lengths than in other variables, so a trust region on the duration of the arcs may be sufficient to prevent unboundedness [136, 137, 137]. Also, the linearization of nonlinear dynamics and constraints may require a trust region on both state and control variables, such as in Refs. [44, 135, 138, 139]. However, if the OCP features control-affine dynamics and no linearized control constraint, then control variables do not need to be constrained through a trust region, as the linearized expressions do not depend on reference controls [140, 141]. Finally, there are some cases where constraining some variables implicitly ensures that also other variables will be constrained. For instance, in Refs. [36, 131, 134] the authors argue that enforcing a trust region solely on the control variables implicitly limits the deviation of the state variables among successive iterations.

It is worth mentioning that Eq. (3.41) serves only as an example, as the trust region constraint can be posed in many ways. Indeed, trust regions can be classified also according to how the deviation from the reference values is evaluated and constrained. It is common to group multiple variables together and constrain the p-norm of their deviation (with $p = \{1, 2, \infty\}$ to ensure convexity) to be smaller than the trust radius. For instance,

$$\|\boldsymbol{x} - \bar{\boldsymbol{x}}\|_p \leq \delta \tag{3.42}$$

Otherwise, quadratic trust regions, such as

$$(\boldsymbol{x} - \bar{\boldsymbol{x}})^T (\boldsymbol{x} - \bar{\boldsymbol{x}}) \leq \delta \tag{3.43}$$

are also quite common in the literature [45, 138, 142].

3.6 Reference Solution Update

Successive convexification methods are based on replacing nonlinear (nonconvex) expressions with linear expressions defined around a reference solution $\{\bar{\boldsymbol{x}}\}$. In the first iteration, the reference values must be provided by the user, and this process is generally called initialization. In later iterations, the reference solution is recursively updated with the computed solutions. Most successive convexification algorithms update the reference solution by replacing it with the last found solution; thus, the i-th problem is formulated using the $(i-1)$-th solution as reference.

A more general update strategy consists in computing the reference solution for the i-th problem as a weighted sum of the K previous solutions. Thus,

$$\bar{x}^{(i)} = \sum_{k=1}^{K} \alpha_k x^{\max\{0,(i-k)\}} \tag{3.44}$$

where α_k are constant weights and $x^{(i)}$ denotes the solution to the i-th problem. Note that if $i < K$ then the initial reference solution $x^{(0)}$ appears multiple times in the sum.

This update method is called *filtering* [38, 136]. Using multiple previous solutions generates a smoother sequence of reference solutions, as the weighted sum of different solutions filters out possible diverging intermediate iterations. As a result, the algorithm is less sensitive to artificial unboundedness, as deviations from the initial reference solutions are intrinsically mitigated by the reduced weight given to every new iteration. Thus, trust regions or other expedients typically necessary to prevent artificial unboundedness can be avoided or used to a lesser extent.

Indeed, filtering can be a greatly appealing alternative to trust regions since it does not introduce any additional constraints or penalty terms to the problem, retaining its original objective and formulation. The downside of using multiple solutions is generally a slower convergence rate, as changes to the reference solution require more iterations than in a last-found-solution update approach. Finally, the filtering technique can be used for the update of all optimization variables: states, controls, and parameters. Some of these variables may be also subject to a trust region constraint, as filtering does not exclude the use of a trust region.

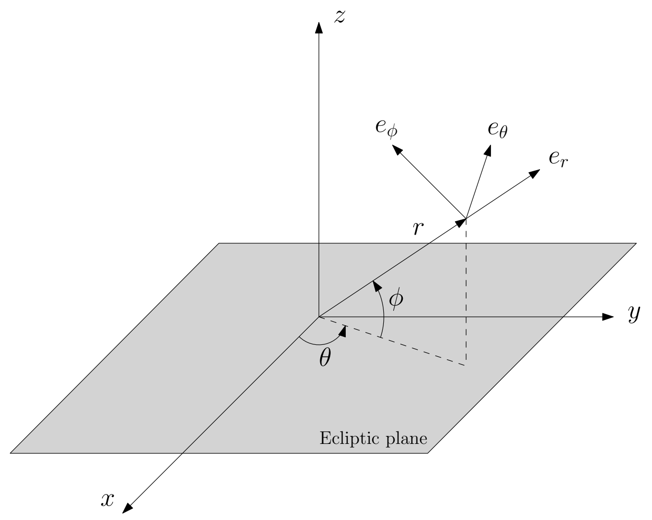


Figure 3.1. Reference frames used for the low-thrust transfer.

3.7 Convergence Criteria

Eventually, the sequential algorithm terminates when all the following criteria are met:

(i) the difference between the computed solution and the reference one converges below an assigned tolerance

$$\|\boldsymbol{x} - \bar{\boldsymbol{x}}\|_\infty < \epsilon_{\text{tol}} \tag{3.45}$$

(ii) the computed solution adheres to the nonlinear dynamics within a tolerance ϵ_f in each phase

$$\max_j \left\| (A_j \boldsymbol{x}_j + B_j \boldsymbol{u}_j + P\boldsymbol{p} + \boldsymbol{c}_j + \boldsymbol{q}_j) - \boldsymbol{f}\left(\boldsymbol{x}_j, \boldsymbol{u}_j, \boldsymbol{p}, \tau_j\right) \right\|_\infty < \epsilon_f \tag{3.46}$$

where j is the discretization node index;

(iii) the virtual buffers of the computed solution are below the dynamics tolerance

$$\|\boldsymbol{w}\|_\infty < \epsilon_f. \tag{3.47}$$

3.8 Example: Low-Thrust Interplanetary Mission

As an example, the use of convex optimization techniques on a low-thrust trajectory design problem is investigated. This mission scenario considers a spacecraft that is leaving the Earth with zero hyperbolic excess velocity and must rendezvous with Mars at a prescribed final time.

3.8.1 System Dynamics

The spacecraft is modeled as a point mass subject to a three-degree-of-freedom translational motion. The state variables are the position vector $\boldsymbol{r}$, the velocity vector $\boldsymbol{v}$, and the spacecraft mass m. Spherical coordinates are adopted for describing the spacecraft position, while velocity is expressed using Cartesian components in a Local-Vertical-Local-Horizontal (LVLH) frame. The reference frames are shown in Fig. 3.1. Therefore, the state vector $\boldsymbol{x}$ is

$$\boldsymbol{x} = [r \quad \theta \quad \varphi \quad v_r \quad v_\theta \quad v_\varphi \quad m] \tag{3.48}$$

where r is the distance from the Sun, θ is the right ascension (i.e., the angular displacement from the initial position), φ is the elevation (i.e., the angle between the position vector and the ecliptic), and v_r, v_θ, and v_φ are the components of velocity along the three axes in the spherical coordinate system (e_r, e_θ, e_ϕ).

The control to be optimized is the thrust vector $\boldsymbol{T}$, which is expressed in the same frame as the velocity,

$$\boldsymbol{T} = [T_r \quad T_\theta \quad T_\varphi] \tag{3.49}$$

The low-thrust engine can regulate the thrust magnitude in a limited range. Thus,

$$\|\boldsymbol{T}\| \leq T_{\max} \tag{3.50}$$

The considered equations of motion for the system are

$$\dot{r} = v_r \tag{3.51}$$
$$\dot{\theta} = \frac{v_\theta}{r\cos\varphi} \tag{3.52}$$
$$\dot{\varphi} = \frac{v_\varphi}{r} \tag{3.53}$$
$$\dot{v}_r = \frac{v_\theta^2 + v_\varphi^2}{r} - \frac{\mu}{r^2} + \frac{T_r}{m} \tag{3.54}$$
$$\dot{v}_\theta = -\frac{v_r v_\theta}{r} + \frac{v_\theta v_\varphi}{r}\tan\varphi + \frac{T_\theta}{m} \tag{3.55}$$
$$\dot{v}_\varphi = -\frac{v_r v_\varphi}{r} - \frac{v_\theta^2}{r}\tan\varphi + \frac{T_\varphi}{m} \tag{3.56}$$
$$\dot{m} = -\frac{T}{c} \tag{3.57}$$

where μ is the gravitational parameter of the Sun and c is the effective exhaust velocity of the engine, defined as the product of the Earth's gravity acceleration at sea level g_0 by the specific impulse I_{sp}.

3.8.2 Optimal Control Problem

The main goal of the optimization is to determine the control law that maximizes the final mass, so the objective function J to minimize is:

$$J = -m(t_f) \tag{3.58}$$

where t_f denotes the (fixed) final time.

The initial state is supposed to be fully assigned

$$\boldsymbol{x}(t_0) = \tilde{\boldsymbol{x}}_0 \tag{3.59}$$

Note that, since the initial mass is prescribed, minimizing Eq. (3.58) is equivalent to minimizing the propellant consumption.

At the final time, the spacecraft must rendezvous with Mars, that is, it must share the same position and velocity of the planet at that time. Thus, the following final conditions are enforced

$$\boldsymbol{r}(t_f) = \tilde{\boldsymbol{r}}_f \tag{3.60}$$

$$\boldsymbol{v}(t_f) = \tilde{\boldsymbol{v}}_f \tag{3.61}$$

Therefore, the resulting optimal control problem, $\mathcal{P}_0$, is

$$\begin{aligned} \mathcal{P}_0 : \quad & \min_{\boldsymbol{x},\, \boldsymbol{T}} && (3.58) \\ & \text{s.t.} && (3.50),\ (3.51)\text{–}(3.57),\ (3.59),\ (3.60),\ (3.61) \end{aligned} \tag{3.62}$$

3.8.3 Convex Transcription

The problem $\mathcal{P}_0$ is not convex due to the nonlinear dynamics in Eqs. (3.51)–(3.57). However, it can be converted into a convex problem through a combination of lossless and successive convexification methods.

First, a change of variables is carried out to replace nonlinear terms in the dynamics by linear terms and obtain a control-affine dynamical system as in Eq. (3.26). The new control variables are introduced:

$$u_r = \frac{T_r}{m} \tag{3.63}$$

$$u_\theta = \frac{T_\theta}{m} \tag{3.64}$$

$$u_\varphi = \frac{T_\varphi}{m} \tag{3.65}$$

$$u_N = \frac{T}{m} \tag{3.66}$$

These can be collected into the new control vector, that is,

$$\boldsymbol{u} = \begin{bmatrix} u_r & u_\theta & u_\varphi & u_N \end{bmatrix} \tag{3.67}$$

Substituting $\boldsymbol{T}$ with $\boldsymbol{u}$ in Eqs. (3.54)–(3.56) directly produces control-affine equations, but the same is not true for Eq. (3.57), which becomes

$$\dot{m} = -\frac{m u_N}{c} \tag{3.68}$$

So, a further change of variables is carried out and the new state variable is introduced

$$z = \ln m \tag{3.69}$$

The introduction of the z variable requires to reformulate the objective function, which can be replaced by

$$J = -z(t_f) \tag{3.70}$$

Note that minimizing the new objective is equivalent to minimizing Eq. (3.58), as the change of variable in Eq. (3.69) is a monotonic transformation.

By replacing the mass in Eq. (3.68) with z, defined in Eq. (3.69), the equations of motion are linear in the control $\boldsymbol{u}$

$$\dot{r} = v_r \tag{3.71}$$

$$\dot{\theta} = \frac{v_\theta}{r \cos\varphi} \tag{3.72}$$

$$\dot{\varphi} = \frac{v_\varphi}{r} \tag{3.73}$$

$$\dot{v}_r = \frac{v_\theta^2 + v_\varphi^2}{r} - \frac{\mu}{r^2} + u_r \tag{3.74}$$

$$\dot{v}_\theta = -\frac{v_r v_\theta}{r} + \frac{v_\theta v_\varphi}{r} \tan\varphi + u_\theta \tag{3.75}$$

$$\dot{v}_\varphi = -\frac{v_r v_\varphi}{r} - \frac{v_\theta^2}{r} \tan\varphi + u_\varphi \tag{3.76}$$

$$\dot{z} = -\frac{u_N}{c} \tag{3.77}$$

Since the equations are still nonlinear in the state variables, Eqs. (3.71)–(3.77) must be linearized around a reference solution $\{\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}}\}$ and the dynamics assume the form

$$\dot{\boldsymbol{x}} = A\boldsymbol{x} + B\boldsymbol{u} + \boldsymbol{c} \tag{3.78}$$

with $\boldsymbol{c} = \bar{\boldsymbol{f}} - (A\bar{\boldsymbol{x}} + B\bar{\boldsymbol{u}})$ and A and B defined as in Eqs. (3.33) and (3.34) and evaluated on the reference solution. The full expressions of matrices A and B are reported in Appendix A.

The new control variables are not independent of each other. In fact, they must satisfy the following constraint

$$u_r^2 + u_\theta^2 + u_\varphi^2 = u_N^2 \tag{3.79}$$

The constraint in Eq. (3.79) is a nonlinear, nonconvex, equality constraint. However, its relaxation, attained by substituting the equality sign with the inequality sign

$$u_r^2 + u_\theta^2 + u_\varphi^2 \le u_N^2 \tag{3.80}$$

is convex and corresponds to a second-order cone constraint.

The convex relaxation defines a feasible set larger than the original one, but the optimal solution satisfies Eq. (3.80) with the equality sign, making the relaxation lossless. Intuitively, this can be understood by noting that the left-hand side represents the acceleration provided by the engine to the system, while the right-hand side is a measure of the propellant mass consumption, as the mass flow rate is proportional to u_N. So, if Eq. (3.80) is satisfied with the inequality sign, the acceleration is smaller than the maximum acceleration that the engine could provide for that value of u_N. Since the goal of the optimization is to minimize the propellant consumption, such a behavior will be automatically discarded by the solver, as only solutions that satisfy Eq. (3.80) is with the equality sign are optimal. The rigorous proof that this relaxation is *exact* is provided in Appendix B for the more general problem of the launch vehicle.

The thrust-magnitude constraint in Eq. (3.50) must be enforced in terms of $\boldsymbol{u}$. Specifically, the following nonlinear path constraint is added to the OCP

$$u_N \le T_{\max} e^{-z} \tag{3.81}$$

The introduction of Eq. (3.81) is the downside of applying this change of variables, as such a constraint is nonconvex and needs to be linearized. Nevertheless, the advantages of solving a control-affine problem outweigh the disadvantages related to the inclusion of this nonlinear constraint, as the overall complexity of the problem

Table 3.1. Low-thrust trajectory design problem data.

Quantity	**Value**	**Unit**
μ	1.3271×10^{11}	km^3/s^2
t_f	253	days
$T_{\max}$	0.55	N
c	32.373	km/s
$m(t_0)$	659.3	kg
$r(t_0)$	1	AU
$\theta(t_0)$	0	deg
$\phi(t_0)$	0	deg
$v_r(t_0)$	0	km/s
$v_\theta(t_0)$	29.784	km/s
$v_\phi(t_0)$	0	km/s
$r(t_f)$	1.52	AU
$\theta(t_f)$	180	deg
$\phi(t_f)$	1.85	deg
$v_r(t_f)$	0	km/s
$v_\theta(t_f)$	24.16	km/s
$v_\phi(t_f)$	0	km/s

is reduced. Equation (3.81) is replaced with its first-order Taylor series expansion around a reference solution, that is

$$u_N \leq T_{\max} e^{-\bar{z}}(1 - (z - \bar{z})) \tag{3.82}$$

To sum up, the following convex optimal control problem is formulated

$$\begin{aligned} \mathcal{P}_1: \ \min_{\boldsymbol{x},\,\boldsymbol{u}} \quad & (3.70) \\ \text{s.t.} \quad & (3.59), (3.60), (3.61), (3.78), (3.80), (3.82) \end{aligned} \tag{3.83}$$

Eventually, to obtain a SOCP program, $\mathcal{P}_1$ must be discretized. To this aim, a trapezoidal collocation scheme, as described in Section 2.3.2 can be employed. The discrete problem is then solved recursively until the convergence criteria Eqs. (3.45) and (3.46) are met. Note that, since no virtual buffers were introduced, the criterion in Eq. (3.47) is neglected.

3.8.4 Numerical Results

The data used to formulate the problem instance are reported in Table 3.1 and are the same used in Ref. [42]. Since normalization is a key point in all numerical solution approaches, in the present application, all the data are scaled by the initial radius, the corresponding circular velocity, and the initial mass of the spacecraft, which are used as nondimensionalization factors. The values used for the tolerances of the successive convexification algorithm are $\epsilon_{\text{tol}} = 10^{-3}$ and $\epsilon_f = 10^{-6}$. Note that no virtual controls or virtual buffers were introduced for this problem, as their use was deemed unnecessary. Also, filtering or trust regions were not implemented.

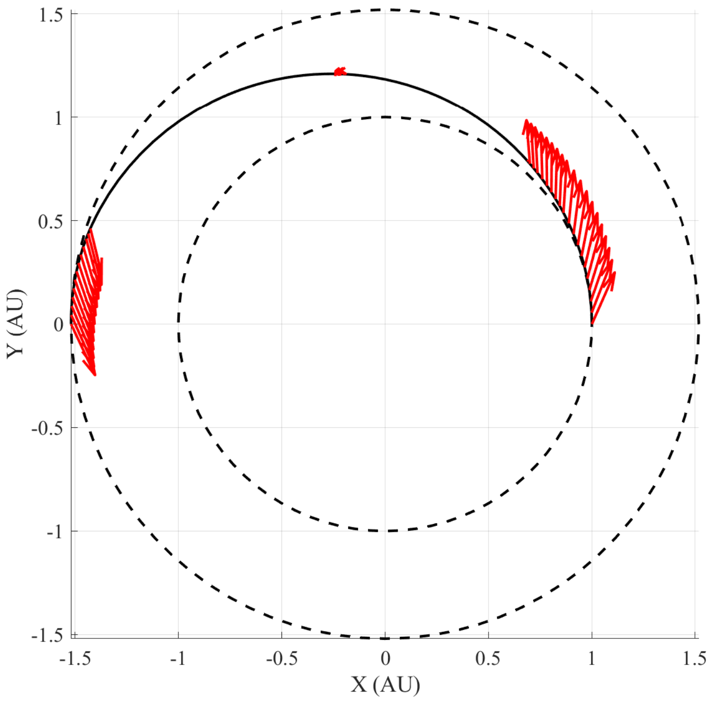
As for the discretization, a starting mesh of $M = 41$ equally spaced nodes in time was used to collocate the dynamics using the trapezoidal rule. Then, an adaptive mesh refinement process, as described in Section 2.3.2 was carried out to add nodes in the regions where the discretization error was above tolerance.

The starting reference solution was generated by forward propagation of the original equations of motion, Eqs. (3.51)–(3.57), starting from the prescribed initial state and assuming identically null thrust. Since at the initial time the spacecraft is supposed to be on the circular Earth's orbit, the first guess trajectory is simply a propagation of such orbit from t_0 to t_f.

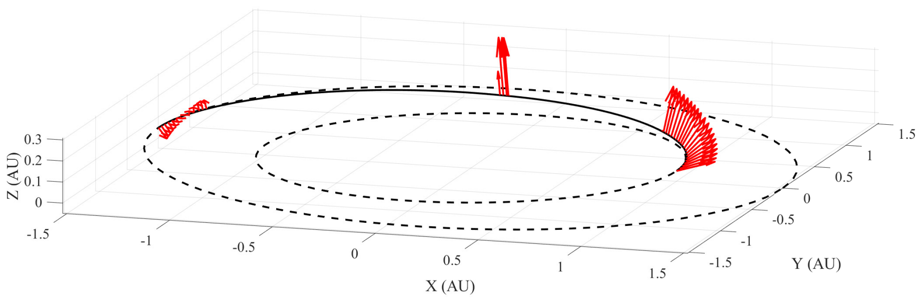
Figure 3.2 shows the optimal trajectory and the thrust direction. The thrust magnitude is reported in Fig. 3.3. The optimal control features a bang-off-bang structure, as usually happens when minimizing the propellant consumption. Specifically, there are two burn phases, at the start and at the end of the transfer, and a short burn at an intermediate time. The whole mass profile is plotted in Fig. 3.4. The overall propellant consumption is 127.45 kg that corresponds to a final mass of 531.85 kg. The results are in good agreement with Ref. [42], where the arrival mass is 530.33 kg. The slight difference (1.52 kg) is ascribable to a numerical error due to the different discretization grid, SOCP solver, or tolerances used.

Convergence was attained in 5 iterations in 0.1 s on the starting grid of 41 nodes. After assessing the discretization accuracy of said grid, a mesh refinement process was carried out. In particular, three refinement steps were needed to achieve a discretization error, estimated as in Eq. (2.24), below 10^{-6} in each segment, using a total of 85 unevenly spaced nodes. Figure 3.5 shows the sequence of used grids. In the first iteration, the new nodes (colored in blue) were added diffusely over the starting grid, except for the interval $t \in [150, 210]$ days that was deemed sufficiently accurate. This is due to the fact that the initial grid is quite coarse, so the discretization is inaccurate in most intervals. In the second and third iterations, fewer nodes are added and only in the neighborhood of the ignition or cut-off of the engine, as the grid is sufficiently accurate to discretize the continuous dynamics, but it needs greater density in the regions where the control is discontinuous.

Figure 3.6 reports the relaxation error due to replacing Eq. (3.79) with Eq. (3.80) to obtain a convex problem. In particular, the plot reports the absolute value of the difference between the left and right-hand side of the constraint. Since the error is always below the solver feasibility tolerance (that is, the maximum tolerated violation of equality constraints, set to 10^{-6}), the relaxation is exact.



(a) Two-dimensional view.



(b) Three-dimensional view.

Figure 3.2. Optimal low-thrust trajectory.

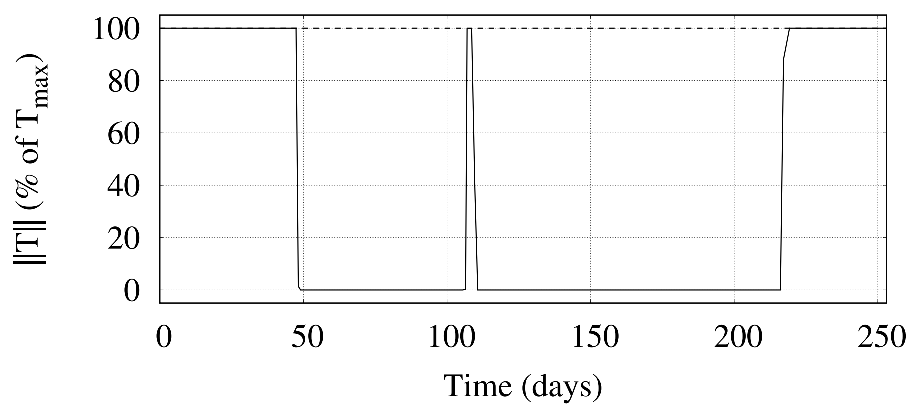


Figure 3.3. Optimal thrust profile.

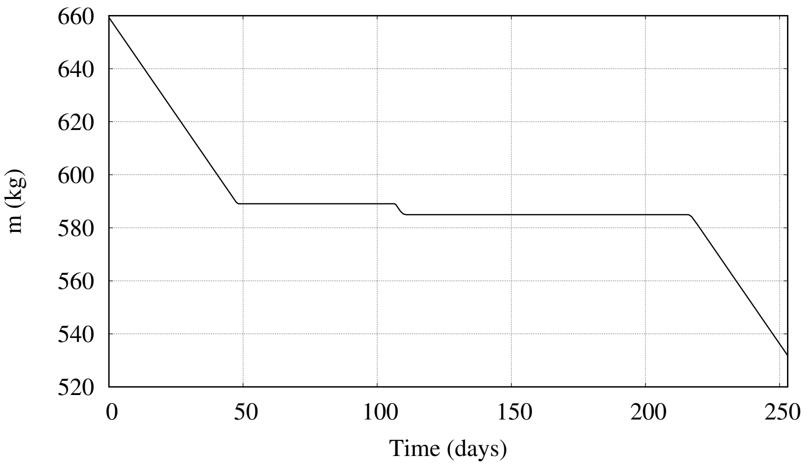


Figure 3.4. Spacecraft mass over time.

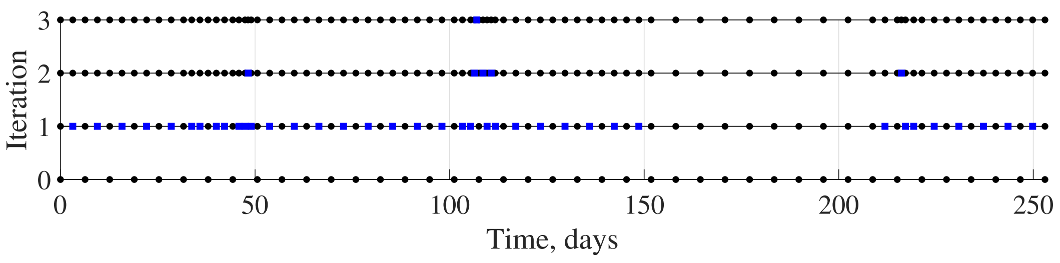


Figure 3.5. Evolution of the discretization grid due to the adaptive mesh refinement process.

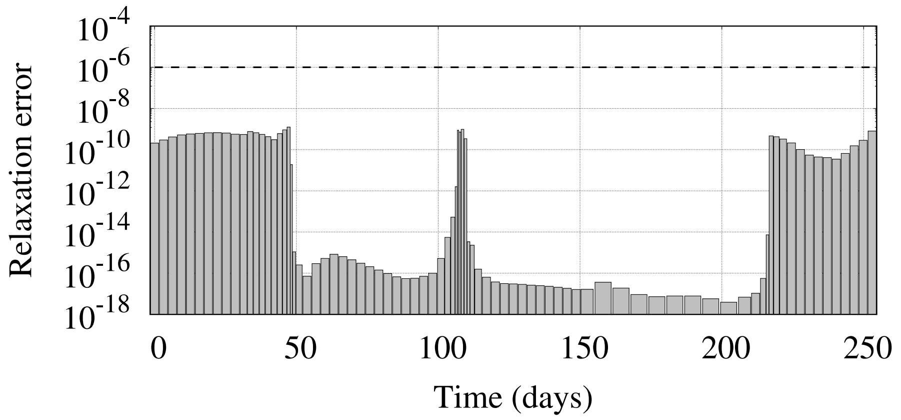


Figure 3.6. Control constraint relaxation error.

Chapter 4

Launch Vehicle Ascent Trajectory Optimization

This chapter introduces the launch vehicle ascent trajectory design problem. First, details on VEGA, which is the considered vehicle in this thesis, are provided. Second, the typical phases of a rocket ascent trajectory are described along with the specific flight strategy of a four-stage vehicle. Then, the considered dynamical model is presented, detailing the assumptions and the forces taken into account. The design of the ascent trajectory is then cast an optimal control problem, which is inherently nonconvex, and is thus later converted into a sequence of convex problems through state-of-the-art convexification methods. Finally, a simple way to initialize the optimization problem and a continuation strategy to reduce its sensitivity to inaccurate initial guesses are presented.

4.1 VEGA Launch Vehicle

In this thesis, the launch vehicle that is being considered is VEGA (acronym for *Vettore Europeo di Generazione Avanzata*). VEGA is an expendable four-stage launch system designed to send small satellites into low Earth orbit (LEO) and developed within a European Space Agency (ESA) program with the support of Italy, Belgium, the Netherlands, Spain, Sweden, Switzerland, and France.

The VEGA program started in the 1990's, when studies were carried on about the possibility of using Ariane solid booster technology to design a small launch vehicle that would complete the Ariane fleet. In fact, for many years, Ariane was Europe's only launch system and was used to guarantee access to space to European countries. This market alone could not sustain the availability and reliability of the service, so Ariane has evolved to meet the needs of the worldwide commercial market, where it has been extremely successful. In 2011, a new launch site was built in French Guiana for the Soyuz rocket in order to complete the performance range offered by Ariane. A comparison between the vehicles currently operated by ESA is illustrated in Fig. 4.1.

VEGA began as a national Italian project aimed at replacing the retired US Scout rocket. In 1998, after the Italian space agency proposed VEGA as an European project, it officially became an ESA project [143]. The launch vehicle prime contractor role was entrusted to the Italian company ELV S.p.A. (renamed Spacelab S.p.A. in 2018), owned jointly by Avio and the Italian space agency (ASI). The maiden flight occurred on 13 February 2012, followed by other 19 flights (counting only two failures).

Figure 4.1. Europe's launch vehicles: VEGA, Soyuz, Ariane 5-GS, and Ariane 5-ECA.

Figure 4.2 illustrates the stage configuration of VEGA. VEGA is one of the few four-stage launch vehicles. The first three stages are solid rocket motors (SRMs), named P80, Z23, and Z9, while the last stage is a small re-ignitable liquid rocket engine, AVUM (Attitude and Vernier Upper Module). The P80 first stage is very similar to the solid rocket boosters used on the Ariane 5. VEGA's second and third stages use Zefiro solid rocket motors, which are smaller both in diameter and in overall length than P80. AVUM has a bipropellant main propulsion system for orbit injection and a monopropellant propulsion system for attitude control. It offers a great flexibility to servicing a wide range of orbits and allows the launch vehicle to deliver payloads to different orbits in case of shared launch.

4.2 Phases

In general, the ascent trajectory of a launch vehicle can be divided into several phases. During each phase, the launch vehicle follows different guidance programs to meet specific mission requirements through the entire ascent. In addition, there are coasting phases that take place after the separation of each stage or between firings of the same engine.

4.2.1 General Phase Sequence

The general phase sequence of an ascent trajectory is illustrated in Fig. 4.3. At the initial time, the rocket stands vertical on the launch pad. After the liftoff, a vertical ascent phase begins since the rocket cannot perform any maneuver and it has to maintain the vertical attitude until it clears the pad and gets past the height of the launch tower. The duration of this phase depends on the launch site structure. For instance, the VEGA rocket lifts off from the Guiana Space Center, so it has to ascend 73 m before rotating.

Once the rocket has cleared the launch pad, it can start rotating through a

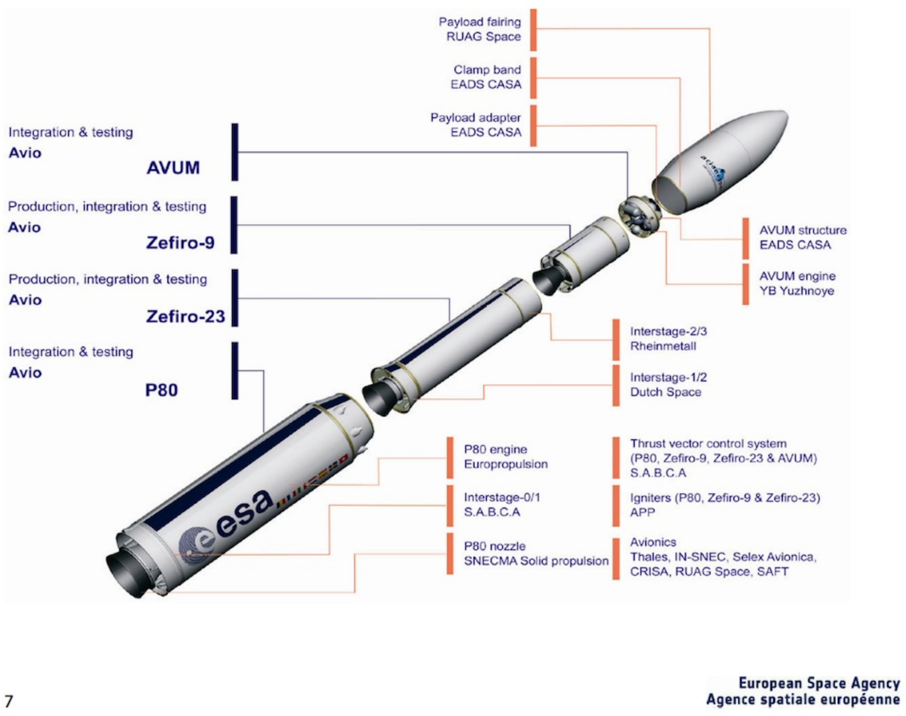


Figure 4.2. VEGA launch vehicle.

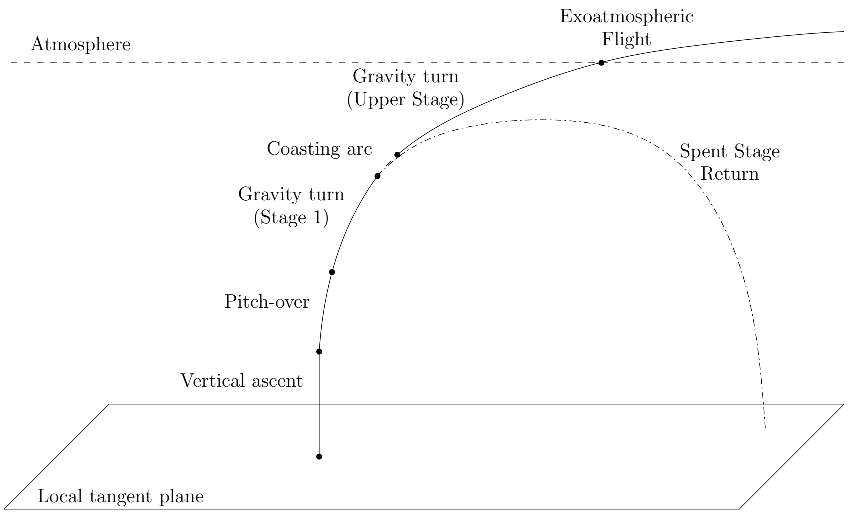


Figure 4.3. Phases of a typical launch vehicle ascent trajectory.

programmed rotation that takes place in a fixed vertical plane. This maneuver is called *pitch-over* and is necessary to start a progressive rotation of the rocket from the initial vertical orientation to the final quasi-horizontal orientation at the burnout of the last stage. The pitch-over lasts only 5 to 20 seconds, depending on the launch system, the target orbit, and other mission characteristics. During this maneuver, the rocket rotates from the vertical orientation to an elevation (i.e., the angle between the rocket axis and the local horizontal) that also depends on the considered mission and rocket. For certain launch vehicles a rotation of few degrees is sufficient, while others require larger programmed rotations. The elevation angle at the end of the pitch-over is usually called *kick angle* and it is an important optimization parameter since small changes in its value can greatly affect the ascent trajectory.

