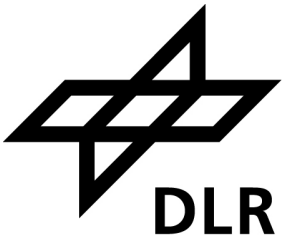
Technische Universität München



Deutsches Zentrum
für Luft- und Raumfahrt
German Aerospace Center

The present work was submitted to the Chair of Space Mobility and Propulsion

presented by

Giuseppe Soldati

Student ID No.: 03773828

Master Thesis

Design of Fault-Tolerant Landing Guidance for Multi-Engine Reusable Launchers

Munich, October 6, 2025

Supervising professor:	Prof. Dr.-Ing. Chiara Manfletti
Assistant supervisor:	Felix Ebert, M.Sc.
	Stefano Farì, M.Sc.
	Dr.-Ing. Marco Sagliano
1st examiner:	Prof. Dr.-Ing. Chiara Manfletti

Statutory Declaration

I hereby declare that the present work was written independently by me, and that no other aids than those specified were used. The sections of the work that have been taken from other works, either verbatim or in essence, are clearly marked as such in each individual case, with an indication of the source. This declaration also extends to graphics, drawings, map sketches, and pictorial representations contained in the work.

I also agree that my Master's, Bachelor's, or term paper may be made accessible by the department to professionally interested persons upon request, including through a library, and that the results contained therein, as well as the developments and programs created during this work, may be used without restriction by the Chair of Space Mobility and Propulsion. (Rights to any programs and inventions that may arise must be clarified in advance.)

Name: Giuseppe Soldati

Matr. Nr.: 03773828

Place, Date _____ Signature _____

Acknowledgements

This remarkable experience is coming to an end, and I want to take a moment to thank the people who made it possible.

My first thoughts go to Stefano and Marco. Thank you, Stefano, for always finding the time to answer my questions and for showing such dedication and hard work. You have been a true example to me. And thank you, Marco, for your guidance and experience, for always finding solutions to any problem, and for fueling my curiosity and drive to learn.

This journey would not have been possible without Felix, who gave me the opportunity to carry out this thesis and was always available when I needed support.

I'm very grateful to everyone in the Navigation and Control department at the Institute of Space Systems. And in particular, to all the "student room" friends, you made the office feel like home.

On a personal note, I wish to express my gratitude to my family for their unwavering support and to Giulia, thank you for your constant presence and for being with me every step of this journey.

Finally, I'd like to thank my lifelong friends from Rimini for all the laughs and fun we have shared, and my friends in Munich for making these master's years truly unforgettable.

Abstract

This work investigates whether, and in what manner, online guidance can provide fault tolerance in the presence of propulsion faults during the landing burn, and to determine the extent to which this contingency guidance remains effective throughout the descent. To this end, the effect of guidance reconfiguration is analyzed in the presence of engine loss, thrust inefficiencies, and thrust vectoring faults. A return-to-launch-site scenario is considered, restricted to the landing phase, where the mitigation of engine faults is paramount to recover the vehicle. In such cases, a new optimal trajectory must be recomputed, constraining the solution to the new degraded vehicle resources.

To address these challenges, a fault-tolerant landing guidance method for a multi-engine Reusable Launch Vehicle is developed, formulated through a sequential convex programming framework, where the optimization problem is updated according to the fault information.

Monte Carlo simulations demonstrate that the algorithm is highly effective in generating feasible recovery trajectories under a wide range of fault scenarios. The results show that the proposed method allows successful recovery where nominal guidance would otherwise fail.

Contents

List of Figures

List of Tables

List of Symbols

General Symbols

$\dot{m}$	[kg/s]	Mass flow rate
$\dot{T}$	[N/s]	Thrust rate
$\mathbf{r}$	[m]	Position vector
$\mathbf{u}$		Control vector
$\mathbf{v}$	[m/s]	Velocity vector
$\mathbf{x}$		State vector
τ		Normalized time
A	[m^2]	Area
g_0	[m/s^2]	Standard gravity
I_{sp}	[s]	Specific impulse
J		Objective / cost
m	[kg]	Mass
Ma	[-]	Mach number
s_η	[-]	Trust region slack variable
$s_\int$	[-]	Integral slack variable
s_κ	[-]	Virtual controls slack variable
s_{pos}	[-]	Final position slack variable
S_{ref}	[m^2]	Reference aerodynamic surface
T	[N]	Thrust
t	[s]	Time
V	[m^3]	Volume

Greek Symbols

α	[°]	Angle of attack
α_{eff}	[°]	Effective angle of attack
α_{tail}	[°]	Tail angle of attack
β	[°]	Sideslip angle
$\boldsymbol{\eta}$	[-]	Trust region vector
$\boldsymbol{\kappa}$	[-]	Virtual controls slack vector
$\boldsymbol{\lambda}$	[-]	Costate vector
$\boldsymbol{\nu}$	[-]	Virtual controls vector
$\Delta\tau$	[-]	Normalized step size
γ_v	[°]	Vertical flight-path angle
μ	[m^3/s^2]	Earth's gravitational parameter
ψ_v	[°]	Vertical azimuth angle
ρ	[kg/m^3]	Atmospheric density

Subscripts

$\square_B$	Body frame
$\square_{CoM}$	Center of mass
$\square_{CoP}$	Center of pressure
$\square_I$	Inertial frame
$\square_k$	Quantity at time step k
$\square_{NED}$	North–East–Down frame
$\square_{ref}$	Reference / nominal
$\square_{UEN}$	Up–East–North frame
$\square_W$	Wind frame

Superscripts

$\square^n$	Quantity at iteration n
$\square^T$	Transpose

Coefficients

C_A	[-]	Axial aerodynamic coefficient

C_N	[-]	Normal aerodynamic coefficient
C_x	[-]	Body-axis aerodynamic coefficient (x)
C_y	[-]	Body-axis aerodynamic coefficient (y)
C_z	[-]	Body-axis aerodynamic coefficient (z)

Special Symbols

$\|\cdot\|_2$	Euclidean norm

Abbrevations

CoM	Center of Mass
CoP	Center of Pressure
DCM	Direction Cosine Matrix
DRL	Downrange Landing
FDD	Fault Detection and Diagnosis
FTC	Fault-Tolerant Control
FTG	Fault-Tolerant Guidance
LQR	Linear Quadratic Regulator
LTI	Linear Time Invariant
NED	North-East-Down
NLP	Nonlinear Program
OCP	Optimal Control Problem
PDL	Powered Descend and Landing
RLV	Reusable Launch Vehicle
RTLS	Return To Launch Site
SCP	Sequential Convex Programming
SCvx	Successive Convexification
SOC	Second Order Cone
SOCP	Second Order Cone Programming
TVC	Thrust Vector Control
UEN	Up–East–North
VTVL	Vertical Takeoff Vertical Landing

1 Introduction

1.1 Reusable Launch Vehicles: Historical Context

Space transportation is experiencing a significant change, as substantial global efforts are directed toward the development of Vertical Takeoff Vertical Landing (VTVL) Reusable Launch Vehicles (RLVs). Reusable rockets are highly valuable assets for the organizations that have the expertise to operate them [9].

The idea of vertical landing dates back to the very beginning of spaceflight. The first spacecraft to achieve a survivable landing on a celestial body was *Luna 9*, which successfully touched down on the Moon on February 3, 1966 [79], after multiple previous Soviet attempts. This marked the first human-made object to achieve a soft landing on another celestial body. The Soviets later expanded this success with *Venera 7*, performing the first soft landing on Venus on December 15, 1970 [76].

On the U.S. side, NASA's *Surveyor I* mission achieved the first American soft landing on the Moon on June 2, 1966 [31], employing a three-legged landing structure and retro-propulsion, unlike the Soviet Luna landers. Later, *Viking I* became the first spacecraft to successfully soft-land on Mars on July 20, 1976 [54].

On Earth, interest in reusable rockets was already explored in the Apollo period and resurfaced with the Space Shuttle program [9]. Recovery concepts can be broadly grouped into two categories: those aiming for a vertical landing of the reusable stage, and those pursuing horizontal landing [26]. The latter include the U.S. Shuttle itself or Europe's *Hermes* spaceplane, although the project was canceled before it ever flew [30]. More recently, uncrewed winged demonstrators such as the Deutsche Zentrum für Luft- und Raumfahrt (DLR) Reusability Flight Experiment (*ReFEx*) [10] continue this line of development.

One of the first demonstrations of vertical landing on Earth was achieved in 1993 by the McDonnell Douglas Delta Clipper Experimental (*DC-X*) vehicle [34]. A key milestone followed with the *G-FOLD* (Guidance for Fuel-Optimal Large Diverts) algorithm [5], first demonstrated in 2013 on the Masten Space Systems *Xombie* VTVL suborbital rocket [74] (see Figure 1.1a), showing the feasibility of onboard real-time computation of large divert maneuvers. Shortly thereafter, private companies began demonstrating VTVL at scale. Blue Origin successfully recovered its *New Shepard* suborbital booster in November 2015 [17] (Figure 1.1b), later reusing it, while SpaceX recovered its first *Falcon 9* stage in December 2015 [91] and reflown it in 2017.

Both companies are now advancing towards heavy-lift reusable launch systems. Blue Origin is developing *New Glenn*, a medium-to-heavy lift rocket with a reusable first stage whose maiden flight occurred on January 16, 2025 [18], reaching orbit though the booster landing attempt failed. SpaceX is developing *Starship* and its *Super Heavy* booster, both fully reusable and capable of autonomous vertical landing.

On October 13, 2024, the Super Heavy booster demonstrated the first successful “catch” landing using mechanical arms [84] (Figure 1.1c).

Figure 1.1: Examples of VTVL vehicles. From the left: (a) Xombie demonstrates the G-FOLD algorithm (2013). (b) Blue Origin’s New Shepard first vertical landing (2015). (c) SpaceX’s Super Heavy booster performs the first successful catch landing with the Mechazilla arms (2024).

Globally, multiple space agencies are pursuing similar technologies. China is developing a reusable variant of its *Long March 8* [42], while India is progressing with the *Reusable Launch Vehicle–Technology Demonstrator (RLV-TD)* [2], aimed at hypersonic flight and autonomous landing validation. In Europe, the European Space Agency (ESA) and ArianeGroup are developing *Themis*, a methane-powered VTVL demonstrator [1], while DLR, the Centre National d’Études Spatiales (CNES), and the Japan Aerospace Exploration Agency (JAXA) are collaborating on *CALLISTO (Cooperative Action Leading to Launcher Innovation in Stage Toss-back Operations)* [29], a small-scale VTVL platform that serves to demonstrate the manoeuvres and operations required for a reusable first stage, together with validating the potential economic benefits such a system could bring to the European space sector.

We can see that the renewed interest in rocket reusability in recent years is translating into numerous products and prototypes that are demonstrating reliable performance. Yet, these efforts may not be directed toward developing methods for fault compensation [32]. An industry-wide study of international launches examined 118 failures from 2000 to mid-2024 and analyzed their statistical distribution across different categories, finding that propulsion issues account for the largest share of mission losses (49.6%) [38]. Consistent with this trend, another analysis reports that propulsion failures represent 54% of cases over the last 15 years [33].
Examples of failed landing attempts due to propulsion system failures include: (i) the Falcon 9 landing attempt on June 15, 2016, where one of the three Merlin engines produced less thrust than expected during the terminal burn, resulting in a crash on the droneship with a consequent loss of the vehicle [22]; and (ii) the Falcon Heavy test flight on February 6, 2018, in which the center core aimed for a droneship return but failed to ignite two of its three engines for the landing burn [23].

1.1.1 Fault-Tolerant Guidance: Motivation and Background

A fault is defined as any unpermitted deviation of at least one characteristic property or parameter of a system from its standard condition [40]. Fault Detection and Isolation (FDI) consists of determining whether a fault is present and, if so, locating and characterizing it. It generally relies on redundancy, which can be provided either by hardware or by analytical methods. Analytical redundancy employs a mathematical model of the system together with estimation techniques, and can be classified into quantitative model-based approaches, which use explicit mathematical models to generate residuals, or qualitative model-based methods, which use artificial intelligence (AI)[39].
On the other hand, hardware redundancy achieves fault detection mainly through cross checks, consistency checks or voting mechanisms [93], for example by comparing measurements of the same signal coming from multiple sensors.
Building upon FDI, Fault Detection, Isolation, and Reconfiguration (FDIR) is a control strategy that maintains safe or acceptable system operation in the presence of faults [39]. Assuming a correct and timely FDI, fault-tolerant control (FTC) enables the system to maintain stability and a limited level of performance despite the reduced resources caused by the fault. This is achieved by using the remaining control authority, allowing recovery from adverse flight conditions caused by faults [80]. For instance, an adaptive controller which generates new control parameters online after a fault occurs is discussed in [56], while [59] presents several strategies for handling thrust and TVC failures depending on the severity of the fault. These include switching to a different control allocation scheme, selecting a precomputed control law designed to be robust against failures, and reconfiguring the TVC inner-loop controller.
When flight conditions are too degraded for FTC to be effective, fault-tolerant guidance (FTG) becomes necessary. FTG refers to a redefinition of the onboard mission objectives when failures render the original plan infeasible. For example, recovery strategies for reentry missions are examined in [52]. While [57] describes online reconfiguration in the presence of a mid-flight change in the performance of the propulsion system of a Vertical Take Off and Landing (VTOL) aircraft. The work in [82] motivates FTG as a recovery strategy for failures that conventional techniques cannot handle, this can be achieved by changing mission objectives and recomputing a feasible trajectory accounting for the reduced resources, reshaping the mission autonomously to avoid the total loss of a launcher during ascent. Along these lines, real-time trajectory replanning for multistage launch vehicles under faults such as thrust drop or mass flow anomalies is studied in [51, 83]. Here, online optimal control is applied to generate a feasible trajectory after a fault occurs. Trajectory regeneration is carried out online, with the strategy determined by the severity of the fault: for less severe cases, the mission goal is preserved but the trajectory is adapted to remain feasible given the degraded resources, while for more severe cases the mission target itself is modified, switching from the original target orbit to a safe rescue orbit. Finally, when FTC is insufficient to address severe faults, FTG methods such as those described in [59] allow the continuation of a mission by regenerating an updated reference trajectory that explicitly accounts for the fault, by computing a feasible and locally optimal solution around the original nominal trajectory but consistent with the new faulty dynamics.

1.2 Guidance Algorithms: State of the Art

Computing guidance means generating a dynamically feasible state and control trajectory that satisfies a set of constraints while optimizing a mission objective, and such a problem is naturally formulated as an optimal control problem (OCP). All terms introduced here are meant in a broad sense, and their precise meaning will be clarified in the remainder of this work.
Trajectory optimization problems can be distinguished between *offline* and *online*. Offline formulations typically rely on indirect optimal control approaches or on direct transcription methods that result in nonlinear programming (NLP) problems. These approaches are categorized as offline because general NLP problems provide no guarantees of finding a feasible solution, may converge only to local minima, and are often sensitive to initial guesses [55]. As a result, they cannot reliably ensure safe guidance in real time. Furthermore, unpredictable computational time and the absence of assured algorithm convergence exclude the use of such methods for real-time applications, which demand both reliability and fast onboard solutions [45]. In contrast, trajectory optimization methods are classified as *online-capable* if they can provide onboard guidance commands in real time that satisfy the problem constraints and terminal conditions, independent of precomputed reference trajectories [81]. These methods generally rely on direct formulations that exploit convexity to achieve predictable runtimes and guaranteed convergence properties.

1.2.1 Offline Optimization

A number of software packages have been developed to address offline optimal control problems (OCPs). *FALCON.m* [61] provides an object-oriented framework within MATLAB [89] for modeling and solving general OCPs. Another widely used tool is *GPOPS-II* [58], which implements variable-order Gaussian quadrature collocation methods to transcribe the continuous-time problem into a sparse nonlinear program. DLR has developed *SPARTAN* [71], a tool based on pseudospectral transcription capable of handling complex multi-phase problems. Other well-established packages include DIDO [62], which enabled the International Space Station (ISS) to execute an optimal maneuver that saved one million dollars and marked the first in-flight use of pseudospectral methods [12], and CASADI [7], a flexible framework for algorithmic differentiation and numerical optimization that has been increasingly adopted for optimal control applications.

1.2.2 Online Optimization

Unlike nonconvex approaches, convex problems can be solved to global optimality under mild assumptions, with guaranteed convergence in polynomial time [20, 41]. This property makes convex optimization especially suitable for real-time applications where reliability and predictability of solve times are critical. Unfortunately, many practical trajectory generation problems, such as powered descend and landing (PDL), are inherently nonconvex and thus not directly compatible with convex optimization methods. Since trajectory generation problems are almost always nonconvex, online guidance methods typically rely on some form of *convexification*[49].

State of the art algorithms address this challenge by reformulating the original nonconvex guidance problem into one that can be solved using a convex optimizer. Two main strategies exist:

1. **Lossless Convexification (LCvx).** LCvx reformulates certain classes of nonconvex (OCPs) into higher-dimensional convex problems. Crucially, it can be shown that the solution of the convex problem is also an optimal solution to the original nonconvex problem, without excluding any feasible solutions [4]. This "lossless" property ensures that if a feasible solution exists, the convexified problem will find it [3]. However, LCvx formulations for more realistic lander models, such as full six-degree-of-freedom (6-DoF) dynamics or cases including complex aerodynamic forces, remain underdeveloped, with only simplified examples studied to date.
2. **Sequential Convex Programming (SCP).** When LCvx cannot be applied, convex optimization can still be used through SCP methods. These approaches iteratively solve a sequence of convex subproblems obtained by linearizing the nonconvexities around the solution at the previous iteration. Well-known algorithms include *Successive Convexification (SCvx)* [50] and *GuSTO* [19], which represent two variants of this general methodology.

Further research has explored hybrid approaches combining pseudospectral optimal control methods with convex optimization techniques. For example, a framework integrating pseudospectral methods with convex formulations for powered descent guidance was introduced in [65].
Several of these algorithms have already been demonstrated on real missions. LCvx was deployed on the Masten Space Systems *Xombie* rocket via the *G-FOLD* algorithm [5], showing that convex optimization could solve precision landing problems in real time. NASA has also tested *SCvx* onboard Blue Origin's *New Shepard* as part of the SPLICE (Safe and Precise Landing Integrated Capabilities Evolution) program [53]. Lastly, the European *CALLISTO* demonstrator is set to carry convex optimization algorithms onboard for guidance and control [69].

1.3 Research Questions

The objective of this thesis is to propose and analyze the effect of guidance reconfiguration in the presence of engine loss, thrust inefficiencies, or thrust vectoring faults. The study focuses on a return to launch site scenario, limited to the landing phase, where the mitigation of engine faults is paramount to recover the vehicle. To this purpose, a new optimal trajectory must be recomputed while constraining the solution to the degraded vehicle resources, which makes the formulation of the underlying optimization problem critical. The thesis builds upon an RLV mission benchmark based on [77], modified to include a cluster of rocket engines rather than a single-engine configuration. This not only provides additional redundancy in thrust capability but also better aligns to the future needs of commercial vehicles. For these reasons, the first research question is:

> **Research Question 1:** How can an online guidance provide fault tolerance in presence of faults affecting the engines in reusable launch vehicles during the landing burn?

Once a functioning algorithm has been established, the next step is to examine both its robustness and its limits. This means subjecting it to a wide range of fault scenarios and operating conditions in order to identify the envelope within which a safe landing can still be guaranteed. To this end, extensive Verification and Validation is carried out, which motivates the second research question.

> **Research Question 2:** In which measure (namely, until when across the descent) is the contingency guidance effective?

1.4 Thesis Overview

Chapter 2 outlines the overall research methodology. Chapter 3 introduces the mathematical preliminaries, beginning with a general classification of optimization problems and concluding with an overview of methods for solving (OCPs). Chapter 4 presents the mission profile and the VTVL model, including the derivation of the equations of motion. Building on this foundation, Chapter 5 defines the reference trajectory, which serves as the basis for the rest of the thesis, and describes the controller used to track it.
The core contribution of the work begins in Chapter 6, where the convex formulation of the guidance algorithm is derived in detail. This formulation is subsequently extended to account for engine and thrust vectoring faults, directly addressing the first research question. Chapter 7 answers the second research question by testing the algorithm through extensive Monte Carlo simulations, leading to conclusions on the impact of faults on mission success and the validation of the algorithm's robustness.
Finally, Chapter 8 states the conclusions of the thesis and points out directions for future work.

2 Methodology

The objective of this chapter is to give an overview on the decisions that have led to the choices made throughout the thesis, how the models, algorithms, and simulation environment are developed in order to addresses the research questions posed in Chapter 1.
Multiple surveys have shown that propulsion-related anomalies represent one of the most frequent causes of launch vehicle failure. For this reason, this work focused on propulsion and TVC related failures.

2.1 Methodology for Research Question 1

> **Research Question 1 (RQ1):** How can an online guidance provide fault tolerance in presence of faults affecting the engines in reusable launch vehicles during the landing burn?

The formulation of RQ1, which explicitly requires an online solution, motivated the investigation of guidance algorithms suitable for real-time implementation, leading to the choice of a convex method. As with all OCPs, the process began with the definition of an appropriate flight dynamics model.
A 3-DoF translational model with a cluster of engines was selected. The multi-engine setup is essential to allow for fault-tolerance, as one engine would not be enough, while the 3-DoF formulation was preferred over a full 6-DoF model for two reasons: (i) real-time applicability requires reduced computational burden, and (ii) the simplified translational model coupled with the engine cluster had not yet been explored in the literature.
Once the nonlinear dynamics had been defined, a convexification strategy was required. Among the available methods, LCvx could not be applied, as existing formulations do not cover the selected flight dynamics model. Therefore, SCP was adopted, since it is a more general method able to tackle the nonconvexities of the problem while still offering opportunities for further development beyond the scope of this thesis.
The guidance problem was thus formulated through SCP, where nonlinearities are linearized at each iteration until convergence is achieved. Details of the algorithm are provided in Chapter 6. First, a standard SCP algorithm was developed, and its correctness was verified by comparing its solutions with those obtained from the well-known OCP solver *GPOPS-II* [58]. The convex optimization problems that are solved in the SCP framework in each iteration were written using *CVX*, a MATLAB library for specifying and solving convex programs [24, 35] and solved using the convex optimization solver *MOSEK* [8]. Once the baseline SCP algorithm was validated, it was then augmented to provide guidance reconfiguration in the presence of faults. This was achieved by appropriately updating the OCP components based on the fault acting on the system. The resulting algorithm, named Fault-Tolerant Sequential Convex Programming (FT-SCP), was then tested on a range of representative cases for all the considered faults, thus addressing RQ1.

2.2 Methodology for Research Question 2

> **Research Question 2 (RQ2):** To what extent, i.e., until when across the descent, is the contingency guidance effective?

Answering RQ2 served multiple purposes. First, it enabled a robustness assessment of the algorithm developed to address RQ1 with respect to dispersions in initial conditions. Second, it allowed an investigation of how the occurrence of different propulsion faults at different times during the descent affects the landing success. RQ2 was addressed in Chapter 7 through a series of Monte Carlo campaigns that combined the FT-SCP algorithm with a feedback controller designed to track the computed trajectories. For this purpose, a simple Linear Quadratic Regulator (LQR) was employed, since the focus of the thesis lies on the guidance aspect rather than on controller design.

In conclusion, this chapter outlined the motivation behind the choices made to address the research questions, which will be explained in greater detail in the following chapters.

3 Mathematical Preliminaries

This chapter introduces the mathematical background required throughout the thesis. It starts with basic notions from optimization theory and then moves to the formulation of OCPs and their numerical solution methods.