After the pitch-over, the rotation must continue, but, due to the presence of the atmosphere, the only practical way to rotate the launch vehicle is a *gravity turn* maneuver. Indeed, launch vehicles are structurally very stiff along their longitudinal axis, but weak along the lateral ones, so that even a small traversal force (i.e., lift) may cause structural failure. Thus, the gravity turn consists in maintaining a null angle of attack by aligning the rocket axis with the relative velocity. In this way, no lift is generated and the lateral load on the rocket is minimized. During the gravity turn, the gravitational acceleration progressively rotates the rocket velocity, as gravity, which acts in the radial direction, is no longer aligned with the thrust (i.e. with the rocket longitudinal axis).

Once the rocket has reached a sufficiently high altitude and the atmosphere is rarefied enough, the launch vehicle can follow a more efficient guidance law, as the lateral aerodynamic load is no longer a concern. Indeed, the gravity turn is not an optimal guidance program, but, rather, a necessary means to ascend through the atmosphere with a slender vehicle.

Every time a stage burns out, it must separate from the rocket and, after a minimum time interval that must elapse to ensure a safe separation, the following stage can ignite. So, during the time interval that immediately follows the separation, no thrust is generated and the trajectory is thus ballistic. These arcs are called *coasting phases*. Coasting phases do not occur only at stage separation, but also when a liquid rocket engine is turned off to be re-ignited later, as usually happens for upper stages. If the coasting phases take place at low altitudes, then their optimal duration corresponds exactly to the minimum value, in order to minimize the gravity losses, otherwise their time-length must be determined by the optimization process. Note that after each stage separation, a return phase also begins. Return arcs are ballistic phases just as the coasting ones, as the spent stage is a passive object subject only to gravity and the atmospheric forces. The difference between coasting arcs and return phases lies in the fact that return phases end when the stage hits the ground or splashes down in the sea.

4.2.2 VEGA's Phase Sequence

The considered phase sequence for a VEGA-like launch vehicle is illustrated schematically in Fig. 4.4 and represents the typical flight strategy of a four-stage configuration. Note that the phases are numbered progressively from 1 to 13 in chronological order, with the relevant exception of the return phase, which, despite being the 13th arc, chronologically starts at the burnout of the third stage, i.e., at the end of Phase 8, and takes place concurrently with Phases 9–12. Hereinafter, let $t_0^{(i)}$ and $t_f^{(i)}$ denote the initial and final time of the i-th phase. For the sake of simplicity, if no phase superscript is specified, then t_0 and t_f denote the liftoff time $t_0^{(1)}$ and the fourth

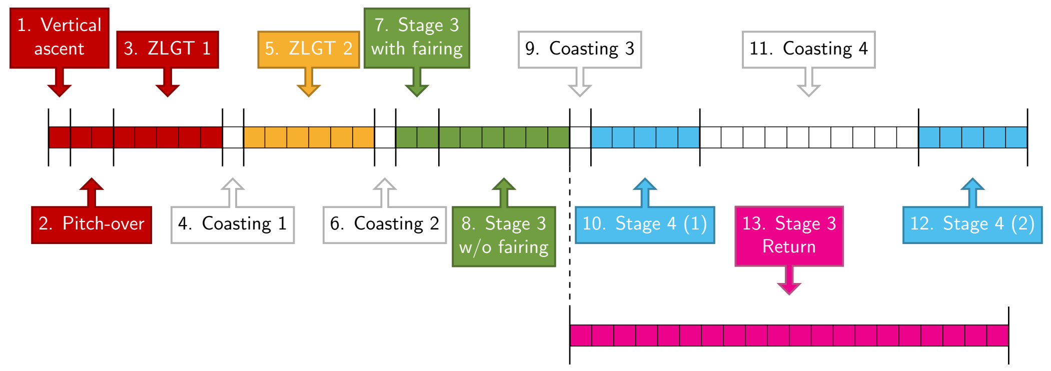


Figure 4.4. Phase sequence of a VEGA-like launch vehicle.

stage burnout $t_f^{(12)}$, respectively. Likewise, let $t_R = t_f^{(13)}$ denote the return time of the spent third stage.

The first stage ascent is divided into three phases to properly account for the different guidance programs, namely, vertical ascent (Phase 1), pitch-over (Phase 2), and gravity turn (Phase 3). The gravity turn maneuver continues for the entire second stage burn, so Phase 5 lasts for the whole operation of Z23. The third stage operates at sufficiently high altitudes and can adopt an optimal guidance program, as aerodynamic loads do not represent a concern anymore. Since also the thermal environment is less critical, during the third stage flight, the payload fairing is jettisoned. Specifically, the fairing is released after an assigned (small) amount of time that guarantees that the engine ignition transient is over and the vehicle attitude is fully controllable at the jettisoning. To efficiently handle the related mass discontinuity, the third stage operation is split into Phases 7 and 8 in correspondence of the jettisoning. VEGA's last stage performs a Hohmann-like maneuver and its flight is conveniently split into two burn arcs, Phases 10 and 12, separated by a coasting one, Phase 11. Note that three other brief coasting arcs (Phases 4, 6, and 9) are included at each stage separation. Finally, the return of the third stage is included as Phase 13 of the OCP. The return of the previous stages is neglected since their splash-down point is relatively close to the launch site and does not represent a concern.

4.3 System Dynamics

The complete description of the rocket dynamics comprises three translational and three rotational degrees of freedom. However, in the early phases of design, the trajectory of the center of mass of the vehicle is of greater interest than its attitude motion. Furthermore, the attitude of the launch vehicle only affects the trajectory in the first phases, while its effect on the remainder of the ascent is minimal. Therefore, in this study, rotational dynamics are neglected and the vehicle is modeled as a point mass, in the context of a three-degree-of-freedom problem.

There are several possible choices for the state variables. The set used in this study relies on the position vector $\boldsymbol{r}$, the inertial velocity vector $\boldsymbol{v}$, and the launch vehicle mass m. Thus, the state vector is

$$\boldsymbol{x} = \begin{bmatrix} x & y & z & v_x & v_y & v_z & m \end{bmatrix}^T \tag{4.1}$$

Note that the rocket position and velocity are expressed in Cartesian coordinates with respect to an Earth-Centered Inertial (ECI) reference frame. In particular, the x axis belongs to the Earth equatorial plane and passes through the meridian of the launch site at the initial time, the z axis is aligned with Earth's angular velocity, and the y axis completes the right-hand frame. The main advantage of using this set of state variables over, for example, a spherical coordinate system [134, 144], is the fact that any trajectory can be studied, even missions to polar and high inclination orbits, which are the typical targets for VEGA, without running into singularities. As a downside, when using Cartesian coordinates, the terminal conditions result in nonlinear expressions of the state variables.

The equations of motion for a non-lifting point mass in flight over a spherical rotating planet are the following

$$\dot{\boldsymbol{r}} = \boldsymbol{v} \tag{4.2}$$

$$\dot{\boldsymbol{v}} = \boldsymbol{g} + \frac{\boldsymbol{T}}{m} + \frac{\boldsymbol{D}}{m} \tag{4.3}$$

$$\dot{m} = -\dot{m}_e \tag{4.4}$$

Indeed, the only forces acting on the system are the gravitational acceleration $\boldsymbol{g}$, the engine thrust $\boldsymbol{T}$, and the atmospheric drag $\boldsymbol{D}$. Also, the mass of the system reduces over time due to the engine mass flow rate $\dot{m}_e$. In the following, each term in Eqs. (4.2)–(4.4) is detailed.

4.3.1 Gravity

Rocket ascent trajectories studies often use a simple gravitational model, as the effect of the gravitational perturbations on a launch vehicle is minimal and complex models are used only in the advanced phases of trajectory design. In this thesis, a basic gravitational model, which assumes a spherical Earth, has been considered. So, the gravitational acceleration is modeled as

$$\boldsymbol{g} = -\frac{\mu}{r^3}\boldsymbol{r} \tag{4.5}$$

where μ is Earth's gravitational parameter.

4.3.2 Atmosphere

The atmosphere plays a major role in the rocket ascent trajectory. Indeed, the choice of the atmospheric model radically affects the final results. An atmospheric model provides the values of density ρ, pressure p, and temperature Θ for a discrete set of altitudes h.

The most basic atmosphere model is the exponential one. The temperature is assumed to be constant, while pressure and density decrease by a factor e over a distance H. Thus,

$$\rho = \rho_0 e^{-\frac{h-h_0}{H}} \tag{4.6}$$

$$p = p_0 e^{-\frac{h-h_0}{H}} \tag{4.7}$$

ρ_0, p_0 and h_0 are reference values. H is called *scale height* and can be computed as

$$H = \frac{R\Theta}{\overline{m}g_0} \tag{4.8}$$

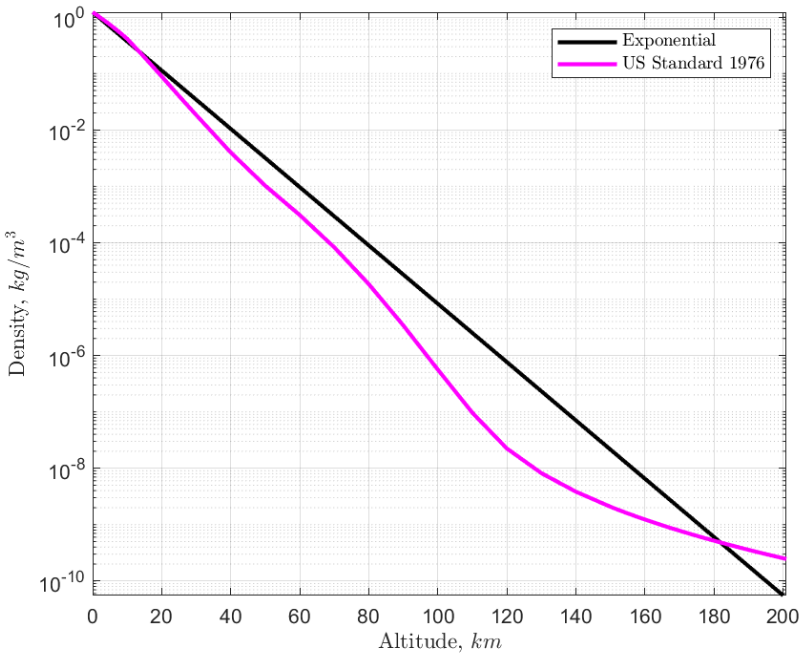


Figure 4.5. Atmospheric density.

where R is the gas constant and $\overline{m}$ is the mean molecular mass. H can be assumed constant over all altitudes or, in more accurate models, its value changes across different altitude ranges. The former approach is of great interest because it can be easily included in analytical solutions. However, it can be used only for problems limited to a narrow range of altitudes since, actually, the scale height is a function of altitude.

Because a launch vehicle altitude ranges from the sea level to orbital altitudes, a more accurate atmospheric model must be used. In this respect, the U.S. Standard Atmosphere 1976 model can be used to evaluate the air density and pressure as functions of the altitude [145]. The latter is compared to the exponential one (with $H = 8.4\,\text{km}$) in Figs. 4.5 and 4.6. The density values of the two models are very similar at lower altitudes (i.e., below 20 km), but at higher altitudes the standard model is characterized by significantly lower densities than the exponential one.

It is worth noting that some atmospheric models are provided only at some discrete altitudes; hence, the data must be interpolated. In particular, while temperature can be simply interpolated linearly, pressure and density should be interpolated exponentially (i.e., their logarithms are interpolated linearly), as this interpolation technique represents quite accurately the real trend of p and ρ even over large intervals of altitudes.

The atmosphere interacts with the rocket by means of the aerodynamic force. The aerodynamic force is proportional to the relative velocity $\boldsymbol{v}_{\text{rel}}$. If the atmosphere is assumed to be rotating along with the Earth, then the relative velocity is given by

$$\boldsymbol{v}_{\text{rel}} = \boldsymbol{v} - \boldsymbol{\omega_E} \times \boldsymbol{r} \tag{4.9}$$

where $\boldsymbol{\omega_E}$ is Earth's angular velocity.

The aerodynamic force component parallel to the relative motion of the rocket is

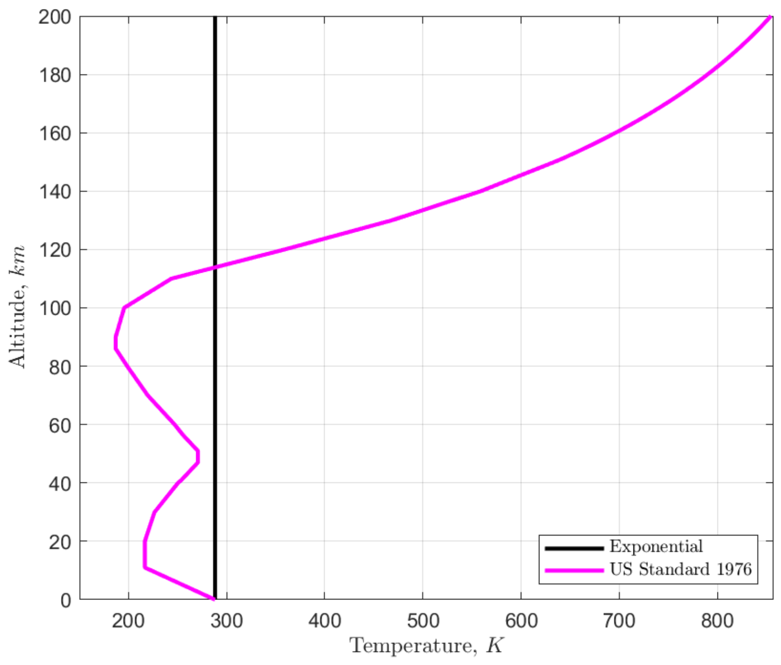


Figure 4.6. Atmospheric temperature.

the drag force D, which is computed as

$$\boldsymbol{D} = -\frac{1}{2} C_D S_{\text{ref}} \rho v_{\text{rel}} \boldsymbol{v}_{\text{rel}} \tag{4.10}$$

where C_D is the drag coefficient, S_{ref} is the reference surface, and ρ is the atmospheric density.

Actually, there is also a component of the aerodynamic force perpendicular to the relative velocity, that is, the lift. However, launch vehicle trajectories are designed to avoid high lateral loads, so the lift is much smaller than the drag and it can be neglected without loss of accuracy.

4.3.3 Propulsion

The last force acting on the launch vehicle is the rocket thrust $\boldsymbol{T}$, whose magnitude T depends on the engine and on the atmospheric conditions. In particular, the engine is characterized by a vacuum thrust T_{vac}, which, for a solid rocket motor, is a prescribed function of time that is unique for each engine. The actual thrust on the vehicle depends also on the external pressure p as

$$T = T_{\text{vac}} - pA_e \tag{4.11}$$

where A_e is the nozzle exit area.

While the thrust magnitude is prescribed by the motor characteristics and the atmospheric conditions, the thrust direction vector $\hat{\boldsymbol{T}}$ must be optimized and represents the control $\boldsymbol{u}$. Its elements are expressed in the ECI frame and, since $\hat{\boldsymbol{T}}$ is a unit vector, the following relationship must be satisfied

$$\hat{T}_x^2 + \hat{T}_y^2 + \hat{T}_z^2 = 1 \tag{4.12}$$

However, since the thrust direction $\hat{\boldsymbol{T}}$ can be optimized only in some propelled arcs, it may be convenient to fictitiously split the thrust magnitude T into three contributions to account for the different guidance programs. For instance, during the gravity turn maneuver (Phases 3 and 5) the rocket axis, which corresponds to the thrust direction in a three-degree-of-freedom model, must be aligned with the relative-to-atmosphere velocity $\boldsymbol{v}_{\text{rel}}$. Likewise, during the liftoff (Phase 1) the thrust must be aligned with the local vertical $\hat{\boldsymbol{r}}$. Thus, let T_a denote the optimally controlled thrust contribution, while T_b and T_c denote the thrust contribution during the gravity turn and vertical ascent, respectively. Only one of these three terms can be non-zero at a given time, depending on the flight phase. Thus,

$$T_a = \begin{cases} T_{\text{vac}} - pA_e & \text{in Phases 2, 7, 8, 10, and 12} \\ 0 & \text{in the other phases} \end{cases} \tag{4.13}$$

$$T_b = \begin{cases} T_{\text{vac}} - pA_e & \text{in Phases 3 and 5} \\ 0 & \text{in the other phases} \end{cases} \tag{4.14}$$

$$T_c = \begin{cases} T_{\text{vac}} - pA_e & \text{in Phase 1} \\ 0 & \text{in the other phases} \end{cases} \tag{4.15}$$

So, the thrust force is expressed as

$$\boldsymbol{T} = T_a \hat{\boldsymbol{T}} + T_b \hat{\boldsymbol{v}}_{\text{rel}} + T_c \hat{\boldsymbol{r}} \tag{4.16}$$

Note that T_b and T_c are multiplied by the relative velocity unit vector and the radial direction vector in Eq. (4.19), to constrain the thrust vector in the gravity turn and vertical ascent directions.

The rocket propulsion is characterized by a continuous propellant mass ejection. The mass flow rate $\dot{m}_e$ is a function of time and depends on the engine. In general, it is related to the thrust magnitude by the specific impulse I_{sp} as

$$\dot{m}_e = \frac{T}{g_0 I_{sp}} \tag{4.17}$$

where g_0 is the gravity acceleration at sea level.

In this thesis, the vacuum thrust T_{vac} and mass flow rate $\dot{m}_e$ time laws of the solid rocket motors are prescribed and modeled as linear functions of time. Instead, AVUM is assumed to generate a constant vacuum thrust and mass flow rate when in operation.

4.3.4 Equations of Motion

The resulting equations of motion are

$$\dot{\boldsymbol{r}} = \boldsymbol{v} \tag{4.18}$$

$$\dot{\boldsymbol{v}} = -\frac{\mu}{r^3}\boldsymbol{r} + \frac{T_a}{m}\hat{\boldsymbol{T}} + \frac{T_b - D}{m}\hat{\boldsymbol{v}}_{\text{rel}} + \frac{T_c}{m}\hat{\boldsymbol{r}} \tag{4.19}$$

$$\dot{m} = -\dot{m}_e \tag{4.20}$$

4.4 Optimal Control Problem

The design of a launch vehicle ascent trajectory is generally carried out by solving an optimal control problem. The goal of the optimization is to find the control law and other mission parameters, such as the duration of free-time arcs, that maximize the payload mass injected into the target orbit.

Besides the payload maximization, the launch vehicle trajectory has to meet several mission requirements that, in an optimal control problem, are expressed as constraints, which are transcribed as differential, boundary, and path constraints. The differential constraints are associated with the equations of motion, Eqs. (4.18)–(4.20), and are transcribed into a set of algebraic constraints via a collocation method, as described in Section 2.3.2. The boundary conditions include the initial, terminal, and linkage constraints. Finally, the path constraints include some aero-mechanical requirements that the trajectory must meet over arcs of finite duration to ensure the integrity of the rocket and payload.

In this section, the assumptions, the objective function, and the constraints that make up the ascent optimal control problem are listed and described.

4.4.1 Objective Function

Let $m_{p,i}$ and $m_{\text{dry},i}$ denote the propellant and inert masses of the i-th stages, for $i = 1, \ldots, 4$. If the propellant and inert masses are assumed to be assigned, one can equivalently decide to maximize the final mass, since it differs from the payload mass by a constant value. Let the OCP be cast as a minimum problem, then the cost function J to minimize is

$$J = -m(t_f) \tag{4.21}$$

4.4.2 Duration of the Phases

In general, the duration of the phases should be optimized in the trajectory design process. However, there are some considerations that constrain (or fix) the duration of some arcs. For instance, the vertical ascent duration is fixed since it should be minimized to reduce the gravitational losses, thus its optimal value is known *a priori* and corresponds to the time required to clear the pad, which depends on the launch site and engine characteristics.

The time-length of the pitch-over maneuver is, in general, another optimization parameter, but, since at the end of the pitch-over the angle of attack must be null to initiate the gravity turn, its duration is fixed to a value that guarantees a small angle of attack at $t_f^{(2)}$. Another approach would consist in including the additional constraint $\alpha(t_f^{(2)}) = 0$ and leaving the pitch-over duration as a decision variable, but this would increase the complexity of the problem and result, at most, in a very slight increase of performance, as the duration of this phase does not affect significantly the carrying capacity of the launch vehicle. The time-lengths of the gravity turn arcs (Phases 3 and 5) are also fixed since they are prescribed by the (fixed) burn time of the first and second stage.

The duration of the third stage flight before jettisoning the payload fairing should be minimized to dispose of the inert mass as soon as possible. Therefore, the time-length of Phase 7 corresponds to the minimum time to ensure full controllability of rocket attitude after the engine ignition transient. As a result, the duration of Phase 8 is also fixed and prescribed by the overall engine burn time. As for AVUM, the overall burn time is assigned, but the subdivision of the firings (Phases 10 and 12) is left to be optimized.

Lastly, the duration of the coasting arcs (Phases 4, 6, 9, and 11) is free in general. However, atmospheric coastings should be minimized in length, thus the duration of Phase 4 is fixed to the minimum value that guarantees a safe separation of Stage 1. It is worth remarking that the duration of the last coasting (Phase 11) has a much greater impact on the performance of VEGA than the previous ones. Thus, in this thesis, the duration of Phases 6 and 9 is fixed, as changes in these variables, which are anyways constrained in narrow ranges due to visibility requirements, do not affect significantly the overall performance [136].

4.4.3 Initial Conditions

The vehicle initial position corresponds to the launch base location at liftoff $\boldsymbol{r}_{\mathrm{LB}}$ and its velocity is equal to the eastward inertial velocity due to Earth rotation. Thus, the following initial conditions must be enforced

$$\boldsymbol{r}(t_0) = \boldsymbol{r}_{\mathrm{LB}} \tag{4.22}$$

$$\boldsymbol{v}(t_0) = \boldsymbol{\omega}_E \times \boldsymbol{r}_{\mathrm{LB}} \tag{4.23}$$

Instead, the initial mass of the system is free to be optimized, as it depends on the payload mass, which is the objective to maximize.

4.4.4 Target Orbit

At the burnout of the last stage, the payload and AVUM must be on the target orbit. An orbit is completely defined by a set of six parameters. These can either be the position and velocity vectors at a certain time or the set of classical orbital elements (COEs). For the sake of simplicity, in this thesis, a circular target orbit is considered; so, fewer parameters need to be imposed, namely, the semi-major axis a, the (null) eccentricity e, and the inclination i. Indeed, any requirement on the right ascension of the ascending node Ω can be easily met by selecting correctly the time window of the launch. As for the argument of the periapsis ω, its value is indifferent, since ω is undefined for circular orbits.

If one considers a circular orbit of prescribed radius r_{des} and inclination i_{des} at t_f, then the terminal conditions are

$$x(t_f)^2 + y(t_f)^2 + z(t_f)^2 = r_{\mathrm{des}}^2 \tag{4.24}$$

$$v_x(t_f)^2 + v_y(t_f)^2 + v_z(t_f)^2 = \mu/r_{\mathrm{des}} \tag{4.25}$$

$$\boldsymbol{r}(t_f) \cdot \boldsymbol{v}(t_f) = 0 \tag{4.26}$$

$$x(t_f)v_y(t_f) - y(t_f)v_x(t_f) = h_{z,\mathrm{des}} \tag{4.27}$$

Equations (4.24) and (4.25) constrain the semi-major axis of the final orbit to be r_{des}. Equation (4.26) guarantees that the radial velocity is null at payload release by imposing that the scalar product of position and velocity is null. Thus, combined with Eqs. (4.24) and (4.25), Eq. (4.26) ensures that the final orbit is circular. Finally, Eq. (4.27) derives from the expression of the inclination in ECI coordinates, that is

$$i = \cos^{-1}\left(\frac{xv_y - yv_x}{h}\right) \tag{4.28}$$

Indeed, since the angular momentum h of the target orbit is known and equal to $\sqrt{\mu r_{\mathrm{des}}}$, Eq. (4.28) can be conveniently expressed as in Eq. (4.27), with

$$h_{z,\mathrm{des}} = \cos i_{\mathrm{des}} \sqrt{\mu r_{\mathrm{des}}}. \tag{4.29}$$

4.4.5 Stage Separation

In general, all state variables are continuous at the phase boundaries. However, for a multi-stage rocket, whenever a stage burns out, it separates from the rocket. Thus, the rocket mass instantaneously reduced by the dry mass of the spent stage. These mass discontinuities are accounted for as linkage constraints between the last propelled arc of that stage and the following coasting phase. Thus,

$$m(t_0^{(4)}) = m(t_f^{(3)}) - m_{\text{dry},1} \tag{4.30}$$

$$m(t_0^{(6)}) = m(t_f^{(5)}) - m_{\text{dry},2} \tag{4.31}$$

$$m(t_0^{(9)}) = m(t_f^{(8)}) - m_{\text{dry},3} \tag{4.32}$$

Likewise, another mass discontinuity occurs at the fairing jettisoning, which results in the following constraint

$$m(t_0^{(8)}) = m(t_f^{(7)}) - m_{\text{fairing}} \tag{4.33}$$

4.4.6 Liquid Rocket Engine Burn Time

As mentioned above, the final stage burn is partitioned in two arcs (Phases 10 and 12). Since we assumed that $m_{p,4}$ is fixed and all the propellant must be consumed, the sum of the time-lengths of the two firings must be equal to the overall stage burn time $t_{b,4}$. Thus,

$$\Delta t^{(10)} + \Delta t^{(12)} = t_{b,4} \tag{4.34}$$

where $\Delta t^{(i)} = t_f^{(i)} - t_0^{(i)}$.

4.4.7 Heat Flux

The high relative-to-the-atmosphere velocity produces an intense thermal flux on the rocket. The payload is protected by the *fairing* during the initial phases. However, once the atmospheric density is sufficiently low, the fairing should be jettisoned in order to reduce the inert mass as soon as possible. As a consequence, the payload is directly exposed to the heat flux, which must not exceed a given value. Thus, the following path constraint is included in the formulation for Phases 8–12

$$\dot{Q} = \frac{1}{2}\rho v_{\text{rel}}^3 \leq \dot{Q}_{\text{max}} \tag{4.35}$$

In line with VEGA's user manual, the heat flux is evaluated according to a simple model involving a free molecular flow acting on a plane surface perpendicular to the relative velocity [58]. For the VEGA launcher, the maximum heat flux tolerated is $\dot{Q}_{\text{max}} = 900\,\text{W/m}^2$.

4.4.8 Maximum Dynamic Pressure

Launch vehicles are not designed to withstand high lateral loads. Therefore, it is important to ensure that, during the ascent, the aerodynamic force does not produce an excessive lift. This constraint can be expressed as

$$q\alpha \leq (q\alpha)_{max} \tag{4.36}$$

where α is the angle of attack and q is the dynamic pressure, which is evaluated as

$$q = \frac{1}{2}\rho v_{\text{rel}}^2 \tag{4.37}$$

The lateral load is actually critical only during the pitch-over maneuver, when α increases and the atmospheric density is high.

The dynamic pressure is an important quantity to evaluate during the trajectory design because the aerodynamic structural loads are proportional to it. For example, consider the zero-lift gravity turn: the nominal angle of attack is null, but during a real ascent the launch vehicle may fail to keep α to zero due to control inaccuracy or unexpected wind. To guarantee always an acceptable lift, a more conservative condition is imposed on the dynamic pressure q, which must remain below a safe threshold

$$q \leq q_{max} \tag{4.38}$$

This constraint extends to the whole trajectory. The point of maximum dynamic pressure is referred to as *max-q* and is one of the most critical load conditions for a launch vehicle.

In this thesis, Eqs. (4.36) and (4.38) are not enforced as explicit constraints in the optimization process, but are checked *a posteriori* on the attained solution. Indeed, the maximum dynamic pressure mainly depends on the first stage thrust profile, which, instead, is assumed to be prescribed in the present study. If the thrust magnitude were an optimization variable, then the inclusion of the constraints on the maximum dynamic pressure would have been mandatory.

4.4.9 Splash-Down Constraint

After the burn out and separation, the spent stages fall down to Earth. During the trajectory design process, it is essential to predict the fallout region of each stage and ensure that it is a safe zone for reentry. In this study, only the return of the third stage is explicitly accounted for in the optimization, as it is the most concerning one and its landing point requires to be actively constrained.

In fact, first and second stage have a relatively small final velocity and end up falling fairly close to the launch site, so the safety constraint reduces to a simple bound on the launch azimuth range. The fourth (and last) stage releases the payload on the target orbit, so, at the burnout, it is in orbit itself and its reentry can be controlled afterwards; after a certain time delay needed to provide a safe distance between the upper stage and the separated spacecraft, AVUM performs an additional burn for a controlled reentry into the Earth's atmosphere. Instead, VEGA's third stage burns out at a velocity that is close to (but lower than) the circular orbit velocity. As a result, it falls far away from the launch site.In this case, the inclusion of Phase 13 is necessary to simulate the return trajectory of the spent stage and predict the impact point, which must be constrained to a safe spot. The latter is generally called a *splash-down* constraint [29, 30].

The initial conditions of the return phase correspond to the conditions at the third stage burnout, so the following linkage constraints must be enforced

$$\boldsymbol{r}(t_0^{(13)}) = \boldsymbol{r}(t_f^{(8)}) \tag{4.39}$$

$$\boldsymbol{v}(t_0^{(13)}) = \boldsymbol{v}(t_f^{(8)}) \tag{4.40}$$

$$m(t_0^{(13)}) = m_{\text{dry},3} \tag{4.41}$$

Also, terminal conditions must be imposed on the return phase to ensure that the splash-down takes place in a safe area. Specifically,

$$x(t_R)^2 + y(t_R)^2 + z(t_R)^2 = R_E^2 \tag{4.42}$$
$$z(t_R) = R_E \sin \varphi_{R,\text{des}} \tag{4.43}$$

where R_E denotes the Earth radius. Equation (4.42) constrains the final altitude of the returned stage to be null, so that the return time t_R corresponds to the instant when the stage hits the ground or splashes down in the sea, while Eq. (4.43) constrains the splash-down location to a given latitude $\varphi_{R,\text{des}}$.

Note that, for missions toward polar or quasi-polar orbits (e.g., Sun-synchronous orbits), constraining the latitude is equivalent to constraining the splash-down distance from the launch base. Indeed, since the orbital plane of the trajectory is selected during the pitch-over maneuver and remains (almost) constant in the remainder of the ascent, as changing the plane later in the flight, at high velocities, would be too costly in terms of propellant, the only quantity that can be efficiently controlled is the downrange distance of the splash-down point from the launch base. This turns out to be a simple, yet effective, way to impose the splash-down constraint, as it consists in assigning just the final value of the z variable, and it well suits the VEGA target orbits, which are typically high inclination orbits.

4.5 Convex Formulation

In this section, the launch vehicle ascent trajectory optimization problem is formulated as a second-order cone programming (SOCP) problem. As detailed in Section 3.2.4, a SOCP problem is characterized by a linear objective, linear equality constraints, and second-order cone constraints. Since the dynamics and constraints outlined in Sections 4.3 and 4.4 do not generate a convex problem, the OCP is converted into an SOCP problem by applying several transformations. First, a convenient change of variables, which produces control-affine dynamics, is exploited. Second, a control constraint is relaxed into a second-order cone constraint and a proof of the lossless property of the relaxation is also provided under mild assumptions. The remaining nonconvexities are then tackled via successive linearization. Virtual controls, buffer zones, and other general convexification expedients are introduced in the formulation to prevent undesired phenomena due to the linearization, such as artificial infeasibility and unboundedness.

4.5.1 Change of Variables

The equations of motion Eqs. (4.18)–(4.20) are nonlinear in both state and control variables, and thus represent a source of nonconvexity. A successive linearization of these equations would produce linear constraints, but, due to the coupling of states and controls, high-frequency jitters would show up in the solution process, hindering its convergence [130]. To prevent this undesired behavior, a change of variables is exploited to obtain a control-affine dynamical system, following the same approach originally proposed by Açıkmeşe and Ploen in Ref. [33] and briefly outlined in Section 3.3.1. Thus, the new control variable is introduced

$$\boldsymbol{u} = \frac{T_a}{m}\hat{\boldsymbol{T}} \tag{4.44}$$

Note that $\boldsymbol{u}$ includes both the thrust-to-mass ratio T_a/m and the thrust direction $\hat{\boldsymbol{T}}$. Replacing the new control in Eqs. (4.18)–(4.20) provides the following dynamics

$$\dot{\boldsymbol{r}} = \boldsymbol{v} \tag{4.45}$$

$$\dot{\boldsymbol{v}} = -\frac{\mu}{r^3}\boldsymbol{r} + \boldsymbol{u} + \frac{T_b - D}{m}\hat{\boldsymbol{v}}_{\text{rel}} + \frac{T_c}{m}\hat{\boldsymbol{r}} \tag{4.46}$$

$$\dot{m} = -\dot{m}_e \tag{4.47}$$

So, the change of variables directly produces control-affine equations in the same form as in Eq. (3.26). Specifically,

$$\frac{d\boldsymbol{x}}{d\tau} = \hat{\boldsymbol{f}}(\boldsymbol{x}, \boldsymbol{p}, \tau) + B\boldsymbol{u}(\tau) \tag{4.48}$$

where the input matrix B is a constant matrix

$$B = \begin{bmatrix} \mathbf{0}_{3\times3} \\ \boldsymbol{I}_{3\times3} \\ \mathbf{0}_{1\times3} \end{bmatrix} \tag{4.49}$$

The new control variables must satisfy Eq. (4.12), which is reformulated as

$$u_x^2 + u_y^2 + u_z^2 = u_N^2 \tag{4.50}$$

where the auxiliary variable u_N represents the norm of the control vector $\boldsymbol{u}$ and corresponds to the magnitude of the thrust-to-mass ratio. Thus, the following constraint must be enforced

$$u_N = \frac{T_a}{m} \tag{4.51}$$

The introduction of this nonlinear path constraint is the price to pay to obtain control-affine dynamics. Nevertheless, the advantages of this change of variables outweigh the disadvantages, since, despite the inclusion of the additional constraint, the complexity of the problem is reduced compared to a coupling of state and control variables in the dynamics.

4.5.2 Constraint Relaxation

The path constraint (4.50) is a nonlinear equality constraint that must be convexified in order to be included in the SOCP formulation. Let us consider its relaxation attained by substituting the equality sign with the inequality sign,

$$u_x^2 + u_y^2 + u_z^2 \leq u_N^2 \tag{4.52}$$

Equation (4.52) is a convex constraint (specifically, a second-order cone constraint). The inequality sign allows the control variables to be located inside a sphere of radius u_N, rather than being constrained on its surface. Therefore, the convex relaxation defines a larger feasible set than the original one. Nevertheless, the following proposition ensures that, under mild assumptions, the resulting OCP has the same solution as the original problem. Note that the return phase can be temporarily removed from the optimal control problem, as, being an uncontrolled phase, it is not affected by the control constraint relaxation.

Assumption 1. Constraint (4.35) is assumed to be inactive almost everywhere[1] in $[t_0, t_f]$.

Remark 1. Assumption 1 states that the heat flux constraint is not active over finite intervals of the solution. This assumption holds almost always for the ascent problem, since typically the heat flux constraint is active only at isolated points in time, e.g., at the fairing jettisoning.

Proposition 1. *Let $\mathcal{P}_A$ be the launch vehicle ascent OCP:*

$$\begin{aligned} \mathcal{P}_A : \quad \min_{\boldsymbol{x},\, \boldsymbol{u},\, t_f} \quad & (4.21) \\ s.t. \quad & (4.22)\text{–}(4.27),\ (4.30)\text{–}(4.35),\ (4.45)\text{–}(4.47), \\ & (4.50),\ (4.51) \end{aligned} \tag{4.53}$$

Let $\mathcal{P}_R$ be the relaxed version of $\mathcal{P}_A$ obtained by substituting Eq. (4.50) *with Eq.* (4.52)*, that is:*

$$\begin{aligned} \mathcal{P}_R : \quad \min_{\boldsymbol{x},\, \boldsymbol{u},\, t_f} \quad & (4.21) \\ s.t. \quad & (4.22)\text{–}(4.27),\ (4.30)\text{–}(4.35),\ (4.45)\text{–}(4.47), \\ & (4.51),\ (4.52) \end{aligned} \tag{4.54}$$

The solution of the relaxed problem $\mathcal{P}_R$ is the same as the solution of $\mathcal{P}_A$. That is, if $\{\boldsymbol{x}^\star; \boldsymbol{u}^\star; t_f^\star\}$ is a solution of $\mathcal{P}_R$, then it is also a solution of $\mathcal{P}_A$ and $u_x^\star(t)^2 + u_y^\star(t)^2 + u_z^\star(t)^2 = u_N^\star(t)^2$ a.e. in $[t_0, t_f^\star]$.

Proof. See Appendix B. □

The proof of Proposition 1 is provided in Appendix B and follows the same reasoning as in Refs. [134, 146]. The intuition that motivates the relaxation is the same as in the seminal work on the planetary landing problem by Açıkmeşe and Ploen [33], later extended to a broader class of problems [32]. When Eq. (4.52) is strictly satisfied, the engine does not provide the maximum attainable acceleration to the rocket and, since the goal of the optimization is to maximize the mass injected into a target orbit, it is apparent that such a behavior is suboptimal, and thus will be automatically discarded by the solution procedure. Finally, note that this relaxation improves the convergence properties of the successive convexification algorithm compared to a linearization of the constraint in Eq. (4.50), since it introduces no approximation and fully preserves the nonlinearity of the original problem. The benefits of performing a change of variables and relaxing the control constraint have also been recently investigated and compared to direct linearization by Yang and Liu [34], showing that keeping some nonlinearity in the convex problem through the relaxation technique and then linearizing a control-affine dynamics achieves better convergence properties.

4.5.3 Successive Linearization

Successive linearization is employed to handle the nonconvexities that cannot be tackled via lossless convexification. In particular, the nonconvex constraints are replaced with the first-order Taylor series expansion around a reference solution, which is recursively updated until convergence is reached.

[1]A condition satisfied almost everywhere (a.e.) means that it can be violated only at a finite number of points (a set of measure zero).

Equations of Motion

The equations of motion Eqs. (4.45)–(4.47) are control-affine but still nonlinear in the state variables, thus they must be linearized as described in Section 3.3.2. So, the dynamics can be written in the same form as Eq. (3.32),

$$\frac{d\boldsymbol{x}}{d\tau} = A\boldsymbol{x} + B\boldsymbol{u} + P\boldsymbol{p} + \boldsymbol{c} \tag{4.55}$$

with $\boldsymbol{x}$ as in Eq. (4.1), $\boldsymbol{u}$ as in Eq. (4.44), and $\boldsymbol{p}$ being a one-dimensional vector that contains the time-length Δt of each phase:

$$\boldsymbol{p}^{(i)} = \left[\Delta t^{(i)}\right] \tag{4.56}$$

In this way, the duration of the free-time arcs can be included as additional variable and optimized. Indeed, Δt corresponds to the time dilation σ previously defined in Eq. (2.17). So, the two symbols will be used interchangeably in the following.

Target Orbit

All terminal conditions at payload release, Eqs. (4.24)–(4.27), are nonlinear in the state variables and must be linearized as

$$\bar{\boldsymbol{r}}(t_f) \cdot \bar{\boldsymbol{r}}(t_f) + 2\bar{\boldsymbol{r}}(t_f) \cdot (\boldsymbol{r}(t_f) - \bar{\boldsymbol{r}}(t_f)) = r_{\text{des}}^2 \tag{4.57}$$

$$\bar{\boldsymbol{v}}(t_f) \cdot \bar{\boldsymbol{v}}(t_f) + 2\bar{\boldsymbol{v}}(t_f) \cdot (\boldsymbol{v}(t_f) - \bar{\boldsymbol{v}}(t_f)) = \mu / r_{\text{des}} \tag{4.58}$$

$$\bar{\boldsymbol{r}}(t_f) \cdot \bar{\boldsymbol{v}}(t_f) + \bar{\boldsymbol{v}}(t_f) \cdot (\boldsymbol{r}(t_f) - \bar{\boldsymbol{r}}(t_f)) + \bar{\boldsymbol{r}}(t_f) \cdot (\boldsymbol{v}(t_f) - \bar{\boldsymbol{v}}(t_f)) = 0 \tag{4.59}$$

$$\bar{v}_y(t_f)(x(t_f) - \bar{x}(t_f)) - \bar{v}_x(t_f)(y(t_f) - \bar{y}(t_f)) - \bar{y}(t_f)v_x(t_f) + \bar{x}(t_f)v_y(t_f) = h_{z,\text{des}} \tag{4.60}$$

Heat Flux

The heat flux constraint of Eq. (4.35) is nonlinear and must be linearized as

$$\dot{Q}(\bar{\boldsymbol{r}}, \bar{\boldsymbol{v}}) + \frac{\partial \dot{Q}}{\partial \boldsymbol{r}}(\bar{\boldsymbol{r}}, \bar{\boldsymbol{v}}) \cdot (\boldsymbol{r} - \bar{\boldsymbol{r}}) + \frac{\partial \dot{Q}}{\partial \boldsymbol{v}}(\bar{\boldsymbol{r}}, \bar{\boldsymbol{v}}) \cdot (\boldsymbol{v} - \bar{\boldsymbol{v}}) \le \dot{Q}_{\max} \tag{4.61}$$

where the partial derivatives of the thermal flux with respect to position and velocity are

$$\frac{\partial \dot{Q}}{\partial \boldsymbol{r}} = \frac{1}{2}\frac{d\rho}{d\boldsymbol{r}} v_{\text{rel}}^3 + \frac{3}{2}\rho v_{\text{rel}} \boldsymbol{\omega}_E \times \boldsymbol{v}_{\text{rel}} \tag{4.62}$$

$$\frac{\partial \dot{Q}}{\partial \boldsymbol{v}} = \frac{3}{2}\rho v_{\text{rel}} \boldsymbol{v}_{\text{rel}} \tag{4.63}$$

Return Terminal Altitude

Also the condition on the return final radius, Eq. (4.42), must be linearized

$$\bar{\boldsymbol{r}}(t_R) \cdot \bar{\boldsymbol{r}}(t_R) + 2\bar{\boldsymbol{r}}(t_R) \cdot (\boldsymbol{r}(t_R) - \bar{\boldsymbol{r}}(t_R)) = R_E^2 \tag{4.64}$$

Control Norm

The auxiliary control variable u_N must be equal to the thrust-to-mass ratio at every time and thus Eq. (4.51) represents a nonlinear path constraint that must be linearized as

$$u_N = \frac{T_{vac} - p(\bar{\boldsymbol{r}})A_e}{\bar{m}}\left(1 - \frac{m - \bar{m}}{\bar{m}}\right) - \frac{A_e}{\bar{m}}\frac{dp(\bar{\boldsymbol{r}})}{d\boldsymbol{r}} \cdot (\boldsymbol{r} - \bar{\boldsymbol{r}}) \tag{4.65}$$

4.5.4 Trust Region on Time-Lengths

Thanks to the change of variables of Eq. (4.44), the dynamics in Eqs. (4.45)–(4.47) are linear in the control variables. Thus, the A and B matrices in Eq. (4.55) do not depend on the reference solution control $\bar{\boldsymbol{u}}$. This provides enhanced robustness to the successive linearization sequence, as intermediate controls can change significantly among the first iterations [38]. However, the linearized dynamics are still function of the reference controls due to the matrix P. Nevertheless, when $\sigma = \bar{\sigma}$, Eq. (4.55) reduces to

$$\frac{d\boldsymbol{x}}{d\tau} = A\boldsymbol{x} + B\boldsymbol{u} + \tilde{\boldsymbol{c}} \tag{4.66}$$

where

$$\tilde{\boldsymbol{c}} = \bar{\sigma}\hat{\boldsymbol{f}}(\bar{\boldsymbol{x}}, \tau) - A\bar{\boldsymbol{x}} \tag{4.67}$$

For arcs of fixed duration, Eq. (4.66) automatically replaces Eq. (4.55), but the other arcs may suffer from unboundedness issues when σ diverges excessively from the reference value, and some expedient may be necessary to ensure convergence.

In the present application, only some phases exhibit an unstable behavior related to σ. Specifically, Phases 10, 11, and 12 need further safeguarding constraints on their duration. Note that since the sum of the time-lengths of Phases 10 and 12 is constrained via Eq. (4.34), there is no need to act on the duration of both phases, and a safeguarding expedient on only one of the two is sufficient to provide algorithmic robustness.

Therefore, a soft trust region constraint is imposed on the duration of Phases 11 and 12,

$$|\sigma^{(i)} - \bar{\sigma}^{(i)}| \leq \delta^{(i)} \qquad i = 11, 12 \tag{4.68}$$

The trust radii $\delta^{(i)}$ are additional optimization variables that are constrained in the interval $[0, \delta^{(i)}_{\max}]$. Generally, a suitable choice of the upper bound is usually somewhere between 1% and 10% of $\bar{\sigma}^{(i)}$.

Moreover, to further incentivize $\sigma \approx \bar{\sigma}$, the trust radii are included in the cost function as (slightly) penalized terms by introducing the penalty terms

$$J^{(i)}_{\delta} = \lambda^{(i)}_{\delta}\delta^{(i)} \qquad i = 11, 12 \tag{4.69}$$

where λ_δ are the penalty weights, which should be as small as possible in order not to shadow the originally intended objective and let the optimization autonomously determine the optimal arc time-lengths.

4.5.5 Virtual Controls and Buffers

Since the linearization of the dynamics may cause artificial infeasibility, a virtual control $\boldsymbol{q}$ is included in the equations of motion, as discussed in Section 3.4, to prevent this undesired phenomenon. Thus, the relaxed linearized dynamics are

$$\frac{d\boldsymbol{x}}{d\tau} = A\boldsymbol{x} + B\boldsymbol{u} + P\boldsymbol{p} + \boldsymbol{c} + \boldsymbol{q} \tag{4.70}$$

Also the linearization of the boundary and path constraints may generate artificial infeasibility, so virtual buffer zones are used to relax the linearized expressions and guarantee the feasibility of the convex problem. In particular, Eqs. (4.57)–(4.65) are grouped into a vector of constraints $\boldsymbol{\chi} = \mathbf{0}$ and then relaxed as $\boldsymbol{\chi} = \boldsymbol{w}$, where $\boldsymbol{w}$ are the virtual buffers.

The following penalty terms are defined to penalize the use of virtual variables

$$J_q = \lambda_q \sum_{i=1}^{N} \sum_{j=1}^{M^{(i)}+1} \sum_{k=1}^{n_x^{(i)}} |q_k(t_j)| \tag{4.71}$$

$$J_w = \lambda_w \sum_{i=1}^{n_w} |w_i| \tag{4.72}$$

where N, M, n_x, and n_w denote the number of phases, discretization segments, state variables, and buffered constraints, respectively.

4.5.6 Augmented Objective Function

The cost function of the convex problem includes, in addition to the final mass term introduced in Eq. (4.21), the trust region penalty terms defined in Eq. (4.69) and the virtual control and the virtual buffer zone penalties defined in Eqs. (3.39) and (3.40). Thus, the augmented objective function to minimize is

$$J = -m(t_f) + J_\delta^{(11)} + J_\delta^{(12)} + J_q + J_w \tag{4.73}$$

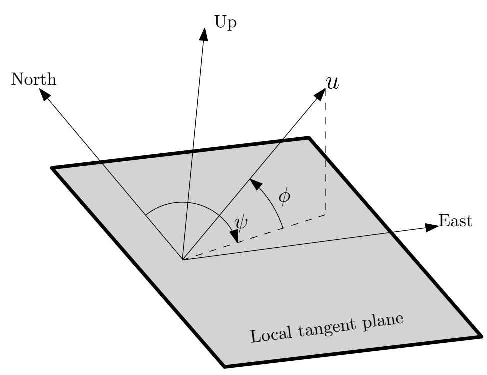
4.6 Initialization Strategy

This section outlines a simple procedure to design a initial reference trajectory that allows for convergence of the successive convexification algorithm.[2] Indeed, the convexification of the original problem nonlinear dynamics and constraints exploits successive linearization, which replaces the original expressions with a first-order Taylor series expansion around a reference solution $\{\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}}, \bar{\sigma}\}$. Therefore, an initial reference solution must be provided.

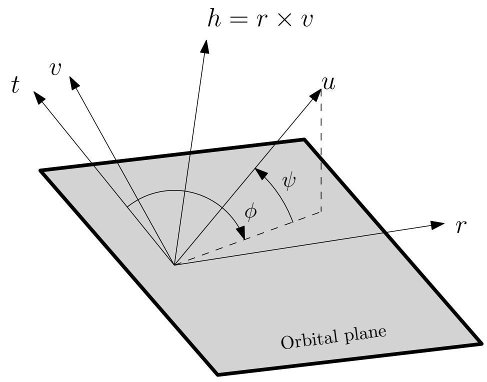
The design of the starting trajectory is a key step, as its choice affects the convergence of the algorithm and the quality of the attained solution. Indeed, sensitivity to the initialization is a major downside of traditional optimization methods. For instance, indirect methods can achieve convergence only if a very accurate first guess is provided. This is a cumbersome drawback as an initialization is required not only for the trajectory, but also for the costate, which often has an unclear physical meaning, and for the structure of the constrained arcs, which is also difficult to guess in general. On the other hand, direct methods exhibit greater robustness to the initialization, but the discretization of highly-sensitive nonconvex OCPs, such as the one at hand, produces a NLP problem whose solution depends significantly on the first guess. These limitations motivate the upstream effort put into the careful convexification process, as a greater robustness is observed in successive convexification algorithms compared to traditional direct optimization methods [134, 136].

The present algorithm does not require an accurate initialization. This is a particularly appealing property, as, in the preliminary phases of design of a launch

[2]A detailed analysis of the convergence of the successive convexification algorithm is provided in Chapter 5.



(a) Pitch-Over phase.



(b) Out-of-atmosphere phases.

Figure 4.7. Angles used to parametrize the control direction.

vehicle trajectory, it can be quite difficult to lay out an accurate ascent trajectory from scratch. Thanks to the convexification process, any starting trajectory with an altitude profile always above sea level is sufficient to achieve convergence. Such trajectories can be easily generated via numerical integration of the original rocket equations of motion, Eqs. (4.18)–(4.20). Only the (unknown) control law, the duration of free-time arcs, and the initial mass must be prescribed.

In general, designing adequate control laws may be a complex task. However, if the atmosphere is removed from the dynamics and considering a small initial mass $m(t_0)$ (i.e., a small payload mass), even trivial control laws can produce acceptable starting trajectories. The control laws are parameterized in terms of two angles: thrust elevation ϕ, which is the angle between the thrust direction and the local horizontal, and thrust azimuth ψ, which, during the pitch-over phase, is the angle measured clockwise from the north direction to the thrust vector and, during the out-of-atmosphere phases, is the angle between the thrust vector and the orbital plane (i.e., the plane where the position and velocity vectors lie). Figure 4.7 illustrates the elevation and azimuth angle.

To design the starting trajectory, the elevation during the pitch-over is assumed to vary linearly from 90° to a final value, the *kick* angle, here denoted as $\phi(t_f^{(2)})$. Since the pitch-over maneuver takes place in a fixed vertical plane, the thrust azimuth law is assumed constant during Phase 2, thus $\psi(t) = \psi^{(2)}$. The choice of the kick angle may be slightly difficult, as it depends on many factors, such as the specific launch vehicle configuration, the target orbit, and other mission parameters. However, a suitable kick angle value can be guessed by a trial-and-error process, using values in the range 70–89 degrees. Instead, there is a systematic way of choosing the value for the pitch-over azimuth. Indeed, the pitch-over rotation places the rocket on the correct heading for its ascent to orbit, since it is the most convenient phase to select the plane of the ascent trajectory due to the reduced velocity in the first seconds after liftoff. So, an accurate first guess for $\psi^{(2)}$ is the value that, under the non-rotating Earth assumption, allows for reaching the target orbit plane without further out-of-plane maneuvers, that is

$$\psi^{(2)} = \sin^{-1}\left(\frac{\cos(i_{\text{des}})}{\cos(\varphi_{LB})}\right) \tag{4.74}$$

where φ_{LB} is the latitude of the launch base. Equation (4.74) is exact only if assuming a non-rotating Earth and neglecting the presence of the atmosphere, but the azimuth angle that it provides represents a suitable guess since it is quite close to the optimal value of $\psi^{(2)}$. Unfortunately, when $i_{\text{des}} < \varphi_{LB}$ Eq. (4.74) does not hold anymore and an *ad hoc* value must be provided for $\psi^{(2)}$ through a trial-and-error process (generally starting from an eastward heading).

For stages 3 and 4, even simpler control laws can be assumed. Indeed, acceptable starting trajectories can be designed by assuming that the orbital plane is kept constant after the second stage burnout. For this reason, in the out-of-atmosphere phases, the thrust azimuth angle ψ is defined as a measure of the thrust out-of-plane component. So, ψ is assumed identically null during the flight of Z9 and AVUM. The only quantity left to define the control direction is the elevation angle ϕ, defined, as in the atmospheric phases, as the angle between the thrust and the local horizontal. In Fig. 4.7b, t denotes the unit vector in the orbital plane that is orthogonal to the radial direction. Since during the operation of upper stages the elevation angle is usually close to zero, an acceptable trajectory can be obtained by assuming $\phi(t) = 0$ in these phases. So, the only parameters to choose to simulate the flight of the upper stages concern the subdivision of the fourth stage burn and the duration of the intermediate coasting (Phase 11).

To sum up, the quantities necessary for generating an acceptable tentative solution are: (i) the initial mass $m(t_0)$ (or, equivalently, the payload mass), (ii) the kick angle $\phi(t_f^{(2)})$, and (iii) the time-lengths of Phases 10, 11, and 12. These values should be selected on the basis of the specific launch vehicle and target orbit. Nevertheless, their choice does not represent an arduous task, since a wide range of values can generate acceptable trajectories.

4.7 Continuation Strategy

Since the original problem is not convex, there is no guarantee that the successive convexification algorithm will converge to the global optimum of the original problem. So, special care should be paid to the initialization procedure. It is apparent that the first guess trajectory must be as close as possible to the optimal solution in order to favor convergence; unfortunately, this is not a trivial task for problems as complex as the one here investigated. The strategy described in Section 4.6 is a simple way to generate a reasonable starting trajectory, but it is still possible that only convergence toward a local optimum (with a significantly worse objective function) or no convergence at all could be achieved. Therefore, it is essential to minimize the sensitivity of the problem to the initial guess and maximize the convergence radius.

The standard way of dealing with the problem at hand is, first, solving the ascent problem without the return phases and the corresponding splash-down constraints. Then, once a solution has been obtained, the return of the spent stages is simulated and, if necessary, the optimization is repeated including the return phases and constraining the splash-downs to safe locations. Indeed, concerns on the splash-down of the spent stages exist only if the simulation of the return trajectory corresponds to an unsafe impact location. So, Phase 13 and the related constraints can be omitted at first, and the focus can be on designing a reference solution for Phases 1–12 only.

To reduce the sensitivity to an inaccurate initial reference solution, a sequential continuation procedure, which consists in solving problems of increasing complexity, can be used, as it significantly improves the convergence properties of the algorithm.[3] Three problem instances are considered:

[3]An analysis of the benefits of using a multi-step continuation procedure is presented in Chapter 5, based on numerical experiments.

1. time-fixed with no atmosphere $\mathcal{P}_1$,
2. time-fixed with atmosphere $\mathcal{P}_2$,
3. free-time with atmosphere $\mathcal{P}_3$.

The latter, $\mathcal{P}_3$, is the complete problem described in Section 4.5 (without the return phase and the relative constraints). $\mathcal{P}_2$ is a slightly easier problem than $\mathcal{P}_3$, as the time-lengths of the free-time phases are fixed to some reference values; so, there are no instability issues related to deviations of time variables σ from the reference values and there is no need to include the trust region constraints as in Eq. (4.68). Instead, $\mathcal{P}_1$ is a much easier problem since, not only the duration of all phases is fixed, but also the atmosphere is removed from the problem, thus reducing the nonlinearity (and nonconvexity) of the underlying problem.

Each problem $\mathcal{P}_k$ uses the solution of the previous one $\mathcal{P}_{k-1}$ as the initial guess. Hence, only the reference trajectory for $\mathcal{P}_1$ must be provided. However, thanks to the convexification process detailed in Section 4.5, $\mathcal{P}_1$ is quite robust to the choice of the initial reference trajectory (even though the same is not true for $\mathcal{P}_2$ and $\mathcal{P}_3$), which, therefore, can be chosen in a large set, without compromising the convergence of the optimization procedure.

It is worth noticing that the success of the multi-step procedure depends on the convergence of all problems, namely, $\mathcal{P}_1$, $\mathcal{P}_2$, and $\mathcal{P}_3$. The feasibility and existence of the solution of each problem is guaranteed by the inclusion of virtual controls and buffer zones. This, in turn, avoids that the solution process is ever prematurely interrupted, and thus allows for the success of the three-step procedure.

Chapter 5

Case Study: Ascent Toward a Low-Earth Polar Orbit

In this chapter, numerical results are presented to discuss the effectiveness of the successive convexification approach for the ascent trajectory optimization of a multistage launch vehicle to a Low-Earth Orbit (LEO). First, the convergence properties of the algorithm are examined, highlighting not only the convergence speed and the number of iterations, but also the quality of the converged solution, which is compared to the solution obtained by means of an unrelated optimization method. Second, the sensitivity of the algorithm to the starting (i.e., first guess) reference solution is investigated by means of a Monte Carlo campaign. Then, the accuracy and performance of the discretization method is assessed by comparing the errors on the final orbit elements and the run times of different meshes. Finally, a parametric study on different splash-down locations is presented to determine how the splash-down constraint affects the carrying capacity of the launch vehicle.

The described algorithm has been implemented in C++ using Gurobi [147] as SOCP solver. All the computational time measurements are relative to tests run on a computer equipped with Intel® Core™ i7-9700K CPU @ 3.60 GHz.

5.1 Problem Statement

The considered case study is a mission toward a 700 km circular polar Earth orbit ($i_{\text{des}} = 90°$), which is the reference target orbit for VEGA [58]. The vehicle is assumed to take off from the equator in correspondence of the Guiana Space Center meridian. Indeed, the Guiana Space Center is located only 500 km north of the equator, at a latitude of 5°.

The data used to model the VEGA-like launch vehicle are summarized in Table 5.1. The main assumption concerns the thrust magnitude and mass flow rate curves ($T(t)$ and $\dot{m}_e(t)$), which are approximated as linear functions of time. These linear laws were designed by retaining the total impulse, which is the quantity that most affects the overall trajectory, so the model of the launch vehicle is representative of the real system performance. Other design values include the fairing mass m_{fairing} (535.3 kg), the drag coefficient C_D (0.381), and the reference surface S (9.079 m^2). Although a realistic aerodynamic model would be needed to accurately predict the splash-down location, for this work, in a simplified manner, the same coefficients are used also for the ballistic return of the spent stage. Notwithstanding, the algorithm can extended to more realistic aerodynamic characterizations of the launch vehicle and stage return. Finally, the U.S. Standard Atmosphere 1976 model is used to evaluate the air density and pressure as functions of the altitude [145].

Table 5.1. VEGA-like rocket data.

Quantity	**Stage 1**	**Stage 2**	**Stage 3**	**Stage 4**	**Unit**
m_p	87 898	23 926	10 006	397	kg
m_{dry}	8417	2563	1326	813	kg
t_b	102.0	75.0	110.0	502.1	s
$T_{\text{vac}}(0)$	2827	1075	299	2.45	kN
$T_{\text{vac}}(t_b)$	1885	717	222	2.45	kN
$\dot{m}_e(0)$	1034	383	105	0.79	kg/s
$\dot{m}_e(t_b)$	689	255	77	0.79	kg/s
A_e	3.09	1.70	1.18	0.07	m^2

The values used for the time-lengths of the arcs of fixed duration are reported in Table 5.2. The duration of the remaining phases, instead, has to be optimized. The threshold on the bearable heat flux $\dot{Q}_{\max}$ is set to $900\,\text{W/m}^2$.

Table 5.2. Time-lengths of time-fixed arcs.

Phase	Δt (s)
1	4.1
2	6.6
3	91.3
4	6.6
5	75.0
6	37.3
7	5.4
8	104.6
9	15.4

Unless otherwise specified, the following values were used to set up the successive convexification algorithm. The default values for the penalty weights of the augmented objective function are $\lambda_\delta^{(11)} = \lambda_\delta^{(12)} = 10^{-4}$ and $\lambda_q = \lambda_w = 10^4$. The convergence and dynamics tolerances were set equal to $\epsilon_{\text{tol}} = 10^{-4}$ and $\epsilon_f = 10^{-6}$. Filtering was used to enhance the convergence properties of the algorithm by considering the weighted sum of $K = 3$ previous solutions. The default weights are $\alpha_1 = 6/\alpha_N$, $\alpha_2 = 3/\alpha_N$, and $\alpha_3 = 2/\alpha_N$ with $\alpha_N = 11$, so that the weight sum is unitary.

As for the discretization, a *hp*-pseudospectral method based on Radau collocation, as described in Section 2.3.2, was used to discretize the OCP. Table 5.3 reports the default values of h and p for every phase of the problem. The values have been devised in a heuristic way in order to meet the desired discretization accuracy. In particular, since Phases 1–12 are relatively brief and do not feature rapidly changing dynamics, no internal subdivision is necessary and h is simply set to 1. Instead, the return phase is split into 10 equally-spaced segments to accurately capture the reentry dynamics while avoiding the use of high-order approximating polynomials. In each segment, the same discretization order p is used.

Finally, all dimensional quantities are scaled in order to improve the numerical behavior of the algorithm. Specifically, the Earth radius, the corresponding circular

Table 5.3. Discretization segments, order, and nodes in each phase.

	Phase												
	1	2	3	4	5	6	7	8	9	10	11	12	13
h	1	1	1	1	1	1	1	1	1	1	1	1	10
p	5	5	17	5	19	14	5	19	9	19	19	19	10
Nodes	6	6	18	6	20	15	6	20	10	20	20	20	101

orbit velocity, and a reference mass of 10 000 kg are used as normalization factors in the optimization.

5.2 Convergence Behavior

In this section, the convergence properties of the successive convexification algorithm are discussed. To this aim, an example reference solution is designed, then the sequence of iterations is reported and commented. For the sake of simplicity, the return phase and the related variables and constraints are omitted from the OCP.

To set up the optimization, a starting reference solution must be provided. This can be handily generated as described in the Section 4.6. For instance, a reasonable reference trajectory can be obtained by using the parameters reported in Table 5.4. Note that a very small payload mass $m_{\mathrm{pl}} = 100$ kg was selected. Note that this mass is significantly lower than the nominal payload that the VEGA rocket can carry to a low-Earth polar orbit (about 1400 kg). This conservative choice allows to design an ascent trajectory that does not fall back to Earth even using trivial control laws in the forward propagation of the dynamics and a rough guess of the other mission parameters, as described in Section 4.6. The duration of Phase 10 is omitted in Table 5.4, as this value comes from Eq. (4.34), once $\Delta t^{(12)}$ and the overall burn time of the stage are assigned.

Table 5.4. Values used for the generation of the starting reference solution.

Quantity	Value	Unit
m_{pl}	100.0	kg
ϕ_{k}	80.0	deg
$\Delta t^{(11)}$	2500.0	s
$\Delta t^{(12)}$	200.0	s

The sequence of iterations obtained by seeding the algorithm with the aforementioned reference solution is illustrated in Fig. 5.1. The dashed black line denotes the initial trajectory, while the intermediate solutions correspond to the solid lines. The color of the solid lines transitions from red to green, according to the iteration number. In this case, the continuation strategy detailed in Section 4.7 was not implemented, so the reference solution was used to solve directly the complete problem, $\mathcal{P}_3$. The termination criteria were met after 22 iterations. This is a relatively short sequence of iterations, if considering that the initial reference solution is quite far from the optimal solution of the problem. The overall computational time is 12.8 s, that is, each iteration requires 0.58 s on average (without any specific code optimization).

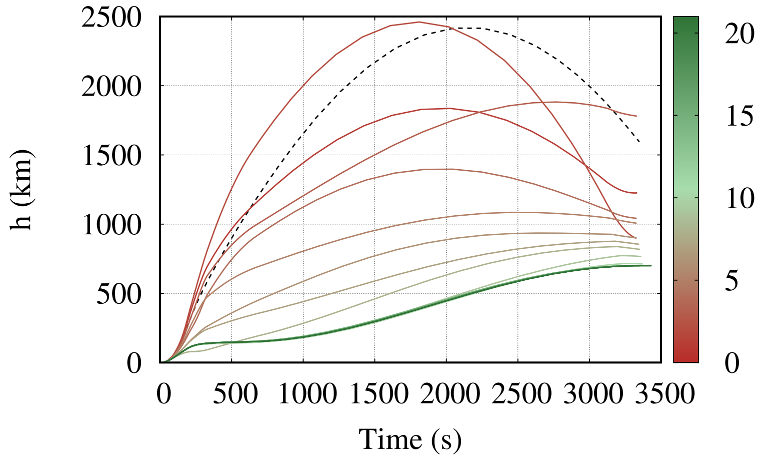


Figure 5.1. Iteration sequence starting from the initial reference solution (dashed black line) and transitioning from red (iter = 1) to green (iter = 22) until convergence.

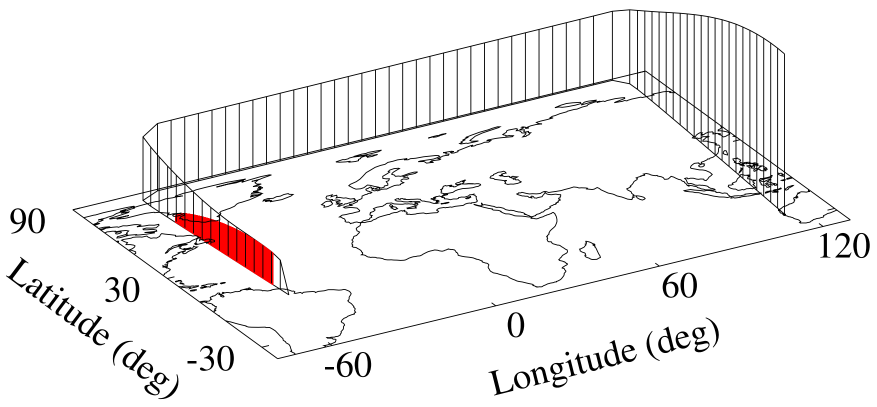
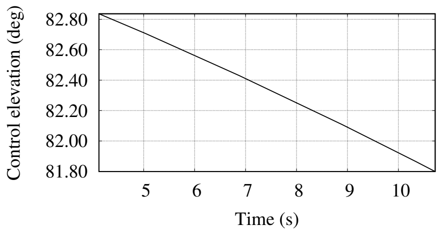


Figure 5.2. Three-dimensional visualization of the optimal trajectory with the simulation of third stage ballistic return colored in red.

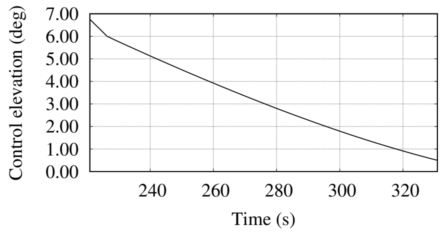
Virtual buffers are actively exploited in the first 10 iterations, as apparent in Fig. 5.1, where the value of the terminal altitude does not match the desired target orbit radius. It was observed that, without the virtual buffers, the intermediate problems would otherwise be infeasible (despite the inclusion of virtual controls). Thus, virtual buffers are essential to ensure the recursive feasibility of the sequential process.

The converged trajectory is illustrated in Fig. 5.2. The figure also reports a simulation of the return phase. This simulation provides a valuable piece of information, as the value of the splash-down latitude, $\varphi_R^\star = 65.79°$, of this unconstrained return corresponds to the optimal splash-down point (i.e., the point that maximizes the payload capacity of the system to the target orbit).