3.1 Parametric Optimization

3.1.1 Nonlinear Optimization

Nonlinear optimization, or nonlinear programming (NLP) problems take the general mathematical form [11]. Let $\mathbf{z} = (z_1, \ldots, z_n) \in \mathbb{R}^n$ denote the vector of decision variables. The problem can be written as

$$\begin{aligned} \text{minimize} \quad & f(\mathbf{z}) \quad \text{(objective function)} \\ \text{subject to} \quad & g_i(\mathbf{z}) \leq 0, \quad i = 1, \ldots, m \quad \text{(inequality constraints)} \\ & h_j(\mathbf{z}) = 0, \quad j = 1, \ldots, p \quad \text{(equality constraints)} \end{aligned} \tag{3.1}$$

where

$$\mathbf{g}(\mathbf{z}) = \begin{bmatrix} g_1(\mathbf{z}) \\ \vdots \\ g_m(\mathbf{z}) \end{bmatrix} \in \mathbb{R}^m, \quad \mathbf{h}(\mathbf{z}) = \begin{bmatrix} h_1(\mathbf{z}) \\ \vdots \\ h_p(\mathbf{z}) \end{bmatrix} \in \mathbb{R}^p. \tag{3.2}$$

and

$$f : \mathbb{R}^n \to \mathbb{R}, \quad \mathbf{g} : \mathbb{R}^n \to \mathbb{R}^m, \quad \mathbf{h} : \mathbb{R}^n \to \mathbb{R}^p. \tag{3.3}$$

The objective of problem (3.1) is to determine the variables $z_1, \ldots, z_n$ that satisfy the equality and inequality constraints defined by $\mathbf{h}$ and $\mathbf{g}$, while minimizing the objective function f.

3.1.2 Convex Optimization

Before discussing what makes an optimization problem convex, it is useful to introduce the basic components of such problems. We begin with the definition of convex sets, followed by the notion of convex functions, and finally arrive at the structure of a convex optimization problem.

Let $\mathbf{z}_1, \mathbf{z}_2 \in \mathbb{R}^n$. Any combination of these points of the form

$$\mathbf{z} = \lambda_1 \mathbf{z}_1 + \lambda_2 \mathbf{z}_2 \tag{3.4}$$

is also an element of $\mathbb{R}^n$. This concept extends naturally to any number of vectors $i = 1, \ldots, K$.

Depending on what restrictions we place on the coefficients λ_i, we obtain different types of combinations [20]:

1. Linear combination: coefficients are arbitrary real numbers

$$\mathbf{z} = \lambda_1 \mathbf{z}_1 + \lambda_2 \mathbf{z}_2, \quad \lambda_i \in \mathbb{R} \tag{3.5}$$

2. Affine combination: coefficients sum to one

$$\mathbf{z} = \lambda_1 \mathbf{z}_1 + \lambda_2 \mathbf{z}_2, \quad \lambda_i \in \mathbb{R}, \quad \lambda_1 + \lambda_2 = 1 \tag{3.6}$$

3. Convex combination: coefficients are nonnegative and sum to one

$$\mathbf{z} = \lambda_1 \mathbf{z}_1 + \lambda_2 \mathbf{z}_2, \quad \lambda_1 + \lambda_2 = 1, \quad \lambda_i \geq 0 \tag{3.7}$$

4. Conic combination: coefficients are nonnegative

$$\mathbf{z} = \lambda_1 \mathbf{z}_1 + \lambda_2 \mathbf{z}_2, \quad \lambda_i \geq 0 \tag{3.8}$$

A set $C \subset \mathbb{R}^n$ is *affine* if, for any $\mathbf{z}_1, \mathbf{z}_2 \in C$ and $\lambda \in \mathbb{R}$,

$$\lambda \mathbf{z}_1 + (1 - \lambda)\mathbf{z}_2 \in C \tag{3.9}$$

That is, C contains all affine combinations of its points [20].

A set $C \subset \mathbb{R}^n$ is *convex* if, for any $\mathbf{z}_1, \mathbf{z}_2 \in C$ and $\lambda_1, \lambda_2 \geq 0$ with $\lambda_1 + \lambda_2 = 1$,

$$\lambda_1 \mathbf{z}_1 + \lambda_2 \mathbf{z}_2 \in C \tag{3.10}$$

Equivalently, C includes the line segment connecting any two of its points [20].

Convexity is preserved under several operations [37]. Let $C_1, C_2 \subset \mathbb{R}^n$ be convex sets, then:

- Intersection:

$$C_1 \cap C_2 \text{ is convex} \tag{3.11}$$

- Algebraic sum:

$$C_1 + C_2 = \{\mathbf{z} \in \mathbb{R}^n : \mathbf{z} = \mathbf{z}_1 + \mathbf{z}_2,\ \mathbf{z}_1 \in C_1,\ \mathbf{z}_2 \in C_2\} \quad \text{is convex} \tag{3.12}$$

- Affine transformation: for $\mathbf{f}(\mathbf{z}) = \mathbf{A}\mathbf{z} + \mathbf{b}$, with $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{b} \in \mathbb{R}^m$,

$$\mathbf{f}(C_1) = \{\mathbf{y} \in \mathbb{R}^m : \mathbf{y} = \mathbf{A}\mathbf{z}_1 + \mathbf{b},\ \mathbf{z}_1 \in C_1\} \quad \text{is convex} \tag{3.13}$$

Having discussed convex sets and operations that preserve convexity, it is natural to extend the concept

to functions defined on these sets. A function $f : C \to \mathbb{R}$, defined on a convex set $C \subset \mathbb{R}^n$, is said to be *convex* [20] if, for all $\mathbf{z}_1, \mathbf{z}_2 \in C$ and $\lambda \in [0, 1]$,

$$f(\lambda \mathbf{z}_1 + (1 - \lambda)\mathbf{z}_2) \leq \lambda f(\mathbf{z}_1) + (1 - \lambda) f(\mathbf{z}_2). \tag{3.14}$$

In the context of the general nonlinear programming problem introduced in (3.1), the optimization problem is classified as *convex* when its components satisfy the following conditions [20]:

- The objective function $f(\mathbf{z})$ is convex.
- Each inequality constraint function $g_i(\mathbf{z})$ is convex.
- Every equality constraint is affine

$$h_j(\mathbf{z}) = \mathbf{a}_j^T \mathbf{z} - b_j, \quad j = 1, \ldots, p, \tag{3.15}$$

  where $\mathbf{a}_j \in \mathbb{R}^n$ and $b_j \in \mathbb{R}$.

3.1.3 Second Order Cone Programming

Second-order cone programming (SOCP) is a subclass of convex optimization in which the objective function is linear, subject to a combination of linear equality constraints, linear inequality constraints and second-order cone (SOC) constraints. A generic SOCP can be written as [20]

$$\begin{aligned} \min_{\mathbf{z} \in \mathbb{R}^n} \quad & \mathbf{f}^\top \mathbf{z} \\ \text{s.t.} \quad & \|\mathbf{A}_i \mathbf{z} + \mathbf{b}_i\|_2 \leq \mathbf{c}_i^\top \mathbf{z} + d_i, \qquad i = 1, \ldots, m, \\ & \mathbf{F}\mathbf{z} = \mathbf{g} \\ & \mathbf{G}\mathbf{z} \leq \mathbf{h} \end{aligned} \tag{3.16}$$

$$\mathbf{A}_i \in \mathbb{R}^{\ell \times n}, \quad \mathbf{b}_i \in \mathbb{R}^\ell, \quad \mathbf{c}_i \in \mathbb{R}^n, \quad d_i \in \mathbb{R}, \quad \mathbf{F} \in \mathbb{R}^{p \times n}, \quad \mathbf{g} \in \mathbb{R}^p, \quad \mathbf{G} \in \mathbb{R}^{q \times n}, \quad \mathbf{h} \in \mathbb{R}^q. \tag{3.17}$$

In problem (3.16), the constraints consist of SOC constraints of the form $\|\mathbf{A}_i \mathbf{z} + \mathbf{b}_i\|_2 \leq \mathbf{c}_i^\top \mathbf{z} + d_i$, together with linear equality constraints $\mathbf{F}\mathbf{z} = \mathbf{g}$ and linear inequality constraints $\mathbf{G}\mathbf{z} \leq \mathbf{h}$. Each SOC constraint is of dimension $\ell + 1$, where the standard second-order cone is defined as

$$\mathcal{K}_{\ell+1} \; = \; \left\{ (\mathbf{u}, t) \in \mathbb{R}^\ell, t \in \mathbb{R} \;\middle|\; \|\mathbf{u}\|_2 \leq t \right\}. \tag{3.18}$$

Thus, each SOC constraint can equivalently be written as [47]

$$\begin{bmatrix} \mathbf{A}_i \\ \mathbf{c}_i^\top \end{bmatrix} \mathbf{z} + \begin{bmatrix} \mathbf{b}_i \\ d_i \end{bmatrix} \in \mathcal{K}_{\ell+1}, \qquad i = 1, \ldots, m. \tag{3.19}$$

This condition defines a second-order cone in $\mathbb{R}^{\ell+1}$ for the variables $(\mathbf{u}, t)$ obtained through the affine transformation

$$\mathbf{u} = \mathbf{A}_i \mathbf{z} + \mathbf{b}_i, \qquad t = \mathbf{c}_i^\top \mathbf{z} + d_i. \tag{3.20}$$

A graphical interpretation is given in Figure 3.1: in the case $\ell = 2$, the admissible pairs $(\mathbf{u}, t)$, with $\mathbf{u} \in \mathbb{R}^2$, occupy the region lying above the boundary of a cone in $\mathbb{R}^3$.

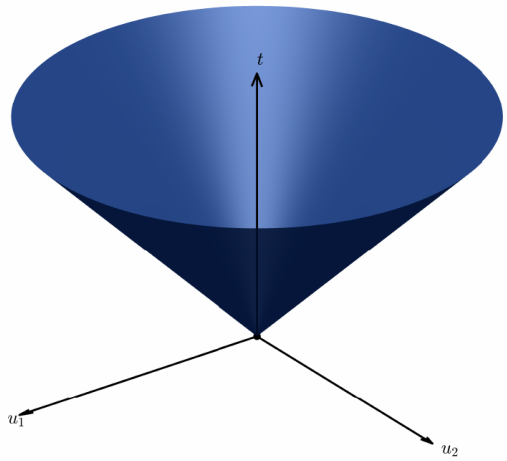


Figure 3.1: Second order cone constraint in $\mathbb{R}^3$: $\|\mathbf{u}\|_2 \leq t$.

3.2 Optimal Control

The trajectory design goal is to determine a sequence of control inputs such that a given performance index is minimized, while ensuring that the resulting state trajectory evolves according to the differential equation governing the system dynamics. Both the state and control trajectories must satisfy boundary conditions and path constraints.

The problem can be formulated as an OCP, whose fundamental building blocks are:

- The state vector $\mathbf{x}(t)$, which defines the state of the system at time t.
- The control vector $\mathbf{u}(t)$, which allows to influence the dynamic behavior of the system at time t.
- The system dynamics defined by the differential equation

$$\dot{\mathbf{x}}(t) = \mathbf{f}(t, \mathbf{x}(t), \mathbf{u}(t)),$$

  which governs the evolution of the state.
- The objective function to be minimized.
- State and control constraints that must be satisfied along the trajectory.
- The boundary conditions.

The optimal control problem is formally stated as [60]:

$$\begin{aligned} \text{Minimize} \quad & \phi\left(\mathbf{x}(t_0), \mathbf{x}(t_f), t_0, t_f\right) + \int_{t_0}^{t_f} f_0\left(t, \mathbf{x}(t), \mathbf{u}(t)\right)\, dt \\ \text{subject to} \quad & \dot{\mathbf{x}}(t) = \mathbf{f}\left(t, \mathbf{x}(t), \mathbf{u}(t)\right) \\ & \mathbf{c}\left(t, \mathbf{x}(t), \mathbf{u}(t)\right) \leq 0 \\ & \boldsymbol{\psi}\left(\mathbf{x}(t_0), \mathbf{x}(t_f), t_0, t_f\right) = 0 \end{aligned} \tag{3.21}$$

OCPs must be solved numerically, and the available numerical methods are commonly divided into two groups: indirect and direct methods, which are described in the following sections.

3.2.1 Indirect Methods

Indirect methods follow a "first optimize, then discretize" approach, they solve optimal control problems by first applying the necessary conditions for optimality from Pontryagin's Minimum Principle [16]. This leads to a boundary value problem (BVP) involving both the state and adjoint differential equations. The main idea is to construct the Hamiltonian, apply the necessary optimality conditions, and then solve the resulting BVP numerically, typically using single or multiple shooting methods [14].
While indirect methods can produce highly accurate solutions, they are often very sensitive to initial guesses and can struggle with path inequalities [15]. In practice, it can be useful to combine direct and indirect methods, for example, using a direct method first to generate a good initial guess for the indirect method.

Indirect methods are outside the scope of this thesis. This section serves only as a brief introduction, and they will not be discussed further.

3.2.2 Direct Methods

Direct methods approximate the state and/or controls of the OCP in a suitable way. A method is referred to as *control parametrization method* when only the control is approximated, while it is denoted as *state and control parametrization method* when both state and control are approximated. In both cases the OCP is discretized and transcribed into a finite-dimensional NLP. A second distinction arises from the manner in which the dynamics are enforced: *shooting methods* advance the dynamics by numerical simulation of the initial value problem (IVP) on some interval, while *collocation methods* impose the dynamics as constraints at a set of collocation points [15, 60].
To introduce these methods, we first recall some fundamental notions from the numerical solution of differential equations.

We consider IVPs of the form

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), t), \qquad \mathbf{x}(t_0) = \mathbf{x}_0. \tag{3.22}$$

and introduce a time grid

$$t_0 < t_1 < \cdots < t_N = t_f, \qquad t_i \in \{t_0, \ldots, t_N\}. \tag{3.23}$$

Numerical methods for solving such problems can be mainly divided into *one-step methods* (e.g., Runge–Kutta methods [63]) and *multi-step methods*. Their distinction lies in what the method exploits to calculate the value of the state at the next time step. For one-step, the value $\mathbf{x}_{i+1}$ depends only on the value $\mathbf{x}_i$ at the previous grid point. On the other hand, multi-step methods also use previous values $\mathbf{x}_{i-1}, \mathbf{x}_{i-2}, \dots$ [14].

A common family of one-step methods are the Runge–Kutta methods, which can be defined as [14, 63]:

$$\mathbf{x}_{i+1} = \mathbf{x}_i + h_i \sum_{j=1}^{K} \beta_j \mathbf{f}_{ij}, \tag{3.24}$$

where

$$\mathbf{f}_{ij} = \mathbf{f}\left(\mathbf{x}_i + h_i \sum_{\ell=1}^{K} \alpha_{j\ell} \mathbf{f}_{i\ell},\, t_i + h_i \rho_j \right). \tag{3.25}$$

The coefficients $\rho_j, \beta_j, \alpha_{j\ell}$ are known constants, typically organized in the Butcher diagram

$$\begin{array}{c|cccc} \rho_1 & \alpha_{11} & \alpha_{12} & \cdots & \alpha_{1k} \\ \rho_2 & \alpha_{21} & \alpha_{22} & \cdots & \alpha_{2k} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \rho_k & \alpha_{k1} & \alpha_{k2} & \cdots & \alpha_{kk} \\ \hline & \beta_1 & \beta_2 & \cdots & \beta_k \end{array}$$

For the trapezoidal method with $k = 2$, the Butcher tableau is

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & \frac{1}{2} & \frac{1}{2} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

which leads to

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \frac{h_i}{2} \left(\mathbf{f}(\mathbf{x}_i, t_i) + \mathbf{f}(\mathbf{x}_{i+1}, t_{i+1}) \right). \tag{3.26}$$

For simplicity, the following discussion will be restricted to one-step methods, although analogous considerations apply to multi-step methods as well.

The control $\mathbf{u}$ must also be discretized on the grid. Common parametrizations include piecewise-constant or piecewise-linear functions, as well as higher-order representations such as B-splines.

For example, a piecewise-constant parametrization results in

$$\mathbf{u}_i \approx \mathbf{u}(t_i), \qquad t_i \in \{t_0, \dots, t_N\} \quad \Rightarrow \quad \mathbf{u}_h = (\mathbf{u}_0, \mathbf{u}_1, \dots, \mathbf{u}_N)^\top \in \mathbb{R}^{(N+1)n_u}. \tag{3.27}$$

Given a choice of control parametrization, the state samples $\mathbf{x}_i \approx \mathbf{x}(t_i)$ are computed using the one-step method, which produces the state approximation

$$\mathbf{x}_h = (\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_N)^\top \in \mathbb{R}^{(N+1)n_x}. \tag{3.28}$$

Direct Collocation

Given the discretization of state and control, *direct collocation* treats both $\{\mathbf{x}_i\}_{i=0}^{N}$ and $\{\mathbf{u}_i\}_{i=0}^{N}$ as decision variables, and enforces the dynamics through a set of defect constraints $\boldsymbol{\xi}_i$ that are imposed on each discretization interval [15].

For example, under trapezoidal discretization, the defect at time step i is given by

$$\boldsymbol{\xi}_i = \mathbf{x}_{i+1} - \mathbf{x}_i - \frac{h}{2}\big(\mathbf{f}(t_i, \mathbf{x}_i, \mathbf{u}_i) + \mathbf{f}(t_{i+1}, \mathbf{x}_{i+1}, \mathbf{u}_{i+1})\big) \tag{3.29}$$

leading to the following NLP:

$$\begin{aligned} &\min_{\mathbf{z}} \; J(\mathbf{z}) \\ \text{s.t.} \quad &\mathbf{H}(\mathbf{z}) = \mathbf{0}, \\ &\mathbf{G}(\mathbf{z}) \leq \mathbf{0}. \end{aligned} \tag{3.30}$$

The equality constraints can be written as

$$H(z) := \begin{pmatrix} \boldsymbol{\xi}_0 = \mathbf{x}_1 - \mathbf{x}_0 - \frac{h}{2}\Big(\mathbf{f}(t_0, \mathbf{x}_0, \mathbf{u}_0) + \mathbf{f}(t_1, \mathbf{x}_1, \mathbf{u}_1)\Big) \\ \vdots \\ \boldsymbol{\xi}_{N-1} = \mathbf{x}_N - \mathbf{x}_{N-1} - \frac{h}{2}\Big(\mathbf{f}(t_{N-1}, \mathbf{x}_{N-1}, \mathbf{u}_{N-1}) + \mathbf{f}(t_N, \mathbf{x}_N, \mathbf{u}_N)\Big) \\ \mathbf{C}_{equality}(\mathbf{z}) \\ \boldsymbol{\Psi}(\mathbf{x}_0, \mathbf{x}_N) \end{pmatrix} = \mathbf{0}. \tag{3.31}$$

while the inequality constraints are

$$\mathbf{G}(\mathbf{z}) := \begin{pmatrix} \mathbf{c}(t_0, \mathbf{x}_0, \mathbf{u}_0) \\ \vdots \\ \mathbf{c}(t_N, \mathbf{x}_N, \mathbf{u}_N) \end{pmatrix} \leq \mathbf{0}. \tag{3.32}$$

The resulting NLP is large-scale but sparse, a property that many NLP solvers are able to exploit for efficiency [60].

Direct Single Shooting

The fully discretized OCP that results from direct collocation can be reduced in size by integrating the differential equations using any numerical method for IVPs. In fact, by solving the IVP, the state trajectory is fully characterized by the initial state $\mathbf{x}_0$ and the control parametrization, leading to the following vector of optimization variables:

$$\mathbf{z} \;=\; (\mathbf{x}_0, \mathbf{u}_0, \ldots, \mathbf{u}_N)^\top, \quad u_i \in \mathbb{R}^{n_u} \tag{3.33}$$

The direct single shooting method proceeds iteratively. Starting from an initial guess for the initial state and controls along the trajectory, the system dynamics are integrated from t_0 to t_f to obtain a candidate trajectory. The resulting state at the final time is then compared with the imposed terminal

state constraint, and the unknown initial state and controls are updated by driving the cost to a lower value. This procedure is repeated until convergence, that is, until the cost is minimized and all constraints are satisfied [60].

Direct Multiple Shooting

The direct single shooting method suffers from convergence problems due to the sensitivity of the states with respect to initial states and preceding control variables. In fact, any slight alteration to these can lead to large, non-linear variations in the subsequent states [14]. To tackle this, in *direct multiple shooting* the time interval is divided into $m+1$ subintervals and the previously mentioned direct single shooting method is used over each subinterval, with the values of the state at the beginning of each shooting segment and the coefficients in the control parametrization being optimization variables [60].

$$\mathbf{z} \;=\; (\mathbf{x}_0, \mathbf{x}_{MS,1}, ..., \mathbf{x}_{MS,m},\, \mathbf{u}_0, ..., \mathbf{u}_N)^\top \tag{3.34}$$

Furthermore, in order to enforce continuity, the shooting defects are imposed to be zero at each shooting node (see Figure 3.2), such that the terminal state of a segment coincides with the initial state at the segment that follows:

$$\mathbf{x}(t_j^-) = \mathbf{x}_{MS,j-1} + \int_{t_{MS,j-1}}^{t_{MS,j}} \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t))\, d\tau \tag{3.35}$$

$$\mathbf{x}(t_j^-) - \mathbf{x}_{MS,j} = \mathbf{0}, \qquad j = 1, \ldots, m \tag{3.36}$$

where $\mathbf{x}_{MS,j}$ denotes the optimization variable for the initial state at shooting node j, and $\mathbf{x}(t_j^-)$ is the solution of the IVP obtained by integrating the dynamics over the interval $[t_{MS,j-1}, t_{MS,j}]$ starting from $\mathbf{x}_{MS,j-1}$. The direct multiple shooting method increases the size of the optimization problem, as the state values at the shooting nodes are introduced as optimization variables. However, this larger problem is acceptable because integrating over shorter subintervals decreases sensitivity to errors in the unknown initial conditions. As a result, the states become sensitive only to the preceding shooting nodes and the control variables within each subinterval [60].

Pseudospectral Methods

Pseudospectral (PS) methods are a global form of orthogonal collocation. In this approach, the state is approximated by a global polynomial, and collocation conditions are enforced at specific points, chosen as the roots of orthogonal polynomials [60]. In PS methods, the number of mesh intervals is fixed, while the degree of the polynomial is varied. Increasing the number of collocation nodes p raises the degree of the polynomial approximation, and the solution accuracy improves accordingly. The main benefit of PS methods is that convergence is spectral (i.e., quasi exponential) with the number of nodes. An additional property of the Gauss pseudospectral method is that the Karush–Kuhn–Tucker (KKT) conditions of the NLP can be mapped to the costates of the continuous-time OCP [13], a property that will be exploited in Chapter 5.

hp pseudospectral methods extend this idea by dividing the time domain into multiple subdomains, over which the equations of motion are collocated. Two parameters are then defined: h, the number of segments, and p, the number of nodes per segment. These methods were first introduced in optimal

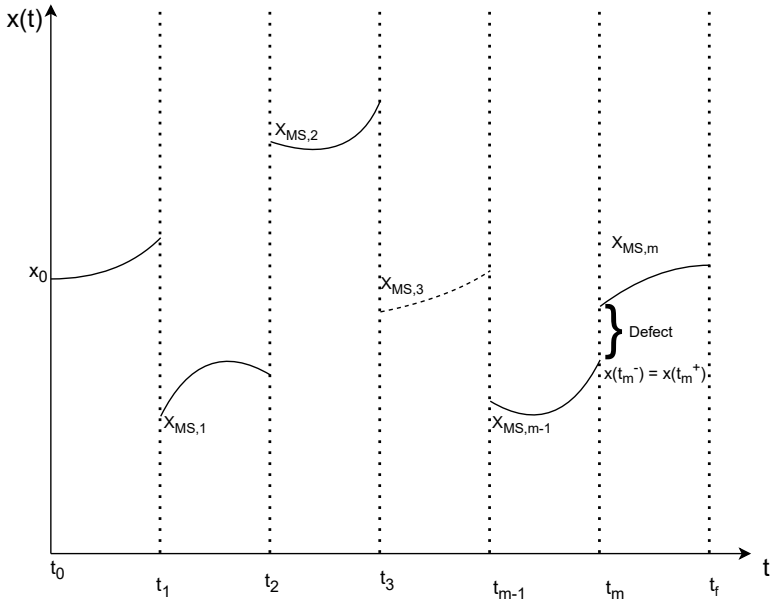


Figure 3.2: Multiple shooting defects.

control by [25], who proposed an adaptive version where the number of segments, segment widths, and polynomial degrees are varied. Such a scheme is implemented, for example, in the optimal control software *GPOPS-II* [58]. More recently, [64] proposed a hybrid framework combining *hp* pseudospectral methods with convex optimization, applied to the powered descent and landing (PDL) problem.