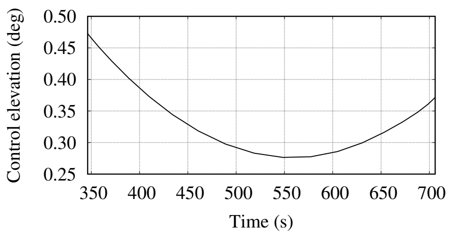
Figure 5.3 shows the optimal control laws of each controlled phase in terms of the elevation angle ϕ, as defined in Section 4.6. As for the pitch-over, one should constrain the initial elevation to 90° to ensure the control continuity with the vertical ascent phase. However, if using Cartesian coordinates, this would require adding a nonlinear constraint to the formulation, so, for the sake of simplicity, the initial



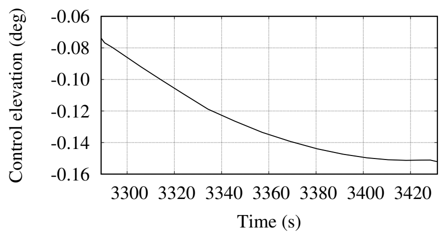
(a) Pitch-Over (Phase 2).



(b) Stage 3 (Phases 7-8).



(c) First burn of stage 4 (Phase 10).



(d) Second burn of stage 4 (Phase 12).

Figure 5.3. Control laws of the optimal solution.

Table 5.5. Optimal solution of the convex approach compared with the EOS solution.

Quantity	Convex	EOS	Unit	Difference (%)
m_{pl}	1400.73	1396.74	kg	0.29
$\Delta t^{(10)}$	359.71	357.60	s	0.59
$\Delta t^{(11)}$	2583.50	2660.70	s	2.90
$\Delta t^{(12)}$	142.39	144.50	s	1.46

pitch-over control direction is unconstrained in the present study. Nevertheless, the initial pitch-over elevation differs from 90° by only a few degrees. All the control laws are regular and very smooth, further proving the optimality of the attained solution.

The discretization accuracy of the converged solution was verified by forward propagation of the original equations of motion (4.2)–(4.4) using the optimal control laws. In particular, the discrepancies in the terminal conditions were inspected. The largest inaccuracy concerns the semi-major axis of the final orbit. However, the error is less than 50 m, which correspond to a relative error equal to 0.0008%. This inaccuracy is in agreement with the finite precision of the SOCP solver. A deeper analysis on the solution accuracy is presented in Section 5.4.

To validate the quality of attained results, the same problem was solved also using EOS [108], a direct shooting algorithm based on differential evolution that was already successfully employed to solve a similar instance of the problem at hand [115]. The comparison between the two solutions is reported in Table 5.5. The payload mass difference is approximately 5 kg and is due to the different sets of time-lengths found. Indeed, the problem features many local optima with different times but similar costs, so the optimization process can converge unpredictably toward one of these. Nevertheless, note that the difference in cost is minimal; so, both solutions are acceptable for any practical purpose. Compared to the convex approach, the main drawback of EOS is the large computational effort required (approximately 20 minutes on the same hardware).

Table 5.6. Optimal solution of the convex approach compared with the EOS solution with free coasting arcs.

Quantity	Convex	EOS	Unit	Difference (%)
m_{pl}	1400.84	1402.36	kg	0.11
$\Delta t^{(4)}$	6.60	6.60	s	<0.01
$\Delta t^{(6)}$	38.40	40.70	s	5.65
$\Delta t^{(9)}$	17.43	6.60	s	164.09
$\Delta t^{(10)}$	359.90	358.10	s	0.50
$\Delta t^{(11)}$	2582.24	2595.40	s	0.51
$\Delta t^{(12)}$	142.20	144.00	s	0.01

As a final remark, the assumption of fixing the duration of the coasting arcs (Phases 6 and 9 in particular) can be easily removed. Indeed, starting from the same reference solution discussed before (hence with the nominal coasting times from Table 5.2), if the OCP is solved with free time-length of Phases 4, 6, and 9 (with the only requirement of guaranteeing a minimum stage separation time of 6.6 s for safety reasons) a slightly improved solution is found, as reported in Table 5.6. The optimal coasting times are slightly different from the nominal values, with the relevant exception of the duration of the first coasting (Phase 4) that is the same as in the fixed-time case and corresponds to the lower bound. However, the change in the payload mass is very small (< 1 kg), thus showing that assuming fixed duration of the coasting arcs does not affect the launch vehicle performance. Also, convergence was attained in a number of iterations (24) comparable to the time-constrained case, showing that the success of the convex optimization method does not depend on this assumption and does not require additional trust regions. Finally, The solution found by the convex approach when freeing the duration of Phases 4, 6, and 9 is in good agreement with the one found by EOS when freeing the same variables, validating the quality of the attained solution (see Table 5.6). In the following, all coasting times, except for Phase 11, are prescribed to the nominal values of Table 5.2 for the sake of simplicity.

5.3 Sensitivity to the Initialization

This section investigates the sensitivity of the algorithm to the starting reference solution and assesses the ability of the algorithm to converge to the optimal solution even when starting from an inaccurate reference trajectory. In particular, the robustness of the algorithm when implementing the filtering technique for updating the reference solution (described in Section 3.6) is compared with the use of traditional methods to prevent artificial unboundedness, namely hard and soft trust regions, through a series of Monte Carlo campaigns. The effect of the numerical continuation strategy presented in Section 4.7 on the success rates of the filtering and trust region approaches is also discussed.

5.3.1 Filtering Approach

A study on the effectiveness of the filtering technique for updating the reference solution was carried out, to assess, in a systematic way, its robustness to a randomly sampled initial guess trajectory and the corresponding computational effort (measured in terms of the average run time). To this end, a Monte Carlo analysis was carried out, using different sets of weights. In addition to the aforementioned set of weights ($\boldsymbol{\alpha} = [6/11,\ 3/11,\ 2/11]$, referred to as *custom* in the following), that was devised in a heuristic way, several distributions of weights inspired by well-known sequences in mathematics were considered. Also, the effect of taking into account fewer or more terms in the weighted sum is investigated by considering different values of K, which denotes the number of previously found solutions used in the recursive update.

Table 5.7. Scattering range of the seeding parameters for the Monte Carlo analysis.

Parameter	**Lower bound**	**Upper bound**	**Unit**
m_{pl}	250	500	kg
ϕ_{k}	75	80	deg
Δt_{11}	2000	2800	s
Δt_{12}	100	300	s

Each Monte Carlo campaign consisted of generating a set of 2000 random initial trajectories as described in Section 4.6 by sampling the initialization parameters in the ranges reported in Table 5.7. Note that the values of the payload mass used for designing the starting trajectories are larger than the one used in Section 5.2. This is due to the fact that lower values (e.g., below 250 kg) were found to generate first guess trajectories for which the algorithm often fails to converge, regardless of the adopted set of weights. The ranges in Table 5.7, instead, are the ones that maximize the convergence rate of the algorithm and allow to better highlight the effects of the filtering method on the convergence rate.

In the present analysis, three families of sets of weights were considered, in addition to the custom set previously introduced. The first family is generated by using a linear sequence of K terms taken in reverse order and normalized by their sum (e.g., $\boldsymbol{\alpha} = [3,\ 2,\ 1]/\alpha_N$, with $\alpha_N = 6$ for $K = 3$). The second family of weights uses a geometric sequence of terms: the weights are the first K powers of two, taken in reverse order and normalized by their sum (e.g., $\boldsymbol{\alpha} = [4,\ 2,\ 1]/\alpha_N$, with $\alpha_N = 7$ for $K = 3$). The last family of weights corresponds to a monotonic subset of the Fibonacci sequence, starting from 1, with terms taken in reverse order and normalized by their sum (e.g., $\boldsymbol{\alpha} = [5,\ 3,\ 2,\ 1]/\alpha_N$, with $\alpha_N = 11$ for $K = 4$).

Table 5.8 presents the results of the Monte Carlo analysis in terms of success rate and run time. A run is considered successful if it converges to a solution with a payload mass value that is greater than 95% of the optimal one reported in Table 5.5. The latter is deemed very close the to global optimum of the problem instance since, despite the high number of runs of the Monte Carlo analysis, none provided a better performing solution. Unsuccessful runs are those for which the algorithm either diverges (i.e., exceeds the maximum number of iterations $i_{\max} = 100$) or it converges to a suboptimal solution (i.e., a feasible solution, but deemed unacceptable since the payload mass is lower than 95% of the optimal value). The reported run time statistics are evaluated by considering successful solutions only.

The results show that the effectiveness of the algorithm strongly depends on K. Indeed, when $K = 1$ the method was never able to achieve convergence, suggesting that if filtering is not employed, then another strategy should be pursued to prevent artificial unboundedness, such as a trust region on state or control variables [131, 132, 134, 148]. By setting $K = 2$, the algorithm can successfully converge to the optimal solution, but only in a limited number of cases. Instead, if $K \geq 3$, the

Table 5.8. Results of the Monte Carlo analysis for several values of K and different families of filtering weights.

K	Distribution of weights	Success rate (%)	Run time (s)		
			Min	Mean	Max
1	Linear[1]	0.0	–	–	–
2	Linear[1]	16.7	4.6	30.3	76.0
3	Linear[2]	55.9	4.4	25.8	75.2
	Geometric	51.7	4.4	25.6	60.4
	Custom	54.4	4.5	27.4	82.0
5	Linear	59.9	6.0	32.2	90.1
	Geometric	58.0	5.2	26.9	78.5
	Fibonacci	61.4	5.9	28.1	74.7
7	Linear	54.4	8.6	32.9	100.2
	Geometric	58.5	5.5	25.2	74.2
	Fibonacci	60.4	6.0	27.6	67.7

[1] For $K = 1$ and 2 the linear set is identical to the geometric and Fibonacci sets.
[2] For $K = 3$ the linear set is identical to the Fibonacci set.

algorithm appears to be much more reliable and it finds the optimal solution in the majority of cases. The best success rates are observed for $K = 5$, but the most efficient setups are the ones that are based on three solutions. This is due to the fact that filters that use more solutions are generally more conservative, as newer solutions are assigned smaller weights, and, despite showing increased ability to compensate for diverging solutions that may appear across the successive iterations, may require more iterations to converge, thus a longer computational time. On the other hand, if fewer solutions are taken into account in the update, the convergence is generally achieved in fewer iterations, as the search space is explored more rapidly, but the algorithm is more sensitive to artificial unboundedness and other diverging phenomena. As for the choice of the weight distribution, the homogeneity of the results for the each K indicates that all the considered families of sets are valuable options. Indeed, only slight differences are observed, suggesting that the geometric set is associated with the shortest mean run times, but the most robust families are the linear one, for $K < 5$, and the Fibonacci one, for $K \geq 5$.

Another Monte Carlo analysis was carried out to assess if the multi-step numerical continuation strategy could improve the success rate of the filtering approach. Besides the 3-step strategy described in Section 4.7, a 1-step approach, which merely consists in solving directly $\mathcal{P}_3$, and a 2-step method, which first solves $\mathcal{P}_1$ and then exploits its solution as a starting guess of $\mathcal{P}_3$, were also considered. Since the performance of all the considered families of weights are comparable, only the custom set of weights was used in this analysis. The same 2000 random initial trajectories as in the previous Monte Carlo campaigns were used. The results of the analysis are reported in Table 5.9. The effect of adopting the multi-step continuation strategy is practically null in terms of success rate (defined as in the previous Monte Carlo campaign). Indeed, the increase in the success rate when solving 2 or 3 problems instead of directly $\mathcal{P}_3$ is minimal. Instead, solving multiple instances of the problem increases significantly the computational time of the optimization process, thus making the use of the continuation strategy unappealing in this case.

Table 5.9. Results of the Monte Carlo analysis on the combined effect of the continuation strategy and the filtering approach.

$N_{\mathcal{P}}$ [†]	Success rate (%)	Run time (s) [‡]		
		Min	Mean	Max
1	54.4	4.5	27.4	82.0
2	54.5	18.8	73.2	201.4
3	54.9	20.0	80.4	196.1

[†] Number of problem instances solved in the continuation strategy.
[‡] The run times are evaluated by considering the successful solutions only.

5.3.2 Trust Region Approach

For the sake of comparison, the effectiveness of the use of a trust region in place of the filtering technique was also investigated. In this respect, a further Monte Carlo campaign was carried out using the same 2000 randomly generated initial reference trajectories employed for evaluating the filtering performance, but considering alternatively (i) an adaptive hard trust region, such as the one used in Refs. [52, 131, 134], or (ii) a soft quadratic trust region, such as in Refs. [135, 141]. Both trust regions were applied only on the control variables, since this was observed to be the most effective strategy for the problem under investigation. Thus, the following constraint was added in each controlled phase i

$$|u_k - u_k^{(i)}| \le \delta_k \qquad \text{for } k = 1, \dots, n_u^{(i)} \tag{5.1}$$

where $n_u^{(i)}$ is the number of control variables.

The initial trust radius of the adaptive hard trust region was set to $10\,\mathrm{m/s^2}$, as it was recognized as the value that granted the best success ratio after a parametric analysis. The trust radius is updated at every iteration j according to a measure of the quality of the linear approximation

$$\rho^{(j)} = \frac{\hat{J}^{(j-1)} - \hat{J}^{(j)}}{L^{(j-1)} - L^{(j)}} \tag{5.2}$$

where $L = -m(t_f) + J_\delta^{(11)} + J_\delta^{(12)} + J_q + J_w$ is the augmented cost function of the convex problem and $\hat{J}$ is an augmented objective function that includes a measure of the linearized dynamics error, which, in case of a *hp*-pseudospectral scheme, is

$$\hat{J} = -m(t_f) + \lambda_f \sum_{l=1}^{N} \sum_{s=1}^{h^{(l)}} \sum_{i=1}^{p_s^{(l)}} \left| \sum_{j=1}^{p_s^{(l)}+1} D_{ij}^{s,l} \boldsymbol{x}_j^{s,l} - \frac{\tau_{s+1}^{(l)} - \tau_s^{(l)}}{2} \boldsymbol{f}_i^{s,l} \right| \tag{5.3}$$

where N denotes the number of phases, $\lambda_f = 10^4$ is the dynamics error penalty weight, and f is the right-hand side of the nonlinear equations of motion (4.45)–(4.47).

It is worth remarking here that $\rho^{(j)}$ provides an estimate of the quality of the convex approximation, as it compares the nonlinear cost reduction $\Delta\hat{J}^{(j)}$ with the

Table 5.10. Results of the Monte Carlo analysis using common trust region algorithms.

$N_{\mathcal{P}}$[†]	Trust region	Success rate (%)	Run time (s)		
			Min	Mean	Max
1	None	0.0	–	–	–
	Hard	0.2	28.9	61.7	131.8
	Soft	0.0	–	–	–
2	None	0.1	48.8	65.0	74.8
	Hard	0.3	45.2	84.9	110.9
	Soft	28.3	24.7	82.5	167.7
3	None	0.1	60.9	68.6	76.3
	Hard	4.3	11.5	67.9	309.2
	Soft	29.1	29.6	90.0	189.1

[†] Number of problem instances solved in the continuation strategy.

convex problem cost reduction $\Delta L^{(j)}$. When $\rho^{(j)} \leq 0$, the trust region is shrunk by a factor 2 and the new solution is rejected, then the iteration is repeated with the new trust radius. If $0 < \rho^{(j)} \leq 0.25$ the iteration is accepted, but the trust radius is still reduced by a factor 2. Instead, the reference solution is updated and the trust radius is retained when $0.25 < \rho^{(j)} \leq 0.7$. Finally, if $\rho^{(j)} > 0.7$ the approximation is supposed to be accurate and the trust radius is increased by a factor 3.2. Note that if $\rho^{(j)}$ is greater than unity the linear cost reduction under-predicts the actual $\Delta\hat{J}$ and is thus conservative, hence the trust radius can be increased.

Instead, the principle of the soft trust region is much easier, as the trust radii δ_k from Eq. (5.1) are nonnegative optimization variables that are included in the augmented cost function of the convex problem through a penalty term $J_\delta = \sum_{k=1}^{n_u} \lambda_{\delta_k} \delta_k$. The penalty weights λ_{δ_k} were supposed equal to 10^{-3} for every control variable u_k and were selected on the basis of a parametric study that determined the most performing value in terms of success rate.

Table 5.10 reports the results of the Monte Carlo analysis. It is apparent that solving the ascent problem is quite challenging if using a trust region and starting from an initial guess generated as described in Section 4.6. Indeed, the success rate is zero or close to zero if no trust region is implemented or if either a soft or a hard one are used.

The success rate can be improved if a continuation strategy, as described in Section 4.7, is adopted. Indeed, the use of a 2-step or 3-step numerical continuation strategy increases the convergence rate of the algorithm and allows to compare the performance of the considered trust regions with greater detail, as solving intermediate problems, which either neglect the atmosphere or fix the time-lengths of the launch vehicle ascent phases, reduces the sensitivity of the algorithm to the initial guess. However, even with this improvement, both the hard and soft trust region appear to be much less robust than the filtering approach, as the success rates are always below 30%. This is due to the fact that, in general, trust regions work better when the initial reference trajectory is closer to the optimal solution, while in this application an accurate guess is supposed not to be available, as a launch vehicle ascent trajectory is quite hard to design from scratch.

Table 5.11. Discretization grids considered in the Monte Carlo analysis.

Grid	Nodes per phase												Total nodes
	1	2	3	4	5	6	7	8	9	10	11	12	
Default	6	6	18	6	20	15	6	20	10	20	20	20	167
A	5	5	9	5	9	9	5	9	5	9	9	9	88
B	5	5	17	5	17	17	5	17	9	17	17	17	148
C	9	9	33	9	33	33	9	33	17	33	33	33	284

5.3.3 Comparison Between Filtering and Trust Regions

Even though advanced trust region implementation demonstrated to be highly effective without particularly good initial guesses in diverse applications, including atmospheric entry [52, 148] and rocket landing [45, 135], a proper tuning of the involved parameters (such as the initial radius of the adaptive hard trust region or the penalty weights of the soft trust region) may require a significant effort. Instead, the filtering approach is much easier to set up, as the method sensitivity to the values of the weights is low and high success rates are achieved by every considered family as long as $K \geq 3$. In this respect, filtering appears as an effective and easy-to-set-up method to improve the convergence of successive convexification algorithms.

Finally, it is worth noting that the circumstances under which the Monte Carlo campaigns were carried out were particularly challenging. Indeed, to assess whether the proposed algorithm is able to solve the optimal ascent trajectory problem even starting from a rough initial guess, the starting trajectories were randomly generated by considering wide ranges of the seeding parameters (i.e., kick angle, payload mass, and coasting times). Greater success rates can be achieved if a more accurate initialization guess is used. A typical scenario is real-time guidance, where a nominal trajectory is already available and the vehicle path usually deviates only slightly from it. In that case, the algorithm converges in fewer iterations and with success rates very close to 100% regardless of using either filtering or a trust region (see Chapter 7).

5.4 Analysis of the Discretization Grid

A further analysis was carried out to investigate the efficiency and accuracy of several *hp* grids. In addition to the "default" mesh introduced in Table 5.3, three additional grids were considered. For each grid, three different values of h (i.e., the number of segments each phase is divided into) were investigated. Table 5.11 reports the number of discretization nodes in each phase for each mesh. Grid B is obtained from the default one by picking for each phase the number of intervals equal to the closest power of 2. This allows to set h equal to 1, 2, or 4, and still use local polynomials of the same order p in each phase segment, as the number of time intervals is a multiple of h. Grid A is obtained by starting from grid B and halving the number of intervals (and thus the local polynomials order p) in each phase, but keeping the minimum number of nodes above 5. Conversely, grid C is designed by doubling the intervals (and correspondingly the order) of each phase of grid B. When increasing h, it is ensured that the number of segments in a phase is less than the number of intervals in order to use local polynomials of, at least, order $p = 2$. Finally, note that, for the sake of simplicity, we did not include the return phase in this analysis.

The same 2000 initial reference trajectories as in Section 5.3 were considered

Table 5.12. Results of the Monte Carlo analysis on different grids.

Grid	h	Success rate (%)	Run time (s)			Average error		
			Min	Mean	Max	a (km)	e	i (deg)
Default	1	55.4	3.4	16.9	71.3	6.6e-02	1.3e-05	1.5e-04
A	1	55.6	1.3	7.3	35.8	7.0e-02	8.4e-05	2.5e-04
	2	56.4	1.3	6.4	40.0	6.1e+00	1.7e-03	3.3e-04
	4	0.0	–	–	–	–	–	–
B	1	55.1	2.9	18.3	55.5	4.1e-02	9.0e-06	5.1e-04
	2	56.1	3.3	15.6	51.9	2.6e-02	2.2e-05	7.2e-04
	4	54.9	2.3	15.5	47.6	1.4e-01	7.0e-05	4.8e-03
C	1	53.9	15.9	65.1	164.3	2.5e-02	4.1e-06	3.7e-04
	2	54.7	5.0	29.2	95.7	2.0e-02	1.4e-05	1.3e-03
	4	54.9	5.0	26.8	77.8	5.6e-02	2.5e-05	3.2e-03

for this campaign. Furthermore, the custom set of filtering weights was used in the update of the reference solution across iterations and no trust region is enforced on the state or control variables. Table 5.12 reports the results of the Monte Carlo analysis. Results are presented in terms of success rate, run time, and errors on the final orbital elements, namely semi-major axis a, eccentricity e, and orbit inclination i. The errors on the orbital elements are evaluated as differences between the desired values and the elements corresponding to the propagated final state, which is computed by numerical integration of the equations of motion of the original problem, that is, Eqs. (4.2)–(4.4), from liftoff to the burnout of the last stage using the optimal control laws. The run time and average error on the terminal value of the orbital elements refer to the successful runs only, as previously done in Section 5.3.1.

As expected, the computational time increases as the total number of nodes increases. For instance, solving the problem with grid C is more demanding than solving it over grid A. Nevertheless, for a given grid, the solution process can be more efficient if the phases are split into smaller segments, i.e., increasing h. On the other hand, increasing h on a given grid implies lower order polynomials, which appear as less accurate. Indeed, the average errors on the propagated orbital elements increase as h increases. This is particularly emphasized for grid A, the one with fewest nodes, which leads to large inaccuracies whenever $h > 1$. Indeed, as a limiting case, when grid A is used with $h = 4$, only unphysical solutions featuring a payload mass 10% greater than the attainable one are obtained; hence, no run is deemed successful. Instead, the success rates of all the other grids are comparable. The default mesh appears as a reasonable trade-off between accuracy and efficiency, as the mean run time is quite small and the errors on the final orbital elements are acceptable for any practical purpose.

5.5 Parametric Analysis of the Splash-Down Constraint

In the results presented in the previous sections, the return of the spent stage (Phase 13) was omitted to focus the analysis on the ascent. Indeed, the inclusion of the return phase increases significantly the complexity of the problem, making the solution procedure less robust to inaccurate starting reference trajectories, such as the ones obtained through the strategy outlined in Section 4.6. So, a common

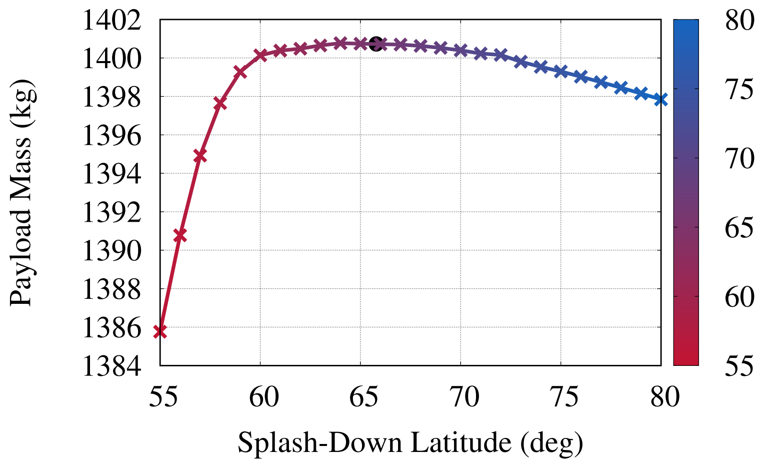


Figure 5.4. Payload mass as function of the splash-down latitude.

approach consists in solving the ascent problem first and then simulate the return trajectory of the spent stages. Eventually, if the splash-down points are not deemed safe, the ballistic reentry of the concerning stages are included in the OCP and constrained to different locations.

In this section, the return of the third stage is included in the OCP as Phase 13. The effect of varying the splash-down point on the trajectory and on the overall system performance is analyzed on the basis of a parametric study, discussing also the interaction between the splash-down and the heat flux constraint.

From a practical standpoint, a grid of impact points was defined within $\varphi_{R,\text{des}} = 55°$ and $\varphi_{R,\text{des}} = 80°$ with an increment step equal to 1°. Since the latitude of the unconstrained (hence, optimal) return corresponds to $\varphi_R^\star = 65.79°$, the unconstrained trajectory was used as initial guess of the problems with splash-down latitudes constrained to the two grid values adjacent to $\varphi_{R,\text{des}}$ (that is, 65° and 66°). Then, the problems constrained to the other splash-down latitudes were sequentially solved by using the converged solution of the OCP with the adjacent $\varphi_{R,\text{des}}$ as starting reference solution. This approach ensures that the initialization is quite accurate, as the difference in the splash-down point is small Indeed, on average only 8 iterations were required to meet the convergence criteria. On the other hand, the inclusion of the return phase significantly increases the problem dimension, thus each iteration was computationally more demanding (requiring 0.95 s on average) than when solving the OCP without Phase 13. Nevertheless, the overall solution process of an OCP required, on average, a computational time of 7.75 s.

The default grid detailed in Table 5.3 to discretize the problem. Numerical integration forward in time of the equations of motion showed that, on this grid, the error on the splash-down point is less than 20 km. This error may appear large compared to the one on the semi-major axis, which is below 100 m, but the return dynamics are much more sensitive due to the nonlinear atmospheric forces that act on the spent stage until the splash-down. Indeed, if the atmospheric drag is removed from the equations of motion, the error on the splash-down location drops to 500 m, which is in agreement with the errors on the payload orbital elements.

The optimal payload mass is plotted in Fig. 5.4 as a function of the splash-down latitude. While moving the spent stage return point significantly changes the trajectory, as shown in Figs. 5.5 and 5.6, it does not necessarily hinder the performance. Indeed, the payload curve is essentially flat in the interval $\varphi_R \in$

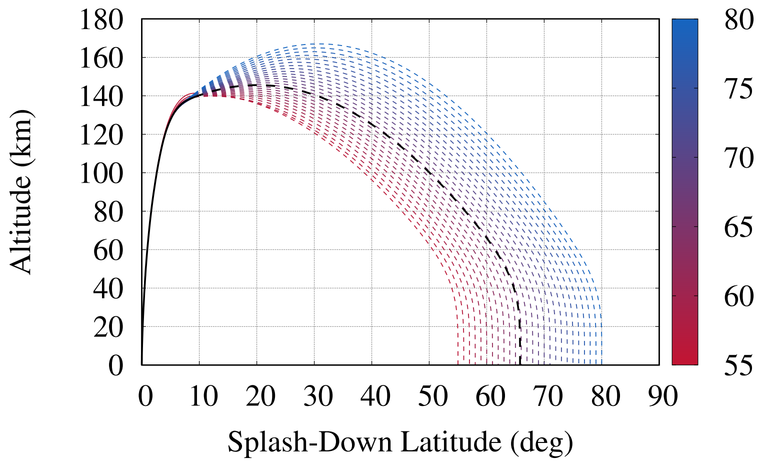


Figure 5.5. Stage 3 trajectories from liftoff to splash-down.

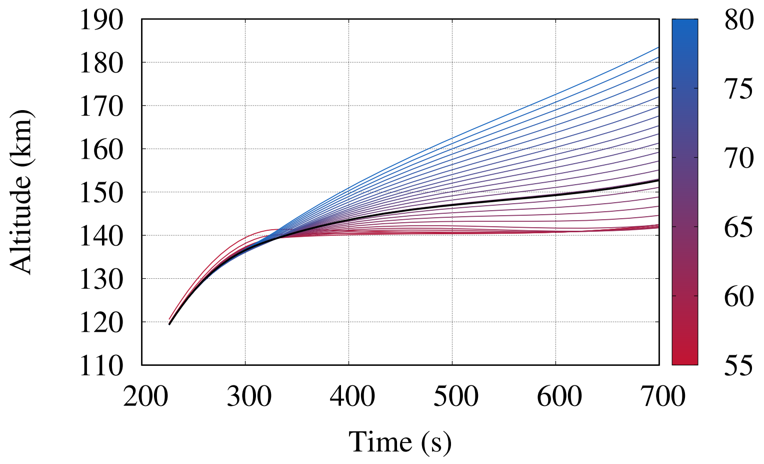


Figure 5.6. Altitude profiles from fairing jettisoning to $t_f^{(10)}$.

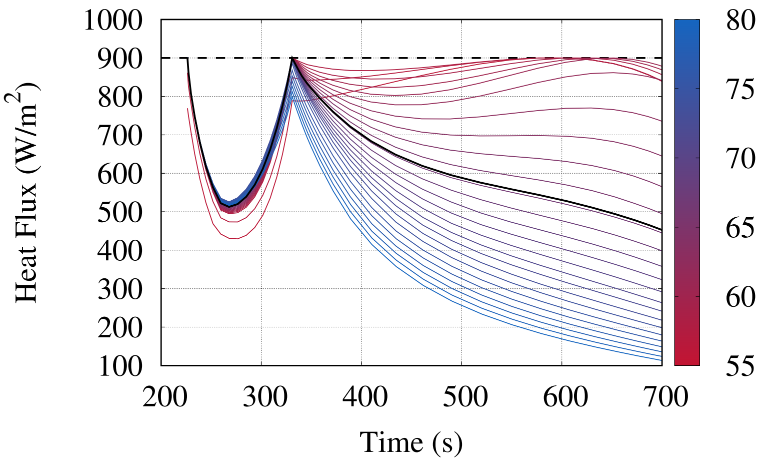


Figure 5.7. Heat flux profiles from fairing jettisoning to $t_f^{(10)}$.

$[60°, 72°]$ and variations only below 1 kg are observed. When the splash-down location is moved beyond 72°, the decrease in performance is more evident, but it still does not represent a concern as a shift of 15° causes a loss of only 3 kg. Instead, moving the splash-down point closer than 60° appears to be more critical, as a greater performance drop is observed. Nevertheless, even constraining the third stage to fall 10° closer than $\varphi_R^\star$ results in a payload reduction by only 1% of its optimal value.

It is worth studying how the heat flux and splash-down constraints interact with each other. Figure 5.7 shows the heat flux history that the payload undergoes from the fairing jettisoning until the end of the first firing of Stage 4. Red curves correspond to splash-down locations at lower latitudes, i.e., closer to the launch site, while blue ones are associated with high-latitude returns. In all trajectories for which $\varphi_R \geq 57°$, the heat flux constraint is active only at the boundaries of Phase 8, so Assumption 1 holds in all these cases. Instead, the heat flux constraint is active over intervals of finite duration when the splash-down point is moved closer than 57°. In particular, the heat flux peak is delayed and occurs during Phase 10. Note that the duration and location of the bounded arc are very difficult to predict, but, thanks to the direct discretization method, the optimal switching structure is automatically determined by the optimization process and no *a priori* guess is required.

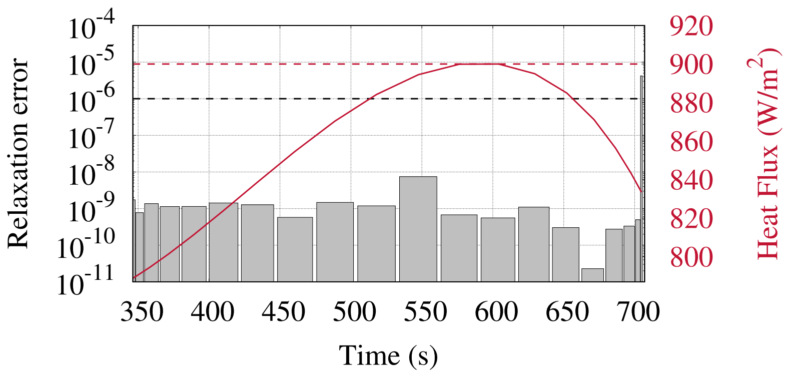


Figure 5.8. Relaxation error during Phase 10 of the trajectory constrained to $\varphi_R = 55°$.

Figure 5.8 reports the relaxation error during Phase 10 of the solution corresponding to $\varphi_R = 55°$. This solution is particularly interesting as Assumption 1 does not hold anymore in the interval $[576.6, 604.2]$ s. The relaxation error is always below the solver feasibility threshold (10^{-6}), except for the final node of the phase that, due to the Radau discretization scheme, is not an optimization variable and is extrapolated *a posteriori* from the approximating polynomial. Nevertheless, even though no theoretical proof can be provided, the relaxation is lossless even in the interval where Assumption 1 does not hold, as the resulting controls satisfy Eq. (4.52) with the equality sign with a precision in agreement with the solver feasibility tolerance.

Chapter 6

Case Study: Ascent of a Two-Stage Vehicle to the ISS

This chapter presents a second case study for the ascent trajectory of a launch vehicle, with the aim of showing that the convex optimization approach here discussed for the design of launch vehicle ascent is sufficiently general to be applied to any launch vehicle mission.

6.1 Two-Stage Launch Vehicle Model

Instead of a VEGA-like configuration, here a two-stage launch vehicle is considered. The vehicle is based on the SpaceX Falcon 9 Full Thrust rocket [149]. The values of propellant and dry masses, burn time, vacuum thrust, mass flow rate, and nozzle exit area adopted in the present study are summarized in Table 6.1. Note that, since the Falcon 9 is equipped with liquid rocket engines, the vacuum thrust and mass flow rate are assumed constant over time. Furthermore, the fairing mass is 1700 kg, the drag coefficient C_D is assumed to be constant and equal to 0.329, while the reference surface S_{ref} is $10.52\,\text{m}^2$. Finally, an isothermal and exponential atmosphere model is considered, with a scale height H equal to 8.4 km for any altitude.