4 Modeling

4.1 Recovery Strategies for Reusable Launch Vehicles

Two primary recovery strategies are commonly considered for the first stage of reusable launch vehicles (RLVs): Return to Launch Site (RTLS) and Downrange Landing (DRL) [26] (Figure 4.1).
The DRL scenario is the simplest to execute and the most fuel-efficient, as the reusable stage follows its ballistic or initial trajectory and lands at a designated point, typically an offshore landing platform. The recovery sequence after Main Engine Cut-Off (MECO) can be summarized as follows:

- **Tilt-over maneuver**: The rocket is reoriented in the direction of flight using gas thrusters.
- **Ballistic flight**: The rocket follows its ballistic trajectory.
- **Entry burn**: The engine is reignited to adjust the ballistic trajectory such that the impact point aligns with the landing pad. The engine is then shut down again.
- **Aerodynamic phase**: The vehicle uses aerodynamic control surfaces to steer and reduce dispersions.
- **Powered descent and landing**: At a low altitude, the engines are reignited to perform a precise, soft landing.

The RTLS scenario involves a more complex sequence of maneuvers but offers the advantage of eliminating the need for an offshore recovery platform. The primary benefit is that the reusable stage can return and land near the launch site. The recovery sequence following MECO is as follows:

- **Tilt-over maneuver**: The rocket is reoriented in the direction of flight using gas thrusters.
- **Boostback burn**: The engine is reignited to reverse the horizontal velocity, redirecting the rocket back toward the launch/landing site.

This is followed by a coast phase, re-entry, and a powered descent and landing phase, which are similar to those in the DRL profile.

4.2 Vehicle Configuration & Mission Profile

The focus of this thesis is on the PDL phase, which begins after the final engine ignition. As such, earlier mission stages such as the boostback burn or aerodynamic coasting are not considered in the analysis. Therefore, the results presented here are not limited to a specific recovery strategy, since the PDL from the guidance and control perspective formulation is identical for both RTLS and DRL. The RLV under study is based on the benchmark proposed in [77]. Within the scope of the mission, only the first stage

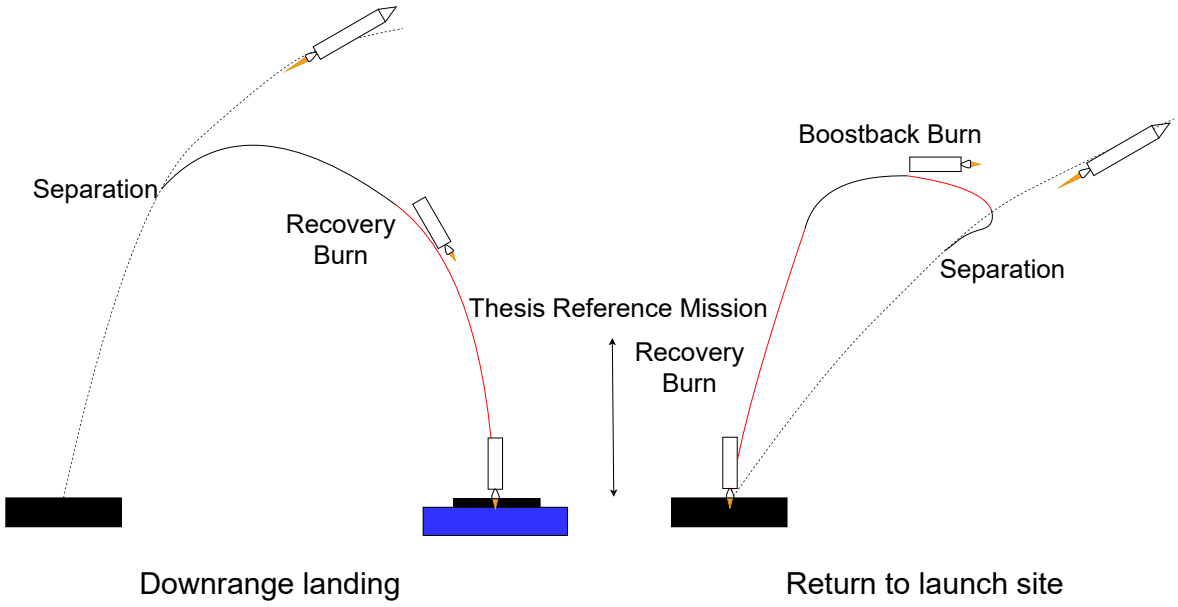


Figure 4.1: Recovery strategies for RLVs.

is considered, with the single-engine configuration replaced by a cluster of five engines. The first stage is shown in Figure 4.2. In addition, the rocket is equipped with two pairs of steerable fins mounted near the top of the vehicle.

The analysis is restricted to the terminal portion of the propulsive landing maneuver, starting at an altitude of 5 km, with the engines assumed to be already ignited. The maximum and minimum total thrust values are evenly distributed among the five engines, obtained by dividing the total values by five. A summary of the main vehicle parameters used in the modeling is provided in Table 4.1. The center of mass (CoM) position is measured from the gimbal point of the central engine. The reference surface area corresponds to the booster cross-section, derived from its diameter. The maximum thrust corresponds to 13.5 times the initial mass, while the minimum thrust is limited to 30% of this value. The thrust rate is obtained from the difference between these bounds, divided by 1.5.

Table 4.1: Rocket parameters

Parameter	Symbol	Value
Dry mass	m_{dry}	2750 kg
Initial mass	m_0	4300 kg
Specific impulse	I_{sp}	282 s
Booster diameter	d	3.0 m
Reference surface area	S_{ref}	7.07 m^2
Booster length	L	11.7 m
CoM position	x_{CoM}	4.6 m
Maximum total thrust	$T_{\max}$	58 kN
Minimum total thrust	$T_{\min}$	17.4 kN
Thrust rate limit	$\dot{T}_{\max}$	27.1 kN/s
Maximum TVC angle	$\beta_{y,i}^{\max}$, $\beta_{z,i}^{\max}$	8°
Maximum TVC rate	$\dot{\beta}_{y,i}^{\max}$, $\dot{\beta}_{z,i}^{\max}$	4°/s

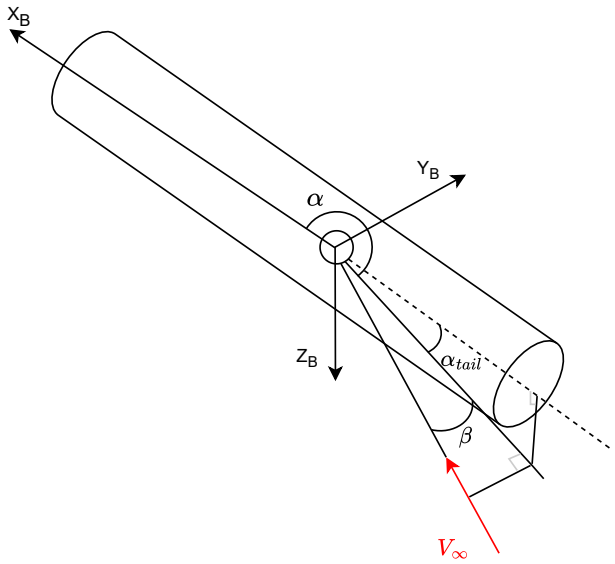


Figure 4.2: Vehicle body axes and aerodynamic angles.

4.3 Equations of Motion

The equations of motion are formulated in an Up-East-North (*UEN*) frame considered inertial and denoted as I, centered at the landing pad (see Figure 4.3). The validity of treating this frame as inertial will be discussed later. External forces are defined in the body–fixed frame B, whose origin is placed at the CoM, which is assumed constant during PDL, since fuel consumption in this phase is small compared to other flight phases, such as ascent. The x_B axis coincides with the rocket longitudinal axis pointing toward the tip, while y_B and z_B complete a right–handed triad. The adopted convention is illustrated in Figure 4.3. Detailed expressions for aerodynamic and propulsive forces are given in the following sections, for now, they are represented in compact form as a resultant force vector.

The rocket is modeled as a point mass with variable mass, moving in three–dimensional space with respect to the inertial reference frame I fixed at the landing pad. The system state is composed of the position, velocity, and mass:

$$\mathbf{r} = \begin{bmatrix} x & y & z \end{bmatrix}^\top \in \mathbb{R}^3 \tag{4.1}$$

$$\mathbf{v} = \begin{bmatrix} v_x & v_y & v_z \end{bmatrix}^\top \in \mathbb{R}^3 \tag{4.2}$$

$$m \in \mathbb{R}_{>0} \tag{4.3}$$

Here, $\mathbf{r}$ and $\mathbf{v}$ denote the rocket position and velocity expressed in the *UEN* frame I, whose axes are aligned with the local up, east, and north directions.

The translational dynamics are expressed as

$$\dot{\mathbf{r}} = \mathbf{v} \tag{4.4}$$

$$\dot{\mathbf{v}} = \mathbf{a}_{\text{grav}} + \frac{1}{m}\mathbf{F}_I \tag{4.5}$$

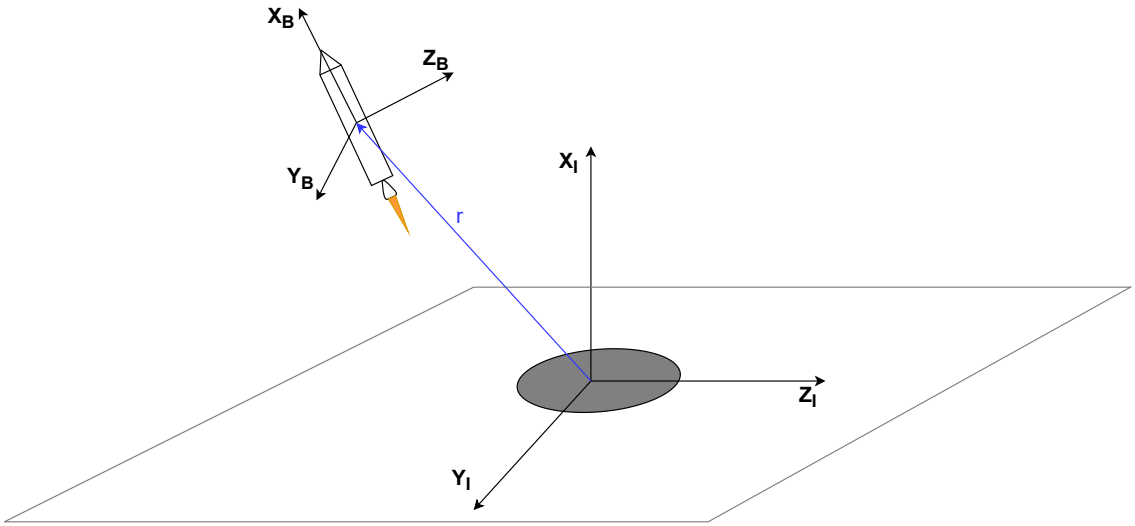


Figure 4.3: Reference frames: inertial *UEN* frame I centered at the landing pad and body–fixed frame B.

where $\mathbf{a}_{\text{grav}}$ is the gravitational acceleration and $\mathbf{F}_I$ is the sum of aerodynamic and propulsive forces expressed in the inertial frame.
The application of thrust leads to propellant consumption, and under the assumption of negligible back-pressure losses in the engines, the corresponding mass variation is described by the classical rocket equation [86]:

$$\dot{m} = -\frac{T}{g_0 I_{sp}} \tag{4.6}$$

where T denotes the total thrust magnitude.
The gravitational acceleration is modeled using the central-body assumption:

$$\mathbf{a}_{\text{grav}} = -\mu \frac{\mathbf{r} + \mathbf{r}_E}{\|\mathbf{r} + \mathbf{r}_E\|_2^3} \tag{4.7}$$

where $\mu = 3.986 \times 10^{14}\,\text{m}^3/\text{s}^2$ is Earth's gravitational parameter, and $\mathbf{r}_E = [R_E,\, 0,\, 0]^T$ is the position vector of the Earth's center, with $R_E = 6371\,\text{km}$ the mean Earth radius.

The orientation of the body frame with respect to the inertial frame can, under the assumption of zero bank angle, be fully described by the angle of attack and sideslip [70], illustrated in Figure 4.2. Based on this, aerodynamic and propulsive forces are expressed in the inertial frame through a Direction Cosine Matrix (DCM), constructed as two consecutive rotations:

$$\mathbf{M}_{IB} = \mathbf{M}_{IW}\mathbf{M}_{WB} \tag{4.8}$$

where $\mathbf{M}_{IW}$ and $\mathbf{M}_{WB}$ represent the DCM from wind axes to inertial reference frame and from body to wind axes, respectively. The transformation $\mathbf{M}_{WB}$ can be written in terms of the aerodynamic angles of

attack α and sideslip β (Figure 4.2) [85]:

$$\mathbf{M}_{WB} = \begin{bmatrix} \cos\alpha\cos\beta & \sin\beta & \sin\alpha\cos\beta \\ -\cos\alpha\sin\beta & \cos\beta & -\sin\alpha\sin\beta \\ -\sin\alpha & 0 & \cos\alpha \end{bmatrix} \tag{4.9}$$

Once the relationship between the body axes and the wind frame is established, the second rotation from wind to inertial must be defined. As anticipated earlier, under the flat–Earth assumption the local *UEN* frame can be treated as inertial, which justifies its use here. This allows the wind–to–inertial transformation to be parameterized by the flight–path and azimuth angles commonly used in aeronautics [70]:

$$\gamma = \tan^{-1}\left(-\frac{v_z}{\sqrt{v_x^2 + v_y^2}}\right), \qquad \psi = \tan^{-1}\left(\frac{v_y}{v_x}\right) \tag{4.10}$$

where v_x, v_y, v_z are the velocity components along the north-east-down *NED* axes. For a rocket landing problem, however, these definitions become unsuitable: during a nearly vertical descent the flight–path angle approaches $-\pi/2$, which makes the associated rotation matrix singular. To avoid this ambiguity, following [70], the angles are redefined as the *vertical flight–path angle* and the *vertical azimuth angle*, which can be used directly to construct the wind–to–*NED* transformation. In this formulation, the orientation of the wind axes is expressed without encountering the singularities of the classical definitions, and the vertical flight–path and vertical azimuth angles (Figure 4.4) are given by

$$\gamma_v = \tan^{-1}\left(\frac{v_z}{v_x}\right), \qquad \psi_v = \tan^{-1}\left(\frac{v_y}{\sqrt{v_x^2 + v_z^2}}\right) \tag{4.11}$$

Since the inertial frame I is taken to coincide with a local *UEN* frame, the conversion from the standard *NED* frame to *UEN* must also be specified. This transformation is represented by the constant matrix

$$\mathbf{M}_{\text{UEN,NED}} = \begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \tag{4.12}$$

With these definitions, the resulting DCM from wind to inertial reference frame is

$$\mathbf{M}_{IW} = \begin{bmatrix} \cos\gamma_v\cos\psi_v & -\sin\psi_v\cos\gamma_v & -\sin\gamma_v \\ \sin\psi_v & \cos\psi_v & 0 \\ \sin\gamma_v\cos\psi_v & -\sin\gamma_v\sin\psi_v & \cos\gamma_v \end{bmatrix} \tag{4.13}$$

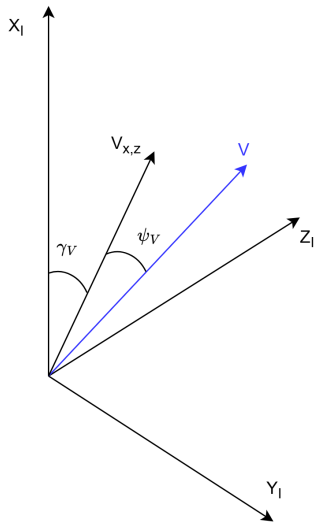


Figure 4.4: Vertical flight-path and azimuth angles.

Finally, by combining the gravitational model with the general representation of aerodynamic and propulsive forces, and applying the frame transformations introduced above, the equations of motion of the system can be written as

$$\begin{cases} \dot{\mathbf{r}} = \mathbf{v} \\ \dot{\mathbf{v}} = \mathbf{a}_{\text{grav}} + \mathbf{M}_{IB}(\mathbf{T}_B + \mathbf{F}_B)/m \\ \dot{m} = -\dfrac{T}{g_0 I_{sp}} \end{cases} \tag{4.14}$$

where $\mathbf{T}_B$ and $\mathbf{F}_B$ denote the propulsive and aerodynamic force vectors expressed in the body frame, respectively.

4.3.1 Aerodynamic Characteristics

The rocket model makes use of lookup tables for aerodynamic parameters, and the corresponding mapping to body forces follows the approach in [72]. The lookup tables define the axial and normal aerodynamic coefficients, as well as the x-component of the center of pressure (CoP) relative to the thrust application point of the vehicle. The aerodynamic data is organized as a two-dimensional set, and due to the axial symmetry of the booster, the coefficients extend to a three-dimensional representation.

The axial coefficient is evaluated using the Mach number and the effective angle of attack, defined as:

$$\alpha_{\text{eff}} = \sqrt{\alpha^2 + \beta^2} \tag{4.15}$$

The axial aerodynamic coefficient is obtained from the lookup table:

$$C_{A,B} = C_A^{\text{LU}}(\alpha_{\text{eff}}, M) \tag{4.16}$$

Since drag acts along the velocity vector, the body-axis x-component of the aerodynamic coefficient is:

$$C_{x,B} = -C_{A,B} \tag{4.17}$$

The y- and z-components of the aerodynamic coefficients are computed using the normal coefficient lookup table and the angles of attack and sideslip:

$$C_{y,B} = -\text{sign}(\beta) \cdot C_N^{\text{LU}}(|\beta|, M) \tag{4.18}$$

$$C_{z,B} = -\text{sign}(\alpha) \cdot C_N^{\text{LU}}(|\alpha|, M) \tag{4.19}$$

The aerodynamic force vector in the body frame is then given by:

$$\mathbf{F}_B = \frac{1}{2}\rho v^2 S_{\text{ref}} \begin{bmatrix} C_{x,B} & C_{y,B} & C_{z,B} \end{bmatrix}^T \tag{4.20}$$

Using the CoP lookup table, two CoP values associated with α and β are computed:

$$X_{\text{CoP}}^{\alpha} = X_{\text{CoP}}^{\text{LU}}(\alpha, M) \tag{4.21}$$

$$X_{\text{CoP}}^{\beta} = X_{\text{CoP}}^{\text{LU}}(\beta, M) \tag{4.22}$$

The corresponding lever arms from the CoM to each CoP are:

$$\mathbf{r}_{\text{CoM}\to\text{CoP}}^{\alpha} = \begin{bmatrix} X_{\text{CoP}}^{\alpha} & 0 & 0 \end{bmatrix}^{\top} - \mathbf{r}_{\text{CoM}} \tag{4.23}$$

$$\mathbf{r}_{\text{CoM}\to\text{CoP}}^{\beta} = \begin{bmatrix} X_{\text{CoP}}^{\beta} & 0 & 0 \end{bmatrix}^{\top} - \mathbf{r}_{\text{CoM}} \tag{4.24}$$

with

$$\mathbf{r}_{\text{CoM}} = \begin{bmatrix} x_{\text{CoM}} & 0 & 0 \end{bmatrix}^{\top} \tag{4.25}$$

The torque contributions with respect to the CoM are then:

$$\boldsymbol{\tau}_B^{\alpha} = \mathbf{r}_{\text{CoM}\to\text{CoP}}^{\alpha} \times \mathbf{F}_B^{\alpha} \tag{4.26}$$

$$\boldsymbol{\tau}_B^{\beta} = \mathbf{r}_{\text{CoM}\to\text{CoP}}^{\beta} \times \mathbf{F}_B^{\beta} \tag{4.27}$$

with the force components defined as:

$$\mathbf{F}_B^{\alpha} = \begin{bmatrix} F_{B,x} & 0 & F_{B,z} \end{bmatrix}^T \tag{4.28}$$

$$\mathbf{F}_B^{\beta} = \begin{bmatrix} F_{B,x} & F_{B,y} & 0 \end{bmatrix}^T \tag{4.29}$$

Finally, the total torque with respect to the CoM is:

$$\boldsymbol{\tau}_{\text{B}}^{\text{aero}} = \boldsymbol{\tau}_B^{\alpha} + \boldsymbol{\tau}_B^{\beta} \tag{4.30}$$

The atmosphere model is the U.S. Standard Atmosphere 1976 [78], which provides atmospheric density, temperature and ambient pressure as a function of altitude.

4.3.2 Propulsive Characteristics

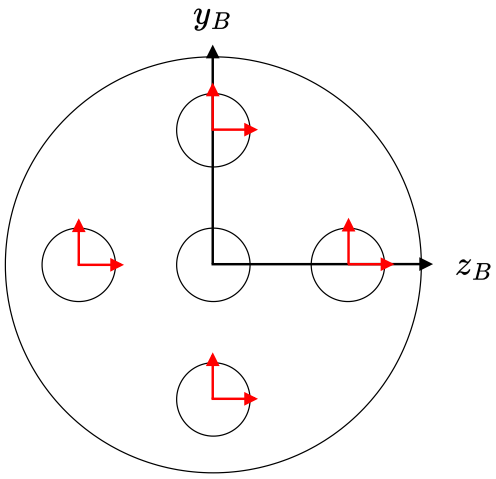


Figure 4.5: Cross-section of the engine cluster.

The vehicle is controlled by adjusting both the magnitude and direction of the thrust vector produced by each of its five rocket engines, which all share the same thrust characteristics with identical limits on both magnitude and rate. The four lateral engines are equipped with TVC, each with two actuators that deflect the nozzle of each engine along the body-frame y- and z-axes by the angles β_y and β_z, respectively.

The central engine's thrust is aligned with the body-frame x-axis, and thus does not contribute to the torque. Its force and torque expressions are given by:

$$\mathbf{T}_{B,1}(t) = T_1(t) \cdot \mathbf{i}_B \tag{4.31}$$

$$\boldsymbol{\tau}_{B,1}^{\text{prop}}(t) = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^\top \tag{4.32}$$

where $\mathbf{i}_B$ is the unit vector along the body-frame x-axis.

The thrust force vector of a lateral engine $i \in \{2, 3, 4, 5\}$, with nozzle deflections $\beta_{y,i}(t)$ and $\beta_{z,i}(t)$, is given by

$$\mathbf{T}_{B,i}(t) = T_i(t) \begin{bmatrix} \cos\left(\beta_{y,i}(t)\right) \cos\left(\beta_{z,i}(t)\right) \\ \cos\left(\beta_{y,i}(t)\right) \sin\left(\beta_{z,i}(t)\right) \\ -\sin\left(\beta_{y,i}(t)\right) \end{bmatrix} \tag{4.33}$$

and the associated torque relative to the CoM is

$$\boldsymbol{\tau}_{B,i}^{\text{prop}}(t) = \begin{bmatrix} \mathbf{r}_{\text{PVP},i} - \mathbf{r}_{\text{CoM}} \end{bmatrix}^\top \times \mathbf{T}_{B,i}(t) \tag{4.34}$$

where $\mathbf{r}_{\text{PVP},i}$ is the position vector of the pivot point of engine i. The total thrust and torque in the body frame are then obtained as the sum of the contributions from all engines:

$$\mathbf{T}_B(t) = \sum_{i=1}^{5} \mathbf{T}_{B,i}(t) \tag{4.35}$$

$$\boldsymbol{\tau}_B^{\text{prop}}(t) = \sum_{i=1}^{5} \boldsymbol{\tau}_{B,i}^{\text{prop}}(t) \tag{4.36}$$

4.4 Equations of Motion Summary and Assumptions

Collecting the gravitational, aerodynamic, and propulsive contributions, and applying the frame transformations introduced above, the complete equations of motion are

$$\begin{cases} \dot{\mathbf{r}} = \mathbf{v} \\ \dot{\mathbf{v}} = \mathbf{a}_{\text{grav}} + \mathbf{M}_{IB}(\mathbf{T}_B + \mathbf{F}_B)/m \\ \dot{m} = -\dfrac{\sum_{i=1}^{5} T_i}{g_0 I_{sp}} \end{cases} \tag{4.37}$$

The following assumptions are made in the modeling framework:

- Only 3-DoF translational dynamics are modeled, attitude dynamics are represented implicitly through aerodynamic angles, whose values are assumed to be instantly tracked by a lower level controller.
- The CoM position is considered static.
- Flexible body effects, propellant sloshing, actuator dynamics, and structural bending modes are neglected.
- The atmosphere is represented by the U.S. Standard Atmosphere 1976 [78].
- Gravity is modeled using the central-body assumption, valid for the short flight times considered.
- Earth is considered to be flat due to limited horizontal distance that the rocket has to cover.

5 Nominal Guidance & Control

5.1 Nominal Guidance

The nominal landing trajectory is obtained by solving a nonlinear OCP. The goal is to compute a feasible trajectory that drives the vehicle from the initial conditions specified in the mission profile (Section 4.2) to a soft landing at the designated pad, located at the origin of the inertial reference frame. The OCP is constructed using the vehicle dynamics defined in Chapter 4, together with path constraints and a suitable cost function. In the following, each of these elements is introduced, after which they are collected into the formal OCP definition. The problem is subsequently scaled and solved numerically to obtain the reference trajectory.

5.1.1 State and Control Augmentation

The equations of motion used for the guidance problem are those defined in (4.37). To enforce explicit bounds on the control rates and to obtain smooth control sequences, the controls are reformulated such that the original control variables become states, while their time derivatives serve as the new control inputs in the OCP.
This augmentation introduces 15 additional first-order differential equations:

$$\dot{T}_i = u_{T_i}, \quad i = 1, \dots, 5 \tag{5.1}$$

$$\dot{\beta}_{y,i} = u_{\beta_{y,i}}, \quad i = 1, \dots, 4 \tag{5.2}$$

$$\dot{\beta}_{z,i} = u_{\beta_{z,i}}, \quad i = 1, \dots, 4 \tag{5.3}$$

$$\dot{\alpha} = u_\alpha \tag{5.4}$$

$$\dot{\beta} = u_\beta \tag{5.5}$$

The angle of attack during the descent phase is defined around π. For the OCP formulation, it is instead expressed in its tail formulation (see Figure 4.2), defined as

$$\alpha_{\text{tail}} = \pi - \alpha \tag{5.6}$$

which shifts the variable close to zero, and the aerodynamic lookup tables are reparameterized accordingly. In the guidance dynamics, the transformation between the body and wind frames, given by (4.9), remains unchanged and continues to use the original angle of attack by remapping α_{tail} back to α. However, α_{tail} is used to access the reparameterized lookup tables. This approach simplifies the OCP implementation, since α can now be constrained to values around zero. In the remainder of this work, the symbol α is

used for simplicity to denote the tail formulation α_{tail}.

5.1.2 Path Constraints

A glideslope constraint is included to guide the rocket during descent by constraining the ratio of horizontal to vertical distance reduction during landing [67]:

$$\frac{x}{\sqrt{y^2+z^2}} \;\geq\; \tan\gamma_{gs}. \tag{5.7}$$

The glideslope angle is fixed at 45°, and the trajectory is constrained to lie within the corresponding cone.

5.1.3 Boundary Conditions

The initial state corresponds to the conditions at the start of the powered descent phase. In particular, the mission begins at an altitude of 5 km with lateral displacements of 200 m and 300 m in the y and z directions, respectively. The terminal state enforces a soft landing at the pad, with the vertical position constrained to 1 m above ground, zero lateral displacements, zero lateral velocities, and a vertical velocity of -1 m/s to avoid a singularity in the definition of the vertical flight path angle (4.11). At landing, the aerodynamic angles and TVC deflections are required to return to zero, while they are unconstrained at the beginning. Both the final mass and final time remain free. All boundary conditions and box constraints are summarized in Table 5.2.

5.1.4 Objective Function

The objective is to minimize the integral of a stage cost composed of several weighted terms:

$$\ell(\mathbf{x},\mathbf{u}) = w_T J_T + w_{\dot{T}} J_{\dot{T}} + w_{\dot{\beta}_y\dot{\beta}_z}\left(J_{\dot{\beta}_y} + J_{\dot{\beta}_z}\right) + w_{\dot{\alpha}\dot{\beta}}\left(J_{\dot{\alpha}} + J_{\dot{\beta}}\right) + w_\tau^{\text{prop}} J_{\tau^{prop}} + w_\tau^{\text{aero}} J_{\tau^{aero}}. \tag{5.8}$$

Each component in (5.8) serves a specific purpose in shaping the solution, and the corresponding weights are listed in Table 5.1.

Table 5.1: NLP integral weights

Weight	Value
w_T	1
$w_{\dot{T}}$	10^{-3}
$w_{\dot{\beta}_y\dot{\beta}_z}$	$5\cdot10^{-1}$
$w_{\dot{\alpha}\dot{\beta}}$	6
w_τ^{prop}	10^{-2}
w_τ^{aero}	10^{-4}

Thrust magnitude: A quadratic penalty on the thrust magnitude leads the optimizer to avoid unnecessarily high thrust levels. Since propellant consumption is directly related to thrust through the mass

depletion equation (4.6), this term effectively promotes fuel efficiency.

$$J_T = \sum_{i=1}^{5} T_i^2. \tag{5.9}$$

Control rates: The sum of squares of the control rates promotes smooth control trajectories:

$$J_{\dot{T}} = \sum_{i=1}^{5} \dot{T}_i^2 \tag{5.10}$$

$$J_{\dot{\beta}_y} = \sum_{i=1}^{4} \dot{\beta}_{y,i}^2, \qquad J_{\dot{\beta}_z} = \sum_{i=1}^{4} \dot{\beta}_{z,i}^2 \tag{5.11}$$

$$J_{\dot{\alpha}} = \dot{\alpha}^2, \qquad J_{\dot{\beta}} = \dot{\beta}^2 \tag{5.12}$$

Torques: Finally, quadratic penalties are imposed on the torques generated by the propulsive and aerodynamic forces:

$$J_{\tau^{prop}} = \sum_{i=1}^{3} \tau_B^{prop}(i)^2 \tag{5.13}$$

$$J_{\tau^{aero}} = \sum_{i=1}^{3} \tau_B^{aero}(i)^2 \tag{5.14}$$

The inclusion of the propulsive and aerodynamic torque penalties in the cost function is motivated by the fact that, despite a 3-DoF formulation, we want to ensure that the resulting solution remains compatible with a full six-degree-of-freedom (6-DoF) model. In other words, the computed trajectory must be trimmable.

Table 5.2: NLP box constraints and boundary conditions

Variable	Units	Lower Bound	Upper Bound	IC	FC
t	s	5	100	0	–
x	m	0	5000	5000	1
y	m	-5000	5000	100	0
z	m	-5000	5000	-200	0
v_x	m/s	-500	500	-219.43	-1
v_y	m/s	-0.1	500	-13.17	0
v_z	m/s	-500	500	-8.78	0
m	kg	m_{dry}	m_0	m_0	–
T_i	kN	$1.1\,T_{\min,i}$	$0.9\,T_{\max,i}$	–	–
$\beta_{y,i}$	deg	-5	5	–	0
$\beta_{z,i}$	deg	-5	5	–	0
α	deg	-10	10	–	0
β	deg	-10	10	–	0
$\dot{T}_i$	kN/s	$-\dot{T}_{\max}$	$\dot{T}_{\max}$	–	–
$\dot{\beta}_{y,i}$	deg/s	$-\dot{\beta}_{y,i}^{\max}$	$\dot{\beta}_{y,i}^{\max}$	–	0
$\dot{\beta}_{z,i}$	deg/s	$-\dot{\beta}_{z,i}^{\max}$	$\dot{\beta}_{z,i}^{\max}$	–	0
$\dot{\alpha}, \dot{\beta}$	deg/s	-5	5	–	0
Integral	–	0	100	–	–

5.1.5 Problem Statement

Collecting the elements defined above yields the following OCP:

Problem 1: Continuous-Time Free-Final-Time Non-Convex Problem

Minimize over $\mathbf{x}, \mathbf{u}, t_f$

$$J = \int_0^{t_f} \ell(\mathbf{x}, \mathbf{u}) d\tau$$

Subject to Dynamics:

$$\dot{\mathbf{r}} = \mathbf{v}$$

$$\dot{\mathbf{v}} = \mathbf{a}_{\text{grav}} + \mathbf{M}_{IB}\big(\mathbf{T}_B + \mathbf{F}_B\big)/m$$

$$\dot{m} = -\frac{\sum_{i=1}^{5} T_i}{g_0 I_{sp}}$$

Path Constraints:

$$\frac{x}{\sqrt{y^2 + z^2}} \geq \tan\gamma_{gs}$$

Box Constraints:

$$\mathbf{x}(t) \in [\mathbf{x}_{\min}, \mathbf{x}_{\max}]$$

$$\mathbf{u}(t) \in [\mathbf{u}_{\min}, \mathbf{u}_{\max}]$$

Boundary Conditions:

$$\begin{aligned} \mathbf{r}(t_0) &= \mathbf{r}_0 & \mathbf{v}(t_0) &= \mathbf{v}_0 & m(t_0) &= m_0 \\ \mathbf{r}(t_f) &= \mathbf{r}_f & \mathbf{v}(t_f) &= \mathbf{v}_f & & \\ \alpha(t_f) &= 0 & \beta(t_f) &= 0 & & \\ \beta_{y,i}(t_f) &= 0 & \beta_{z,i}(t_f) &= 0 & i &= 1, \ldots, 4 \end{aligned}$$

5.1.6 Initialization

For the initial guess, the position and velocity states are obtained by linear interpolation between the boundary conditions, while the mass is initialized by linear interpolation between the initial and dry mass. The thrust states are also initialized by linear interpolation between their minimum and maximum bounds, and all remaining variables are set to zero when no prior information is available.

5.1.7 Scaling

One important aspect that strongly influences the performance of an optimization algorithm is the scaling of the problem. A problem is considered poorly scaled if changes in one optimization variable lead to significantly larger variations in the cost function than those caused by variations in another variable [55]. In such cases, the numerical optimizer may struggle, leading to slow convergence or even failure to converge. In comparison, a properly scaled NLP can be solved much more efficiently.
Scaling issues typically arise when the relative magnitudes of variables or constraints differ significantly, or when their values deviate significantly from unity. A standard approach is to apply a transformation

of units so that optimization variables lie approximately within the range $[-1, 1]$.
Four fundamental scaling quantities are typically selected: length, speed, time, and mass. All other scales are derived from these [60]. In this work, the length, speed, and mass scales are set to their maximum values, while scales for other variables are obtained by combining these fundamental quantities. The adopted scaling is summarized in Table 5.3.

Table 5.3: Scaling of fundamental and derived quantities

Quantity	Symbol	Definition	Value
Length	L	-	$5000\,\mathrm{m}$
Speed	V	-	$500\,\mathrm{m/s}$
Time	T	L/V	$10\,\mathrm{s}$
Mass	M	-	$4300\,\mathrm{kg}$
Acceleration	a	$V/(L/V)$	$50\,\mathrm{m/s^2}$
Area	A	L^2	$2.5 \times 10^7\,\mathrm{m^2}$
Volume	V_{ol}	L^3	$1.25 \times 10^{11}\,\mathrm{m^3}$
Density	ρ	M/V_{ol}	$3.44 \times 10^{-8}\,\mathrm{kg/m^3}$
Thrust	F	$MV/(L/V)$	$2.1 \times 10^5\,\mathrm{N}$

Once the scaling factors are defined, all bounds, initial guesses, and problem parameters are expressed in the scaled domain, which allows the NLP solver to handle variables of comparable magnitudes and thereby improves convergence.

5.1.8 Nominal Trajectory Solution

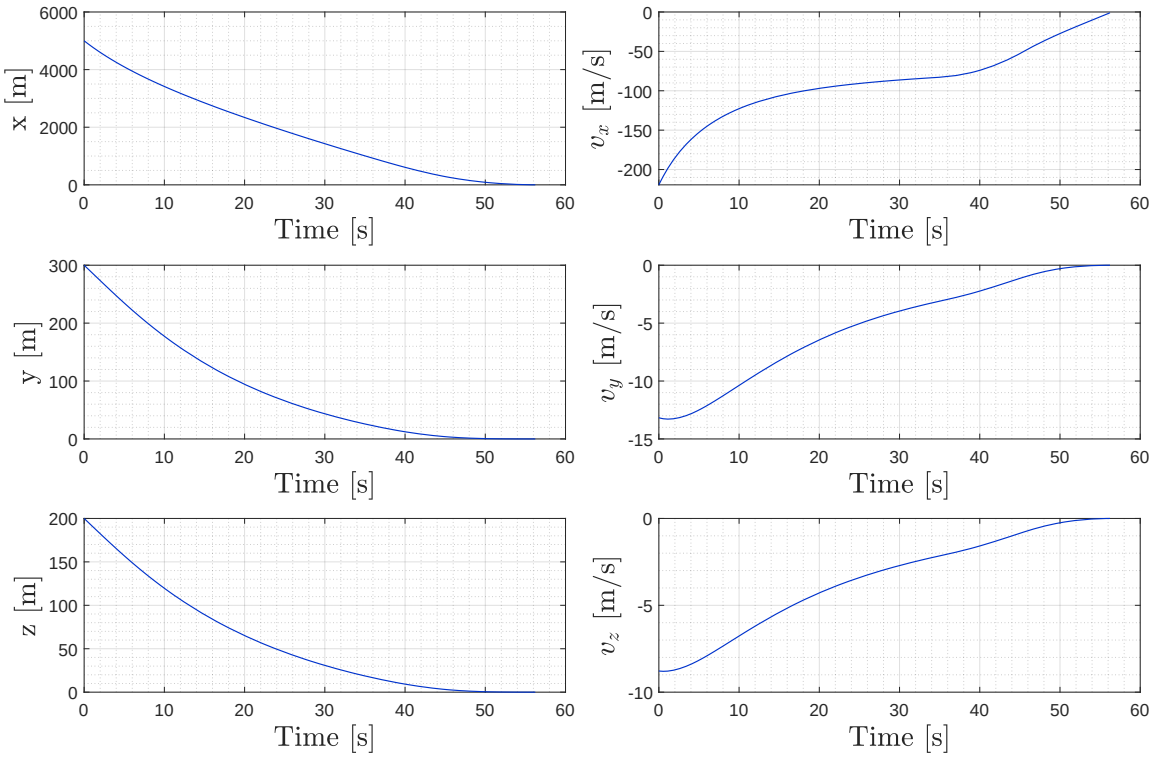
The OCP defined in Problem 1 is solved using *GPOPS-II* [58], a MATLAB package that implements variable-order Gaussian quadrature collocation methods, transforming the continuous-time OCP into a sparse NLP, which is then solved using *IPOPT* [90]. Although *GPOPS-II* provides automatic scaling options, manual scaling is often preferred as it ensures better conditioning of the problem.
Figure 5.1a shows the position and velocity profiles, while Figure 5.1b illustrates the mass consumption. Figures 5.2a and 5.2b show the propulsive and aerodynamic augmented states, which correspond to the actual control inputs to the system as the augmentation introduced in Section 5.1.1 is applied solely for numerical reasons and does not alter the physical interpretation of the controls. The vehicle lands with approximately 970 kg of fuel remaining, corresponding to about 1% of the total propellant mass, which is consistent with the 0.7% margin reported in the benchmark [77].

5.1.9 Solution Verification

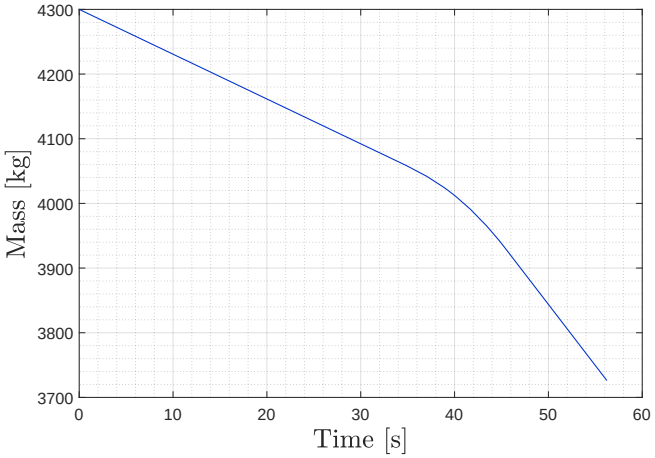
Optimality Check

One of the defining properties of the Gauss pseudospectral method is that it enables the Karush–Kuhn–Tucker (KKT) of the NLP to be mapped to the costates of the continuous-time OCP [13]. Therefore, the optimality of the solution can be verified a posteriori via the same procedure as in [70]. The costate vector is defined as

$$\boldsymbol{\lambda} = \begin{bmatrix} \boldsymbol{\lambda}_r^T & \boldsymbol{\lambda}_v^T & \lambda_m & \boldsymbol{\lambda}_T^T & \boldsymbol{\lambda}_{\beta_y}^T & \boldsymbol{\lambda}_{\beta_z}^T & \lambda_\alpha & \lambda_\beta \end{bmatrix}^T \in \mathbb{R}^{22} \tag{5.15}$$

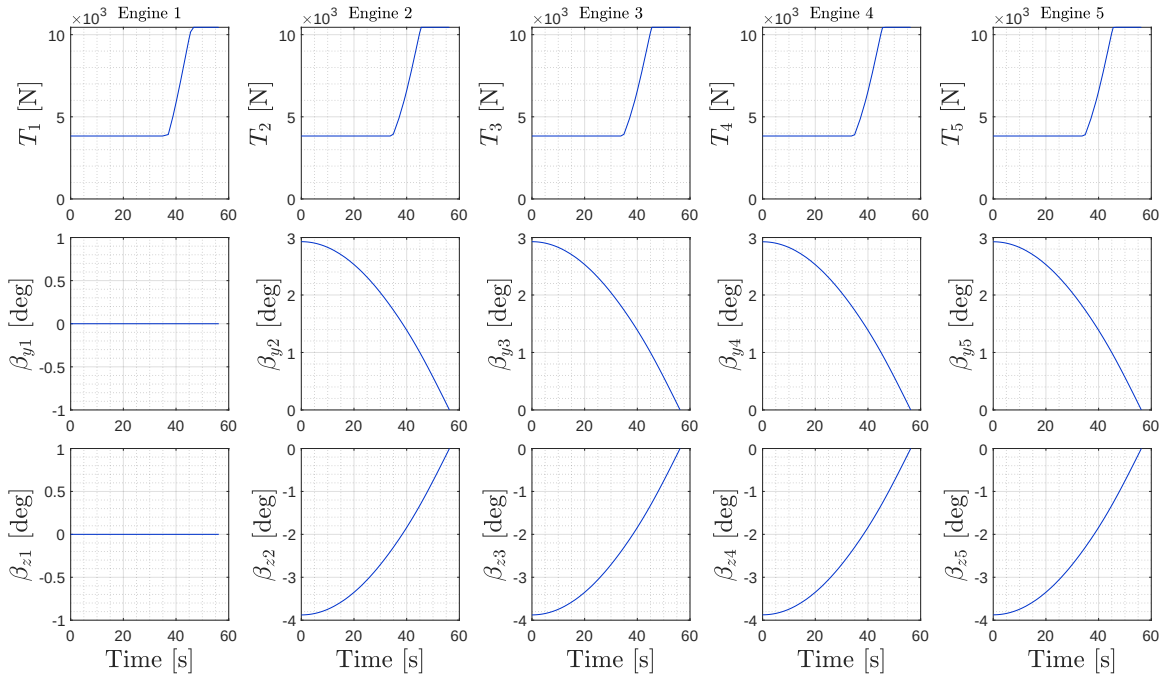


(a) Position and velocity.

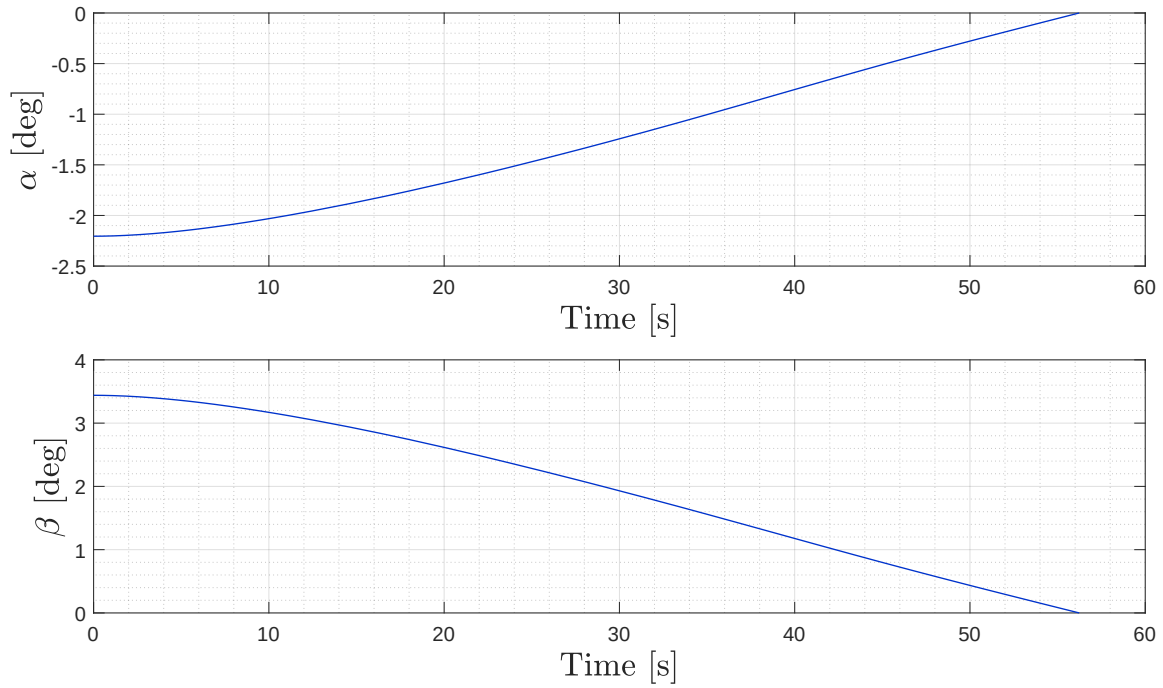


(b) Mass.

Figure 5.1: *GPOPS* states: position, velocity, and mass.



(a) Propulsive controls.



(b) Aerodynamic controls.

Figure 5.2: *GPOPS* controls.

and the Hamiltonian can be formulated as

$$H = \boldsymbol{\lambda}_r^T \mathbf{v} + \boldsymbol{\lambda}_v^T \mathbf{f}_\mathbf{v}(\mathbf{x}, \mathbf{u}) - \lambda_m \frac{\sum_{i=1}^{5} T_i}{I_{sp} g_0} + \sum_{i=1}^{5} \lambda_{T,i}\, \dot{T}_i + \sum_{i=1}^{4} \lambda_{\beta_y,i}\, \dot{\beta}_{y,i} + \\ + \sum_{i=1}^{4} \lambda_{\beta_z,i}\, \dot{\beta}_{z,i} + \lambda_\alpha\, \dot{\alpha} + \lambda_\beta\, \dot{\beta} + \ell(\mathbf{x}, \mathbf{u}) \tag{5.16}$$

with $\mathbf{f}_v(\mathbf{x}, \mathbf{u})$ given by (4.5).
Since the Hamiltonian does not depend explicitly on time, it remains constant along the optimal trajectory. Moreover, as this is a free final time problem, the transversality condition [21] requires

$$H(t_f) = 0 \tag{5.17}$$

Hence, the Hamiltonian along the optimal trajectory satisfies

$$H(t) = 0, \quad t \in [t_0, t_f] \tag{5.18}$$

The optimality check therefore consists of reconstructing the Hamiltonian using the costates returned by *GPOPS-II* and verifying that it remains close to zero. As these costates are approximations of those of the original OCP, this procedure provides an a posteriori validation of the computed solution. Figure 5.3 shows the reconstructed Hamiltonian, which remains close to zero throughout the trajectory, thereby confirming the optimality of the solution.