Table 6.1. Two-Stage Rocket Data

Quantity	**Stage 1**	**Stage 2**	**Unit**
m_p	410 843	107 509	kg
m_{dry}	22 200	4000	kg
t_b	162	397	s
T_{vac}	8227	934	kN
$\dot{m}_e$	2536	270.8	kg/s
A_e	11.039	1.227	m^2

6.2 Phases

As described in Section 4.2.1, the ascent trajectory of a launch vehicle is a succession of different guidance programs, namely vertical ascent, gravity turn, and out-of-atmosphere phases, possibly separated by coasting arcs. The phase structure used to model the ascent trajectory of a two-stage vehicle is reported in Fig. 6.1. Differently

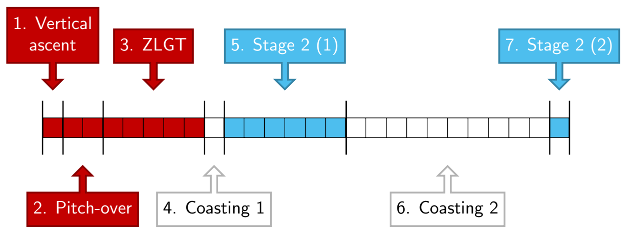


Figure 6.1. Phases of the optimal control problem for the two-stage vehicle.

from the VEGA-like rocket, there is no need to include (nor to constrain) the return phases of the spent stages, as the first stage burns out at a relatively small velocity and, if uncontrolled, splashes down relatively close to the coast, while the second stage burns out in orbit, so it can re-enter the Earth with a programmed maneuver after payload release.

6.3 Problem Statement

The goal of the problem is to maximize the final mass of the system. Thus,

$$J = -m(t_f) \tag{6.1}$$

The launch vehicle dynamics are the same as for the VEGA-like case. The rocket is supposed to lift off from Cape Canaveral (located at a latitude equal to $\varphi_{\text{LB}} = 28.585°$) and must release the payload into the ISS's orbit, which is assumed to be a circular 400 km altitude orbit with an inclination of 51.6°. So, the initial and final conditions are

$$\boldsymbol{r}(t_0) = \boldsymbol{r}_{\text{LB}} \tag{6.2}$$

$$\boldsymbol{v}(t_0) = \boldsymbol{\omega}_E \times \boldsymbol{r}_{\text{LB}} \tag{6.3}$$

$$x(t_f)^2 + y(t_f)^2 + z(t_f)^2 = r_{\text{des}}^2 \tag{6.4}$$

$$v_x(t_f)^2 + v_y(t_f)^2 + v_z(t_f)^2 = \mu/r_{\text{des}} \tag{6.5}$$

$$\boldsymbol{r}(t_f) \cdot \boldsymbol{v}(t_f) = 0 \tag{6.6}$$

$$x(t_f)v_y(t_f) - y(t_f)v_x(t_f) = h_{z,\text{des}} \tag{6.7}$$

The fairing is assumed to be jettisoned as soon as the first stage separates from the rocket. Thus,

$$m(t_0^{(4)}) = m(t_f^{(3)}) - m_{\text{dry},1} - m_{\text{fairing}} \tag{6.8}$$

For the sake of simplicity, no heat flux constraint is considered in the present case, even though it could be easily included as for the VEGA-like case.

The time-lengths of Phases 1–3 and the duration of the first coasting are assumed as fixed parameters and their values are reported in Table 6.2. Instead, the time-lengths of the last three phases are free. Since the total propellant mass of the upper stage is fixed, and fully consumed, the sum of the duration of Phases 5 and 7 must equal the overall burn time $t_{b,2}$

$$\Delta t^{(5)} + \Delta t^{(7)} = t_{b,2} \tag{6.9}$$

To enhance convergence, a soft trust region constraint was posed on the duration of Phases 6 and 7

$$|\sigma^{(i)} - \bar{\sigma}^{(i)}| \leq \delta^{(i)} \qquad i = 6, 7 \tag{6.10}$$

The trust radii are penalized through the additional terms

$$J_\delta^{(i)} = \lambda_\delta^{(i)} \delta^{(i)} \qquad i = 6, 7 \tag{6.11}$$

and the corresponding penalty weights were set equal to $\lambda_\delta^{(6)} = 10^{-4}$ and $\lambda_\delta^{(7)} = 10^{-3}$.

Table 6.2. Nominal time-lengths of time-fixed arcs

Phase	Δt (s)
1	5.00
2	6.00
3	151.00
4	1.00

Analogously to the VEGA-like case, a *hp*-pseudospectral discretization method, based on Radau collocation, was used to discretize the OCP. Table 6.3 reports the values of h and p for every phase of the problem. Differently from the VEGA case, where only the atmospheric return required splitting the domain in multiple segments, the ascent phases of the Falcon 9 rocket require a greater number of nodes because of longer time-lengths. To avoid the use of high-order polynomials, Phases 3, 5, and 6 were split into $h = 2$ subintervals.

Table 6.3. Discretization segments, order, and nodes in each phase.

	Phase						
	1	2	3	4	5	6	7
h	1	1	2	1	2	2	1
p	5	5	20	5	20	20	10
Nodes	6	6	41	6	41	41	11

Virtual controls and buffers were added to the formulation to relax the linearized dynamics and terminal constraints. The values of the penalty weights are the same as for the VEGA-like case, that is, $\lambda_q = \lambda_w = 10^4$. Also, the same nondimensionalization factors were used to scale the data, that is, the Earth radius is the reference length, the corresponding circular orbit velocity is the reference velocity, and a reference mass of 10 000 kg is considered.

6.4 Initialization

To find a suitable initial guess, a reference trajectory is obtained according to the strategy described in Section 4.6. Indeed, by assuming no atmosphere and assigned values of the unknown initial mass, control laws, and arc time-lengths, a trajectory can be designed through forward propagation of the rocket equations of motion. Simple parametric control laws are adopted in the pitch-over (Phase 2) and the two second stage burns (Phases 5 and 7). The control laws are expressed in terms of the elevation angle ϕ and the azimuth angle ψ, defined as in Section 4.6. During the pitch-over phase, the elevation varies linearly from 90° to $\phi(t_f^{(2)})$, which denotes the kick angle and must be guessed. Since the maneuver takes place in an inertially-fixed plane, the azimuth is kept equal to a constant value $\psi^{(2)}$, which is picked as in Eq. (4.74). For the upper stage phases, the tentative control laws are designed such that the orbital plane is kept constant after the first stage burnout, hence $\psi \equiv 0$ in the second stage phases. On the other hand, the elevation is set to be identically null in the last burn, while it is assumed to vary linearly from a given value $\phi(t_f^{(5)})$ to zero in the first firing of the second stage.

To sum up, the only variables to guess in order to generate a tentative solution are the initial mass m_0, the kick angle $\phi(t_f^{(2)})$, the initial elevation of the upper stage $\phi(t_f^{(5)})$, and the time-lengths of the second stage phases. The values to pick depend on the specific launcher and target orbit. Nevertheless, coming up with suitable values is not an arduous task, especially if a three-step continuation strategy, as outlined in Section 4.7, is adopted since the solution of $\mathcal{P}_1$ does not require an extremely accurate initialization. For instance, the first guess values used to generate the starting reference trajectory are reported in Table 6.4.

Table 6.4. Values used for the generation of the first guess trajectory of the two-stage rocket.

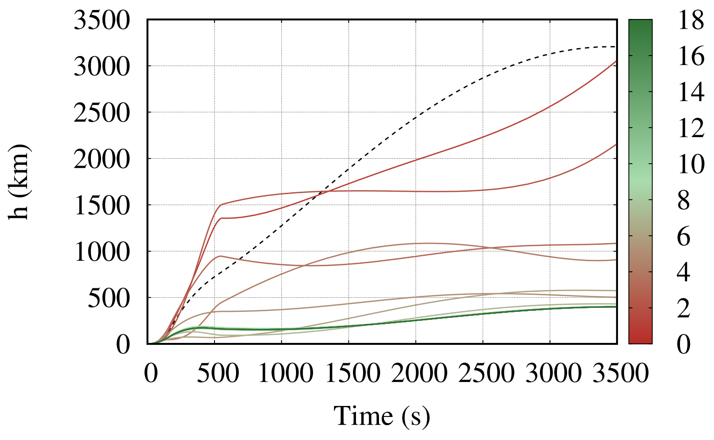
Quantity	**Value**	**Unit**
m_0	558 000	kg
$\phi(t_f^{(2)})$	89.5	deg
$\phi(t_0^{(5)})$	20.0	deg
$\Delta t^{(5)}$	394.5	s
$\Delta t^{(6)}$	3000.0	s
$\Delta t^{(7)}$	2.5	s

It is worth noticing that the upper stage is expected to perform a Hohmann-like maneuver, but the propellant distribution among the two firings of the second stage is quite different from VEGA's last stage, as most of the propellant is consumed during the first burn, leaving only a very small fraction of the overall impulse for the injection maneuver. This is due to the greater acceleration (i.e., thrust-to-mass ratio) provided by the upper stage engine of the Falcon 9 in the last seconds of operation compared to VEGA. Another difference concerns the kick angle value, which, for the Falcon 9, is much closer to 90°, as this kind of launch vehicle rotates much more slowly than lighter vehicles such as VEGA.

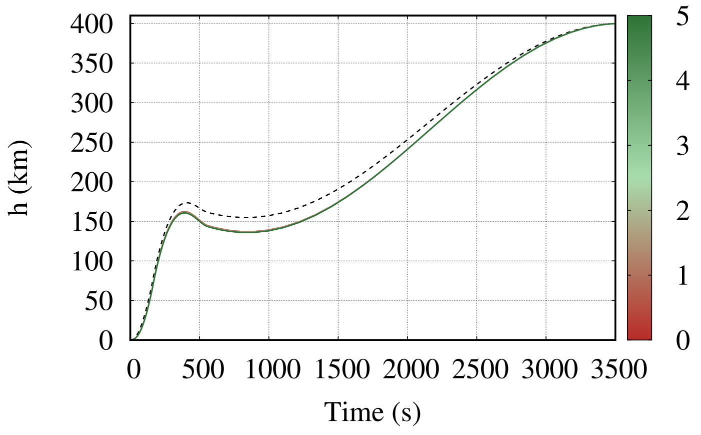
6.5 Convergence Behavior

In this section, the convergence of the successive convexification algorithm for the two-stage rocket problem is investigated. Differently from Section 5.2, where the complete problem $\mathcal{P}_3$ was solved directly starting from the first guess solution, here the convergence in each step of the three-step continuation strategy is discussed. The tolerances on the convergence and dynamics were set equal to $\epsilon_{\text{tol}} = 10^{-4}$ and $\epsilon_f = 10^{-6}$. Also, filtering was used to enhance convergence, based on $K = 3$ previous solutions, with weights $\boldsymbol{\alpha} = [6,\, 3,\, 2]/\alpha_N$ and $\alpha_N = 11$.

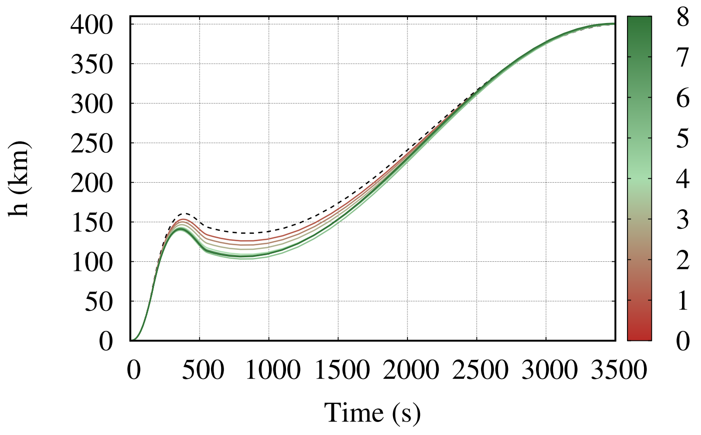
Figure 6.2 reports the altitude profiles throughout the successive iterations for all the continuation steps. The dashed lines denote the starting trajectories, while the solid lines gradually transition from red to green to illustrate the convergence process. The largest variations occur in the first sequence (Fig. 6.2a). This is due to the rough tentative trajectory, which is quite distant from the solution of $\mathcal{P}_1$. Nevertheless, the process converges rapidly: after just 10 iterations, the difference between one



(a) Starting from the first guess trajectory and solving the time-fixed, with no atmosphere problem.



(b) Starting from the no-atmosphere solution and solving the time-fixed atmospheric problem.



(c) Starting from the atmospheric time-fixed solution and solving the free-time problem.

Figure 6.2. Altitude profile across the iteration sequences for the ISS mission.

solution and the next one is minimal, and the altitude profiles in Fig. 6.2a start overlapping. Convergence within the desired tolerance is attained in 18 iterations.

When the problems $\mathcal{P}_2$ and $\mathcal{P}_3$ are considered, the differences between one iteration and the next one are smaller. Indeed, when the atmosphere is added (Fig. 6.2b), the most evident change concerns the maximum altitude reached before the long coasting phase, which is reduced by a few tens of kilometers. This is mainly due to the decrease in the effective engine thrust caused by the (non-null) external pressure. In this step, convergence is attained in 5 iterations. Finally, in the third step, where the time-lengths of the second stage arcs are also optimized, the differences among the profiles plotted in Fig. 6.2c are almost unnoticeable. Only slight changes in the time-lengths are needed to find the optimal solution. The reason is twofold: (i) the solution of $\mathcal{P}_2$ is an accurate initial guess for $\mathcal{P}_3$ and (ii) the penalty terms J_δ on time-length deviations favor solutions in the neighborhood of the reference trajectory. Indeed, the ascent problem features many locally optimal solutions, but, for any practical purpose, some of these solutions are deemed as performing as the global optimum, as they feature similar values of the payload mass. In order to let the algorithm adjust the time-lengths of free-time arcs to the optimal values and, thus, prevent convergence toward a local optimum with a payload mass significantly smaller than the global optimum, it is key to pick values as small as possible for the time-deviations penalty weights λ_δ. This third step requires 8 iterations to achieve convergence.

It is worth remarking that the successful convergence of the successive convexification process strongly depends on the presence of virtual controls and virtual buffers, as their inclusion was observed to be necessary to prevent the formulation of infeasible intermediate problems, which would otherwise interrupt the sequence prematurely. Indeed, it has been observed that, during the iterative process, the virtual controls $\boldsymbol{q}$ and the virtual buffers $\boldsymbol{w}$ are actively used in the first iterations, then rapidly approach zero and stay below the desired threshold value (10^{-6}) in almost all the remaining iterations.

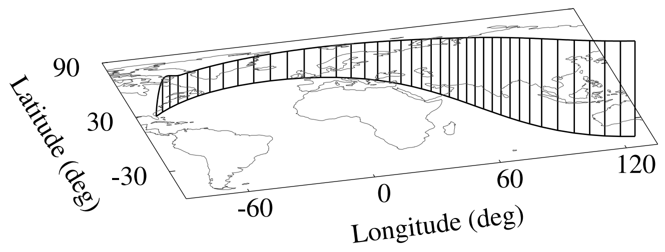
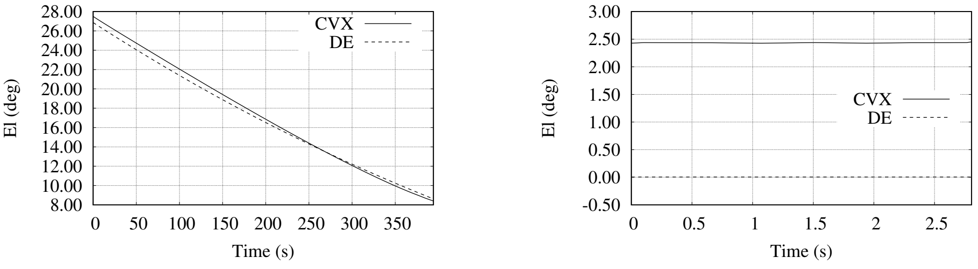


Figure 6.3. Visualization of the ISS ascent trajectory.

The final trajectory is visualized in Fig. 6.3 and the converged solution is summarized in Table 6.5. For the sake of comparison, this case study was also solved by means of the differential evolution (DE) algorithm EOS [108]. The two solutions are in good agreement: the difference in payload mass is less than 50 kg, which corresponds to 0.18% of the payload mass m_{pl} and 0.01% of the lift-off mass m_0, thus in accordance with the prescribed tolerances. The burn times of the second stage firings are substantially the same, as the difference is only of 0.01 s. Instead, a larger discrepancy is observed in the duration of the second coasting arc. The cause for the additional 34 seconds is related to the provided initial guess (3000 s). Indeed,

Table 6.5. Final solution compared with the DE solution.

Quantity	Convex	DE	Diff. (%)	Unit
m_0	572 507.53	572 556.36	0.01	kg
m_{pl}	26 256.59	26 304.45	0.18	kg
$\Delta t^{(5)}$	394.19	394.20	<0.01	s
$\Delta t^{(6)}$	2954.90	2920.72	1.17	s
$\Delta t^{(7)}$	2.81	2.80	0.20	s



(a) Second stage first firing **(b)** Second stage second firing

Figure 6.4. Optimal control laws in the ISS mission.

the solution of the free-time convex problem is handled by (slightly) penalizing the Δt deviation from the reference value via Eq. (6.10); thus, sometimes, locally optimal solutions in the neighborhood of the reference times may be preferred if they feature a payload mass close to the global optimum.

Figure 6.4 shows the time history of the optimal thrust direction elevation angle, defined as the angle between the in-plane component of the thrust and the local radial direction, during the two upper stage firings. In both phases, the time-laws appear quite regular, suggesting the optimality of the attained solution. Specifically, during the first firing (Fig. 6.4a) the trend is almost linear; instead, in the second firing (Fig. 6.4b), the control angle is kept constant. The second burn angle appears to be shifted with respect to the DE solution one, which is kept almost to zero for the whole firing. This is due to the different coasting duration, which affects the rocket orientation at the time of the second ignition. Finally, it is worth noting that the convex approach achieves a speed-up factor of about 100x–300x compared to the fully optimized parallel-execution DE solver.

The exactness of the relaxation of the control constraint (4.50) into Eq. (4.52) can be empirically verified by inspecting the relaxation error $e_R = |u_x^2 + u_y^2 + u_z^2 - u_N^2|$ at each discretization node. As an example, Fig. 5.8 reports the relaxation error during Phase 5, clearly showing that the equality constraint is always satisfied within the prescribed tolerance (here set equal to 10^{-6}).

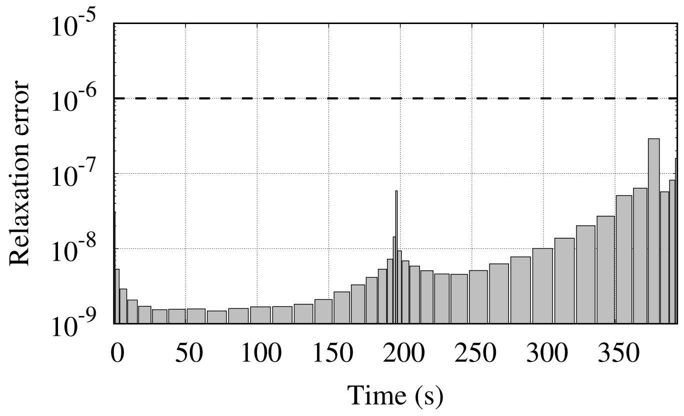


Figure 6.5. Relaxation error across Phase 5 of the ISS solution.

Chapter 7

Application to Upper Stage Guidance

This chapter investigates a closed-loop guidance, based on convex optimization and model predictive control, for the upper stage of a launch vehicle. Specifically, two guidance strategies are considered. The first one, which recursively solves an optimal control problem that comes directly from the launch vehicle ascent trajectory optimization problem discussed in Chapter 4, provides robustness to model errors and random external disturbances. The second one, instead, incorporates an additional return phase in the OCP to reduce the scattering of the splash-down location of the spent stage even in the case of uncertain stage cut-off time. Numerical results are presented to assess and compare the performance and robustness of both approaches with reference to VEGA's third stage case study.

7.1 Motivation for Upper Stage Guidance

Robust and efficient guidance algorithms are paramount in the final phases of a launch vehicle ascent to guarantee the accurate injection of the payload into the target orbit, even in presence of non-negligible uncertainties, and to enhance the vehicle responsiveness in off-nominal conditions. Indeed, launch vehicles feature significant model uncertainties with respect to the performance of the propulsion system, dispersions of the aerodynamic coefficients, and unpredicted variations of local atmospheric conditions, to name a few. Therefore, the design of a reliable guidance algorithm, capable of working in a not fully deterministic scenario, is mandatory to ensure the system in-flight autonomy.

While safety-related considerations must be necessarily taken into account when dealing with vehicles that operate in uncertain environments, the design of aerospace systems should also be performance-oriented. To this end, the definition of *optimal* guidance strategies is key to maximizing the launch vehicle performance (e.g., in terms of carrying capacity).

This thesis investigates the optimal guidance of the third stage of a VEGA-like launch vehicle as a case study. This peculiar rocket configuration is such that the velocity of the third stage at separation is quite high, hence the burned-out stage ends up falling far away from the launch site. Two major concerns arise from this aspect: (i) an accurate prediction of the (nominal) impact point of the spent stage is required, and (ii) the actual impact point must be sufficiently close to the nominal one, despite external disturbances, model uncertainties, and the launch vehicle intrinsic dispersions.

At present, the guidance of Z9 relies on a neutral axis maneuver in the last seconds before cut-off to meet these requirements. Similarly to the *null miss condition* developed for ballistic missiles [89], the neutral axis maneuver (NAM) prescribes the attitude of the rocket, so that the ground impact point is retained regardless of any additional velocity increments. Under the assumption of Keplerian motion (i.e., subject only to inverse-square gravity) during the return phase, an analytical guidance law can be derived. However, due to the presence of the atmosphere, such a model is inexact and the guidance algorithm requires a consistent effort for trajectory validation and verification to ensure robustness before each launch, making this approach cumbersome and time-consuming.

Instead, if a more realistic dynamical model were used in the guidance scheme, the onboard algorithm would provide a much more reliable prediction of the splash-down location and thus reduce the dependance on time-consuming pre-flight procedures. Also, a closed-loop architecture provides greater robustness to random in-flight disturbances and model uncertainties than an open-loop one, being based on the encountered flight conditions, and the recursive computation of the optimal trajectory and control signal guarantees better system efficiency than tracking a pre-scheduled flight plan. Therefore, motivated by the computational efficiency and reliability of the convex approach to the launch vehicle ascent trajectory optimization, even when considering a realistic dynamical model and practical constraints, a model predictive control framework that incorporates convex optimization is investigated as a closed-loop strategy for a safe and performing upper stage guidance.

7.2 Model Predictive Control Algorithms

MPC consists essentially of a feedback controller that generates the control signal by solving an OCP with updated initial conditions and parameters in a receding-horizon manner. The flowchart in Fig. 7.1 represents a general MPC algorithm. The loop starts with the reading of the data coming from the navigation system, which concern the present state of the system and other parameters that characterize the OCP (e.g., the local atmospheric conditions). Then, the optimal control problem instance is formulated on the basis of the measured state, which is used to set up the initial condition of the problem and adjust the dynamical model and constraints. The output of the optimization is a control signal that maximizes the system performance according to the OCP merit index and ensures that all constraints will be satisfied. The OCP is defined over a time domain, referred to as the *prediction horizon*, that is longer than the *control horizon*, which is the time interval T that elapses between one control step and the next one and determines the MPC update frequency. Indeed, the prediction horizon must last long enough to evaluate the system performance and account for all the mission constraints. The extent of the control horizon, on the other hand, is much shorter. Over the control horizon, the optimal control signal is used for actuation purpose, then the MCP cycle repeats until the stopping criterion is met.

It is worth noting that, in a theoretical study as the present thesis, the actuation of the optimal control law and the subsequent interaction with the environment is replaced with a simulation of the system dynamical model, sometimes referred to as *truth model*, perturbed by a noise signal that approximates the expected external disturbances and allows to take into account effects due to unmodeled dynamics.

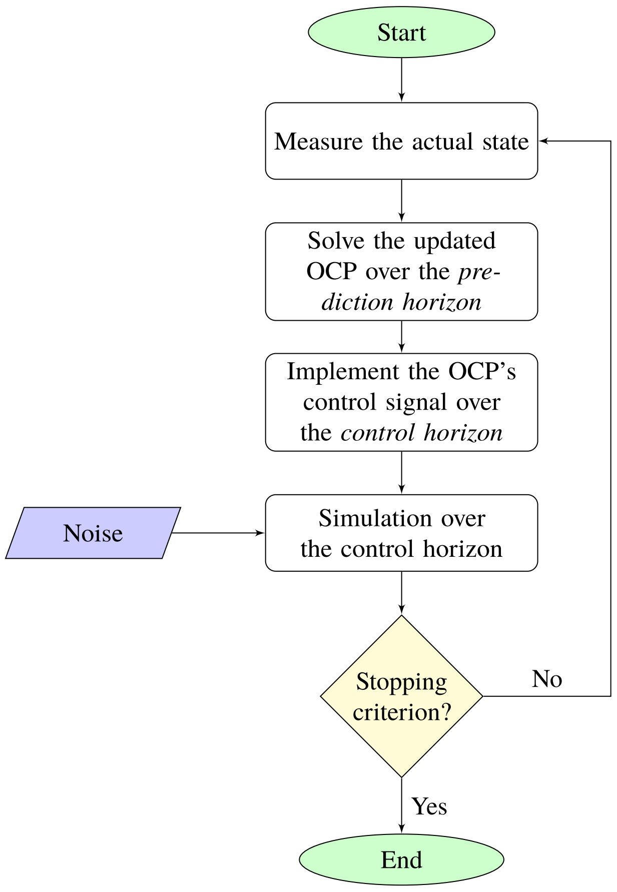

Start
Measure the actual state
Solve the updated OCP over the *prediction horizon*
Implement the OCP's control signal over the *control horizon*
Noise
Simulation over the control horizon
Stopping criterion?
No
Yes
End

Figure 7.1. MPC flowchart.

7.3 Upper Stage Guidance Strategy

An MPC scheme provides inherent robustness to model errors and additive noise thanks to the closed-loop architecture. In most applications, these classes of uncertainties cover all the significant sources of mismatch between the predictive model and the truth one. However, if the system is subject to a different class of uncertainty, then some further expedients must be devised to make the MPC algorithm robust.

For instance, upper stages equipped with a solid rocket motor, such as VEGA's third stage, feature a non-negligible dispersion on their cut-off time. Indeed, the thrust profile that a SRM will provide is difficult to predict and the burn time can be estimated only with a limited accuracy. Since even a few seconds of extra operation of the stage may move the return point of the spent stage by hundreds of kilometers, a guidance strategy has to explicitly account for such an uncertainty to ensure robustness.

Two guidance strategies are investigated in this thesis, both of which make use of an MPC framework to guide the flight of the third stage of VEGA in a closed-loop fashion. The two strategies mainly differ by the OCP that is recursively solved. The first strategy relies on the definition of an OCP that directly comes from the solution of the ascent problem presented in Section 4.4, where only the flight of the third and fourth stages (Phases 7–13) are considered. The second strategy, instead, assumes that, in the last few seconds of operation, Z9 switches to an open-loop control law. To increase the robustness of the splash-down point to uncertainties on the SRM cut-off time, an additional return phase is included in the OCP, and the open-loop control is tailored to minimizing the dispersion of the splash-down point by evaluating (and updating) directly onboard and in a systematic way the neutral axis direction. Due to this main difference, in the following, the former strategy is referred to as *single-return* strategy and the latter as *multi-return* strategy.

It is worth noticing that the simulation of two return trajectories (instead of one) increases the problem dimension and complexity, requiring a larger computational effort. Therefore, the multi-return strategy should be implemented only if the system features a significant uncertainty on the cut-off time of the stage. Instead, if the stage were equipped with a liquid rocket engine, which can be turned off more easily than a SRM, the single-return strategy is a more convenient approach, as it is inherently robust and allows for higher update frequencies.

7.3.1 Single-Return Strategy

Figure 7.2 illustrates the phases of the ascent of a VEGA-like rocket from the third stage ignition until payload release into orbit, while accounting also for the return of the spent third stage. The phases of the flight are identified by letters instead of numbers to avoid confusion with the phases of the OCP.

The closed-loop MPC guidance is implemented for the entire operation of the third stage, that is, Phases A and B. The stopping criterion of the MPC algorithm is the burnout of the stage, which is supposed to take place at a well-known time. Then, an estimate of the necessary propellant consumption to release the payload in the desired orbit can be retrieved by solving the ascent OCP for the remainder of the flight (i.e., Phases C–F)[1]. Likewise, the final position and velocity at the cut-off can be used to simulate the return of the spent third stage (Phase G).

[1]The guidance of the fourth stage flight is out of the scope of this dissertation.

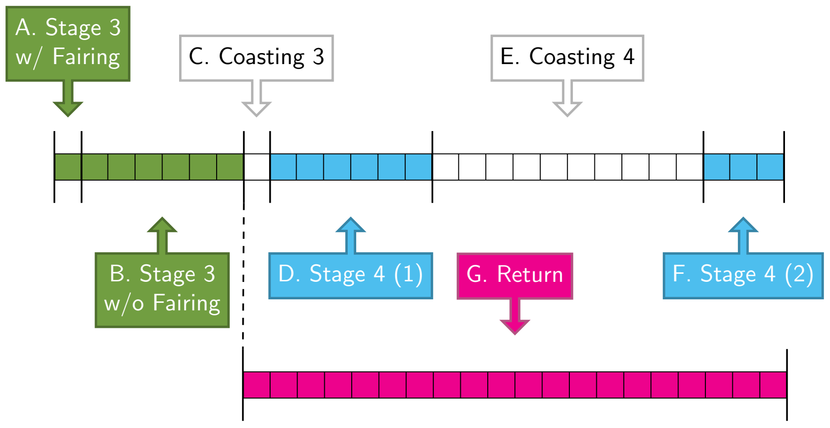


Figure 7.2. Phases of the upper stage single-return guidance strategy.

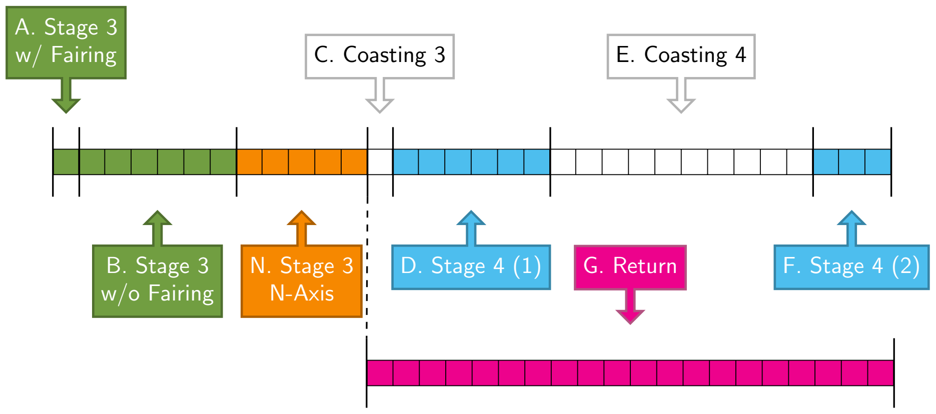


Figure 7.3. Phases of the upper stage multi-return guidance strategy.

7.3.2 Multi-Return Strategy

To design a guidance algorithm that is robust not only to generic in-flight disturbance and model errors but also to an uncertain cut-off time of the solid rocket motor, the guidance strategy must explicitly incorporate, in some form, an open-loop neutral axis maneuver.

Figure 7.3 reports the phase sequence of the multi-return strategy. Differently from the single-return approach, the third stage flight is here split in three phases. The first two phases (Phases A and B) model the vehicle flight before and after the payload fairing jettisoning, respectively, while the third one (Phase N) models the neutral axis maneuver. The neutral axis phase is quite different from the rest of the stage operation, as, during this phase, the attitude of the rocket (hence, the thrust vector) is prescribed in a constant, open-loop neutral axis direction.

The MPC guidance scheme is applied only in Phases A and B, that is, from the third stage ignition until the moment when the neutral axis maneuver starts, which is the MPC stopping criterion and corresponds to the minimum cut-off time of the SRM. From that point on, the open-loop neutral axis maneuver (Phase N) begins and the rocket holds on to a constant attitude, which is the same attitude of the vehicle during the last MPC cycle. Phase N terminates as soon as the SRM actually

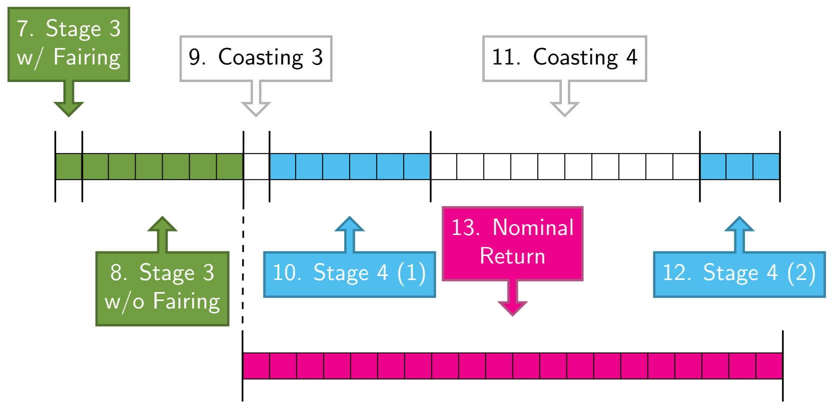


Figure 7.4. Phases of the OCP recursively solved by the single-return MPC algorithm.

burns out.

Also in this case, a prediction of the propellant consumption of the fourth stage can be retrieved by solving the ascent OCP relative to Phases C-F and the splash-down point can be estimated by simulating the return of the spent stage with initial position and velocity corresponding to the measured burnout conditions.

7.4 Single-Return Optimal Control Problem

The MPC controller generates at every time step a control law by solving in real time a finite-horizon optimal control problem. In this section, the upper stage guidance problem considered in the single-return algorithm is described and formulated as a multi-phase OCP. In particular, the OCP is obtained essentially by removing Stages 1 and 2 from the complete ascent OCP described in Section 4.4. First, the phase sequence of the OCP is outlined; then, the considered dynamical model is presented; and finally, the objective function and all the constraints of the OCP are reported.

7.4.1 Phase Sequence

Figure 7.4 illustrates schematically the considered phase sequence of the OCP. It is obtained simply by starting from the phase sequence of the complete ascent problem (reported in Fig. 4.4) and removing all the arcs that precede the third stage ignition. Indeed, Phases 7–13 are the same as for the complete ascent.

7.4.2 System Dynamics

The dynamical model here considered is the same as described in Sec. 4.3. The vehicle state $\boldsymbol{x}$ is described by its position $\boldsymbol{r}$, velocity $\boldsymbol{v}$, and mass m,

$$\boldsymbol{x} = \begin{bmatrix} x & y & z & v_x & v_y & v_z & m \end{bmatrix}^T \tag{7.1}$$

The thrust direction $\hat{\boldsymbol{T}}$ corresponds to the control of the system, whose elements, being a unit vector, must satisfy the following identity at any time

$$\hat{T}_x^2 + \hat{T}_y^2 + \hat{T}_z^2 = 1 \tag{7.2}$$

The resulting equations of motion are

$$\dot{\boldsymbol{r}} = \boldsymbol{v} \tag{7.3}$$

$$\dot{\boldsymbol{v}} = -\frac{\mu}{r^3}\boldsymbol{r} + \frac{T_a}{m}\hat{\boldsymbol{T}} - \frac{D}{m}\hat{\boldsymbol{v}}_{\text{rel}} \tag{7.4}$$

$$\dot{m} = -\dot{m}_e \tag{7.5}$$

Note that, since the upper stage does not perform the gravity turn maneuver or a vertical ascent, the only thrust contribution that appears in Eqs. (7.3)–(7.5) is the optimally controlled one, T_a, while T_b and T_c are dropped.

7.4.3 Objective and Constraints

The goal of the optimization is to minimize the propellant consumed during the fourth stage operation. Indeed, if the guidance algorithm minimizes the propellant mass, not only it will provide better responsiveness in off-nominal conditions, but it will also increase the launch vehicle nominal payload, as smaller fuel tanks will lead to an increment of the vehicle carrying capacity. The objective can be equivalently formulated as maximizing the final mass (or minimizing its opposite). Thus, the considered cost function is

$$J = -m(t_f) \tag{7.6}$$

In an MPC framework, the initial condition of the OCP is updated at every control step with the real-time measurements $\tilde{\boldsymbol{x}}_0$, therefore the initial state is completely assigned

$$\boldsymbol{x}(t_0) = \tilde{\boldsymbol{x}}_0 \tag{7.7}$$

Differently from the ascent problem considered before, in the upper stage problem the initial mass is prescribed. So, to maximize the final mass, the assumption of fixed propellant consumption must be dropped for the fourth stage. Otherwise, $m(t_f)$ would be determined as the diffeence between the initial mass and the propellant and inert masses, making the optimization pointless. Therefore, the constraint in Eq. (4.34) is removed from the problem and the overall burn time of AVUM is a free optimization variable. If the optimization provides burn times shorter than nominal, then the unused propellant can be considered as additional payload mass. Conversely, if longer than nominal burn times are found, less payload can be loaded on the vehicle. Note that the propellant mass of the third stage is still supposed prescribed.

The final conditions of Phase 12 are the same as detailed in Section 4.4. Indeed, if considering a circular target orbit with semi-major axis a_{des} and inclination i_{des}, the following constraints are imposed

$$x(t_f)^2 + y(t_f)^2 + z(t_f)^2 = a_{\text{des}}^2 \tag{7.8}$$

$$v_x(t_f)^2 + v_y(t_f)^2 + v_z(t_f)^2 = v_{\text{des}}^2 \tag{7.9}$$

$$\boldsymbol{r}(t_f) \cdot \boldsymbol{v}(t_f) = 0 \tag{7.10}$$

$$x(t_f)v_y(t_f) - y(t_f)v_x(t_f) = h_{z,\text{des}} \tag{7.11}$$

To constrain the splash-down point of the spent stage, the following conditions are imposed at the end of Phase 13

$$x(t_R)^2 + y(t_R)^2 + z(t_R)^2 = R_E^2 \tag{7.12}$$

$$z(t_R) = z_{R,\text{des}} \tag{7.13}$$

Since the OCP is a multi-phase problem, proper linkage conditions must be imposed at each internal boundary. In particular, all state variables are continuous across phases, with the relevant exception of the mass, which is discontinuous at the fairing jettisoning and at the third stage separation. Thus,

$$m(t_0^{(8)}) = m(t_f^{(7)}) - m_{\text{fairing}} \tag{7.14}$$

$$m(t_0^{(9)}) = m(t_f^{(8)}) - m_{\text{dry},3} \tag{7.15}$$

As for Phase 13, its initial position and velocity are equal to the burnout conditions of the third stage, while its mass is equal to the dry mass of the stage. Thus,

$$\boldsymbol{r}(t_0^{(13)}) = \boldsymbol{r}(t_f^{(8)}) \tag{7.16}$$

$$\boldsymbol{v}(t_0^{(13)}) = \boldsymbol{v}(t_f^{(8)}) \tag{7.17}$$

$$m(t_0^{(13)}) = m_{\text{dry},3} \tag{7.18}$$

Finally, the constraint on the maximum heat flux is included in the formulation for Phases 8–12

$$\dot{Q} = \frac{1}{2}\rho v_{\text{rel}}^3 \leq \dot{Q}_{\max} \tag{7.19}$$

7.4.4 Convex Formulation

The same convexification strategy devised to convert the ascent OCP into a sequence of convex problems that quickly converges to the optimal solution can be used on the upper stage problem. This is a crucial step in the design of the guidance algorithm, as it greatly reduces the cost of the onboard optimization and enables the model predictive controller to attain high update frequencies.

First, lossless convexification strategies, namely a change of variables and a control constraint relaxation, are employed to obtain control-affine equations of motion. So, the new control is introduced

$$\boldsymbol{u} = \frac{T_a}{m}\hat{\boldsymbol{T}} \tag{7.20}$$

which includes both the thrust-to-mass ratio T_a/m and the thrust direction vector $\hat{\boldsymbol{T}}$. By replacing $\hat{\boldsymbol{T}}$ with $\boldsymbol{u}$ in Eqs. (7.3)–(7.5), control-affine dynamics are obtained

$$\dot{\boldsymbol{r}} = \boldsymbol{v} \tag{7.21}$$

$$\dot{\boldsymbol{v}} = -\frac{\mu}{r^3}\boldsymbol{r} + \boldsymbol{u} - \frac{D}{m}\hat{\boldsymbol{v}}_{\text{rel}} \tag{7.22}$$

$$\dot{m} = -\dot{m}_e \tag{7.23}$$

The new control vector must satisfy Eq. (7.2), which becomes

$$u_x^2 + u_y^2 + u_z^2 = u_N^2 \tag{7.24}$$

The additional variable u_N represents the thrust-to-mass ratio. Thus, the following nonlinear path constraint is included

$$u_N = \frac{T}{m} \tag{7.25}$$

Equation (7.24) is a nonlinear equality constraint that is suitable for a lossless relaxation, as Proposition 1 from Section 4.5.2 holds also for the upper stage problem. So, by replacing the equality with an inequality, a second-order cone constraint is formulated

$$u_x^2 + u_y^2 + u_z^2 \leq u_N^2 \tag{7.26}$$

The equations of motion (7.21)–(7.23) are linearized around a reference solution $\{\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}}, \bar{s}\}$ and augmented with a virtual control signal $\boldsymbol{q}$ to prevent artificial infeasibility

$$\frac{d\boldsymbol{x}}{d\tau} = A\boldsymbol{x} + B\boldsymbol{u} + P\boldsymbol{p} + \boldsymbol{c} + \boldsymbol{q} \tag{7.27}$$

The same penalty term J_q as in Eq. (4.71) is be added to the cost function to penalize the use of virtual controls.

The final conditions at the payload release, Eqs. (7.8)–(7.11), are linearized as

$$\bar{\boldsymbol{r}}(t_f) \cdot \bar{\boldsymbol{r}}(t_f) + 2\bar{\boldsymbol{r}}(t_f) \cdot (\boldsymbol{r}(t_f) - \bar{\boldsymbol{r}}(t_f)) = a_{\text{des}}^2 \tag{7.28}$$

$$\bar{\boldsymbol{v}}(t_f) \cdot \bar{\boldsymbol{v}}(t_f) + 2\bar{\boldsymbol{v}}(t_f) \cdot (\boldsymbol{v}(t_f) - \bar{\boldsymbol{v}}(t_f)) = v_{\text{des}}^2 \tag{7.29}$$

$$\bar{\boldsymbol{r}}(t_f) \cdot \bar{\boldsymbol{v}}(t_f) + \bar{\boldsymbol{v}}(t_f) \cdot (\boldsymbol{r}(t_f) - \bar{\boldsymbol{r}}(t_f)) + \bar{\boldsymbol{r}}(t_f) \cdot (\boldsymbol{v}(t_f) - \bar{\boldsymbol{v}}(t_f)) = 0 \tag{7.30}$$

$$\bar{v}_y(t_f)(x(t_f) - \bar{x}(t_f)) - \bar{v}_x(t_f)(y(t_f) - \bar{y}(t_f)) - \bar{y}(t_f)v_x(t_f) + \bar{x}(t_f)v_y(t_f) = h_{z,\text{des}} \tag{7.31}$$

Likewise, the constraint on the terminal radius at the splash-down, Eq. (7.12), is reformulated as

$$\bar{\boldsymbol{r}}(t_f^{(13)}) \cdot \bar{\boldsymbol{r}}(t_f^{(13)}) + 2\bar{\boldsymbol{r}}(t_f^{(13)}) \cdot (\boldsymbol{r}(t_f^{(13)}) - \bar{\boldsymbol{r}}(t_f^{(13)})) = R_E^2 \tag{7.32}$$

The heat flux constraint in Eq. (4.35) is another nonlinear expression that is replaced by its Taylor series expansion reported in Eq. (4.61). Analogously, Eq. (7.25) is linearized as

$$u_N = \frac{T_{vac} - p(\bar{\boldsymbol{r}})A_e}{\bar{m}} \left(1 - \frac{m - \bar{m}}{\bar{m}}\right) - \frac{A_e}{\bar{m}} \frac{dp(\bar{\boldsymbol{r}})}{d\boldsymbol{r}} \cdot (\boldsymbol{r} - \bar{\boldsymbol{r}}) \tag{7.33}$$

Virtual buffer zones are introduced to relax the linearized constraints and prevent artificial infeasibility. So, Eqs. (7.28)–(7.33) are grouped into a vector $\boldsymbol{\chi} = \boldsymbol{0}$ that is then relaxed as $\boldsymbol{\chi} = \boldsymbol{w}$. The vector $\boldsymbol{w}$ holds all the virtual buffers, which are highly penalized by adding the following penalty term to the cost function

$$J_w = \lambda_w \left\|\boldsymbol{w}\right\|_1 \tag{7.34}$$

with λ_w denoting the penalty weight of the virtual buffers.

Finally, to prevent excessive changes from the reference value, a soft trust region constraint is imposed on the time-lengths σ of Phases 11 and 12

$$|\sigma^{(i)} - \bar{\sigma}^{(i)}| \leq \delta^{(i)} \qquad i = 11, 12 \tag{7.35}$$

The trust radii $\delta^{(i)}$ are added to the set of optimization variables and bounded in a fixed interval $[0, \delta_{\max}^{(i)}]$. The soft trust region penalty terms are

$$J_\delta^{(i)} = \lambda_\delta^{(i)} \delta^{(i)} \qquad i = 11, 12 \tag{7.36}$$

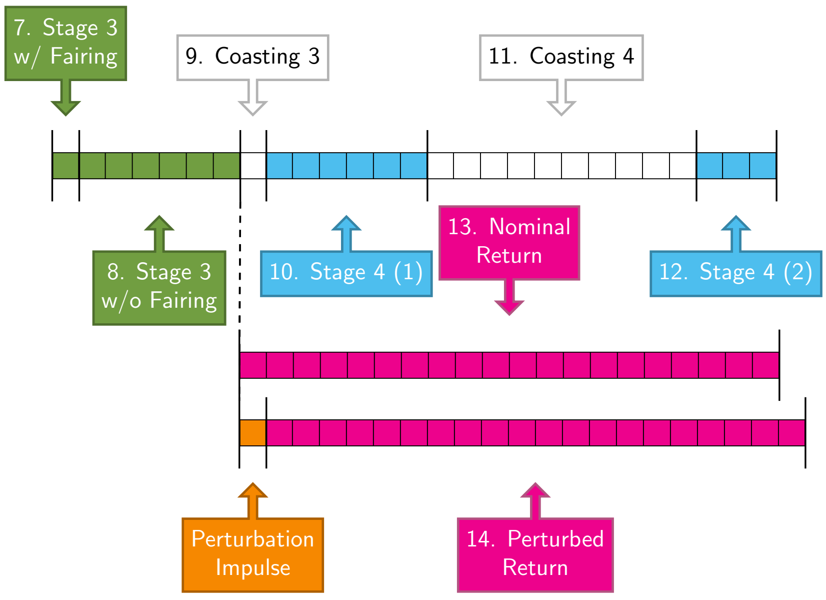


Figure 7.5. Phases of the OCP recursively solved by the multi-return MPC algorithm.

The augmented cost function of the convex problem includes the trust radii, the virtual control, and the virtual buffer penalties

$$J = -m(t_f) + J_\delta^{(11)} + J_\delta^{(12)} + J_q + J_w \tag{7.37}$$

To sum up, the convex problem to solve in the single-return strategy, denoted as $\mathcal{P}_S$, is

$$\mathcal{P}_S : \quad \begin{aligned} \min_{\boldsymbol{x},\, \boldsymbol{u}\, \boldsymbol{p}} \quad & (7.37) \\ \text{s.t.} \quad & (7.7),\ (7.13)\text{–}(7.18),\ (7.27)\text{–}(7.33),\ (7.35) \end{aligned} \tag{7.38}$$

7.5 Multi-Return Optimal Control Problem

In the multi-return MPC algorithm, a different optimal control problem is formulated. The OCP is analogous to the single-return OCP, but it includes an additional return phase, which relates to a fictitious object, with the same position and mass of the third stage at cut-off, but a different velocity obtained by a slight perturbation of fixed magnitude and unknown direction. This direction corresponds to the direction of the velocity perturbation such that the splash-down locations of the two return phases will match. Note that, by definition, such a direction is the neutral axis direction, which the algorithm computes as a by-product of the optimization at each control cycle.

7.5.1 Phase Sequence

Figure 7.5 reports the phases of the multi-return OCP. The difference with the phase sequence of the single-return OCP is the inclusion of a second return (Phase 14) perturbed with respect to the nominal return (Phase 13) by a velocity increment.

Phase 13 refers to the *nominal* return phase, that is, the one with initial position and velocity corresponding exactly to the state values at the end of Phase 8 (i.e., the nominal burnout of the SRM). Instead, Phase 14 simulates a *perturbed* return, that is, the initial conditions differ from the nominal values by a velocity increment $\boldsymbol{\Delta V}_{\mathrm{NA}}$, which has fixed magnitude but unknown direction. The magnitude of the velocity perturbation is in the order of the maximum extra ΔV that the SRM can provide after the nominal cut-off, while the direction is an optimization variable, which is further constrained to be aligned with the control direction at the end of Phase 8. In this way, at the nominal burnout, the rocket is oriented in the $\boldsymbol{\Delta V}_{\mathrm{NA}}$ direction and, since both returns are constrained to fall in the same splash-down point, $\boldsymbol{\Delta V}_{\mathrm{NA}}$ corresponds to the neutral axis direction, which is thus an output of the optimization process.

Thus, the perturbation that differentiate the perturbed return from the nominal one is an instantenous ΔV that models the effect of extra seconds of operation of the third stage compared to the nominal (i.e., minimum) cut-off time. Including the direction of $\boldsymbol{\Delta V}_{\mathrm{NA}}$ as an optimization parameter represents an expedient to account for and optimize the neutral axis maneuver in the OCP. Indeed, modeling the manevuer as a finite-duration arc in the OCP is not straightforward, as its duration depends on the SRM actual performance, which is unknown *a priori* and uncontrollable.

7.5.2 Objective and Constraints

The multi-return OCP, denoted as $\mathcal{P}_M$ shares the same objective, Eq. (7.6), and dynamics, Eqs. (7.3)–(7.5), of the single return problem $\mathcal{P}_S$. In addition, $\mathcal{P}_M$ incorporates all the constraints of $\mathcal{P}_S$ and it introduces more constraints related to the perturbed return (Phase 14).

The splash-down point of the perturbed return is constrained to the same point as the nominal return (Phase 13). Thus, the following conditions are imposed at the end of Phase 14

$$x(t_{PR})^2 + y(t_{PR})^2 + z(t_{PR})^2 = R_E^2 \tag{7.39}$$

$$z(t_{PR}) = z_{R,\mathrm{des}} \tag{7.40}$$

with t_{PR} denoting the splash-down time of the perturbed return trajectory (Phase 14).

The initial position and velocity of Phase 14 are equal to the burnout conditions of the third stage, except for a velocity increment, denoted as $\boldsymbol{\Delta V}_{\mathrm{NA}}$. Instead, the mass is equal to the dry mass of the stage. Thus,

$$\boldsymbol{r}(t_0^{(14)}) = \boldsymbol{r}(t_f^{(8)}) \tag{7.41}$$

$$\boldsymbol{v}(t_0^{(14)}) = \boldsymbol{v}(t_f^{(8)}) + \boldsymbol{\Delta V}_{\mathrm{NA}} \tag{7.42}$$

$$m(t_0^{(14)}) = m_{\mathrm{dry},3} \tag{7.43}$$

A continuity constraint between the velocity perturbation $\boldsymbol{\Delta V}_{\mathrm{NA}}$ and the final control direction at the nominal burnout of the third stage is imposed to ensure that, from the nominal burnout time onwards, the rocket is in the neutral axis direction. Thus,

$$\boldsymbol{\Delta V}_{\mathrm{NA}} / \Delta V_{\mathrm{NA}} = \hat{\boldsymbol{T}}(t_f^{(8)}) \tag{7.44}$$

Table 7.1. Initial number of discretization segments, order, and nodes in each phase of the single-return OCP.

	Phase						
	7	8	9	10	11	12	13
h	1	1	1	1	1	1	5
p	5	19	9	19	19	19	20
Nodes	6	20	10	20	20	20	101

7.5.3 Convex Formulation

The convexification of the multi-return OCP follows the same rationale used to convexify $\mathcal{P}_S$. So, all the common constraints, including the dynamics, are tackled in the same way as detailed in Section 7.4.4. The only constraint introduced in $\mathcal{P}_M$ that needs to be linearized is the constraint on the terminal radius of the perturbed return, Eq. (7.39), which is reformulated as

$$\bar{\boldsymbol{r}}(t_f^{(14)}) \cdot \bar{\boldsymbol{r}}(t_f^{(14)}) + 2\bar{\boldsymbol{r}}(t_f^{(14)}) \cdot (\boldsymbol{r}(t_f^{(14)}) - \bar{\boldsymbol{r}}(t_f^{(14)})) = R_E^2 \tag{7.45}$$

The virtual controls, virtual buffers, and the trust region on the time-lengths of Phases 10 and 11 defined for $\mathcal{P}_S$ are introduced also in $\mathcal{P}_M$. Thus, the resulting convex OCP to solve in the multi-return algorithm is

$$\begin{aligned} \mathcal{P}_M : \quad \min_{\boldsymbol{x},\, \boldsymbol{u}\, \boldsymbol{p}} \quad & (7.37) \\ \text{s.t.} \quad & (7.7),\ (7.13)\text{–}(7.15),\ (7.27)\text{–}(7.33),\ (7.35),\ (7.40)\text{–}(7.45) \end{aligned} \tag{7.46}$$

7.6 Update of the OCP

At the end of every control cycle, the OCP must be updated with the measurements coming from the navigation system. Therefore, the initial condition, Eq. (7.7), is updated with the measured (or simulated) state at that time. Also, the starting reference solution $\{\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}}, \bar{s}\}$ is replaced with the optimal solution computed at the previous time, removing the portion of flight elapsed between the two control steps.

Since the extent of the OCP time domain (i.e., the prediction horizon) reduces at each update due to the receding-horizon implementation of the MPC algorithm, the size of the discretization grid may also be reduced at every step to save computational resources. So, starting from the *hp* grids in Tables 7.1 and 7.2, the number of nodes reduces linearly over time until a minimum value of 5 nodes per segment. Analogously, the phases of the OCP should be removed as soon as they are elapsed, thus Phase 7 is removed from the OCP as soon as the fairing is jettisoned.

In the multi-return problem, Phase 8 is split into two uneven segments of different order p, as a small segment is introduced near the end of the phase to prevent numerical issues to the discretization of rapidly changing dynamics. Indeed, during the transition to the neutral axis attitude, the rocket rotates by approximately 40° in a few seconds, thus a global interpolating polynomial would feature high-frequency oscillations in that region to model such a rapid maneuver.

It is worthwhile mentioning that in the present study the time to solve the optimization problem is supposed null, but actually the simulation should introduce a delay between the measurement of the updated state and the actuation of the optimal control law. This delay may introduce significant deviations from the predicted

Table 7.2. Initial number of discretization segments, order, and nodes in each phase of the multi-return OCP.

	Phase							
	7	8	9	10	11	12	13	14
h	1	2	1	1	1	1	5	5
p	5	[19, 5]	9	19	19	19	20	20
Nodes	6	25	10	20	20	20	101	101

trajectory if the solution time is long. Nevertheless, thanks to the convexification strategy, the time required to solve the OCPs is assumed to be sufficiently brief to neglect the computation delay. Preliminary numerical results confirm the validity of this hypothesis.

7.7 Softening the Heat Flux Constraint

Recursive feasibility of the OCP is key to the design of an MPC controller, as the optimization procedure must provide a control signal at every control step. The formulated convex problem includes virtual controls and buffer zones to ensure the feasibility of each instance. Under nominal circumstances, these expedients would guarantee the feasibility of the OCP, as proved in Chapter 5. However, when the system operates in off-nominal conditions, satisfying some constraints may be impossible, and, as a result, the optimization may provide unphysical solutions that actively exploit virtual variables or no solution at all. A common strategy to avoid this undesired phenomenon is relaxing the initial condition and turn the hard constraint of Eq. (7.7) into a soft constraint with a corresponding penalty term in the augmented objective function (e.g., as proposed in Ref. [69]). This is a general approach that can be readily employed regardless of the application. However, as such, it does not exploit any knowledge on the problem characteristics. Instead, if possible, identifying the critical mission requirements and formulating a properly relaxed problem is a more efficient approach.

For the problem under investigation, only the constraint on maximum thermal flux (4.61) may cause an infeasible problem instance. Indeed, the nominal trajectory is such that the heat flux is greater than or equal to the threshold just until the fairing jettisoning. So, due to off-nominal initial conditions or external disturbances in the dynamics, the heat flux may be greater than expected at some points in the simulation and keeping it below threshold may be impossible. The infeasibility is solved by softening the heat flux constraint Eq. (4.61) as

$$\bar{\dot{Q}} + \frac{\partial \bar{\dot{Q}}}{\partial \boldsymbol{r}} \cdot (\boldsymbol{r} - \bar{\boldsymbol{r}}) + \frac{\partial \bar{\dot{Q}}}{\partial \boldsymbol{v}} \cdot (\boldsymbol{v} - \bar{\boldsymbol{v}}) \leq \dot{Q}_{\max}\left(1 + \delta_{\dot{Q}}\right) \tag{7.47}$$

where $\delta_{\dot{Q}}$ is a non-negative optimization variable that is penalized in the cost function via the term

$$J_{\dot{Q}} = \lambda_{\dot{Q}}\delta_{\dot{Q}} \tag{7.48}$$

Conversely to penalty terms on virtual variables, high values should not be assigned to the penalty weight $\lambda_{\dot{Q}}$. Indeed, if the relaxation of the heat flux constraint is highly penalized, then the optimization would recognize virtual controls as a more convenient option and thus violate the dynamics and never meet the converge criterion (3.46). Instead, a better approach consists in considering a conservative nominal threshold on the bearable heat flux $\dot{Q}_{\max}$ and accepting a (small) violation of such a threshold due to the external disturbances. So, in practice, assigning a small value to the penalty weight $\lambda_{\dot{Q}}$ appears as the most effective strategy.

7.8 Perturbation Model

To simulate the actual stage operation, a numerical integration of the original dynamical model, Eqs. (7.3)–(7.5), is carried out. To account for model uncertainty and external disturbance, white Gaussian noise is added to the equations of motion, resulting in the following stochastic differential equation (SDE)

$$d\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t)dt + G(\boldsymbol{x}, \boldsymbol{u}, t)d\boldsymbol{B}_t \tag{7.49}$$

where $\boldsymbol{B}_t$ is a n_B-dimensional Wiener process (standard Brownian motion) and the matrix G determines how the external disturbances affect the system.

In the present case, the external disturbance is supposed to be a random acceleration with zero mean and a standard deviation proportional to the thrust level by a factor α_G. In particular, two different noise contributions are considered: the first one is aligned with the thrust direction, while the second one is in a random direction. The intensity of the former is supposed to be $\alpha_{G_\parallel}$ times larger than the one of the latter, as most of the noise is usually in the thrust direction. The resulting G matrix is

$$G = \alpha_G \frac{T}{m} \begin{bmatrix} \mathbf{0}_{3\times1} & \mathbf{0}_{3\times3} \\ \alpha_{G_\parallel}\hat{\boldsymbol{T}} & \boldsymbol{I}_{3\times3} \\ \mathbf{0}_{1\times1} & \mathbf{0}_{1\times3} \end{bmatrix} \tag{7.50}$$

Note that the dimension of the Wiener process is $n_B = 4$.

The initial conditions at the stage ignition are perturbed to account for possible performance dispersions of the previous stages. The position error is modeled as a perturbation of the altitude only, as perturbations in other directions do not significantly affect performance. Errors on r are uniformly sampled in the range $[-\Delta_r, \Delta_r]$. Instead, the scattered initial conditions on the velocity are generated by uniformly sampling noise from a sphere of radius Δ_v.

As for the SRM thrust profile used in the simulations, every run considers a perturbed thrust law that features an additional burn time Δt_b of up to 5 s and a small increase in the engine total impulse ΔI_{tot} up to 0.5% with respect to the nominal one used as a reference in the OCP. By randomly scattering these quantities, a constant, nonzero thrust level is assumed in the extra seconds of operation

$$T_{\text{vac}}(t > t_{b,\text{nom}}) = \frac{\Delta I_{\text{tot}}}{\Delta t_b} \tag{7.51}$$

The nominal thrust profiles and some examples of perturbed laws are shown in Fig. 7.6.

7.9 Numerical Results

In this section, numerical results are presented to assess the performance and robustness of the MPC framework. The single and multi-return guidance strategies are compared and the advantages of including a perturbed return phase in the guidance OCP are investigated.

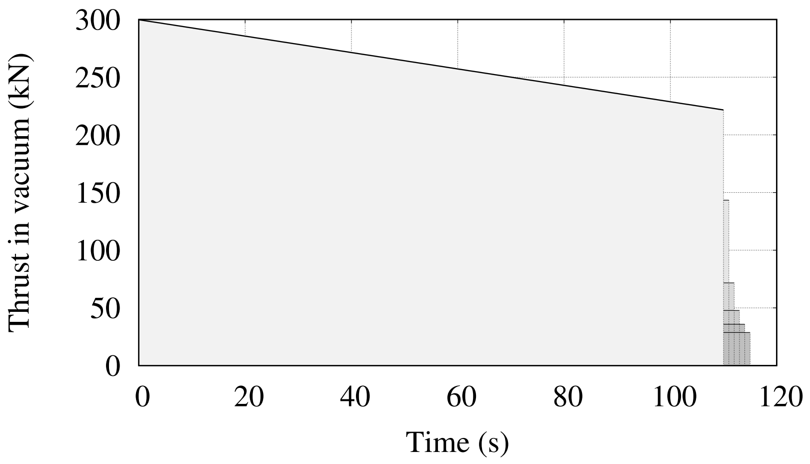


Figure 7.6. Nominal and perturbed thrust profiles.

The algorithms were implemented in C++ and the OCP was solved using Gurobi [147] as second-order cone programming solver. All the simulations ran on a computer equipped with Intel® Core™ i7-9700K CPU @ 3.60 GHz.

7.9.1 Problem Data

The considered case study is the same as in Chapter 5, that is, an ascent trajectory from an equatorial launch base to a 700 km circular orbit with $i_{\text{des}} = 90°$, with the relevant difference that the splash-down latitude of the third stage is arbitrarily constrained to $\varphi_{R,\text{des}} = 60°$. The same data were used to model the launch vehicle stages and aerodynamic coefficients, with the exception of the uncertain additional burn time of the third stage, which is modeled as detailed in Section 7.8.

The duration of the phases is the same as reported in Table 5.2. In particular, the third stage nominal burn time is $t_{b,3} = 110$ s, which is also the minimum burn time considered in the simulations, while delayed cut-offs (up to 5 s) can provide a maximum velocity increment of $\Delta V_{\text{NA}} = 25$ m/s. Since the fairing is released $\Delta t^{(7)} = 5.4$ s after ignition, Phase 8 lasts $\Delta t^{(8)} = 104.6$ s. The duration of the coasting arc that follows the third stage separation (Phase 9) is fixed to $\Delta t^{(9)} = 15.4$ s, while the time-lengths of all the other phases are unconstrained.

In the OCP, the threshold on the heat flux used to compute the solution is set to $\dot{Q}_{\max} = 900\,\text{W/m}^2$. However, $\dot{Q}_{\max}$ is a nominal value and is lower than the actual bearable heat flux that must not be exceeded in simulations ($1135\,\text{W/m}^2$) [58]. Therefore, the softening of the heat flux constraint as in Eq. (7.47) with a small value of the penalty coefficient ($\lambda_{\dot{Q}} = 10^{-2}$) is justified.

The values of the other parameters of the successive convexification algorithm are the same used in Chapter 5. Specifically, the penalty weights on the time-lengths, $\lambda_\delta^{(11)}$ and $\lambda_\delta^{(12)}$, were set to 10^{-4}, while, the penalty weights of the virtual variables were set to $\lambda_q = \lambda_w = 10^4$. The tolerances on the convergence criteria, Eqs. (3.45)–(3.47), were prescribed as $\epsilon_{\text{tol}} = 10^{-4}$ and $\epsilon_f = 10^{-6}$.

7.9.2 Nominal Trajectories

The *nominal* trajectories are the optimal trajectories computed in a fully deterministic scenario (i.e., without accounting for uncertainties or disturbances). These are used as initial reference solutions in the first MPC cycle. The nominal trajectories of the single and multi-return approaches can be computed via the convex optimization approach described in Chapter 4, with the only difference of including Phase 14 and the related constraints in the OCP for the multi-return scenario.

The nominal solutions also provide valuable information on the (nominal) performance of the two strategies. For instance, the achievable payload mass considering only the nominal return in the optimization is $m_{\text{pl}}^{\star} = 1400.1\,\text{kg}$. Instead, when including a perturbed return, the payload reduces to $m_{\text{pl}}^{\star} = 1328.3\,\text{kg}$. The carrying capacity of the multi-return strategy is smaller due to the fact that constraining also the perturbed return forces the third stage to be oriented in the neutral axis direction at burnout, while the single-return problem does not take into account such a requirement. Thus, the difference in the payload mass represents the (nominal) cost of performing the neutral axis maneuver.

7.9.3 Monte Carlo Campaigns

A Monte Carlo analysis on the combined effect of off-nominal initial conditions, in-flight disturbance, and uncertain thrust profiles, as described in Section 7.8, was carried out for both the single-return strategy and the multi-return one. When relying on the single-return guidance strategy, in case of longer-than-nominal cut-off time, the control direction is kept constant and equal to the attitude at the nominal burnout for all the extra seconds of operation. This allows to compare the multi-return guidance strategy with the single-return one, which optimizes only the nominal return of the spent stage.

The initial radius error is scattered uniformly in a range of values $[-\Delta_r, \Delta_r]$ with $\Delta_r = 500\,\text{m}$ and the initial velocity error is sampled from a sphere of radius $\Delta_v = 40\,\text{m/s}$. Three levels of in-flight disturbance were considered as reported in Table 7.3 and called Low (L), Medium (M), and High (H). In each case, the standard deviation of the noise in the thrust direction is 5 times greater than the one in a random direction, thus $\alpha_{G_{\|}} = 5$.

Table 7.3. Standard deviations of the random-direction Gaussian in-flight disturbance in terms of T/m percentage.

Parameter	Case L	Case M	Case H
α_G	0.25 ‰	0.5 ‰	1 ‰

In each simulation, the additional burn time Δt_b is sampled uniformly in the range $[0, 5]$s, while the additional total impulse ΔI_{tot} is a fraction of the nominal one uniformly sampled in the range $[0, 5]$‰. The update frequency of the MPC is set to 1 Hz, meaning that the OCP is solved every $T = 1\,\text{s}$.

Four hundred independent MPC simulations were carried out for each disturbance intensity level α_G for both the single and multi-return approaches. The stats on the achieved payload mass of the Monte Carlo campaigns are reported in Table 7.4 in terms of minimum, mean, maximum, and standard deviation σ. The attainable payload mass is estimated for each run by using the conditions at the burnout of the SRM as initial conditions and solving the ascent OCP relative to Phases 9–12. As expected, for both algorithms, as the intensity of the in-flight disturbance increases, larger dispersions on the payload mass are observed. The payload mass ranges of the two algorithms are of comparable size, but the multi-return average payload is smaller by approximately 75 kg due to the cost of the (induced) neutral axis

Table 7.4. Results of the Monte Carlo campaigns (payload mass).

Method	**Case**	**Payload mass (kg)**			
		Min	Mean	Max	σ
Single return	L	1359.19	1409.74	1449.70	17.10
	M	1335.17	1405.21	1482.21	24.41
	H	1254.00	1407.49	1503.37	39.77
Multi return	L	1278.39	1331.44	1375.78	18.16
	M	1266.97	1329.81	1389.62	23.56
	H	1215.90	1331.64	1456.62	41.06

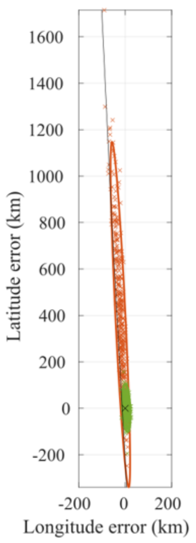
Table 7.5. Results of the Monte Carlo campaigns (splash-down point).