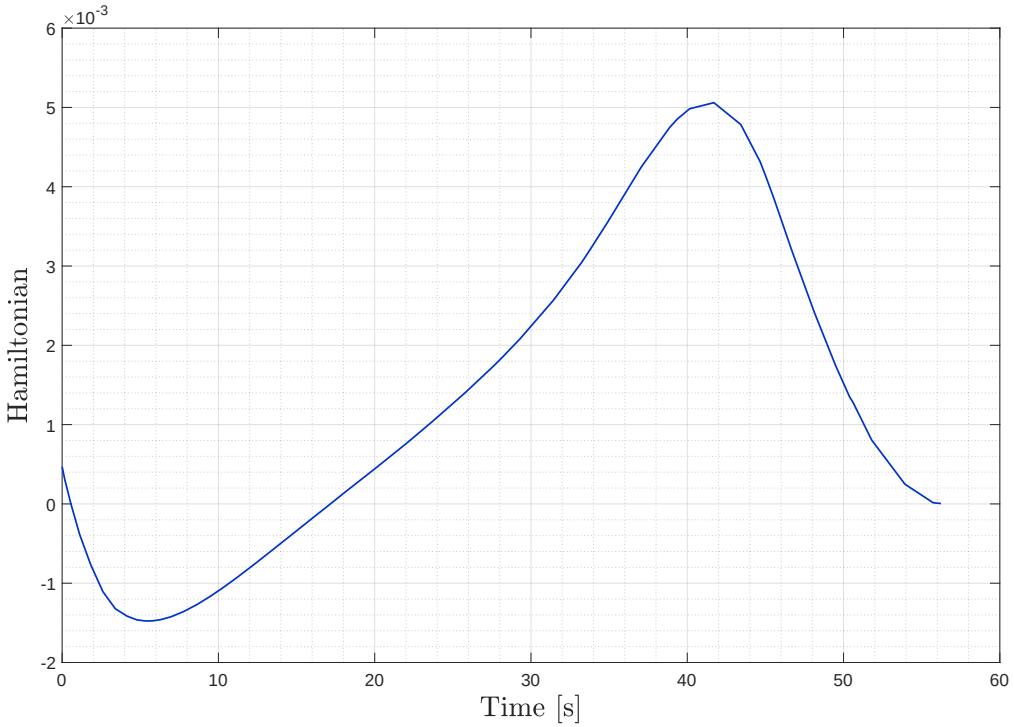


Figure 5.3: Optimality check: reconstructed Hamiltonian.

Feasibility Check

An a posteriori feasibility check of the solution consists in verifying that the state trajectories obtained through a Runge–Kutta 45 propagation (ode45 in MATLAB, implementing the Dormand–Prince method [27, 75]) of the initial conditions, using the interpolated controls from the numerical solution of the OCP, remain within a specified accuracy of those computed by the numerical optimization [43]. This validation confirms that the optimized trajectory satisfies the full nonlinear dynamics, with errors on the order of 2 m in position, as shown in Figure 5.5. Figure 5.4 compares the Runge–Kutta propagation with the *GPOPS-II* solution for the position and velocity states.

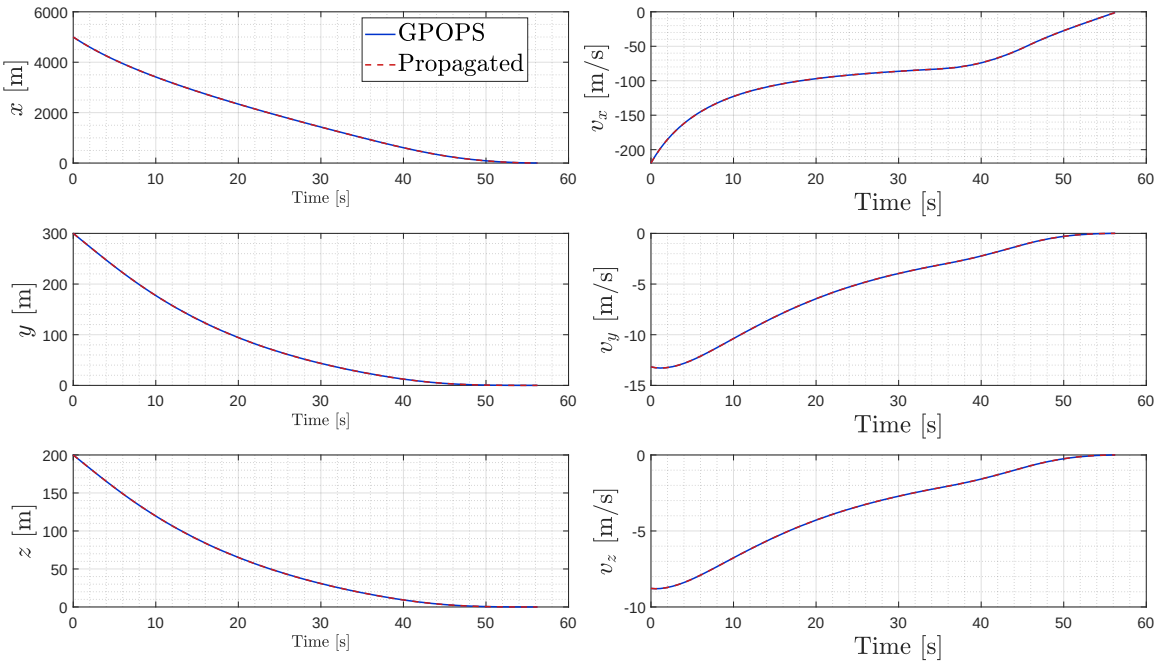


Figure 5.4: Validation of *GPOPS* solutions: discrepancy between Runge-Kutta and *GPOPS* states.

5.2 Nominal Controller

The trajectory optimization described in the previous section provides an open-loop guidance solution, i.e., a sequence of states and control inputs that satisfy the dynamics. However, in practice such an open-loop approach cannot be followed accurately due to modeling limitations, parameter uncertainties, and external disturbances. A feedback controller is therefore required to track the reference trajectory and ensure robust performance under these nonideal conditions, as illustrated in the block diagram of Figure 5.6. The design of the controllers follows a methodical procedure [68], summarized below:

1. Select the number of operating points at which controllers will be synthesized.
2. Identify a parameter that changes monotonically along the trajectory to serve as the scheduling variable. In this work, time is chosen.
3. Sample the states and control inputs of the reference trajectory at discrete points along the scheduling parameter.

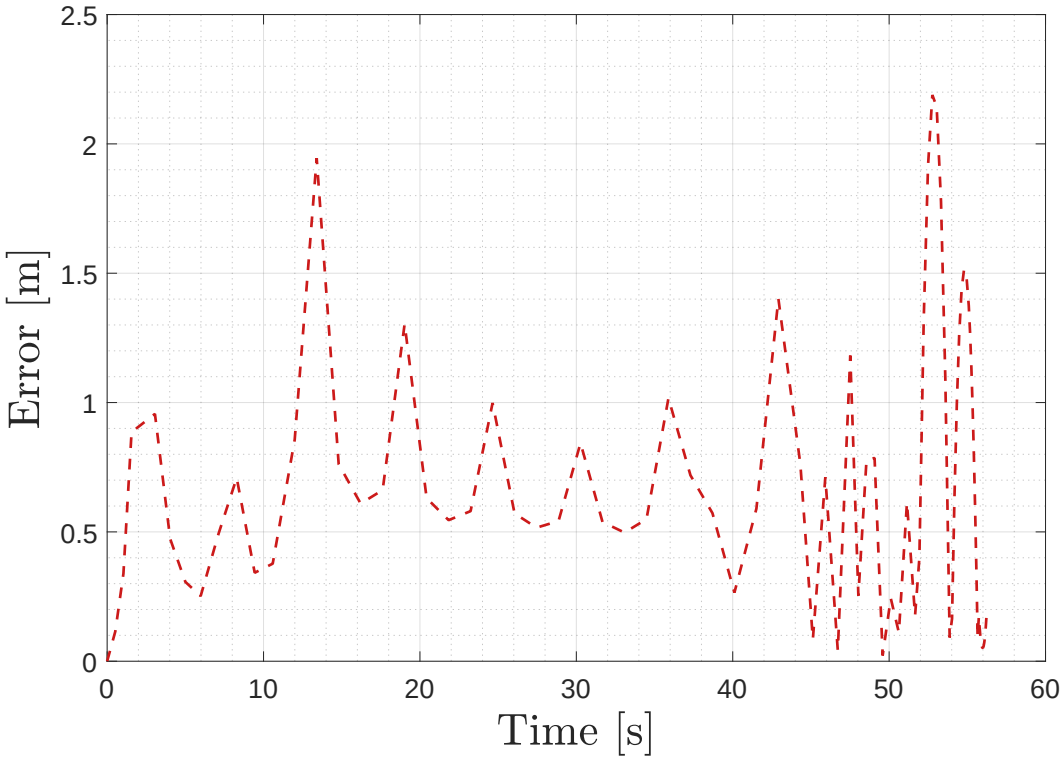


Figure 5.5: Validation of GPOPS solutions: position error over time.

4. Linearize the nonlinear equations of motion numerically at each sampled point, yielding a family of linear time-invariant (LTI) systems.
5. Design and analyze controllers for each of these LTI models.
6. Evaluate the individual controllers in both frequency and time domains.
7. Implement a gain-scheduled controller by interpolating between the gain matrices according to the current time.
8. Assess the closed-loop performance of the overall nonlinear system under the scheduled control law.

In this work, 13 operating points are distributed along the reference trajectory, at which a Linear Quadratic Regulator (LQR) [6] is synthesized. LQR is adopted due to its simplicity, as the primary focus of this thesis lies on the guidance aspects. Its objective is to track the nominal trajectory by minimizing a quadratic performance index that penalizes deviations in the states as well as the control effort. The weighting matrices Q and R determine the trade-off between tracking performance and actuation usage. The resulting feedback gains are applied in a state-feedback fashion, where the control action depends on the deviation from the reference trajectory. The controller consists of a single loop regulating translational motion by commanding thrust magnitudes, TVC deflections, and aerodynamic angles based on errors in position and velocity, while actuator saturation limits are imposed separately on each control channel as LQR does not explicitly account for constraints.
The controller design is based on a reduced-order system, obtained by eliminating the mass row and column from the A matrix and the mass row from the B matrix of each LTI model derived in Step 4. This

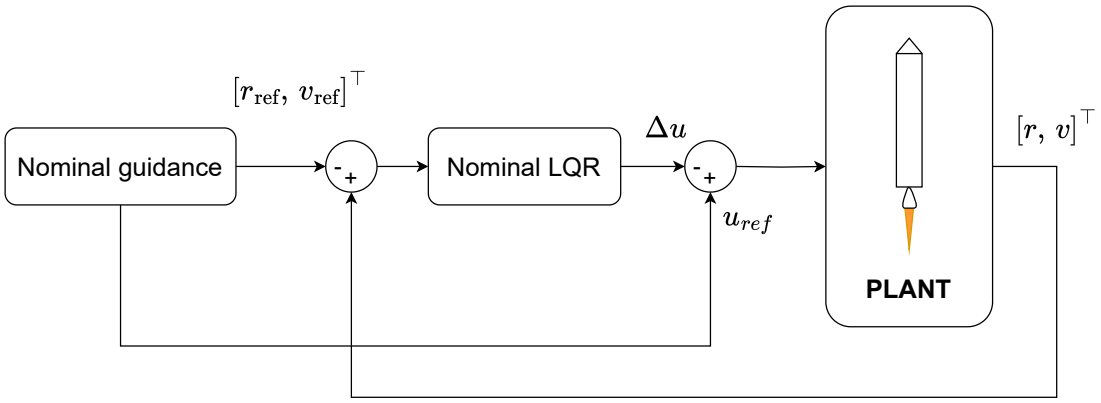


Figure 5.6: Block diagram: nominal closed-loop.

reduction is motivated by the fact that vehicle mass cannot be directly measured or influenced through feedback, and its inclusion in the control law would therefore be meaningless. From this procedure, 13 gain matrices of dimension 15×6 are obtained, leading to the following control law:

$$\mathbf{u} = \mathbf{u}_{\text{ref}} - \mathbf{K}_{\text{LQR}} \left(\mathbf{x} - \mathbf{x}_{\text{ref}}\right), \tag{5.19}$$

where the reference control vector is

$$\mathbf{u}_{\text{ref}} = \left[T_1,\, T_2,\, T_3,\, T_4,\, T_5,\, \beta_{y,1},\, \beta_{y,2},\, \beta_{y,3},\, \beta_{y,4},\, \beta_{z,1},\, \beta_{z,2},\, \beta_{z,3},\, \beta_{z,4},\, \alpha,\, \beta\right]^\top, \tag{5.20}$$

and the state vector is

$$\mathbf{x} = \begin{bmatrix}\mathbf{r}^T & \mathbf{v}^T\end{bmatrix}^\top, \quad \mathbf{x}_{\text{ref}} = \begin{bmatrix}\mathbf{r}_{\text{ref}}^T & \mathbf{v}_{\text{ref}}^T\end{bmatrix}^\top. \tag{5.21}$$

5.2.1 Linear Analysis of the Closed-Loop System

The sensitivity and complementary sensitivity transfer functions of the closed-loop system, consisting of the controller in feedback with the plant, are considered.

Frequency Domain Analysis

The Nichols chart is a standard tool for assessing the stability and robustness of a control system, as it allows the gain and phase margins to be read directly [36]. These margins quantify how close the system operates to instability, which occurs when the gain reaches 0 dB or higher while the phase simultaneously equals $-180°$. By providing a visual representation of this relationship, the Nichols chart offers a convenient way to evaluate robustness. Figure 5.7 presents the Nichols chart for the synthesized controllers. No instabilities are observed, and, because the system dynamics resemble those of a double integrator, the gain margins for all input channels are infinite.

The output sensitivity transfer function S_o indicates good disturbance rejection at low frequencies, with no significant peaks present (Figures 5.8a and 5.8b).

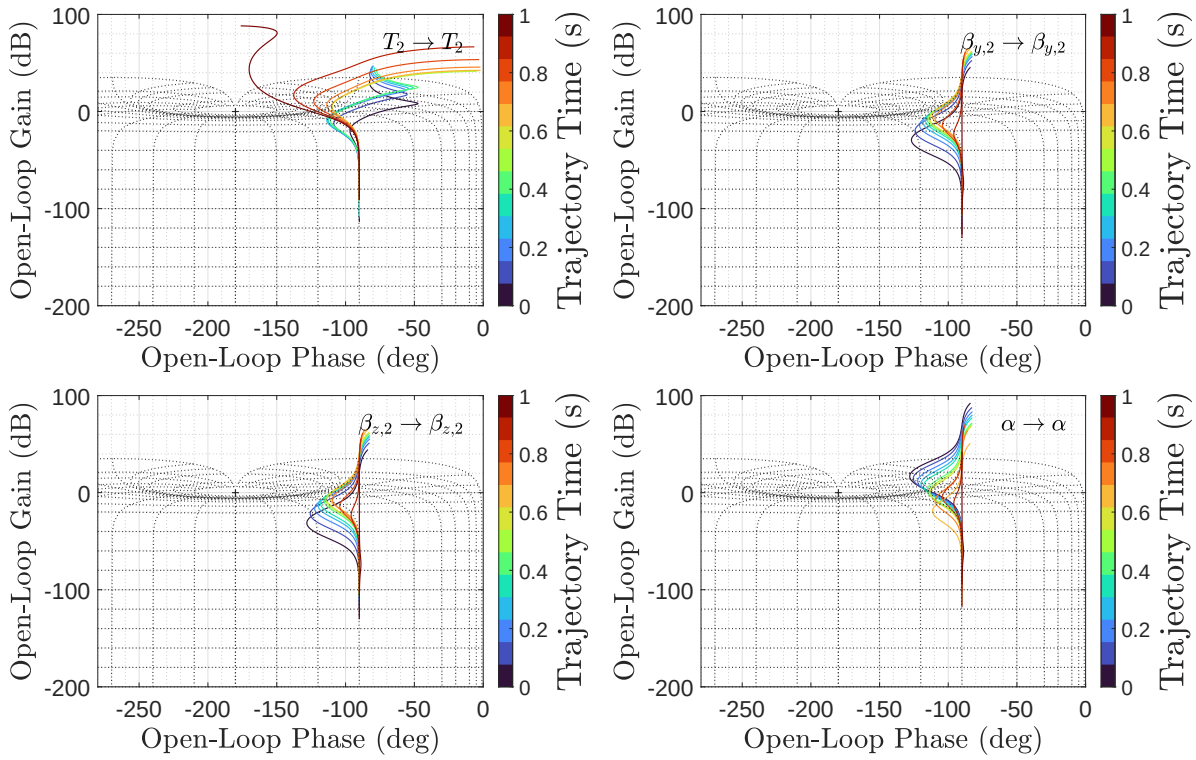


Figure 5.7: Closed-loop stability analysis: Nichols charts.

Time Domain Analysis

Step responses of the position and velocity outputs are computed to evaluate the closed-loop behavior in the time domain. For command tracking, the steady-state values are expected to converge to unity, whereas for disturbance rejection they should remain close to zero. Figures 5.9a and 5.9b confirm these expectations, showing accurate tracking performance in both position and velocity for the vertical and lateral channels.

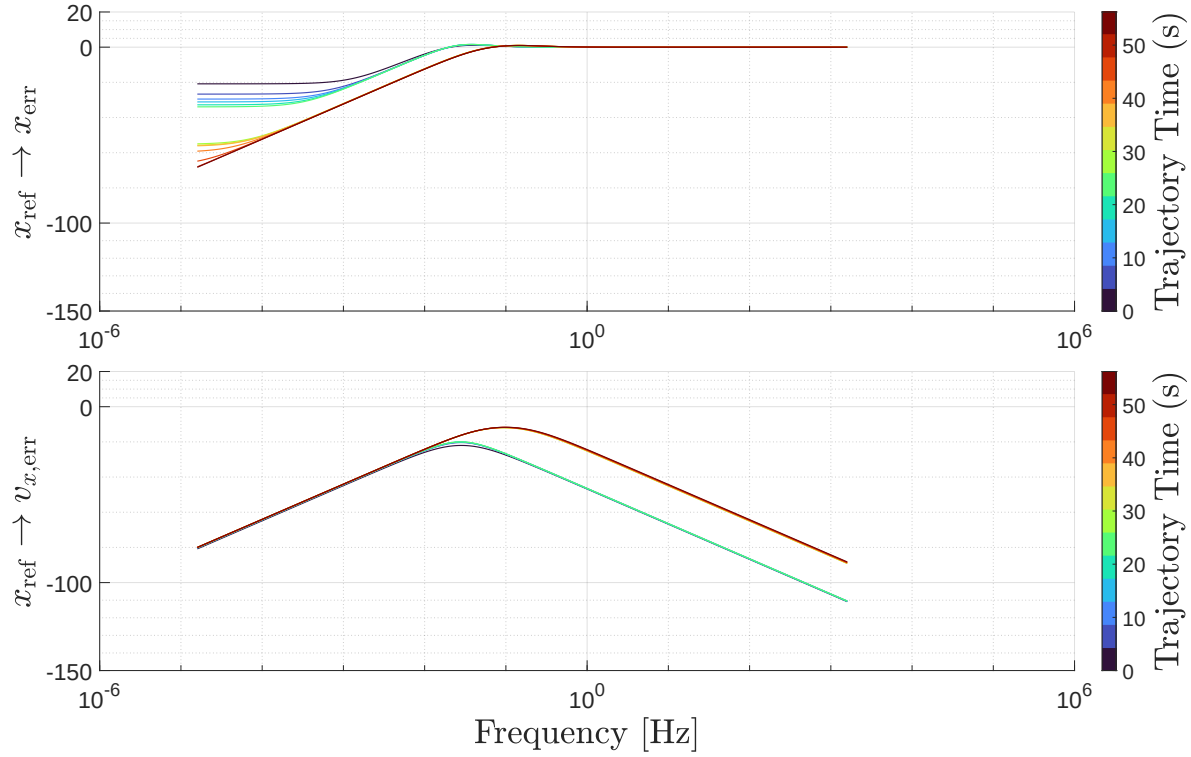
5.2.2 Nonlinear Analysis

The controller is evaluated in a simulated closed-loop environment on a plant that evolves according to (4.37) and accounts for uncertainties in the initial conditions and aerodynamic parameters, which are summarized in Table 5.4, with $\mathcal{N}$ denoting normally distributed variables and $\mathcal{U}$ denoting uniformly distributed variables.

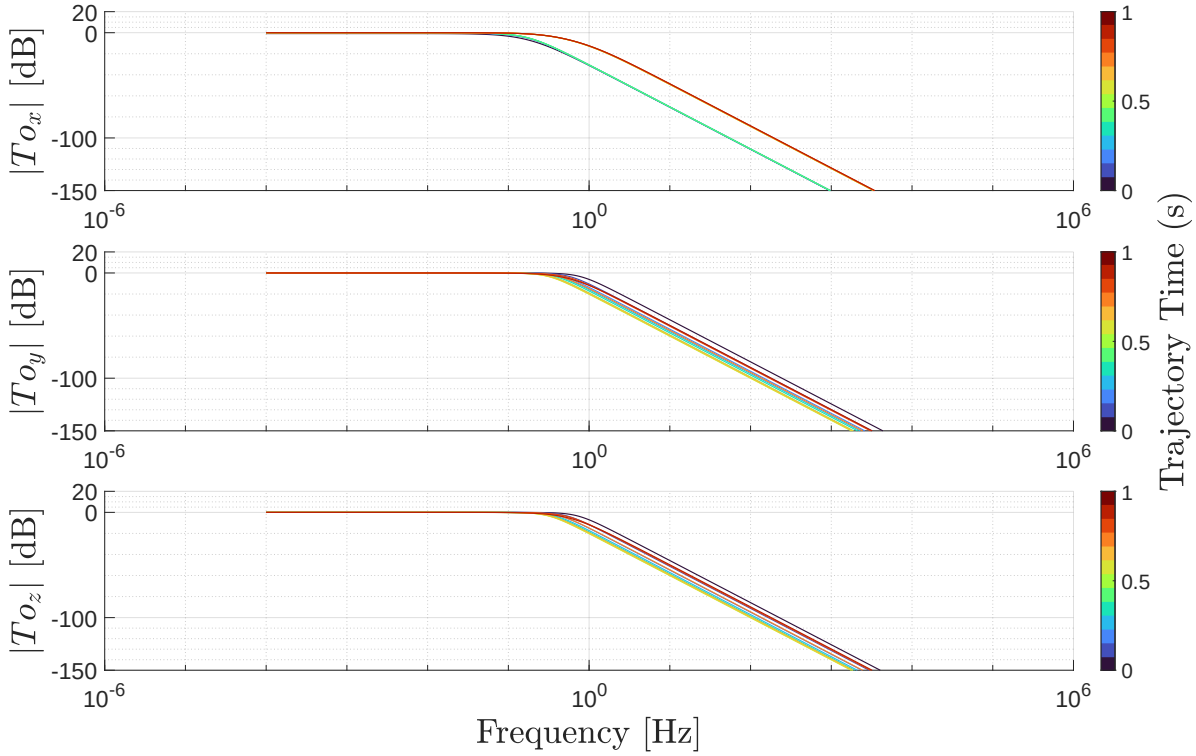
Uncertainties in the initial conditions are summed to the nominal ones while perturbations in air density and aerodynamic coefficients are introduced multiplicatively:

$$\rho = (1 + \Delta\rho)\,\rho_{\text{nom}} \tag{5.22}$$

$$C_x = (1 + \Delta C_x)\,C_{x,\text{nom}}, \quad C_y = (1 + \Delta C_y)\,C_{y,\text{nom}}, \quad C_z = (1 + \Delta C_z)\,C_{z,\text{nom}} \tag{5.23}$$

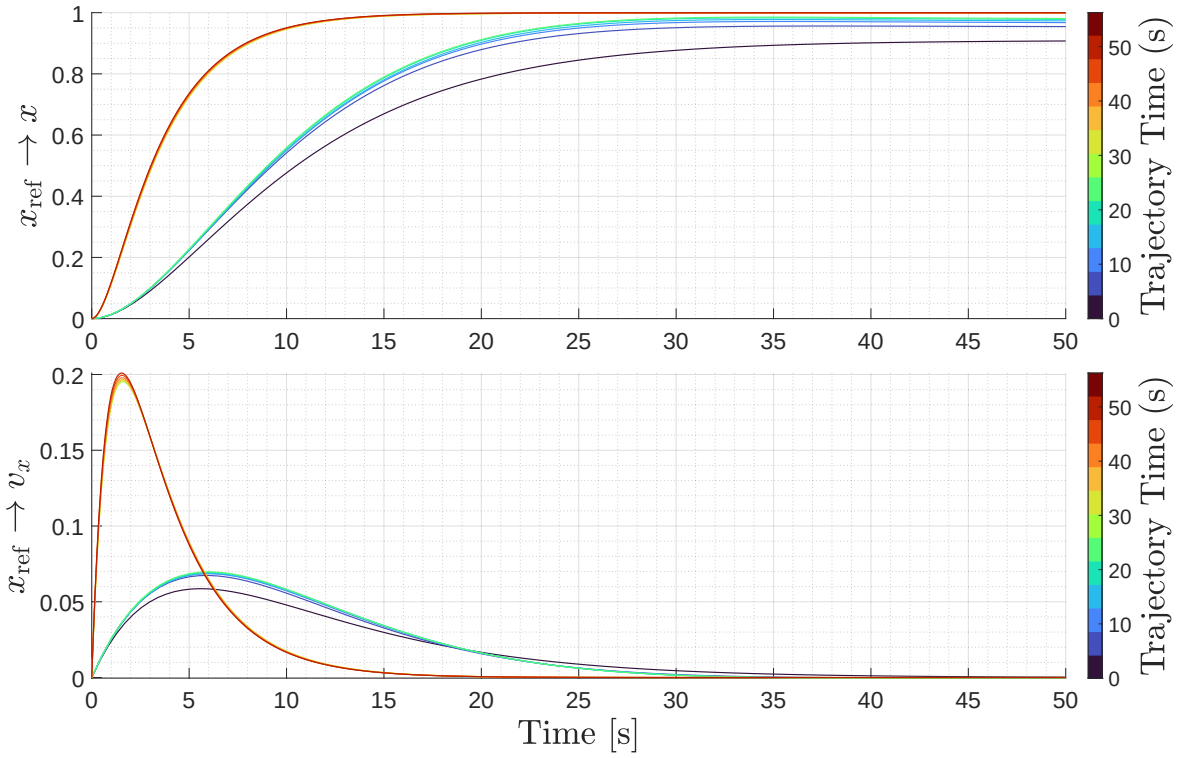


(a) Output sensitivity for x-position.