Method	**Case**	φ_R **(deg)**				**Footprint size (km)**
		Min	Mean	Max	σ	
Single return	L	59.74	63.60	68.15	1.87	1021.88
	M	59.30	63.33	68.65	2.00	1093.53
	H	57.78	63.63	75.40	2.73	1488.32
Multi return	L	59.72	60.00	60.21	0.09	50.38
	M	59.17	60.01	60.62	0.20	106.44
	H	58.25	60.02	61.41	0.38	209.48

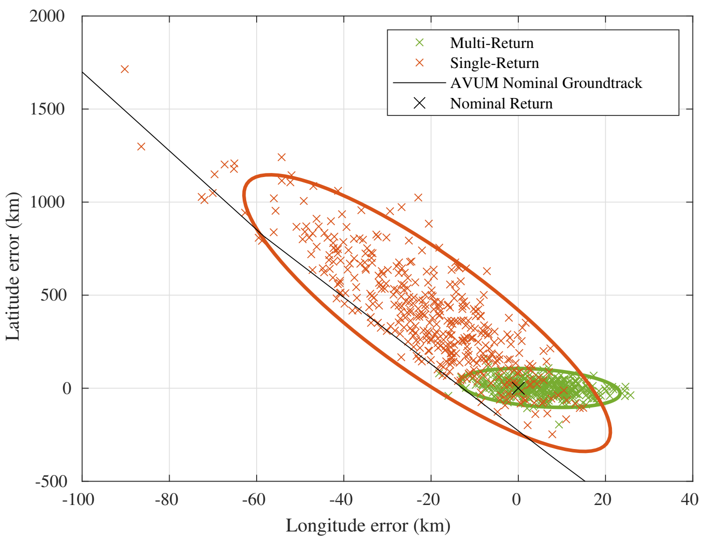
maneuver.

It is worth noting that the average payload mass attained by both algorithms is slightly greater than the nominal values ($m^\star_{\text{pl}} = 1400.1\,\text{kg}$ for the single-return and $m^\star_{\text{pl}} = 1328.3\,\text{kg}$ for the multi-return) due to the additional I_{tot} that provides more energy to the system. The gain in payload mass of the multi-return strategy is smaller due to the fact that the additional impulse is provided in the neutral axis direction, which is not optimal.

The major difference in the behavior of the two strategies lies in the splash-down point, as reported in Table 7.5. The splash-down points were evaluated by forward propagation of the equations of motion, Eqs. (7.3)–(7.5), starting from the simulated burnout state. The additional impulse provided by the SRM shifts the mean splash-down point attained by the single-return strategy farther by approximately 3.6° regardless of the disturbance intensity level, meaning that this deviation depends only on the SRM performance dispersion. Instead, the multi-return algorithm retains the mean splash-down location with great accuracy (the largest error is 0.02° in case H) thanks to the fact that the thrust vector is oriented in the neutral axis direction during the extra seconds of operation. The range of the spent stage impact latitude increases for both approaches as the disturbance intensity increases, but, while the lower bounds are comparable, the upper bounds of the return points corresponding to the single-return strategy are much more distant from the desired value. This asymmetry is due to the fact that the additional SRM impulse moves the return point only farther; thus, the two disturbances add up for the single-return approach. Instead, the neutral axis maneuver compensates for the SRM performance uncertainty and the range of the attained φ_R is symmetric. Also, the dispersion of



(a) Same scale for latitude and longitude.



(b) Zoomed scale of the longitude error.

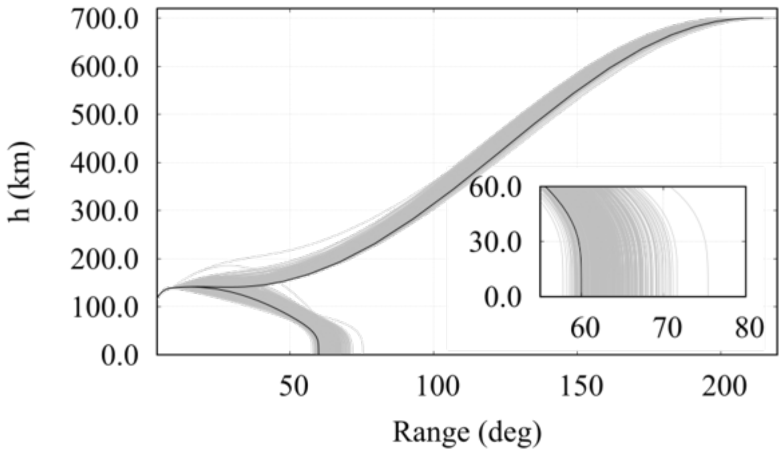
Figure 7.7. Splash-down footprints for Case H.

the return point is significantly reduced when employing the multi-return algorithm, with standard deviations σ one order of magnitude smaller than in the single-return approach.

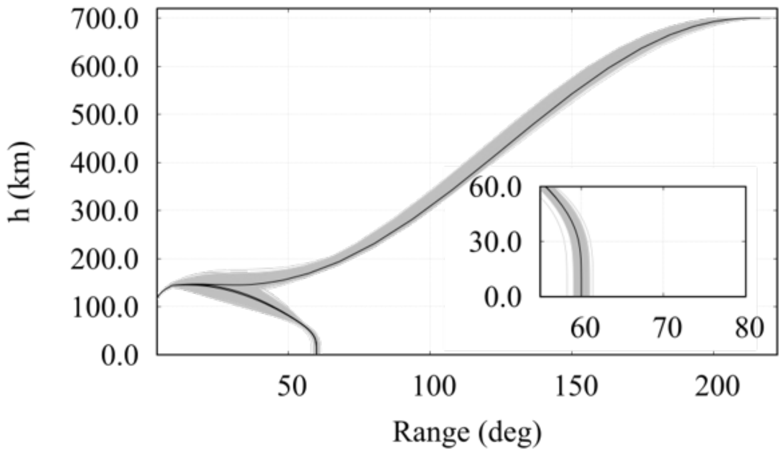
Table 7.5 also reports the footprint length, which is computed as the major axis of the 95% confidence ellipse of the simulated splash-down points. The footprints for case H are illustrated in Figure 7.7 for both approaches. The figure clearly shows that the single-return strategy leads to a greater dispersion of the splash-down point. Figure 7.7a shows that both footprints are almost aligned with the South-North direction, as the dispersion of the latitude is much greater than the one of the longitude. Indeed, the latter is only due to the Earth's rotation, since longer return trajectories fall further west and shorter ones further east, due to the different return times, causing a slight horizontal dispersion. In Figure 7.7b, the axes use different scales to better visualize the dispersion of the impact points.

The trajectories for case H, which is associated with the largest envelopes, are reported in Fig. 7.8 and compared with the nominal one, which is denoted by the black line. Besides the altitude profile of the simulated third stage operation, the figure also illustrates the optimal solution of the ascent OCP defined by Phases 9 to 12 and the simulation of the spent stage return with the burnout state as initial condition. The trajectories of both guidance strategies deviate significantly from the nominal trajectory during the central portion of the flight, as the MPC approach autonomously recomputes the optimal path to compensate for the encountered disturbances. Also, the single-return strategy features a much greater dispersion on the return point, as shown in Fig. 7.8a.

The implemented control histories during the third stage flight are shown in Fig. 7.9 in terms of the elevation angle, which is defined as the angle between the thrust direction $\hat{\boldsymbol{T}}$ and the local horizontal. The elevation profiles are close to the nominal controls for most of the stage operation, with larger deviations toward the end. Indeed, due to the external disturbance, meeting the splash-down constraint requires larger control actions as the burnout time approaches. Figure 7.9b shows

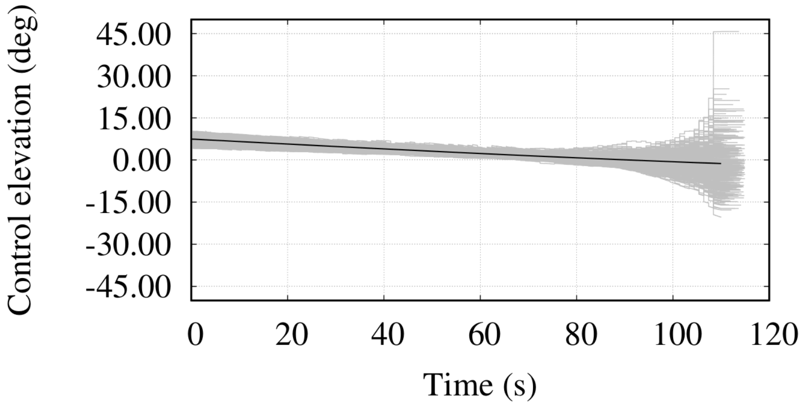


(a) Single-return algorithm.

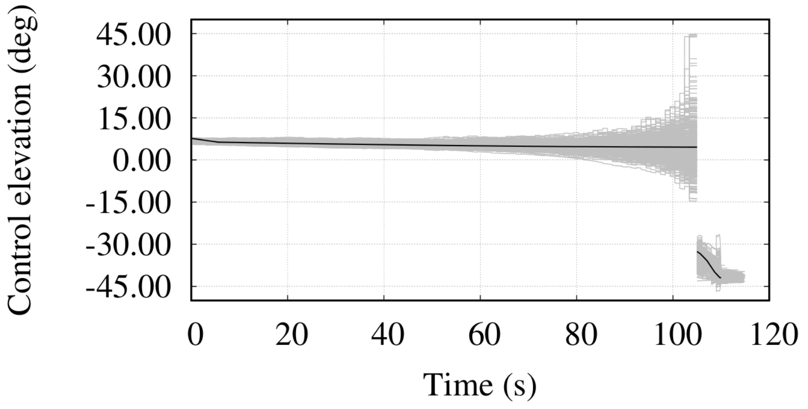


(b) Multi-return algorithm.

Figure 7.8. Altitude profiles for Case H.



(a) Single-return algorithm.



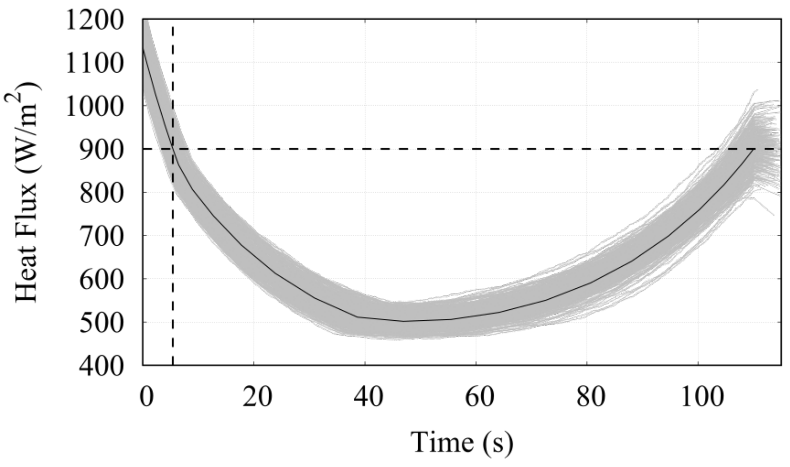
(b) Multi-return algorithm.

Figure 7.9. Control elevation profiles for Case H.

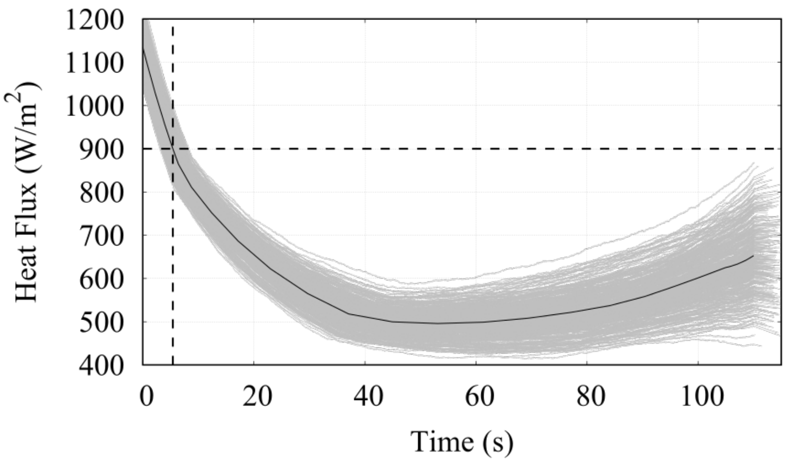
the different orientation of the neutral axis direction compared to the optimal one, as the maneuver requires the rocket to quickly rotate by approximately 40° downward.

The encountered heat flux profiles are illustrated in Fig. 7.10. Due to the relaxation introduced in Eq. (7.47), the nominal threshold $\dot{Q}_{\max}$ is violated at some points in the flight. Nevertheless, the heat flux is never greater than $1000\,\mathrm{W/m^2}$, except for a few simulations of the single-return strategy in the proximity of the burnout where, due to the additional seconds of SRM operations, the relative velocity increases more than predicted. Considering that the payload can bear a flux of up to $1135\,\mathrm{W/m^2}$, the thermal requirement is satisfied within an acceptable error. It is worth remarking that the violation is not avoidable by the guidance strategy, as it depends on the encountered in-flight disturbance and the scattering of the initial conditions, since increased initial velocity and reduced altitude expose the system to more critical thermal conditions.

As for the algorithm computational efficiency, the average solution time of the single-return OCP was 1.50 s, while solving the multi-return problem required 3.20 s on average. The multi-return OCP is thus more expensive to solve, due to the additional return phase and the related constraints that significantly increase the problem dimension and complexity. However, the multi-return strategy is the only one that is robust to dispersion of the SRM cut-off time. So, the multi-return strategy is the unique option when dealing with a system such as VEGA's third stage, for



(a) Single-return algorithm.



(b) Multi-return algorithm.

Figure 7.10. Heat flux profiles for Case H.

which the uncertainty on the SRM burn time is significant. Nevertheless, if the burnout time is known in advance with greater accuracy, the single-return strategy is the best candidate, as it is computationally less demanding, thus allows for higher update frequencies and thus more robustness than a multi-return approach.

It is worth mentioning that at every control step of every simulation, the successive convexification algorithm always converged to a solution, proving the great robustness of the devised algorithm to off-nominal starting conditions. Indeed, compared to the sensitivity study in Section 5.3, the initial guess in real-time guidance applications is much more accurate, as the mismatch between the flown and nominal trajectory, which is the initial reference solution of the successive convexification algorithm, is small, thus a success rate of 100% is achieved. This achievement is also due to the inclusion of virtual variables and the softening of the heat flux constraint, which are essential to prevent the formulation of an infeasible problem instance.

As a final remark, the solution times of both OCPs appear incompatible with a 1 Hz update frequency, but this issue could be easily addressed by using optimized software, dedicated hardware, and custom convex solvers, for which the computation times are expected to be at least one order of magnitude lower. Also, previous investigations on this topic show that good MPC performance could be obtained even with less frequent updates, in the range 1–3 s [150].

Chapter 8

Conclusions

This thesis investigated a convex approach to the optimization of the ascent trajectory of a multistage launch vehicle. To tackle the intrinsic nonconvexities of the problem, several convexification methods were examined. These include a convenient change of variables, to produce control-affine dynamics, combined with a lossless relaxation of a nonlinear control constraint into a second-order cone constraint. Leveraging optimal control theory, the thesis provided rigorous proof of the exactness of said relaxation approach, showing the equivalence between the relaxed problem and the original one. Thus, the benefits of such a convexification method were apparent, as the novel formulation reduces the complexity of the problem and shares the same solution as the original problem.

Successive linearization was essential to tackle the nonlinear dynamics and constraints in a systematic way. However, differently from lossless convexification methods, the linearization introduces an approximation error and convergence of the resulting sequential algorithm cannot be theoretically guaranteed. Thus, several methods to enhance the convergence of successive linearization were examined and compared, including the addition of virtual variables, traditional trust regions, and a novel method for the update of the reference solution called filtering. The latter is an original contribution of the present research, which proved to be greatly effective for the launch vehicle problem but is a general strategy that can be beneficial to any application. Indeed, filtering does not depend on or modify the problem formulation by any means, yet it provides algorithmic robustness to the sequential procedure thanks to the reduced weight assigned to new, potentially diverging, solutions in the update of the reference one.

A VEGA-like launch vehicle was taken as a case study due to the particularly arduous requirements it poses to the trajectory design task. The main difficulty is related to the need to predict and actively constrain the splash-down location of the third spent stage to a safe region. To account for such a requirement, a simulation of the reentry trajectory of the burned-out stage was included in the optimal control problem. Despite the additional phase increases the complexity and dimension of the optimization problem, the convexification strategy proved to be reliable and computationally efficient, and an analysis on the launch vehicle performance sensitivity to the splash-down location of the third stage was carried out, showing that moving the return point of the spent stage can significantly change the mission profile and is thus a major criticality of the VEGA rocket. Also, the ascent of a two-stage launch vehicle, inspired to SpaceX's Falcon 9 rocket, was considered as a second case study to assess the effectiveness of the convex optimization approach in a different mission scenario and even for a different launch vehicle configuration, eventually proving the generality of the investigated methodology.

The convex approach features two key merits: first, the computational burden of the overall procedure is significantly reduced compared to traditional direct optimization methods; second, the sensitivity to the initialization does not represent a concern, as convergence can be achieved even starting from a rough reference trajectory. Therefore, it represents a fast and reliable alternative to traditional optimization methods, which, in turn, often manifest high sensitivity to the supplied first guess solution or require a large computational effort to achieve convergence.

These beneficial properties make the convex approach potentially suitable for optimization-based real-time guidance, as brief computational times and reliability are key elements in time-critical applications. So, this thesis investigated the implementation of a closed-loop guidance algorithm, based on embedding the convex optimization approach into a model predictive control framework, for the upper stage of a launch vehicle.

In particular, two guidance strategies were devised according to the in-flight uncertainties that the system is expected to be subject to. The first one was proven to provide robustness to generic model errors and random external disturbances by recursively solving an optimal control problem that comes directly from the complete ascent trajectory optimization problem. However, to robustly ensure a splash-down constraint in the presence of uncertainty on the stage cut-off time, which is often non-negligible for solid rocket motors, the optimal control problem associated with the second guidance strategy was modified to incorporate an additional return phase, perturbed by a velocity increment and constrained to the same splash-down point as the nominal return. In this way, the solution of the OCP provides the neutral axis attitude, which can thus be reliably computed and updated onboard and with minor computational cost, resulting in better system performance, as such a strategy leverages the information on the encountered flight conditions, and cut the lengthy and time-consuming validation and verification tasks that a pre-scheduled neutral axis maneuver would require before each launch to ensure robustness.

Monte Carlo campaigns have been carried out to investigate the effectiveness of the proposed guidance strategies in presence of in-flight disturbances, off-nominal operating conditions, and uncertain SRM performance. The optimal control of the third stage of a VEGA-like launch vehicle was investigated as case study. The results of the Monte Carlo campaigns provide significant evidence of the robustness of the MPC framework. In this respect, the reliability of the convex optimization approach plays a pivotal role since it can autonomously recompute a revised optimal trajectory even starting from an inaccurate initial guess. Furthermore, the computational efficiency of the solution process allows for high update frequencies that allow to effectively compensate for mismatches between the predicted state and the measured one and, thus, set up a robust and efficient guidance algorithm.

8.1 Future Work

Future research may be directed toward studying even more realistic scenarios, incorporating additional nonconvex boundary or path constraints. For instance, visibility requirements were not accounted for in this study, but are essential to ensure that the launch vehicle is trackable during the most critical phases of the ascent. These may be posed as (concave) minimum altitude requirements that the rocket must meet to be visible from a given ground station or may require the addition of nonlinear mission-dependent constraints in the formulation. Also, the flexibility of the considered convexification approach to different target orbits, including non-circular ones, and launch vehicle configurations can be investigated with minor modifications to the OCP. An interesting research direction would be the concurrent optimization of the thrust law of the stages and the ascent trajectory. In this way, the optimization would not be limited to the trajectory design task, but it would aim at designing a comprehensively optimal launch system. Naturally, all these additional requirements would increase the complexity of the problem; so, mindful convexification approaches should be devised to retain the overall computational efficiency of the solution process.

As for the performance of the successive convexification algorithm, greater convergence rates can be achieved if better initialization strategies are devised. Indeed, due to the high sensitivity to the decision parameters, designing a launch vehicle ascent trajectory from scratch is quite arduous. The thesis investigated a three-step continuation strategy that, however, only partially mitigated the need for a good initialization. So, more robust strategies to easily design a suitable initial guess for the optimization process should be investigated. Also, further speed-ups are expected if custom convex programming solvers are used instead of off-the-shelf libraries and the algorithm runs on dedicated hardware.

Finally, even though the preliminary results on the effectiveness of the devised closed-loop guidance of a launch vehicle upper stage are encouraging, an extensive validation and verification campaign is still necessary to demonstrate the practical real-time implementation of the proposed MPC algorithms. Nevertheless, the continuous effort to develop more performing computing hardware, which will imply higher update frequencies, combined with the ongoing research in convex optimization, allows to look forward to the future with great confidence on the use of computational, optimization-based algorithms for the guidance of launch vehicles to come.

Appendix A

Linearization Matrices for the Low-Thrust Problem

With reference to Eq. (3.78), A is a 7×7 matrix and the analytical expressions of its non-null elements of are reported below:

$$A_{14} = 1 \tag{A.1}$$

$$A_{21} = -\frac{v_\theta}{r^2 \cos\varphi} \tag{A.2}$$

$$A_{23} = \frac{v_\theta \tan\varphi}{r \cos\varphi} \tag{A.3}$$

$$A_{25} = \frac{1}{r \cos\varphi} \tag{A.4}$$

$$A_{31} = -\frac{v_\varphi}{r^2} \tag{A.5}$$

$$A_{36} = \frac{1}{r} \tag{A.6}$$

$$A_{41} = \frac{2\mu}{r^3} - \frac{v_\varphi^2 + v_\theta^2}{r^2} \tag{A.7}$$

$$A_{45} = \frac{2v_\theta}{r} \tag{A.8}$$

$$A_{46} = \frac{2v_\varphi}{r} \tag{A.9}$$

$$A_{51} = -\frac{v_\varphi v_\theta \tan\varphi}{r^2} + \frac{v_r v_\theta}{r^2} \tag{A.10}$$

$$A_{53} = \frac{v_\varphi v_\theta}{r \cos^2\varphi} \tag{A.11}$$

$$A_{54} = -\frac{v_\theta}{r} \tag{A.12}$$

$$A_{55} = \frac{v_\varphi \tan\varphi}{r} - \frac{v_r}{r} \tag{A.13}$$

$$A_{56} = \frac{v_\theta \tan\varphi}{r} \tag{A.14}$$

$$A_{61} = \frac{v_\theta^2 \tan\varphi}{r^2} + \frac{v_\varphi v_r}{r^2} \tag{A.15}$$

$$A_{63} = -\frac{v_\theta^2}{r \cos^2\varphi} \tag{A.16}$$

$$A_{64} = -\frac{v_\varphi}{r} \tag{A.17}$$

$$A_{65} = -\frac{2v_\theta \tan\varphi}{r} \tag{A.18}$$

$$A_{66} = -\frac{v_r}{r} \tag{A.19}$$

Analogously, the B matrix is

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -\frac{1}{c} \end{bmatrix} \tag{A.20}$$

Appendix B

Proof of Lossless Relaxation

The proof that the optimal solution of $\mathcal{P}_R$, defined in Eq. (4.54), is the optimal solution also of problem $\mathcal{P}_A$, defined in Eq. (4.53), is here provided. Proposition 1 is proved by contradiction, using the direct adjoint approach [151]. First, the Hamiltonian H must be introduced, which is

$$H = \boldsymbol{\lambda}\cdot\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u},t) = \boldsymbol{\lambda}_r\cdot\boldsymbol{v}+\boldsymbol{\lambda}_v\cdot\left(-\frac{\mu}{r^3}\boldsymbol{r}+\boldsymbol{u}+\frac{T_b-D}{m}\hat{\boldsymbol{v}}_{\text{rel}}+\frac{T_c}{m}\hat{\boldsymbol{r}}\right)+\lambda_m\left(-\dot{m}_e\right) \tag{B.1}$$

where $\boldsymbol{\lambda}(t) = \begin{bmatrix}\boldsymbol{\lambda}_r^T & \boldsymbol{\lambda}_v^T & \lambda_m\end{bmatrix}^T$ is the costate vector, conveniently split in position, velocity, and mass subvectors. To take into account the path constraints, the Lagrangian L is defined as

$$L = H - \mu_Q(\dot{Q}-\dot{Q}_{\max}) - \mu_u(u_x^2+u_y^2+u_z^2-u_N^2) \tag{B.2}$$

where $\mu_u(t)$ and $\mu_Q(t)$ are the Lagrange multipliers associated with the constraints from Eqs. (4.52) and (4.35), respectively. The optimal solution must satisfy the complementary slack conditions. Thus,

$$\mu_Q \geq 0, \quad \mu_Q(\dot{Q}-\dot{Q}_{\max}) = 0 \tag{B.3}$$

$$\mu_u \geq 0, \quad \mu_u\left(u_x^2+u_y^2+u_z^2-u_N^2\right) = 0 \tag{B.4}$$

According to Pontryagin's minimum principle, the optimal control $\boldsymbol{u}^\star$ must be such that

$$\boldsymbol{u}^\star = \arg\min_{\boldsymbol{u}\in\mathcal{U}} H(\boldsymbol{x}^\star,\boldsymbol{u},\boldsymbol{\lambda},t) \tag{B.5}$$

where $\mathcal{U}$ is the set of controls that satisfies Eq. (4.52). Thus,

$$\mathcal{U}(\boldsymbol{x},t) = \left\{(u_x,u_y,u_z) : u_x^2+u_y^2+u_z^2 \leq \left(\left(T_{\text{vac}}-pA_e\right)/m\right)^2\right\} \tag{B.6}$$

where u_N was replaced with Eq. (4.51). The Karush–Kuhn–Tucker (KKT) condition for minimizing the Hamiltonian over $\boldsymbol{u}\in\mathcal{U}$ is:

$$\frac{\partial L}{\partial \boldsymbol{u}} = \boldsymbol{\lambda}_v - 2\mu_u\boldsymbol{u} = 0 \tag{B.7}$$

The costate equations are:

$$\dot{\boldsymbol{\lambda}} = -\frac{\partial L}{\partial \boldsymbol{x}} = -\left[\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right]^T\boldsymbol{\lambda} - \mu_Q\frac{\partial \dot{Q}}{\partial \boldsymbol{x}} - 2\mu_u u_N\frac{\partial u_N}{\partial \boldsymbol{x}} \tag{B.8}$$

The Jacobian matrix of the dynamics has the following structure:

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} = \begin{bmatrix} \mathbf{0}_{3\times3} & I_{3\times3} & \mathbf{0}_{3\times1} \\ A_{vr} & A_{vv} & \boldsymbol{a}_{vm} \\ \mathbf{0}_{1\times3} & \mathbf{0}_{1\times3} & \mathbf{0}_{1\times1} \end{bmatrix} \tag{B.9}$$

where A_{vr} and A_{vv} are full (i.e., without any identically null elements) 3×3 matrices and $\boldsymbol{a}_{vm}$ is a full 3×1 vector. The costate equations expand to:

$$\dot{\boldsymbol{\lambda}}_r = -A_{vr}^T \boldsymbol{\lambda}_v - \mu_Q \frac{\partial \dot{Q}}{\partial \boldsymbol{r}} - 2\mu_u u_N \frac{\partial u_N}{\partial \boldsymbol{r}} \tag{B.10}$$

$$\dot{\boldsymbol{\lambda}}_v = -\boldsymbol{\lambda}_r - A_{vv}^T \boldsymbol{\lambda}_v - \mu_Q \frac{\partial \dot{Q}}{\partial \boldsymbol{v}} \tag{B.11}$$

$$\dot{\lambda}_m = -\boldsymbol{a}_{vm} \cdot \boldsymbol{\lambda}_v - 2\mu_u u_N \frac{\partial u_N}{\partial m} \tag{B.12}$$

A transversality condition useful for the present demonstration is

$$\lambda_m(t_f) = 1 \tag{B.13}$$

Finally, since the terminal time t_f is free, the Lagrangian is null at the final boundary,

$$L(t_f) = 0 \tag{B.14}$$

Now, we show that the constraint in Eq. (4.52) must be active almost everywhere. In particular, first we assume that (4.52) is strictly satisfied, i.e., it is not active, and then argue that it is not possible since it generates a contradiction. When Eq. (4.52) is not active, the complementary condition in Eq. (B.4) requires that $\mu_u = 0$. Thus, the KKT condition from Eq. (B.7) becomes

$$\boldsymbol{\lambda}_v(t) = \mathbf{0} \tag{B.15}$$

Leveraging Assumption 1, the heat flux constraint is inactive almost everywhere. Thus, according to Eq. (B.3):

$$\mu_Q = 0 \tag{B.16}$$

Replacing Eqs. (B.15), (B.16), and $\mu_u = 0$ in the costate equations (B.10)–(B.12) leads to

$$\boldsymbol{\lambda}_r(t) = \mathbf{0} \tag{B.17}$$

$$\dot{\lambda}_m = 0 \tag{B.18}$$

Because of Eqs. (B.15) and (B.17), the Lagrangian condition (B.14) requires that

$$\lambda_m(t_f) = 0 \tag{B.19}$$

However, this violates Eq. (B.13), thus generating a contradiction. This contradiction proves that the optimal solution of $\mathcal{P}_R$ must satisfy Eq. (4.52) with the equality sign. □

Bibliography

[1] Goddard, R. H., "A Method of Reaching Extreme Altitudes," *Nature*, Vol. 105, No. 2652, 1920, pp. 809–811. https://doi.org/10.1038/105809a0.

[2] Tsien, H. S., and Evans, R. C., "Optimum Thrust Programming for a Sounding Rocket," *Journal of the American Rocket Society*, Vol. 21, No. 5, 1951, pp. 99–107. https://doi.org/10.2514/8.4372.

[3] Lawden, D. F., "Stationary Rocket Trajectories," *The Quarterly Journal of Mechanics and Applied Mathematics*, Vol. 7, No. 4, 1954, pp. 488–504. https://doi.org/10.1093/qjmam/7.4.488.

[4] Tsiotras, P., and Kelley, H. J., "Goddard Problem with Constrained Time of Flight," *Journal of Guidance, Control, and Dynamics*, Vol. 15, No. 2, 1992, pp. 289–296. https://doi.org/10.2514/3.20836.

[5] Seywald, H., and Cliff, E. M., "Goddard Problem in Presence of a Dynamic Pressure Limit," *Journal of Guidance, Control, and Dynamics*, Vol. 16, No. 4, 1993, pp. 776–781. https://doi.org/10.2514/3.21080.

[6] Spurlock, F., and Williams, C. H., "DUKSUP: A Computer Program for High Thrust Launch Vehicle Trajectory Design & Optimization," *50th AIAA/ASME/SAE/ASEE Joint Propulsion Conference*, 2014. https://doi.org/10.2514/6.2014-3671.

[7] Jurovics, S., "Optimum Steering Program for the Entry of a Multistage Vehicle Into a Circular Orbit," *ARS Journal*, Vol. 31, No. 4, 1961, pp. 518–523. https://doi.org/10.2514/8.5545.

[8] Spurlock, O. F., and Teren, F., "Payload Optimization of Multistage Launch Vehicles," Tech. Rep. D-3191, NASA, Cleveland, Ohio, 1966.

[9] Spurlock, O. F., and Teren, F., "Optimum Launch Trajectories for the ATS-E Mission," *Journal of Spacecraft and Rockets*, Vol. 8, No. 12, 1971, pp. 1202–1208. https://doi.org/10.2514/3.59787.

[10] Colasurdo, G., Pastrone, D., and Casalino, L., "Optimization of Rocket Ascent Trajectories Using an Indirect Procedure," *Guidance, Navigation, and Control Conference*, 1995. https://doi.org/10.2514/6.1995-3323.

[11] Martinon, P., Bonnans, F., Laurent-Varin, J., and Trelat, E., "Numerical Study of Optimal Trajectories with Singular Arcs for an Ariane 5 Launcher," *Journal of Guidance, Control, and Dynamics*, Vol. 32, No. 1, 2009, pp. 51–55. https://doi.org/10.2514/1.37387.

[12] Casalino, L., and Pastrone, D., "Optimization of Hybrid Propellant Mars Ascent Vehicle," *50th AIAA/ASME/SAE/ASEE Joint Propulsion Conference*, 2014. https://doi.org/10.2514/6.2014-3953.

[13] Hanson, J., Shrader, M., and Cruzen, C., "Ascent Guidance Comparisons," *AIAA Guidance, Navigation, and Control Conference*, Scottsdale, AZ, 1994. https://doi.org/10.2514/6.1994-3568.

[14] Smith, I. E., "General Formulation of the Iterative Guidance Mode," Tech. Rep. TM X-53414, NASA, 1966.

[15] McHenry, R. L., Long, A. D., Cockrell, B. F., Thibodeau III, J. R., and Brand, T. J., "Space Shuttle Ascent Guidance, Navigation, and Control," *Journal of the Astronautical Sciences*, Vol. 27, No. 1, 1979, pp. 1–38.

[16] Chang, H. P., "Spherical Atmospheric Linear Tangent (SATLIT) Guidance," Tech. Rep. NAS 8-37814, NASA, 1993.

[17] Leung, M. S. K., and Calise, A. J., "Hybrid Approach to Near-Optimal Launch Vehicle Guidance," *Journal of Guidance, Control, and Dynamics*, Vol. 17, No. 5, 1994, pp. 881–888. https://doi.org/10.2514/3.21285.

[18] Calise, A. J., Melamed, N., and Lee, S., "Design and Evaluation of a Three-Dimensional Optimal Ascent Guidance Algorithm," *Journal of Guidance, Control, and Dynamics*, Vol. 21, No. 6, 1998, pp. 867–875. https://doi.org/10.2514/2.4350.

[19] Gath, P. F., and Calise, A. J., "Optimization of Launch Vehicle Ascent Trajectories with Path Constraints and Coast Arcs," *Journal of Guidance, Control, and Dynamics*, Vol. 24, No. 2, 2001, pp. 296–304. https://doi.org/10.2514/2.4712.

[20] Calise, A. J., and Brandt, N., "Generation of Launch Vehicle Abort Trajectories Using a Hybrid Optimization Method," *Journal of Guidance, Control, and Dynamics*, Vol. 27, No. 6, 2004, pp. 929–929. https://doi.org/10.2514/1.7989.