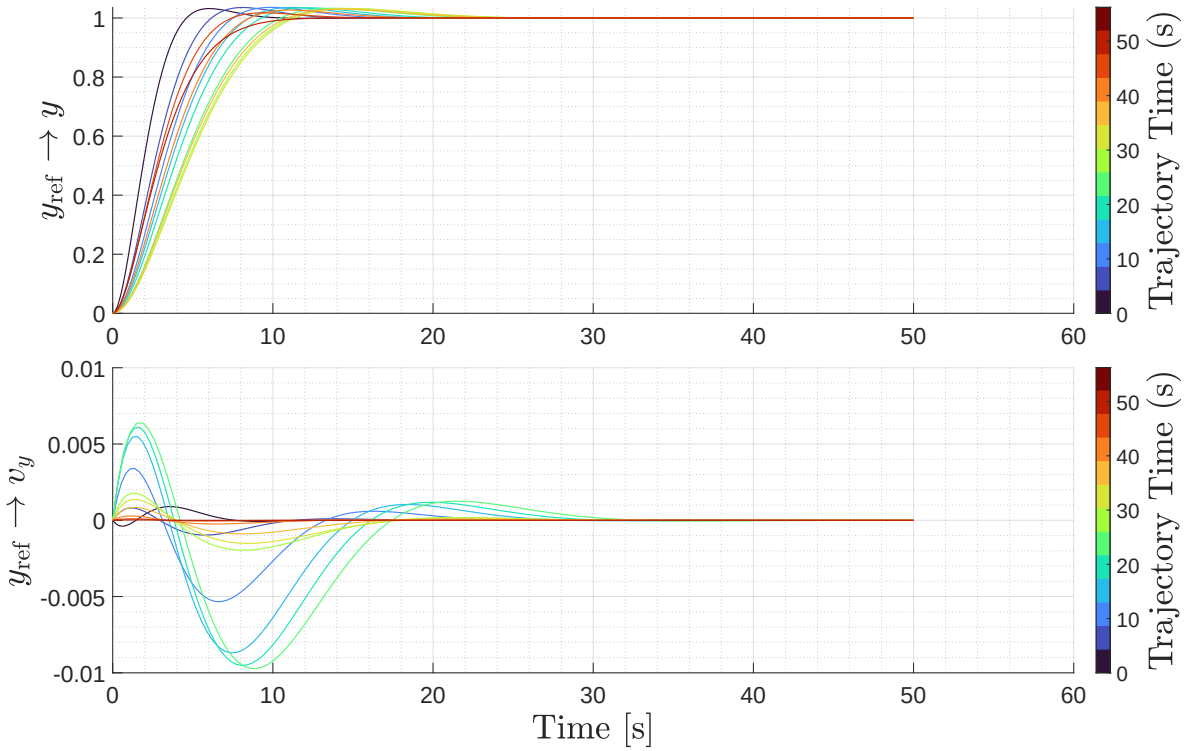


(b) Complementary sensitivity for position channels.

Figure 5.8: Frequency domain analysis.



(a) Step response for x-position.



(b) Step response for y-position.

Figure 5.9: Time domain analysis: step responses for x- and y-position.

Table 5.4: Initial conditions and parameter uncertainties

Variable	Distribution	Units
$\Delta r_{y,z}$	$\mathcal{N}(0, 100)$	m
$\Delta v_{x,y,z}$	$\mathcal{N}(0, 20)$	m/s
Δm	$\mathcal{U}(-2\%, 2\%)$	kg
$\Delta \rho$	$\mathcal{N}(0, 10\%)$	kg/m^3
$\Delta C_{x,y,z}$	$\mathcal{N}(0, 10\%)$	-

To evaluate the performance and robustness of the controller, a Monte Carlo campaign of 1000 cases is conducted with uncertainties sampled according to Table 5.4. A simulation is considered successful if the landing requirements listed in Table 5.5 are satisfied.
Figure 5.10 illustrates the results, showing adequate trajectory tracking under uncertainties, with approximately 70% of the cases resulting in a successful landing.

Table 5.5: Successful landing requirements

Requirement	Value
Vertical velocity at touchdown	$v_x(t_f) \geq -2.5\,\mathrm{m/s}$
Lateral velocities at touchdown	$\lvert v_y(t_f)\rvert, \lvert v_z(t_f)\rvert \leq 1.5\,\mathrm{m/s}$
Lateral position error at touchdown	$\lvert y(t_f)\rvert, \lvert z(t_f)\rvert \leq 10\,\mathrm{m}$

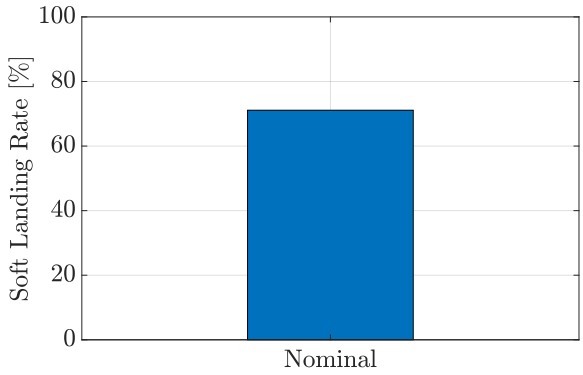


Figure 5.10: Closed-loop Monte Carlo campaign: successful landings.

In conclusion, the nominal trajectory provides the baseline against which the solutions of the following chapter are compared, while the gain-scheduled LQR controller is included to enable closed-loop simulations presented in Chapter 7.

6 Online Guidance

6.1 Sequential Convex Programming: Theoretical Overview

Sequential Convex Programming (SCP) is an iterative approach that tackles nonconvex optimization problems by repeatedly solving convex subproblems. The key idea is to retain the convex parts and replace nonconvex components with convex approximations [28]. Consider a general nonlinear optimization problem of the form

$$\begin{aligned} \min_{x} \quad & f(x) \\ \text{subject to} \quad & g_i(x) \leq 0, \quad i = 1, \ldots, m \\ & h_j(x) = 0, \quad j = 1, \ldots, p \end{aligned} \tag{6.1}$$

In this formulation, the objective function $f(x)$ and the inequality constraints $g_i(x)$ can be nonconvex, and the equality constraints $h_j(x)$ are generally non-affine, making the resulting optimization problem non-convex. SCP solves this problem iteratively by constructing an approximation of the solution $x^{(k)}$, while confining the optimization problem to a convex trust region around it [28]. Within this trust region:

- The nonconvex functions $f(x)$, $g_i(x)$ are replaced by convex approximations $\hat{f}(x)$, $\hat{g}_i(x)$ within the trust region $\mathcal{P}^{(k)}$.
- The nonlinear equality constraints $h_j(x)$ are replaced by affine approximations $\hat{h}_j(x)$ within the trust region $\mathcal{P}^{(k)}$.

These approximations define the convex optimization subproblem:

$$\begin{aligned} \min_{x} \quad & \hat{f}(x) \\ \text{s.t.} \quad & \hat{g}_i(x) \leq 0, \quad i = 1, \ldots, m \\ & \hat{h}_j(x) = 0, \quad j = 1, \ldots, p \\ & x \in \mathcal{P}^{(k)} \end{aligned} \tag{6.2}$$

with the trust region defined as

$$\mathcal{P}^{(k)} \;=\; \left\{ x \in \mathbb{R}^n \;\middle|\; \|x - x^{(k)}\|_2 \leq \rho^{(k)} \right\}. \tag{6.3}$$

While affine approximations obtained through linearization are the most common approach to handle nonlinear equality constraints, linearization about the current estimate does not always guarantee convergence of the iterative scheme. To address the shortcomings of pure linearization, the convex–concave decomposition (CCD) method has been shown to perform well for certain classes of problems with non-

linear equality constraints [48], however, for other types of problems it may suffer from infeasibility, particularly in the early SCP iterations. To overcome this limitation, [73] introduced the augmented convex–concave decomposition (ACCD), which guarantees feasibility from the very first iteration. Although these methods represent important developments, they are outside the scope of this thesis, and the present work adopts the standard SCP framework instead.

6.2 Algorithm Derivation

This section derives the SCP algorithm step by step from the nonconvex OCP introduced in Chapter 5 and summarized in Problem 1, and Figure 6.1 provides an overview of the procedure.

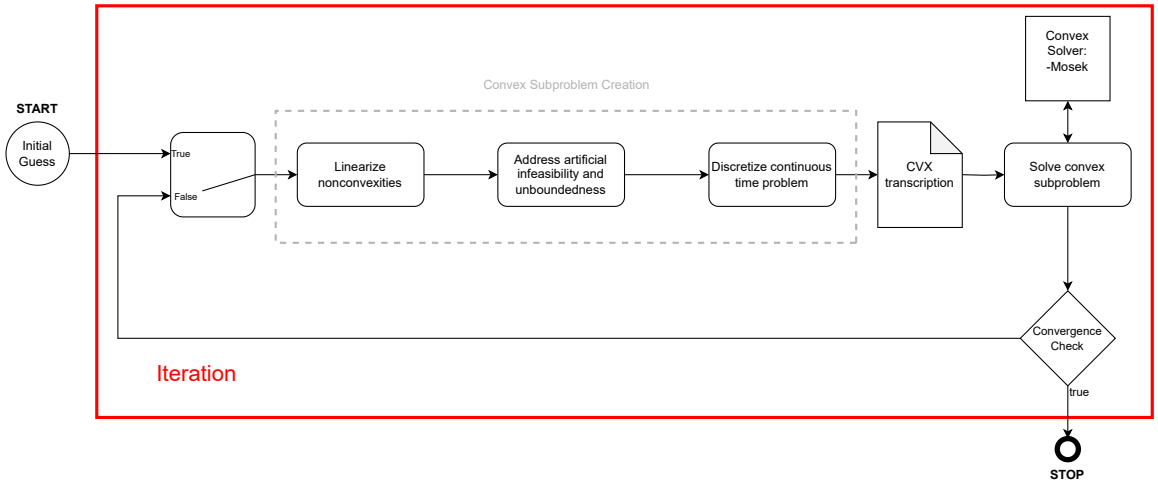


Figure 6.1: SCP algorithm schematic.

Problem 1 is a nonconvex optimal control problem because of nonlinearities in the system dynamics, and cost function. To understand which parts make the problem nonconvex, we analyze each component separately, separating convex and nonconvex terms. The analysis proceeds in three parts: the system dynamics, the cost function, and the path constraints. Finally, the formulation is extended to account for the free–final–time nature of the problem.

The system dynamics can be equivalently written by explicitly separating the convex and nonconvex terms:

$$\dot{\mathbf{x}} = \mathbf{f}_{\mathrm{nc}}(\mathbf{x}) + \mathbf{f}_{\mathrm{c}}(\mathbf{x}) + \mathbf{B}\mathbf{u} \tag{6.4}$$

where the convex terms are represented by

$$\mathbf{f}_{\mathrm{c}}(\mathbf{x}) = \begin{bmatrix} \mathbf{v} \\ \mathbf{0}_{3\times 1} \\ -\frac{\sum_{i=1}^{5} T_i}{I_{\mathrm{sp}} g_0} \\ \mathbf{0}_{15\times 1} \end{bmatrix} = \mathbf{A}_{\mathbf{c}}\mathbf{x} \tag{6.5}$$

with

$$\mathbf{A}_c = \left[\begin{array}{c|c|c} \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} & \mathbf{0}_{3\times16} \\ \hline \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times16} \\ \hline \mathbf{0}_{1\times7} & -\frac{1}{I_{\mathrm{sp}}g_0}\,\mathbf{1}_{1\times5} & \mathbf{0}_{1\times10} \\ \hline \mathbf{0}_{15\times3} & \mathbf{0}_{15\times3} & \mathbf{0}_{15\times16} \end{array}\right] \tag{6.6}$$

The control input matrix is defined as

$$\mathbf{B} = \begin{bmatrix} \mathbf{0}_{7\times15} \\ \mathbf{I}_{15\times15} \end{bmatrix} \tag{6.7}$$

Since $\mathbf{A}_c$ and $\mathbf{B}$ only contain constant terms, they need to be computed only once.
In Chapter 5, we motivated the choice of augmented states and control rates as inputs by noting that this formulation allows us to impose explicit bounds on the rates and to obtain smooth control sequences. In SCP, an additional benefit of using a system that is affine in the controls is that there is no need to compute Jacobians with respect to the control variables. According to [46], linearization with respect to the control variables introduces high-frequency chatter in the control profile due to numerical instability, which is generally detrimental to the convergence of the successive solution process. Small oscillations in $\mathbf{u}^{n-1}$ propagate into the elements of the linearized dynamics, and as a result the updated control $\mathbf{u}^n$ is likely to amplify these chattering effects.

The nonconvex terms are therefore limited to the velocity states

$$\mathbf{f}_{\mathrm{nc}}(\mathbf{x}) = \begin{bmatrix} \mathbf{0}_{3\times1} \\ \mathbf{a}_{\mathrm{grav}} + \mathbf{a}_{\mathrm{prop}} + \mathbf{a}_{\mathrm{aero}} \\ \mathbf{0}_{15\times1} \end{bmatrix} \tag{6.8}$$

and finally, the overall dynamics can be expressed as

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x},\mathbf{u}) = \mathbf{A_c}\mathbf{x} + \mathbf{f}_{\mathrm{nc}}(\mathbf{x}) + \mathbf{B}\mathbf{u} \tag{6.9}$$

The glideslope constraint (5.7) is conic by nature and therefore does not require convexification, it can in fact be expressed as SOC constraint [66].

$$\left(\left[y,\, z\right],\, \frac{x}{\tan\gamma_{gs}} \right) \in \mathcal{K}_3 \tag{6.10}$$

which, as discussed in Section 3.1.3, is equivalent to

$$\left\| \begin{bmatrix} y & z \end{bmatrix}^\top \right\|_2 \leq \frac{x}{\tan\gamma_{gs}} \tag{6.11}$$

The box constraints are linear inequalities and therefore fully compatible with convex optimization.
The remaining source of nonconvexity arises from the objective function, which, as with the dynamics, can be separated into convex and nonconvex terms:

$$\ell(\mathbf{x},\mathbf{u}) = \ell_c(\mathbf{x},\mathbf{u}) + \ell_{nc}(\mathbf{x},\mathbf{u}) \tag{6.12}$$

The convex term includes all sums of squares of states and controls (5.9)–(5.12) and since each term is a convex function, their sum preserves convexity (3.12).

$$\ell_{\mathrm{c}}(\mathbf{x},\mathbf{u}) = w_T J_T + w_{\dot{T}} J_{\dot{T}} + w_{\dot{\beta}_y\dot{\beta}_z}\left(J_{\dot{\beta}_y} + J_{\dot{\beta}_z}\right) + w_{\dot{\alpha}\dot{\beta}}\left(J_{\dot{\alpha}} + J_{\dot{\beta}}\right) \tag{6.13}$$

On the other hand, the propulsive and aerodynamic torques are formulated through nonlinear expressions of the optimization variables, thus their relative cost terms represent another source of nonconvexity.

$$\ell_{nc}(\mathbf{x},\mathbf{u}) = \ell_\tau(\mathbf{x},\mathbf{u}) = w_\tau^{\mathrm{prop}} J_\tau^{prop} + w_\tau^{\mathrm{aero}} J_\tau^{aero} \tag{6.14}$$

Lastly, to account for the free final time in the problem, the dynamics are expressed in terms of a normalized trajectory time $\tau \in [0,1]$, following the approach in [87]. The original time derivatives can then be related to the normalized time via the chain rule:

$$\frac{d\mathbf{x}(t)}{dt} = \frac{d\tau}{dt}\cdot\frac{d\mathbf{x}}{d\tau} \tag{6.15}$$

Introducing the time–dilation coefficient σ, which maps τ to the actual time t, gives

$$\sigma := \left(\frac{d\tau}{dt}\right)^{-1} = \frac{dt}{d\tau} = t_f - t_0 = t_f \tag{6.16}$$

With this definition, the dynamics in normalized time become

$$\mathbf{x}'(\tau) = \sigma\cdot\mathbf{f}(\mathbf{x}(\tau),\mathbf{u}(\tau)) \tag{6.17}$$

where the time–dilation coefficient σ is treated as an optimization variable.

6.2.1 Convexification Procedure

To construct a convex subproblem, we begin by eliminating all nonconvex components of the original problem. This is achieved by replacing each nonconvex term with its first-order approximation around a reference denoted as $\{\tilde{\mathbf{x}}(\tau),\tilde{\mathbf{u}}(\tau),\tilde{\sigma}\}$, quantities that will be explicitly defined in Section 6.2.2. Through the separation of convex and non-convex terms, we can apply a partial linearization and apply sequential convex programming by leveraging the difference between convex and non-convex terms.
Therefore, the normalized time dynamics (6.17) can be rewritten as:

$$\dot{\mathbf{x}}(\tau) = \mathbf{f}(\tilde{\mathbf{x}}(\tau),\tilde{\mathbf{u}}(\tau))\,\sigma + \mathbf{A}(\tau)\,\mathbf{x}(\tau) + \mathbf{B}_\sigma\,\mathbf{u}(\tau) + \mathbf{z}(\tau) \tag{6.18}$$

with

$$\mathbf{A}(\tau) = \left(\mathbf{A}_{\mathrm{nc}}(\tau) + \mathbf{A}_{\mathbf{c}}\right)\tilde{\sigma} \tag{6.19}$$

$$\mathbf{B}_\sigma = \mathbf{B}\,\tilde{\sigma} \tag{6.20}$$

and

$$\mathbf{A}_{\mathrm{nc}}(\tau) = \left.\frac{\partial\mathbf{f}_{\mathrm{nc}}(\mathbf{x},\mathbf{u})}{\partial\mathbf{x}}\right|_{\tilde{\mathbf{x}}(\tau),\,\tilde{\mathbf{u}}(\tau)} \tag{6.21}$$

$$\mathbf{z}(\tau) = -\mathbf{A}(\tau)\,\tilde{\mathbf{x}}(\tau) - \mathbf{B}_\sigma(\tau)\,\tilde{\mathbf{u}}(\tau) \tag{6.22}$$

Because of linearization, the resulting subproblem may become infeasible, even if the original nonconvex problem is feasible. This phenomenon, known as artificial infeasibility [49], typically occurs when the intersection of the convexified constraints is empty, which would prematurely terminate the algorithm. Artificial infeasibility is most common in the early iterations, often caused by a poor initial guess. For instance, if the initial estimate of the final time is far from a realistic value, the linearized dynamics may fail to admit any feasible solution.
A common remedy is to relax the linearized dynamics equality constraints by introducing slack variables, known as a virtual controls, which are left unconstrained but penalized heavily in the cost function [87].

$$\boldsymbol{\nu}(\tau) \in \mathbb{R}^{22} \tag{6.23}$$

With this modification, the linearized dynamics become

$$\dot{\mathbf{x}}(\tau) = \mathbf{f}(\tilde{\mathbf{x}}(\tau), \tilde{\mathbf{u}}(\tau))\,\sigma + \mathbf{A}(\tau)\,\mathbf{x}(\tau) + \mathbf{B}_\sigma\,\mathbf{u}(\tau) + \mathbf{z}(\tau) + \mathbf{C}\boldsymbol{\nu} \tag{6.24}$$

where the matrix C is a parameter that determines which virtual controls are introduced to assist the convergence process. In our case, we set

$$\mathbf{C} = \mathbf{I}_{22} \tag{6.25}$$

which implies that the virtual controls are applied to all the states. Intuitively, the virtual controls serve as auxiliary inputs that are only activated when necessary to prevent infeasibility. For the final solution to be feasible and physically consistent with the original problem, the virtual control terms must vanish. Following [87], each virtual control vector is bounded by a corresponding slack variable $\boldsymbol{\kappa}$:

$$\|\boldsymbol{\nu}(\tau)\|_2 \leq \boldsymbol{\kappa}(\tau) \tag{6.26}$$

and to progressively eliminate them during the optimization, an additional slack variable is used to bound the norm of $\boldsymbol{\kappa}$:

$$\|\boldsymbol{\kappa}\|_2 \leq s_\kappa, \quad s_\kappa \in \mathbb{R} \tag{6.27}$$

where s_κ is included in the cost function and weighted according to its corresponding penalty term.
The remaining nonconvex terms, specifically the torques in the cost function, can also be expressed in a convex form through linearization around the reference. The integrand is convexified by expanding the nonlinear torque terms to first order:

$$\ell_{\text{convex}}(\mathbf{x}, \mathbf{u}) = \ell_{\text{c}}(\mathbf{x}, \mathbf{u}) + \ell_\tau(\tilde{\mathbf{x}}, \tilde{\mathbf{u}}) + \left.\frac{\partial \ell_\tau}{\partial \mathbf{x}}\right|_{\tilde{\mathbf{x}},\tilde{\mathbf{u}}} (\mathbf{x} - \tilde{\mathbf{x}}) + \left.\frac{\partial \ell_\tau}{\partial \mathbf{u}}\right|_{\tilde{\mathbf{x}},\tilde{\mathbf{u}}} (\mathbf{u} - \tilde{\mathbf{u}}) \tag{6.28}$$

Since the torques only depend on the augmented states, the final integral expression becomes

$$\ell_{\text{convex}}(\mathbf{x}, \mathbf{u}) = \ell_{\text{c}}(\mathbf{x}, \mathbf{u}) + (\mathbf{A}_\tau^{\text{prop}} + \mathbf{A}_\tau^{\text{aero}})(\mathbf{x} - \hat{\mathbf{x}}) \tag{6.29}$$

with

$$\mathbf{A}_\tau^{\text{prop}} = \left.\frac{\partial \boldsymbol{\tau}_{\text{B}}^{\text{prop}}(\mathbf{x})}{\partial \mathbf{x}}\right|_{\tilde{\mathbf{x}}}, \quad \boldsymbol{\tau}_{\text{B}}^{\text{prop}}(\mathbf{x}) = \begin{bmatrix} \mathbf{0}_{3\times 8} & * & \mathbf{0}_{3\times 2} \end{bmatrix} \tag{6.30}$$

where the nonzero block $*$ corresponding to the columns from 9 to 20, and

$$\mathbf{A}_{\tau}^{\text{aero}} = \left.\frac{\partial \boldsymbol{\tau}_{\text{B}}^{\text{aero}}(\mathbf{x})}{\partial \mathbf{x}}\right|_{\tilde{\mathbf{x}}}, \quad \boldsymbol{\tau}_{\text{B}}^{\text{aero}}(\mathbf{x}) = \begin{bmatrix}\mathbf{0}_{3\times 20} & * \end{bmatrix} \tag{6.31}$$

where the nonzero block $*$ corresponding to the columns from 21 to 22.
Linear approximations are only accurate in the vicinity of the trajectory about which the system is linearized. For this reason, the subproblem solution must be kept reasonably close to the linearization point characterized by the reference solution, which in the case of SCP, is the solution at the previous iteration. This motivation leads to the use of trust region constraints. Different formulations have been proposed in the literature. One option is to prescribe a constant, user-specified trust region radius [44]. This approach decreases the optimization subproblem size since the radius is treated as fixed input rather than an optimization variable. Another strategy is to define update rules for the trust region radius, expanding or contracting it according to measures of how well the linear model captures the nonlinear dynamics [50, 92]. This approach comes with the benefit of still keeping the optimization problem size contained, as the radius is constant with respect to the optimization problem that is being solved, while taking into account other information and updating its value between iterations. A further alternative is to embed the trust region bounds directly into the optimization problem itself [67, 88].
Adopting the latter idea, we constrain the change between consecutive solutions by augmenting the problem with a slack variable that limits the maximum allowable deviation. In particular, we require

$$\left\| \mathbf{X}(\tau) - \tilde{\mathbf{X}}(\tau) \right\| \leq \boldsymbol{\eta}(\tau) \tag{6.32}$$

where $\mathbf{X}(\tau) = [\mathbf{x}(\tau), \mathbf{u}(\tau)]$ collects states and controls, $\hat{\mathbf{X}}(\tau) = [\tilde{\mathbf{x}}(\tau), \tilde{\mathbf{u}}(\tau)]$ denotes the corresponding values at the reference about which the linearization occurs, and $\boldsymbol{\eta}(\tau)$ specifies the allowable deviation, which is penalized in the cost through a corresponding slack variable:

$$\|\boldsymbol{\eta}\|_2 \leq s_\eta, \quad s_\eta \in \mathbb{R} \tag{6.33}$$

Lastly, the time-dilation coefficient is bounded through a fixed size trust region.

$$\|\sigma - \tilde{\sigma}\| \leq \eta_\sigma = const. \tag{6.34}$$

In order to obtain a fully linear cost function, as required for the SOCP formulation, the integral term is bounded by an additional slack variable $s_\int$, which is then penalized in the cost through a weighting factor $w_\int$:

$$\int_0^1 \ell_{\text{convex}}(\tau)\, d\tau \;\leq\; s_\int \tag{6.35}$$

The augmented cost function then becomes

$$J = w_\int\, s_\int + w_\eta\, s_\eta + w_\kappa\, s_\kappa + w_\sigma\, \sigma \tag{6.36}$$

where the last term penalizes the time–dilation coefficient σ, whose role will be clarified later.
With these modifications, the formulation results in a free-final-time, continuous-time convex optimal control problem, summarized as Problem 2.

Problem 2: Continuous-Time Fixed-Final-Time Convex Problem

Minimize over $\mathbf{x}, \mathbf{u}, \boldsymbol{\nu}, \boldsymbol{\eta}, \boldsymbol{\kappa}, s_\eta, s_\kappa, \sigma$

$$J = w_\int s_\int + w_\eta \cdot s_\eta + w_\kappa \cdot s_\kappa + w_\sigma \cdot \sigma$$

Subject to Linearized Dynamics:

$$\dot{\mathbf{x}}(\tau) = \mathbf{f}(\hat{\mathbf{x}}(\tau), \hat{\mathbf{u}}(\tau))\,\sigma + \mathbf{A}(\tau)\,\mathbf{x}(\tau) + \mathbf{B}_\sigma\,\mathbf{u}(\tau) + \mathbf{z}(\tau) + \mathbf{C}\boldsymbol{\nu}$$

Path Constraints:

$$\left([y,\, z],\, \frac{x}{\tan\gamma_{gs}}\right) \in \mathcal{K}_3$$

State and Control constraints:

$$\mathbf{x}(\tau) \in [\mathbf{x}_{\min}, \mathbf{x}_{\max}]$$

$$\mathbf{u}(t) \in [\mathbf{u}_{\min}, \mathbf{u}_{\max}]$$

Virtual Control and Trust Region Constraints:

$$\|\boldsymbol{\nu}(\tau)\|_2 \leq \boldsymbol{\kappa}(\tau)$$

$$\|\boldsymbol{\kappa}\|_2 \leq s_\kappa$$

$$\left\|\mathbf{X}(\tau) - \tilde{\mathbf{X}}(\tau)\right\| \leq \boldsymbol{\eta}(\tau)$$

$$\|\boldsymbol{\eta}\|_2 \leq s_\eta$$

$$\|\sigma - \tilde{\sigma}\| \leq \eta_\sigma$$

$$\int_0^1 \ell_{\text{convex}}(\tau)\,d\tau \;\leq\; s_\int$$

Boundary conditions:

$$\begin{aligned}
&\mathbf{r}(0) = \mathbf{r}_0, && \mathbf{v}(0) = \mathbf{v}_0 && m(0) = m_0 \\
&\mathbf{r}(1) = \mathbf{r}_f && \mathbf{v}(1) = \mathbf{v}_f && \\
&\alpha(1) = 0 && \beta(1) = 0 && \\
&\beta_{y,i}(1) = 0 && \beta_{z,i}(1) = 0 && i = 1, \ldots, 4
\end{aligned}$$

6.2.2 Discretization

To solve the continuous-time optimal control problem, we transcribe it into a finite-dimensional parameter optimization problem. This is achieved through temporal discretization, carried out using the trapezoidal scheme. The normalized time variable $\tau \in [0, 1]$ is partitioned uniformly into $K - 1$ intervals with K

nodes $\{\tau_1, \tau_2, \ldots, \tau_{K-1}, \tau_K\}$, where $\tau_1 = 0$, $\tau_K = 1$. The step size is therefore

$$\Delta\tau = \frac{1}{K-1}. \tag{6.37}$$

and normalized time at node k is

$$\tau_k = \frac{k-1}{K-1},\ k = 1, 2, \ldots, K \tag{6.38}$$

Here, the subscript k denotes the time step, while the superscript n indicates the iteration number. The discretized optimal control problem at iteration n is defined by the following dynamics:

$$\begin{aligned}\mathbf{x}_{k+1} = \mathbf{x}_k + \frac{\Delta\tau}{2}\Big[&\mathbf{f}_k^{n-1}\sigma + \mathbf{A}_k^{n-1}\mathbf{x}_k + \mathbf{B}_\sigma\mathbf{u}_k + \mathbf{z}_k^{n-1} + \mathbf{C}\boldsymbol{\nu}_k \\ &+ \mathbf{f}_{k+1}^{n-1}\sigma + \mathbf{A}_{k+1}^{n-1}\mathbf{x}_{k+1} + \mathbf{B}_\sigma\mathbf{u}_{k+1} + \mathbf{z}_{k+1}^{n-1} + \mathbf{C}\boldsymbol{\nu}_{k+1}\Big], \\ &k = 1, \ldots, K-1\end{aligned} \tag{6.39}$$

with

$$\mathbf{f}_k^{n-1} = \mathbf{f}(\mathbf{x}_k^{n-1}, \mathbf{u}_k^{n-1}) \tag{6.40}$$

$$\mathbf{A}_k^{n-1} = \left(\mathbf{A}_{\mathrm{nc},k}^{n-1} + \mathbf{A}_c\right)\sigma^{n-1} \tag{6.41}$$

$$\mathbf{B}_\sigma = \mathbf{B}\,\sigma^{n-1} \tag{6.42}$$

$$\mathbf{A}_{\mathrm{nc},k}^{n-1} = \left.\frac{\partial \mathbf{f}_{\mathrm{nc}}(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}^{n-1}} \tag{6.43}$$

$$\mathbf{z}_k^{n-1} = -\mathbf{A}_k^{n-1}\mathbf{x}_k^{n-1} - \mathbf{B}_\sigma\mathbf{u}_k^{n-1} \tag{6.44}$$

The continuous-time glideslope constraint in (6.10) is enforced at each discretization node through K SOC constraints:

$$\left([y_k,\ z_k],\ \frac{x_k}{\tan\gamma_{gs}}\right) \in \mathcal{K}_3 \tag{6.45}$$

As highlighted in Rao's survey [60], the quadrature used for the Lagrange term, that is, the integral term in the cost function, should match the numerical integration scheme adopted for the dynamics. Accordingly, the trapezoidal rule is also used to approximate the integral term.

$$\sum_{k=1}^{K-1} \frac{\Delta\tau}{2}\left[\ell_{\mathrm{convex},k}^n(\mathbf{x}_k, \mathbf{u}_k) + \ell_{\mathrm{convex},k+1}^n(\mathbf{x}_{k+1}, \mathbf{u}_{k+1})\right] \leq s_\int \tag{6.46}$$

with

$$\ell_{\mathrm{convex},k}^n(\mathbf{x}, \mathbf{u}) = \ell_c(\mathbf{x}_k, \mathbf{u}_k) + \boldsymbol{\tau}_{\mathrm{B}}(\mathbf{x}_k^{n-1})\sigma + \mathbf{A}_{\tau,k}^{n-1}\mathbf{x}_k + \mathbf{z}_{\tau,k}^{n-1} \tag{6.47}$$

$$\mathbf{A}_{\tau,k}^{n-1} = \left(\mathbf{A}_{\tau,k}^{n-1,\mathrm{prop}} + \mathbf{A}_{\tau,k}^{n-1,\mathrm{aero}}\right)\sigma^{n-1} \tag{6.48}$$

$$\mathbf{z}_{\tau,k}^{n-1} = -\mathbf{A}_{\tau,k}^{n-1}\mathbf{x}_k^{n-1} \tag{6.49}$$

$$\mathbf{A}_{\tau,k}^{n-1,\mathrm{prop}} = \left.\frac{\partial \boldsymbol{\tau}_{\mathrm{B}}^{\mathrm{prop}}(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}_k^{n-1}} \tag{6.50}$$

$$\mathbf{A}_{\tau,k}^{n-1,\text{aero}} = \left.\frac{\partial \boldsymbol{\tau}_{\text{B}}^{\text{aero}}(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}_k^{n-1}} \tag{6.51}$$

The virtual control vector at time step k, denoted by $\boldsymbol{\nu}_k \in \mathbb{R}^{22}$, is bounded by the corresponding slack variable κ_k, where $\boldsymbol{\kappa} = [\kappa_1, \ldots, \kappa_K]^\top \in \mathbb{R}^K$. The vector $\boldsymbol{\kappa}$ is itself bounded by the slack variable $s_\kappa \in \mathbb{R}$:

$$\|\boldsymbol{\nu}_k\|_2 \leq \kappa_k \tag{6.52}$$

$$\|\boldsymbol{\kappa}\|_2 \leq s_\kappa \tag{6.53}$$

Similarly, the trust region variable η_k at time step k bounds the deviation between the current optimization variables and their values at the previous iteration $n-1$. Collecting all η_k into the vector $\boldsymbol{\eta} \in \mathbb{R}^K$, we impose the following bounds:

$$\|\mathbf{x}_k - \mathbf{x}_k^{n-1}\|_2^2 + \|\mathbf{u}_k - \mathbf{u}_k^{n-1}\|_2^2 \leq \eta_k \tag{6.54}$$

$$\|\boldsymbol{\eta}\|_2 \leq s_\eta \tag{6.55}$$

The complete set of optimization variables, together with their dimensions, is summarized in Table 6.1, whereas Problem 3 presents the final discrete-time, fixed-final-time convex formulation, and 6.2.2 provides a high-level description of the SCP algorithm.

SCP Pseudocode

Input: Select weights $w_\eta, w_\kappa, w_\int, w_\sigma$, tolerances $\epsilon_\eta, \epsilon_\nu$, maximum iterations $N_{\max}$, and initial guess $\mathbf{z}^0 = \{\mathbf{x}^0, \mathbf{u}^0, \sigma^0\}$.

for $n \in \{2, \ldots, N_{\max}\}$ **do:**

1. Linearize torques about $\mathbf{z}^{n-1} = \{\mathbf{x}^{n-1}, \mathbf{u}^{n-1}, \sigma^{n-1}\} \rightarrow \mathbf{A}_\tau^{n-1}, \mathbf{z}_\tau^{n-1}$ using (6.48)–(6.51)
2. Linearize dynamics about $\mathbf{z}^{n-1} \rightarrow \mathbf{A}^{n-1}, \mathbf{f}^{n-1}, \mathbf{z}^{n-1}$ using (6.40)–(6.44)
3. Solve Problem (3) $\rightarrow \mathbf{z}^n = \{\mathbf{x}^n, \mathbf{u}^n, \sigma^n\}$
4. **if** (6.56) & (6.57):
   STOP
5. **else:**
   continue with the next iteration

6.2.3 Convergence Check

After each convex subproblem is solved, convergence is verified. If the criteria are satisfied, the algorithm terminates and the current solution is accepted. Convergence is declared when both of the following conditions hold:

$$s_\kappa \leq \epsilon_\nu \tag{6.56}$$

$$s_\eta \leq \epsilon_\eta \tag{6.57}$$

Condition (6.56) ensures that the virtual controls vanish, reflecting the fact that they are introduced only to aid convergence and to prevent infeasibility in the early iterations. Condition (6.57) requires the solution to remain sufficiently close to the one obtained at the previous iteration, which guarantees that the update lies within the region where the linear approximation is valid.

Table 6.1: SCP Optimization variables

Variable	Dimension	Description
$\mathbf{x}$	$[22 \times K]$	State trajectory
$\mathbf{u}$	$[15 \times K]$	Control trajectory
σ	$[1]$	Time-dilation coefficient
$\boldsymbol{\nu}$	$[22 \times K]$	Virtual controls
$\boldsymbol{\eta}$	$[K]$	Trust region size
$\boldsymbol{\kappa}$	$[K]$	Virtual controls vector slack
s_κ	$[1]$	Virtual controls slack
s_η	$[1]$	Trust region slack
$s_\int$	$[1]$	Integral slack

6.2.4 Initialization

The initial guess for the position and velocity states is obtained by linearly interpolating between the initial and final boundary conditions. The thrust magnitudes are initialized and held constant at their minimum value, while the TVC and aerodynamic angles are set to zero. All control rates are also initialized to zero. This simple initialization scheme is chosen to validate the performance of the algorithm by comparing it with the results obtained with *GPOPS-II* in Chapter 5.

Problem 3: Discrete-Time Fixed-Final-Time Convex Problem

Minimize over $\mathbf{x}, \mathbf{u}, \boldsymbol{\nu}, \boldsymbol{\kappa}, \boldsymbol{\eta}, s_\eta, s_\kappa, s_\int, \sigma$

$$J = w_\int s_\int + w_\eta \cdot s_\eta + w_\kappa \cdot s_\kappa + w_\sigma \cdot \sigma$$

Subject to linearized dynamics:

$$\begin{aligned}\mathbf{x}_{k+1} = \mathbf{x}_k + \frac{\Delta\tau}{2}\Big[&\mathbf{f}_k^{n-1}\sigma + \mathbf{A}_k^{n-1}\mathbf{x}_k + \mathbf{B}_\sigma\mathbf{u}_k + \mathbf{z}_k^{n-1} + \mathbf{C}\boldsymbol{\nu}_k \\ &+ \mathbf{f}_{k+1}^{n-1}\sigma + \mathbf{A}_{k+1}^{n-1}\mathbf{x}_{k+1} + \mathbf{B}_\sigma\mathbf{u}_{k+1} + \mathbf{z}_{k+1}^{n-1} + \mathbf{C}\boldsymbol{\nu}_{k+1}\Big], \\ &k = 1, \ldots, K-1\end{aligned}$$

Box constraints:

$$\mathbf{x}_k \in [\mathbf{x}_{\min}, \mathbf{x}_{\max}], \quad k = 1, \ldots, K$$

$$\mathbf{u}_k \in [\mathbf{u}_{\min}, \mathbf{u}_{\max}], \quad k = 1, \ldots, K$$

Path constraints:

$$\left([y_k,\, z_k],\, \frac{x_k}{\tan\gamma_{gs}}\right) \in \mathcal{K}_3, \quad k = 1, \ldots, K$$

Virtual control and trust region constraints:

$$\|\boldsymbol{\nu}_k\|_2 \leq \kappa_k, \quad k = 1, \ldots, K$$

$$\|\boldsymbol{\kappa}\|_2 \leq s_\kappa$$

$$(\mathbf{x}_k - \mathbf{x}_k^{n-1})^\top(\mathbf{x}_k - \mathbf{x}_k^{n-1}) + (\mathbf{u}_k - \mathbf{u}_k^{n-1})^\top(\mathbf{u}_k - \mathbf{u}_k^{n-1}) \leq \eta_k, \quad k = 1, \ldots, K$$

$$\|\boldsymbol{\eta}\|_2 \leq s_\eta$$

$$\|\sigma - \tilde{\sigma}\| \leq \eta_\sigma$$

$$\sum_{k=1}^{K-1} \frac{\Delta\tau}{2}\left[\ell_{\text{convex},k}^n(\mathbf{x}_k, \mathbf{u}_k) + \ell_{\text{convex},k+1}^n(\mathbf{x}_{k+1}, \mathbf{u}_{k+1})\right] \leq s_\int$$

Boundary conditions:

$$\mathbf{r}(0) = \mathbf{r}_0, \qquad \mathbf{v}(0) = \mathbf{v}_0, \qquad m(0) = m_0$$

$$\begin{aligned}\mathbf{r}(1) &= \mathbf{r}_f & \mathbf{v}(1) &= \mathbf{v}_f \\ \alpha(1) &= 0 & \beta(1) &= 0 \\ \beta_{y,i}(1) &= 0 & \beta_{z,i}(1) &= 0 \qquad i = 1, \ldots, 4\end{aligned}$$

6.2.5 Nominal SCP Validation

The problem is initialized with the line approach described earlier, and the resulting trajectories are evaluated to assess the performance of the SCP algorithm. Position, velocity and mass (Figures 6.2a and 6.2b) together with propulsive and aerodynamic controls (Figures 6.3a and 6.4a) closely match the nominal results. All constraints remain satisfied, and the final integral value differs by only 0.48% as shown in Table 6.3, confirming the validity of the proposed approach.

Small discrepancies appear in the control profiles: most notably, a lower thrust rate during the first and only thrust bang, which results from the higher penalization weight for the thrust derivative in SCP (Table 6.2), and small differences in the TVC and aerodynamic angles at the beginning of the trajectory, which remain well below 0.5°. Both the thrust and TVC rate profiles (Figure 6.3b) appear noisier than in the nominal case, despite the thrust rate integral being penalized more heavily in SCP. These differences can be attributed to several factors, including the distinct linearization and dynamics approximations, the convergence conditions, and the discretization schemes; in particular, whereas SCP applies a uniform trapezoidal discretization, *GPOPS-II* employs an *hp*-adaptive pseudospectral method and directly solves the original nonconvex formulation using the NLP solver *IPOPT* [90].

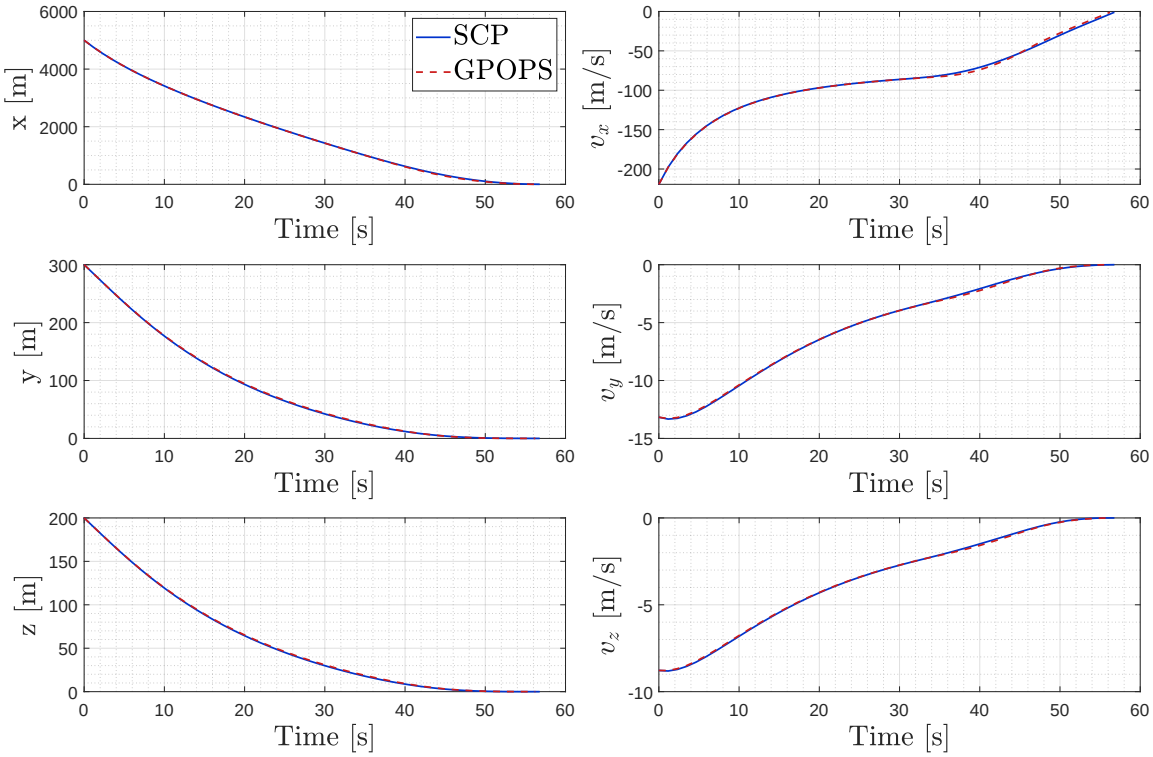
A further distinction arises in the cost function formulation. While the NLP directly minimizes the integral cost in (5.8), the SCP objective also includes penalty terms associated with virtual controls and trust region slack. In addition, a small penalty on the final time is introduced, as numerical experimentation showed that this improves the agreement with the *GPOPS-II* solution. The treatment of the integral itself differs in two ways: the torque contributions are included in linearized form, and the weight on the thrust-rate term is increased from 10^{-3} to 10^{-1}. Figure 6.5 shows the evolution of the slack variables for the virtual controls, trust region, and integral, together with the value of σ, across the SCP iterations. Starting from the initialization at $\sigma = 1$, the time-dilation coefficient increases to its realistic value of about 6 within the first eight iterations. During this phase, the virtual-control slack variable remains non-negligible, since the discretized dynamics constraint in (6.39) cannot be satisfied without artificial inputs. Once σ stabilizes around 6, the magnitude of the virtual controls rapidly vanishes to essentially zero (numerical values on the order of 10^{-18}, i.e. machine precision). From that point onward, both the integral slack and the trust region slack decrease, leading to convergence at iteration 13.