[21] Lu, P., Sun, H., and Tsai, B., "Closed-Loop Endoatmospheric Ascent Guidance," *Journal of Guidance, Control, and Dynamics*, Vol. 26, No. 2, 2003, pp. 283–294. https://doi.org/10.2514/2.5045.

[22] Lu, P., and Pan, B., "Highly Constrained Optimal Launch Ascent Guidance," *Journal of Guidance, Control, and Dynamics*, Vol. 33, No. 2, 2010, pp. 404–414. https://doi.org/10.2514/1.45632.

[23] Kelley, H. J., "Gradient Theory of Optimal Flight Paths," *ARS Journal*, Vol. 30, No. 10, 1960, pp. 947–954. https://doi.org/10.2514/8.5282.

[24] Brauer, G. L., Cornick, D. E., and Stevenson, R., "Capabilities and applications of the Program to Optimize Simulated Trajectories (POST). Program summary document," Tech. Rep. CR-2770, NASA, 1977.

[25] Vlases, W. G., Paris, S. W., Lajoie, R. M., Martens, M. J., and Hargraves, C. R., "Optimal Trajectories by Implicit Simulation," Tech. Rep. TR WRDC-TR-90-3056, Boeing Aerospace and Electronics, Wright-Patterson Air Force Base, Ohio, 1990.

[26] Hargraves, C., and Paris, S., "Direct Trajectory Optimization Using Nonlinear Programming and Collocation," *Journal of Guidance, Control, and Dynamics*, Vol. 10, No. 4, 1987, pp. 338–342. https://doi.org/10.2514/3.20223.

[27] Betts, J. T., and Huffman, W. P., "Sparse Optimal Control Software SOCS," Tech. Rep. MEA-LR-085, Boeing Information and Support Services, Seattle, Washington, 1997.

[28] Wiegand, A., "ASTOS User Manual," *Unterkirnach, Germany: Astos Solutions GmbH*, Vol. 17, 2010.

[29] Spangelo, I., and Well, K. H., "Rocket Ascent with Heat-Flux and Splash Down Constraints," *Automatic Control in Aerospace 1994*, IFAC Post-print Volume, Pergamon, Oxford, 1995, pp. 9–15. https://doi.org/10.1016/B978-0-08-042238-1.50005-7.

[30] Weigel, N., and Well, K. H., "Dual Payload Ascent Trajectory Optimization with a Splash-Down Constraint," *Journal of Guidance, Control, and Dynamics*, Vol. 23, No. 1, 2000, pp. 45–52. https://doi.org/10.2514/2.4485.

[31] Liu, X., Lu, P., and Pan, B., "Survey of Convex Optimization for Aerospace Applications," *Astrodynamics*, Vol. 1, No. 1, 2017, pp. 23–40. https://doi.org/10.1007/s42064-017-0003-8.

[32] Açıkmeşe, B., and Blackmore, L., "Lossless Convexification of a Class of Optimal Control Problems with Non-Convex Control Constraints," *Automatica*, Vol. 47, No. 2, 2011, pp. 341–347. https://doi.org/10.1016/j.automatica.2010.10.037.

[33] Acikmese, B., and Ploen, S. R., "Convex Programming Approach to Powered Descent Guidance for Mars Landing," *Journal of Guidance, Control, and Dynamics*, Vol. 30, No. 5, 2007, pp. 1353–1366. https://doi.org/10.2514/1.27553.

[34] Yang, R., and Liu, X., "Comparison of Convex Optimization-Based Approaches to Solve Nonconvex Optimal Control Problems," *AIAA Scitech 2019 Forum*, 2019. https://doi.org/10.2514/6.2019-1666.

[35] Liu, X., and Lu, P., "Solving Nonconvex Optimal Control Problems by Convex Optimization," *Journal of Guidance, Control, and Dynamics*, Vol. 37, No. 3, 2014, pp. 750–765. https://doi.org/10.2514/1.62110.

[36] Mao, Y., Szmuk, M., and Açıkmeşe, B., "Successive Convexification of Non-Convex Optimal Control Problems and Its Convergence Properties," *55th Conference on Decision and Control (CDC)*, IEEE, 2016, pp. 3636–3641. https://doi.org/10.1109/CDC.2016.7798816.

[37] Bonalli, R., Cauligi, A., Bylard, A., Lew, T., and Pavone, M., "Trajectory Optimization on Manifolds: A Theoretically-Guaranteed Embedded Sequential Convex Programming Approach," *Proceedings of Robotics: Science and Systems*, 2019. https://doi.org/10.15607/RSS.2019.XV.078.

[38] Benedikter, B., Zavoli, A., and Colasurdo, G., "A Convex Optimization Approach for Finite-Thrust Time-Constrained Cooperative Rendezvous," *Advances in the Astronautical Sciences*, Vol. 171, 2019, pp. 1483–1498.

[39] Lu, P., and Liu, X., “Autonomous Trajectory Planning for Rendezvous and Proximity Operations by Conic Optimization,” *Journal of Guidance, Control, and Dynamics*, Vol. 36, No. 2, 2013, pp. 375–389. https://doi.org/10.2514/1.58436.

[40] Morgan, D., Chung, S.-J., and Hadaegh, F. Y., “Model Predictive Control of Swarms of Spacecraft Using Sequential Convex Programming,” *Journal of Guidance, Control, and Dynamics*, Vol. 37, No. 6, 2014, pp. 1725–1740. https://doi.org/10.2514/1.G000218.

[41] Wang, Z., and Grant, M. J., “Optimization of Minimum-Time Low-Thrust Transfers Using Convex Programming,” *Journal of Spacecraft and Rockets*, Vol. 55, No. 3, 2018, pp. 586–598. https://doi.org/10.2514/1.A33995.

[42] Wang, Z., and Grant, M. J., “Minimum-Fuel Low-Thrust Transfers for Spacecraft: A Convex Approach,” *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 54, No. 5, 2018, pp. 2274–2290. https://doi.org/10.1109/TAES.2018.2812558.

[43] Wu, Y., Wang, Z., Benedikter, B., and Zavoli, A., “A Convex Approach to Multi-phase Trajectory Optimization of eVTOL Vehicles for Urban Air Mobility,” *AIAA SciTech Forum*, 2022. https://doi.org/10.2514/6.2022-2159.

[44] Foust, R., Chung, S.-J., and Hadaegh, F. Y., “Optimal Guidance and Control with Nonlinear Dynamics Using Sequential Convex Programming,” *Journal of Guidance, Control, and Dynamics*, Vol. 43, No. 4, 2020, pp. 633–644. https://doi.org/10.2514/1.G004590.

[45] Szmuk, M., Acikmese, B., and Berning, A. W., “Successive Convexification for Fuel-Optimal Powered Landing with Aerodynamic Drag and Non-Convex Constraints,” *AIAA Guidance, Navigation, and Control Conference*, 2016. https://doi.org/10.2514/6.2016-0378.

[46] Sagliano, M., “Pseudospectral Convex Optimization for Powered Descent and Landing,” *Journal of Guidance, Control, and Dynamics*, Vol. 41, No. 2, 2018, pp. 320–334. https://doi.org/10.2514/1.G002818.

[47] Sagliano, M., “Generalized hp Pseudospectral-Convex Programming for Powered Descent and Landing,” *Journal of Guidance, Control, and Dynamics*, Vol. 42, No. 7, 2019, pp. 1562–1570. https://doi.org/10.2514/1.G003731.

[48] Liu, X., “Fuel-Optimal Rocket Landing with Aerodynamic Controls,” *Journal of Guidance, Control, and Dynamics*, Vol. 42, No. 1, 2019, pp. 65–77. https://doi.org/10.2514/1.G003537.

[49] Wang, Z., and Grant, M. J., “Constrained Trajectory Optimization for Planetary Entry via Sequential Convex Programming,” *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 10, 2017, pp. 2603–2615. https://doi.org/10.2514/1.G002150.

[50] Wang, Z., and Grant, M. J., “Autonomous Entry Guidance for Hypersonic Vehicles by Convex Optimization,” *Journal of Spacecraft and Rockets*, Vol. 55, No. 4, 2018, pp. 993–1006. https://doi.org/10.2514/1.A34102.

[51] Sagliano, M., and Mooij, E., "Optimal Drag-Energy Entry Guidance via Pseudospectral Convex Optimization," *AIAA Guidance, Navigation, and Control Conference*, 2018. https://doi.org/10.2514/6.2018-1315.

[52] Calabuig, G. J. D., and Mooij, E., "Optimal On-board Abort Guidance based on Successive Convexification for Atmospheric Re-Entry," *AIAA Scitech Forum*, 2021. https://doi.org/10.2514/6.2021-0860.

[53] Pinson, R., and Lu, P., "Rapid Generation of Optimal Asteroid Powered Descent Trajectories via Convex Optimization," *AAS/AIAA Astrodynamics Specialist Conference*, 2015, pp. 2655–2672.

[54] Zhang, K., Yang, S., and Xiong, F., "Rapid Ascent Trajectory Optimization for Guided Rockets via Sequential Convex Programming," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, Vol. 233, No. 13, 2019, pp. 4800–4809. https://doi.org/10.1177/0954410019830268.

[55] Li, Y., Guan, Y., Wei, C., and Hu, R., "Optimal Control of Ascent Trajectory for Launch Vehicles: A Convex Approach," *IEEE Access*, Vol. 7, 2019, pp. 186491–186498. https://doi.org/10.1109/ACCESS.2019.2960864.

[56] Li, Y., Pang, B., Wei, C., Cui, N., and Liu, Y., "Online Trajectory Optimization for Power System Fault of Launch Vehicles via Convex Programming," *Aerospace Science and Technology*, Vol. 98, 2020, p. 105682. https://doi.org/10.1016/j.ast.2020.105682.

[57] Cheng, X., Li, H., and Zhang, R., "Efficient Ascent Trajectory Optimization Using Convex Models Based on the Newton-Kantorovich/Pseudospectral Approach," *Aerospace Science and Technology*, Vol. 66, 2017, pp. 140 – 151. https://doi.org/10.1016/j.ast.2017.02.023.

[58] *Vega User's Manual*, Arianespace, 2014.

[59] Kouvaritakis, B., and Cannon, M., *Model Predictive Control*, Advanced Textbooks in Control and Signal Processing, Springer International Publishing, 2016. https://doi.org/10.1007/978-3-319-24853-0.

[60] Cutler, C. R., and Ramaker, B. L., "Dynamic Matrix Control – A Computer Control Algorithm," *Joint Automatic Control Conference*, Vol. 17, 1980, p. 72.

[61] Prett, D. M., and Gillette, R. D., "Optimization and Constrained Multivariable Control of a Catalytic Cracking Unit," *Joint Automatic Control Conference*, Vol. 17, 1980, p. 73.

[62] Mayne, D., Rawlings, J., Rao, C., and Scokaert, P., "Constrained Model Predictive Control: Stability and Optimality," *Automatica*, Vol. 36, No. 6, 2000, pp. 789 – 814. https://doi.org/10.1016/S0005-1098(99)00214-9.

[63] Eaton, J. W., and Rawlings, J. B., "Model-Predictive Control of Chemical Processes," *Chemical Engineering Science*, Vol. 47, No. 4, 1992, pp. 705 – 720. https://doi.org/10.1016/0009-2509(92)80263-C.

[64] Matsko, T., "Internal Model Control for Chemical Recovery," *Chemical Engineering Progress*, Vol. 81, No. 12, 1985, pp. 46–51.

[65] Hrovat, D., Di Cairano, S., Tseng, H. E., and Kolmanovsky, I. V., "The Development of Model Predictive Control in Automotive Industry: A Survey," *2012 IEEE International Conference on Control Applications*, 2012, pp. 295–302. https://doi.org/10.1109/CCA.2012.6402735.

[66] Eren, U., Prach, A., Koçer, B. B., Raković, S. V., Kayacan, E., and Açıkmeşe, B., "Model Predictive Control in Aerospace Systems: Current State and Opportunities," *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 7, 2017, pp. 1541–1566. https://doi.org/10.2514/1.G002507.

[67] Pascucci, C. A., Bennani, S., and Bemporad, A., "Model Predictive Control for Powered Descent Guidance and Control," *European Control Conference (ECC)*, 2015, pp. 1388–1393. https://doi.org/10.1109/ECC.2015.7330732.

[68] Jung, Y., and Bang, H., "Mars Precision Landing Guidance Based on Model Predictive Control Approach," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, Vol. 230, No. 11, 2016, pp. 2048–2062. https://doi.org/10.1177/0954410015607893.

[69] Wang, J., Cui, N., and Wei, C., "Optimal Rocket Landing Guidance Using Convex Optimization and Model Predictive Control," *Journal of Guidance, Control, and Dynamics*, Vol. 42, No. 5, 2019, pp. 1078–1092. https://doi.org/10.2514/1.G003518.

[70] Carson, J., and Acikmese, A., "A Model-Predictive Control Technique with Guaranteed Resolvability and Required Thruster Silent Times for Small-Body Proximity Operations," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2006. https://doi.org/10.2514/6.2006-6780.

[71] Reynolds, T., and Mesbahi, M., "Small Body Precision Landing via Convex Model Predictive Control," *AIAA SPACE and Astronautics Forum and Exposition*, 2017. https://doi.org/10.2514/6.2017-5179.

[72] Hartley, E. N., "A Tutorial on Model Predictive Control for Spacecraft Rendezvous," *European Control Conference (ECC)*, 2015, pp. 1355–1361. https://doi.org/10.1109/ECC.2015.7330727.

[73] Weiss, A., Baldwin, M., Erwin, R. S., and Kolmanovsky, I., "Model Predictive Control for Spacecraft Rendezvous and Docking: Strategies for Handling Constraints and Case Studies," *IEEE Transactions on Control Systems Technology*, Vol. 23, No. 4, 2015, pp. 1638–1647. https://doi.org/10.1109/TCST.2014.2379639.

[74] Keviczky, T., Borrelli, F., Fregene, K., Godbole, D., and Balas, G. J., "Decentralized Receding Horizon Control and Coordination of Autonomous Vehicle Formations," *IEEE Transactions on Control Systems Technology*, Vol. 16, No. 1, 2008, pp. 19–33. https://doi.org/10.1109/TCST.2007.903066.

[75] Mayne, D., "Nonlinear Model Predictive Control: Challenges and Opportunities," *Nonlinear Model Predictive Control*, Springer, 2000, pp. 23–44. https://doi.org/10.1007/978-3-0348-8407-5_2.

[76] De Nicolao, G., Magni, L., and Scattolini, R., "Stability and Robustness of Nonlinear Receding Horizon Control," *Nonlinear Model Predictive Control*, Springer, 2000, pp. 3–22. https://doi.org/10.1007/978-3-0348-8407-5_1.

[77] Qin, S. J., and Badgwell, T. A., “An Overview of Nonlinear Model Predictive Control Applications,” *Nonlinear Model Predictive Control*, Springer, 2000, pp. 369–392. https://doi.org/10.1007/978-3-0348-8407-5_21.

[78] Boyd, S., Boyd, S. P., and Vandenberghe, L., *Convex Optimization*, Cambridge University Press, 2004. https://doi.org/10.1017/CBO9780511804441.

[79] Bemporad, A., and Morari, M., “Robust Model Predictive Control: A Survey,” *Robustness in identification and control*, 1999, pp. 207–226. https://doi.org/10.1007/BFb0109870.

[80] Chisci, L., Rossiter, J., and Zappa, G., “Systems with Persistent Disturbances: Predictive Control with Restricted Constraints,” *Automatica*, Vol. 37, No. 7, 2001, pp. 1019–1028. https://doi.org/10.1016/S0005-1098(01)00051-6.

[81] Langson, W., Chryssochoos, I., Raković, S., and Mayne, D., “Robust Model Predictive Control Using Tubes,” *Automatica*, Vol. 40, No. 1, 2004, pp. 125–133. https://doi.org/10.1016/j.automatica.2003.08.009.

[82] Mesbah, A., “Stochastic Model Predictive Control: An Overview and Perspectives for Future Research,” *IEEE Control Systems Magazine*, Vol. 36, No. 6, 2016, pp. 30–44. https://doi.org/10.1109/MCS.2016.2602087.

[83] Lew, T., Lyck, F., and Müller, G., “Chance-Constrained Optimal Altitude Control of a Rocket,” *European Conference for Aeronautics and Space Sciences (EUCASS)*, 2019. https://doi.org/10.13009/EUCASS2019-388.

[84] Oguri, K., Ono, M., and McMahon, J. W., “Convex Optimization over Sequential Linear Feedback Policies with Continuous-time Chance Constraints,” *2019 IEEE 58th Conference on Decision and Control (CDC)*, 2019, pp. 6325–6331. https://doi.org/10.1109/CDC40024.2019.9029604.

[85] Chen, Y., Georgiou, T. T., and Pavon, M., “Optimal Steering of a Linear Stochastic System to a Final Probability Distribution, Part I,” *IEEE Transactions on Automatic Control*, Vol. 61, No. 5, 2016, pp. 1158–1169. https://doi.org/10.1109/TAC.2015.2457784.

[86] Chen, Y., Georgiou, T. T., and Pavon, M., “Optimal Steering of a Linear Stochastic System to a Final Probability Distribution, Part II,” *IEEE Transactions on Automatic Control*, Vol. 61, No. 5, 2016, pp. 1170–1180. https://doi.org/10.1109/TAC.2015.2457791.

[87] Benedikter, B., Zavoli, A., Wang, Z., Pizzurro, S., and Cavallini, E., “Covariance Control for Stochastic Low-Thrust Trajectory Optimization,” *AIAA SciTech Forum*, 2022. https://doi.org/10.2514/6.2022-2474.

[88] Giannini, M., and Cruciani, I., “VEGA LV Qualification Process: GNC aspects on HWIL Testing and Analysis,” *5th European Conference for Aeronautics and Space Sciences*, 2013.

[89] Regan, F. J., and Anandakrishnan, S. M., *Dynamics of Atmospheric Re-Entry*, American Institute of Aeronautics and Astronautics, 1993, Chap. 6, pp. 175–177. https://doi.org/10.2514/4.861741.

[90] Martens, G., Vellutini, E., and Cruciani, I., "Innovative Strategy for Z9 Reentry," *6th International Conference on Astrodynamics Tools and Techniques (ICATT)*, 2016.

[91] Willems, J. C., "1696: The Birth of Optimal Control," *Decision and Control, 1996., Proceedings of the 35th IEEE Conference on*, Vol. 2, IEEE, 1996, pp. 1586–1587. https://doi.org/10.1109/CDC.1996.572753.

[92] Goldstine, H. H., *A History of the Calculus of Variations from the 17th through the 19th Century*, Vol. 5, Springer Science & Business Media, 2012.

[93] Euler, L., "Elementa Calculi Variationum," *Novi commentarii academiae scientiarum Petropolitanae*, 1766, pp. 51–93.

[94] Betts, J. T., *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*, Vol. 19, Siam, 2010. https://doi.org/10.1137/1.9780898718577.

[95] Kelly, M. P., "Transcription Methods for Trajectory Optimization," *Tutorial, Cornell University, Ithaca, New York*, 2015.

[96] Betts, J. T., and Huffman, W. P., "Mesh Refinement in Direct Transcription Methods for Optimal Control," *Optimal Control Applications and Methods*, Vol. 19, No. 1, 1998, pp. 1–21. https://doi.org/10.1002/(SICI)1099-1514(199801/02)19:1<1::AID-OCA616>3.0.CO;2-Q.

[97] Darby, C. L., Hager, W. W., and Rao, A. V., "An hp-Adaptive Pseudospectral Method for Solving Optimal Control Problems," *Optimal Control Applications and Methods*, Vol. 32, No. 4, 2011, pp. 476–502. https://doi.org/10.1002/oca.957.

[98] Garg, D., "Advances in Global Pseudospectral Methods for Optimal Control," Ph.D. thesis, University of Florida, Gainesville, FL, 2011.

[99] Elnagar, G., Kazemi, M. A., and Razzaghi, M., "The Pseudospectral Legendre Method for Discretizing Optimal Control Problems," *IEEE Transactions on Automatic Control*, Vol. 40, No. 10, 1995, pp. 1793–1796. https://doi.org/10.1109/9.467672.

[100] Huntington, G. T., "Advancement and Analysis of a Gauss Pseudospectral Transcription for Optimal Control Problems," Ph.D. thesis, Massachusetts Institute of Technology, 2007.

[101] Garg, D., Patterson, M., Hager, W. W., Rao, A. V., Benson, D. A., and Huntington, G. T., "A Unified Framework for the Numerical Solution of Optimal Control Problems Using Pseudospectral Methods," *Automatica*, Vol. 46, No. 11, 2010, pp. 1843 – 1851. https://doi.org/10.1016/j.automatica.2010.06.048.

[102] Patterson, M. A., and Rao, A. V., "GPOPS-II: A MATLAB Software for Solving Multiple-Phase Optimal Control Problems Using hp-Adaptive Gaussian Quadrature Collocation Methods and Sparse Nonlinear Programming," *ACM Trans. Math. Softw.*, Vol. 41, No. 1, 2014. https://doi.org/10.1145/2558904.

[103] Ross, I. M., and Fahroo, F., "Pseudospectral Knotting Methods for Solving Nonsmooth Optimal Control Problems," *Journal of Guidance, Control, and Dynamics*, Vol. 27, No. 3, 2004, pp. 397–405. https://doi.org/10.2514/1.3426.

[104] Sagliano, M., Theil, S., D'Onofrio, V., and Bergsma, M., "SPARTAN: A Novel Pseudospectral Algorithm for Entry, Descent, and Landing Analysis," *Advances in Aerospace Guidance, Navigation and Control*, Springer International Publishing, Cham, 2018, pp. 669–688. https://doi.org/10.1007/978-3-319-65283-2_36.

[105] Berrut, J.-P., and Trefethen, L. N., "Barycentric Lagrange Interpolation," *SIAM Review*, Vol. 46, No. 3, 2004, pp. 501–517. https://doi.org/10.1137/S0036144502417715.

[106] Izzo, D., Märtens, M., and Pan, B., "A Survey on Artificial Intelligence Trends in Spacecraft Guidance Dynamics and Control," *Astrodynamics*, Vol. 3, No. 4, 2019, pp. 287–299. https://doi.org/10.1007/s42064-018-0053-6.

[107] Zavoli, A., Federici, L., Benedikter, B., and Colasurdo, G., "Comparative Analysis of Genetic Crossover Operators for the Optimization of Impulsive Multi-Rendezvous Trajectories," *AIDAA 2019 - XXV International Congress*, Rome, Italy, 2019.

[108] Federici, L., Benedikter, B., and Zavoli, A., "EOS: a Parallel, Self-Adaptive, Multi-Population Evolutionary Algorithm for Constrained Global Optimization," *2020 IEEE Congress on Evolutionary Computation (CEC)*, 2020, pp. 1–10. https://doi.org/10.1109/CEC48606.2020.9185800.

[109] Pontani, M., and Conway, B. A., "Particle Swarm Optimization Applied to Space Trajectories," *Journal of Guidance, Control, and Dynamics*, Vol. 33, No. 5, 2010, pp. 1429–1441. https://doi.org/10.2514/1.48475.

[110] Radice, G., and Olmo, G., "Ant Colony Algorithms for Two Impluse Interplanetary Trajectory Optimization," *Journal of Guidance, Control, and Dynamics*, Vol. 29, No. 6, 2006, pp. 1440–1444. https://doi.org/10.2514/1.20828.

[111] Hokamoto, S., and Murakami, J., "Genetic-Algorithm-Based Rendezvous Trajectory Design for Multiple Active Debris Removal," *28th International Symposium on Space Technology and Science*, 2011.

[112] Zona, D., Federici, L., and Zavoli, A., "Preliminary Design of Multi-Chaser Active Debris Removal Missions with Evolutionary Algorithms," *AAS/AIAA Astrodynamics Specialist Conference*, Virtual, 2021.

[113] Federici, L., Zavoli, A., and Colasurdo, G., "Evolutionary Optimization of Multirendezvous Impulsive Trajectories," *International Journal of Aerospace Engineering*, Vol. 2021, 2021. https://doi.org/10.1155/2021/9921555.

[114] Federici, L., Zavoli, A., and Colasurdo, G., "Preliminary Capture Trajectory Design for Europa Tomography Probe," *International Journal of Aerospace Engineering*, Vol. 2018, 2018. https://doi.org/10.1155/2018/6890173.

[115] Federici, L., Zavoli, A., Colasurdo, G., Mancini, L., and Neri, A., "Integrated Optimization of First-Stage SRM and Ascent Trajectory of Multistage Launch Vehicles," *Journal of Spacecraft and Rockets*, Vol. 58, No. 3, 2021, pp. 786–797. https://doi.org/10.2514/1.A34930.

[116] Izzo, D., Öztürk, E., and Märtens, M., "Interplanetary Transfers via Deep Representations of the Optimal Policy and/or of the Value Function," *Proceedings*

of the Genetic and Evolutionary Computation Conference Companion, Association for Computing Machinery, New York, NY, USA, 2019, p. 1971–1979. https://doi.org/10.1145/3319619.3326834.

[117] Sánchez-Sánchez, C., and Izzo, D., "Real-Time Optimal Control via Deep Neural Networks: Study on Landing Problems," *Journal of Guidance, Control, and Dynamics*, Vol. 41, No. 5, 2018, pp. 1122–1135. https://doi.org/10.2514/1.G002357.

[118] Cheng, L., Wang, Z., Song, Y., and Jiang, F., "Real-Time Optimal Control for Irregular Asteroid Landings Using Deep Neural Networks," *Acta Astronautica*, Vol. 170, 2020, pp. 66–79. https://doi.org/10.1016/j.actaastro.2019.11.039.

[119] Shi, Y., and Wang, Z., "A Deep Learning-Based Approach to Real-Time Trajectory Optimization for Hypersonic Vehicles," *AIAA Scitech Forum*, 2020. https://doi.org/10.2514/6.2020-0023.

[120] Shi, Y., and Wang, Z., "Onboard Generation of Optimal Trajectories for Hypersonic Vehicles Using Deep Learning," *Journal of Spacecraft and Rockets*, Vol. 58, No. 2, 2021, pp. 400–414. https://doi.org/10.2514/1.A34670.

[121] Gaudet, B., and Furfaro, R., "Adaptive Pinpoint and Fuel Efficient Mars Landing Using Reinforcement Learning," *IEEE/CAA Journal of Automatica Sinica*, Vol. 1, No. 4, 2014, pp. 397–411. https://doi.org/10.1109/JAS.2014.7004667.

[122] Furfaro, R., Scorsoglio, A., Linares, R., and Massari, M., "Adaptive Generalized ZEM-ZEV Feedback Guidance for Planetary Landing via a Deep Reinforcement Learning Approach," *Acta Astronautica*, Vol. 171, 2020, pp. 156–171. https://doi.org/10.1016/j.actaastro.2020.02.051.

[123] Holt, H., Armellin, R., Scorsoglio, A., and Furfaro, R., "Low-Thrust Trajectory Design Using Closed-Loop Feedback-Driven Control Laws and State-Dependent Parameters," *AIAA Scitech Forum*, 2020. https://doi.org/10.2514/6.2020-1694.

[124] Miller, D., Englander, J. A., and Linares, R., "Interplanetary Low-Thrust Design Using Proximal Policy Optimization," *AAS/AIAA Astrodynamics Specialist Conference*, 2019.

[125] Federici, L., Scorsoglio, A., Zavoli, A., and Furfaro, R., "Autonomous Guidance for Cislunar Orbit Transfers via Reinforcement Learning," *2021 AAS/AIAA Astrodynamics Specialist Conference*, Big Sky, virtual, 2021.

[126] Broida, J., and Linares, R., "Spacecraft Rendezvous Guidance in Cluttered Environments via Reinforcement Learning," *29th AAS/AIAA Space Flight Mechanics Meeting*, American Astronautical Society Ka'anapali, Hawaii, 2019, pp. 1–15.

[127] Federici, L., Scorsoglio, A., Zavoli, A., and Furfaro, R., "Meta-Reinforcement Learning for Adaptive Spacecraft Guidance during Multi-Target Missions," *72nd International Astronautical Congress (IAC)*, Dubai (UAE), 2021.

[128] Federici, L., Benedikter, B., and Zavoli, A., "Deep Learning Techniques for Autonomous Spacecraft Guidance During Proximity Operations," *Journal of Spacecraft and Rockets*, 2021, pp. 1–12. https://doi.org/10.2514/1.A35076.

[129] Federici, L., Scorsoglio, A., Ghilardi, L., D'Ambrosio, A., Benedikter, B., Zavoli, A., and Furfaro, R., "Image-based Meta-Reinforcement Learning for Autonomous Terminal Guidance of an Impactor in a Binary Asteroid System," *AIAA Scitech 2022 Forum*, San Diego (CA), USA, 2022. https://doi.org/10.2514/6.2022-2270.

[130] Liu, X., Shen, Z., and Lu, P., "Entry Trajectory Optimization by Second-Order Cone Programming," *Journal of Guidance, Control, and Dynamics*, Vol. 39, No. 2, 2016, pp. 227–241. https://doi.org/10.2514/1.G001210.

[131] Mao, Y., Szmuk, M., and Acikmese, B., "Successive convexification: A superlinearly convergent algorithm for non-convex optimal control problems," *arXiv preprint*, 2018.

[132] Szmuk, M., Reynolds, T. P., and Açıkmeşe, B., "Successive Convexification for Real-Time Six-Degree-of-Freedom Powered Descent Guidance with State-Triggered Constraints," *Journal of Guidance, Control, and Dynamics*, Vol. 43, No. 8, 2020, pp. 1399–1413. https://doi.org/10.2514/1.G004549.

[133] Wang, Z., and McDonald, S. T., "Convex Relaxation for Optimal Rendezvous of Unmanned Aerial and Ground Vehicles," *Aerospace Science and Technology*, Vol. 99, 2020, p. 105756. https://doi.org/10.1016/j.ast.2020.105756.

[134] Benedikter, B., Zavoli, A., Colasurdo, G., Pizzurro, S., and Cavallini, E., "Convex Approach to Three-Dimensional Launch Vehicle Ascent Trajectory Optimization," *AIAA Journal of Guidance, Control, and Dynamics*, Vol. 44, No. 6, 2021, pp. 1116–1131. https://doi.org/10.2514/1.G005376.

[135] Sagliano, M., Heidecker, A., Hernández, J. M., Farì, S., Schlotterer, M., Woicke, S., Seelbinder, D., and Dumont, E., "Onboard Guidance for Reusable Rockets: Aerodynamic Descent and Powered Landing," *AIAA Scitech Forum*, 2021. https://doi.org/10.2514/6.2021-0862.

[136] Benedikter, B., Zavoli, A., Colasurdo, G., Pizzurro, S., and Cavallini, E., "Convex Optimization of Launch Vehicle Ascent Trajectory with Heat-Flux and Splash-Down Constraints," 2022. https://doi.org/10.2514/1.A35194.

[137] Benedikter, B., Zavoli, A., Colasurdo, G., Pizzurro, S., and Cavallini, E., "Autonomous Upper Stage Guidance with Robust Splash-Down Constraint," *72nd International Astronautical Congress (IAC)*, Dubai, UAE, 2021.

[138] Szmuk, M., Eren, U., and Acikmese, B., "Successive Convexification for Mars 6-DoF Powered Descent Landing Guidance," *AIAA Guidance, Navigation, and Control Conference*, 2017. https://doi.org/10.2514/6.2017-1500.

[139] Szmuk, M., and Acikmese, B., "Successive Convexification for 6-DoF Mars Rocket Powered Landing with Free-Final-Time," *AIAA Guidance, Navigation, and Control Conference*, 2018. https://doi.org/10.2514/6.2018-0617.

[140] Chen, Y., Cutler, M., and How, J. P., "Decoupled Multiagent Path Planning via Incremental Sequential Convex Programming," *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 5954–5961. https://doi.org/10.1109/ICRA.2015.7140034.

[141] Malyuta, D., Reynolds, T., Szmuk, M., Acikmese, B., and Mesbahi, M., "Fast Trajectory Optimization via Successive Convexification for Spacecraft Rendezvous with Integer Constraints," *AIAA Scitech Forum*, 2020. https://doi.org/10.2514/6.2020-0616.

[142] Malyuta, D., Reynolds, T., Szmuk, M., Mesbahi, M., Acikmese, B., and Carson, J. M., "Discretization Performance and Accuracy Analysis for the Rocket Powered Descent Guidance Problem," *AIAA Scitech Forum*, 2019. https://doi.org/10.2514/6.2019-0925.

[143] Manual, V. U., "Arianespace," , 2006.

[144] Benedikter, B., Zavoli, A., and Colasurdo, G., "A Convex Approach to Rocket Ascent Trajectory Optimization," *8th European Conference for Aeronautics and Space Sciences (EUCASS)*, 2019. https://doi.org/10.13009/EUCASS2019-430.

[145] NOAA, NASA, and USAF, *U.S. Standard Atmosphere*, U.S. Government Printing Office, 1976.

[146] Liu, X., Shen, Z., and Lu, P., "Exact Convex Relaxation for Optimal Flight of Aerodynamically Controlled Missiles," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 52, No. 4, 2016, pp. 1881–1892. https://doi.org/10.1109/TAES.2016.150741.

[147] Gurobi Optimization, LLC, *Gurobi Optimizer Reference Manual*, 2020.

[148] Wang, Z., and Lu, Y., "Improved Sequential Convex Programming Algorithms for Entry Trajectory Optimization," *Journal of Spacecraft and Rockets*, Vol. 57, No. 6, 2020, pp. 1373–1386. https://doi.org/10.2514/1.A34640.

[149] SpaceX, *Falcon 9 Launch Vehicle Payload User's Guide*, SpaceX, 2019.

[150] Benedikter, B., Zavoli, A., Colasurdo, G., Pizzurro, S., and Cavallini, E., "Autonomous Upper Stage Guidance Using Convex Optimization and Model Predictive Control," *AIAA ASCEND*, 2020. https://doi.org/10.2514/6.2020-4268.

[151] Hartl, R. F., Sethi, S. P., and Vickson, R. G., "A Survey of the Maximum Principles for Optimal Control Problems with State Constraints," *SIAM Review*, Vol. 37, No. 2, 1995, pp. 181–218. https://doi.org/10.1137/1037043.