Table 6.2: SCP parameters

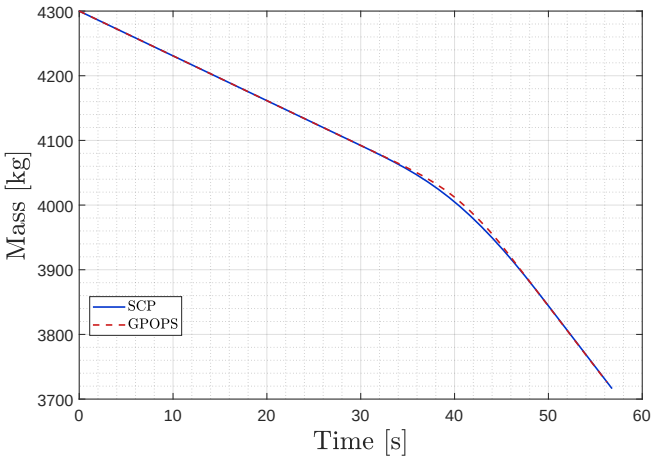
Integral weights		Slack weights		Other parameters	
w_T	1	w_η	10^{-1}	ϵ_η	10^{-2}
$w_{\dot{T}}$	2×10^{-1}	w_ν	10^{5}	ϵ_ν	10^{-9}
$w_{\dot{\beta}_y \dot{\beta}_z}$	5×10^{-1}	w_σ	10^{-2}	$N_{\max}$	30
$w_{\dot{\alpha}\dot{\beta}}$	6	$w_\int$	10	η_σ	0.5
w_τ^{prop}	10^{-2}				
w_τ^{aero}	10^{-4}				

Table 6.3: Integral values comparison

Term	GPOPS	SCP	Difference (%)
Total	3.35×10^{-2}	3.33×10^{-2}	-0.48
T	2.39×10^{-2}	2.41×10^{-2}	$+0.63$
$\dot{T}$	9.76×10^{-4}	5.95×10^{-4}	-39.05
TV̇C	2.64×10^{-3}	2.49×10^{-3}	-5.79
aėro	5.91×10^{-3}	6.14×10^{-3}	$+3.79$
Propulsive torque	2.7×10^{-5}	2.4×10^{-5}	-11.10
Aero torque	1.0×10^{-6}	1.0×10^{-6}	$+2.66$

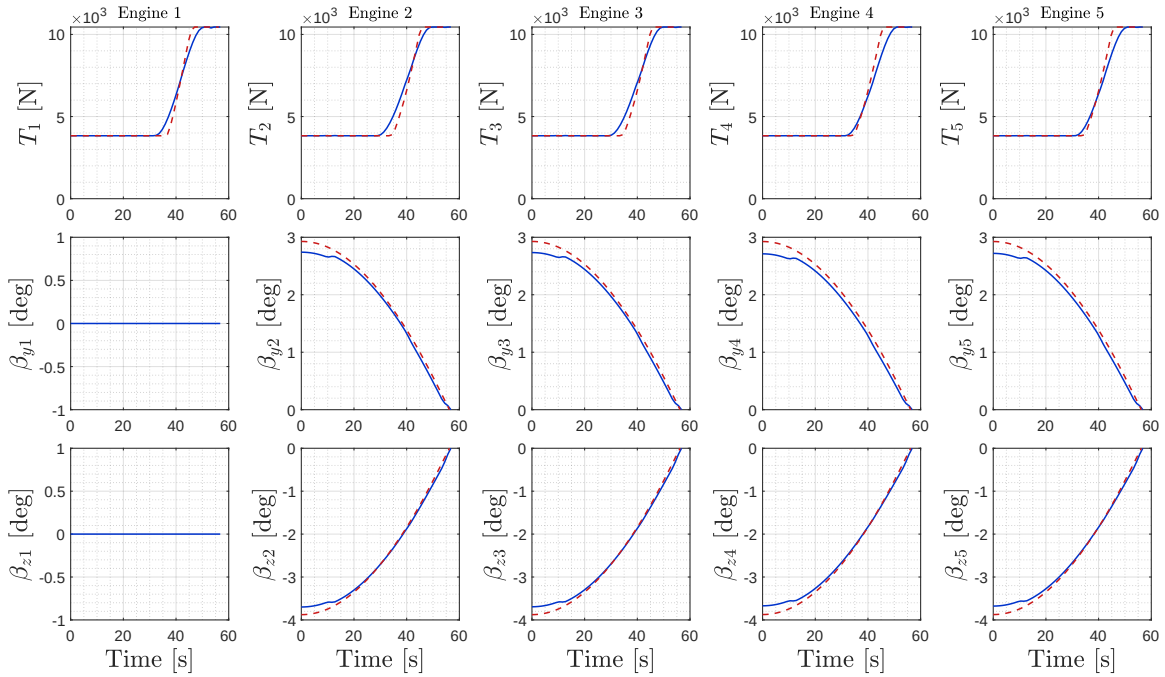


(a) Position and velocity.

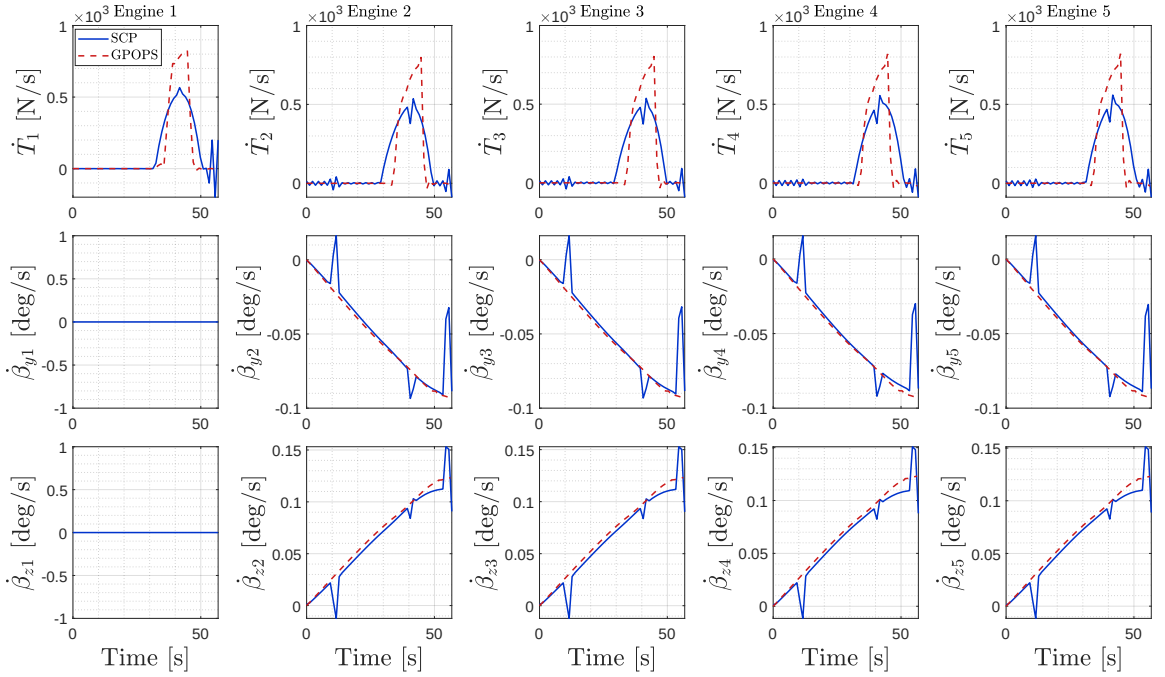


(b) Mass.

Figure 6.2: Comparison of position, velocity and mass states.

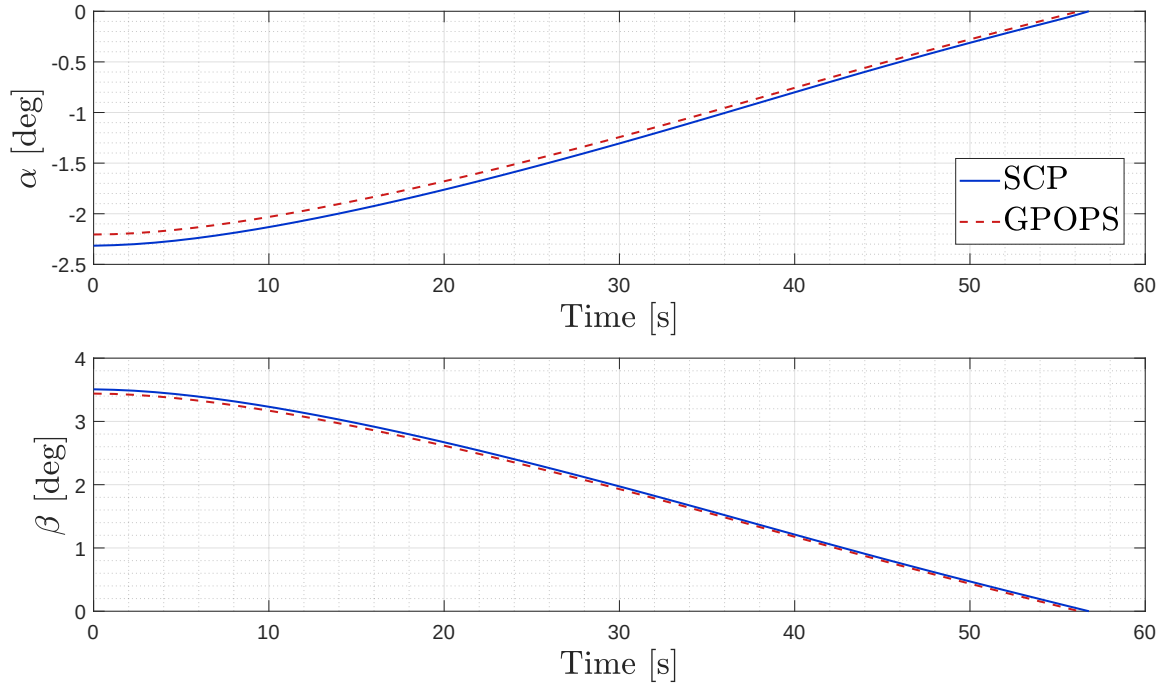


(a) Thrust and TVC.

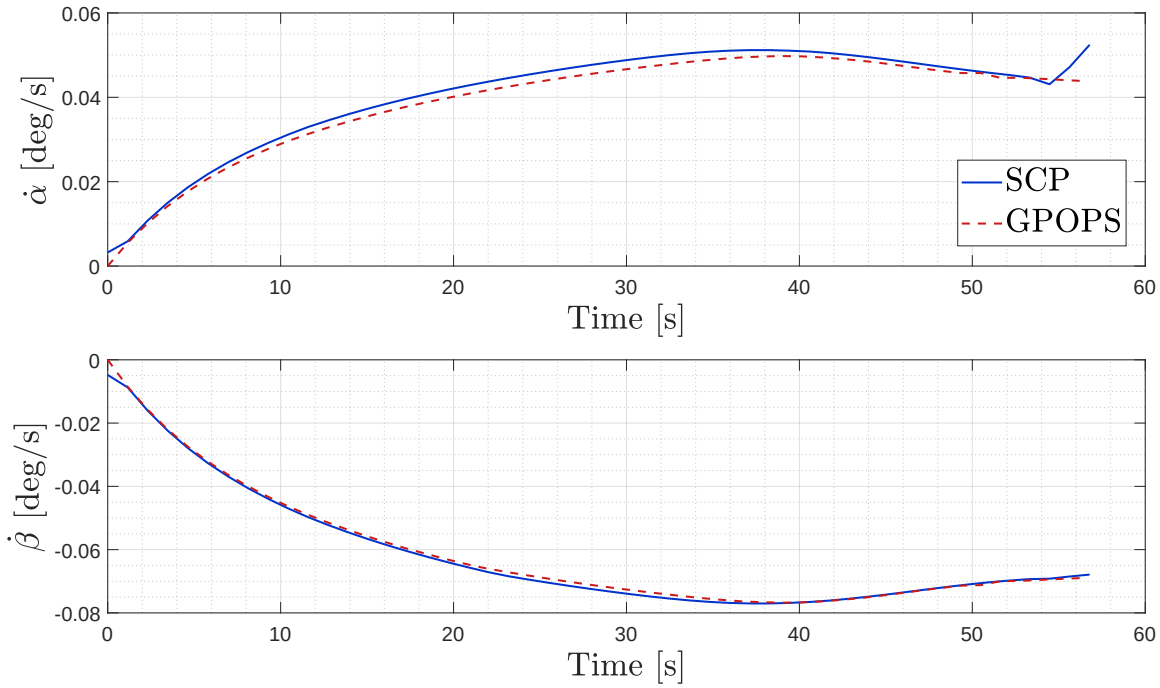


(b) Thrust and TVC rates.

Figure 6.3: Comparison of propulsive quantities: thrust magnitudes, TVC deflections, and rates.



(a) Aerodynamic angles.



(b) Aerodynamics rates.

Figure 6.4: Comparison of aerodynamic quantities: aerodynamic angles and rates.

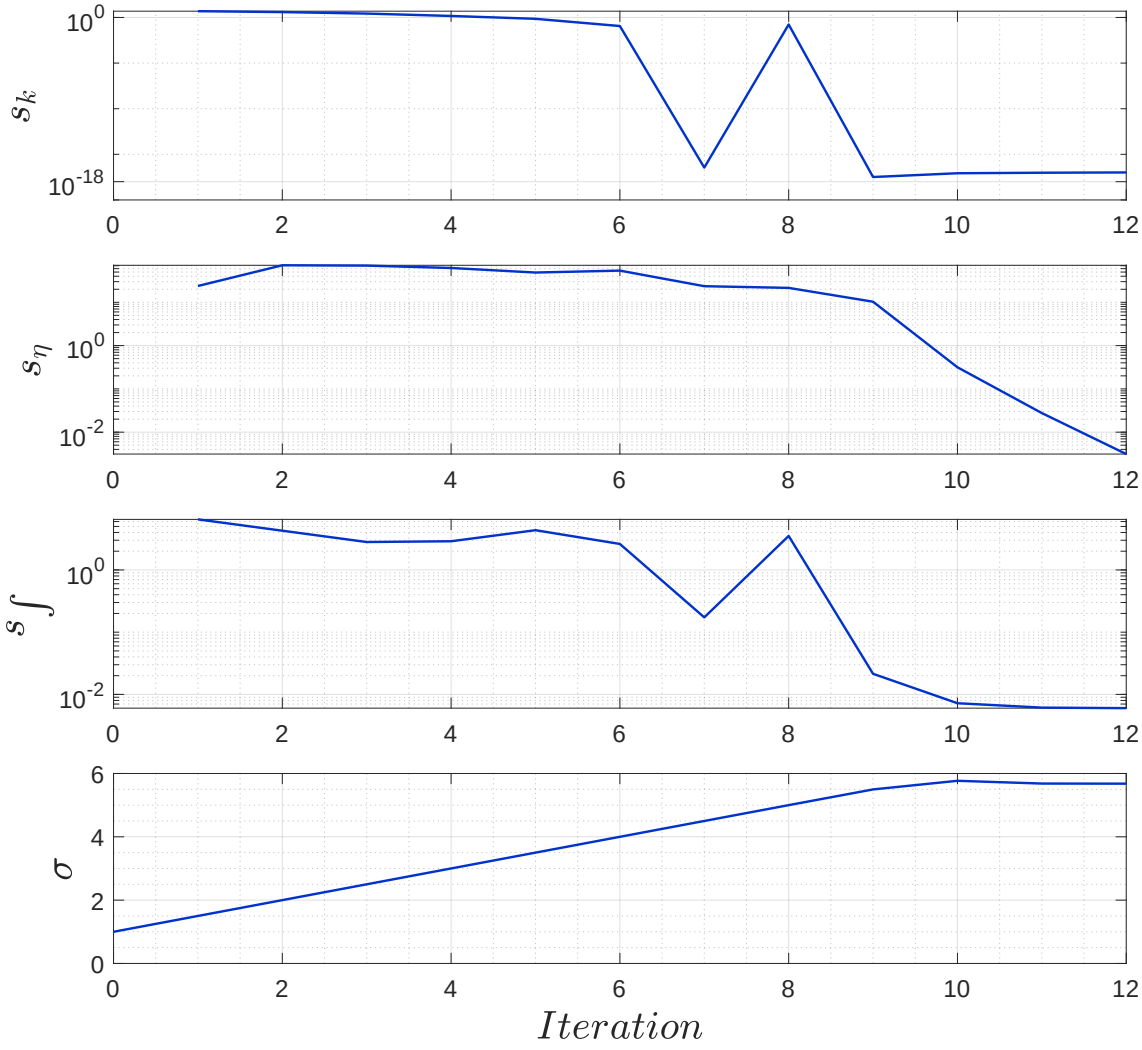


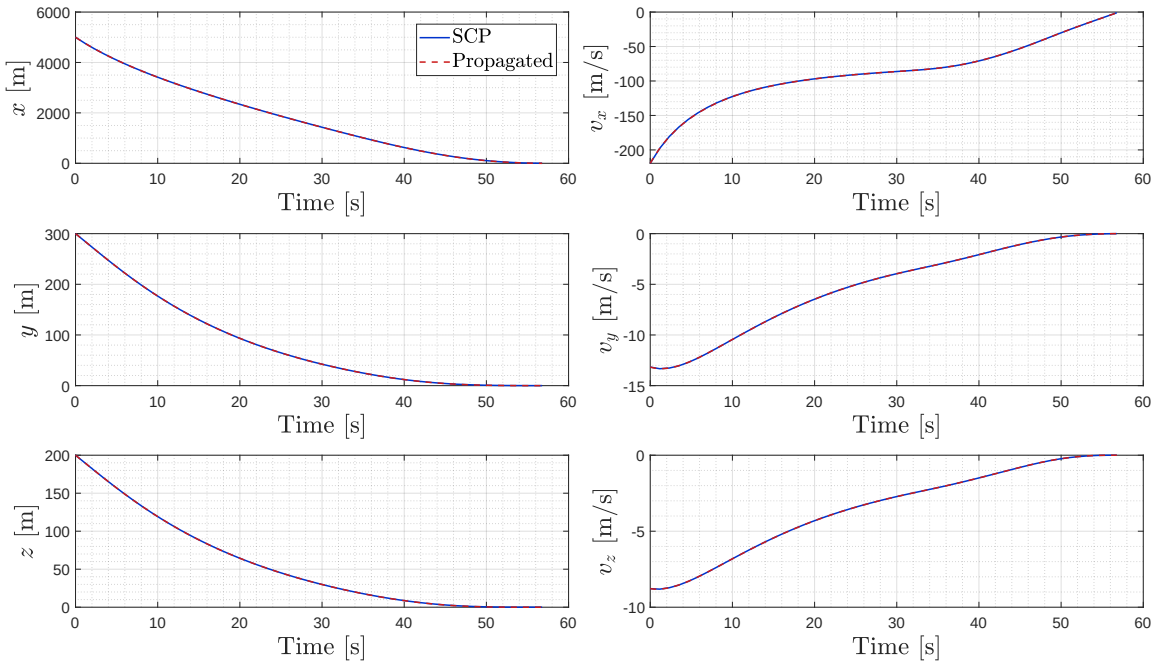
Figure 6.5: Evolution of the cost function components across SCP iterations.

6.2.6 Feasibility Check

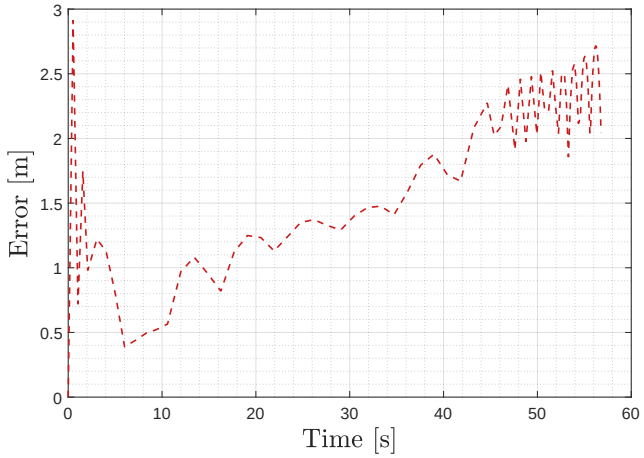
The same feasibility check described in Section 5.1.9 confirms that the optimized trajectory satisfies the full nonlinear dynamics, with errors on the order of 3m in position as shown in Figure 6.6a and 6.6b.

6.3 Fault-Tolerant Sequential Convex Programming

In the following, the Fault-Tolerant Sequential Convex Programming (FT-SCP) scheme is introduced. The objective is to extend the standard SCP formulation so that it can accommodate the degraded conditions resulting from a fault. The description begins with the definition of the fault cases, the information available at the time of the fault, followed by the modifications applied to the optimization problem in order to ensure feasibility and robustness under the new system constraints, ultimately allowing the computation of a trajectory consistent with the degraded system.



(a) Position and velocity propagation.



(b) Position error.

Figure 6.6: Validation of SCP solution: discrepancy between Runge-Kutta and SCP states.

6.3.1 Fault Scenarios

Based on the trends highlighted in [38], and similarly to [59], the failure cases considered in this work are listed in Table 6.4. Catastrophic failures are excluded, since both FTC and FTG would not be effective in such cases.

Propulsion failures are modeled as deviations from nominal thrust delivery by the propulsion system. Two distinct types are considered: total thrust failures (F1), where one engine shuts down and is unable to deliver any thrust, and partial thrust failures (F2), where one engine operates with degraded performance, being able to deliver only a certain percentage of its nominal maximum thrust. In addition, TVC actuator failures with jamming behavior are considered. These are represented as both TVC deflection angles being stuck at a fixed position (F3).

In [59], severe thrust loss scenarios are handled through an FTG approach, while all other failures, as well as their combinations, are managed by FTC strategies that integrate control allocation recovery, inner-loop TVC control recovery, and FTC recovery. In the latter case, the nominal TVC controller is replaced by an FTC-based controller designed to be robust against propulsion failures. In the present work, the emphasis is placed on the guidance aspect. Consequently, all failure modes are handled exclusively through the recovery guidance action.

For the purposes of this study, it is assumed that all types of failures are detected, isolated, and assessed instantly. In other words, FDI is considered ideal, with the fault type and its location identified without delay. Furthermore, guidance reconfiguration is assumed to occur immediately after FDI, and the recovery guidance is considered capable of delivering a solution with no time delay.

Table 6.4: Failure Cases

Name	Identifier
Total engine loss	F1
Thrust degradation	F2
TVC jamming	F3

To inform the guidance algorithm, the relevant information is passed at the time of the fault. This is achieved through the *Fault Information Bus*, which collects the parameters required for the problem reconfiguration and is summarized in Table 6.5.

Table 6.5: Information contained in the fault information bus.

Parameter	Units	Description
Fault time	s	Time of fault occurrence
Thrust motor identifier	–	Index of faulty engine (1–5)
TVC motor identifier	–	Index of faulty actuator (1–4)
Thrust status	–	0 = Nominal, 1 = F2, 2 = F1
TVC status	–	0 = Nominal, 1 = F3
Thrust degradation	%	Value between $\left(\frac{T_{\min,\text{nom}}}{T_{\max,\text{nom}}}, 0.95\right)$ of nominal thrust
TVC stuck angles	deg	Fixed deflections $[\beta_y^{jam}, \beta_z^{jam}]$

6.3.2 Final Position Relaxation

Similarly to the approach in [66], the standard SCP formulation is modified in the treatment of the horizontal components of the final position. Instead of enforcing them as hard constraints, they are relaxed into soft constraints of the form

$$|y(t_f)| \leq s_{\mathrm{y}} \tag{6.58}$$

$$|z(t_f)| \leq s_{\mathrm{z}} \tag{6.59}$$

with the corresponding slack variables added to the cost function multiplied by a weight:

$$J = w_{pos}(s_{\mathrm{y}} + s_{\mathrm{z}}) + w_\eta s_\eta + w_\kappa s_\kappa + w_\sigma \sigma + w_\int s_\int \tag{6.60}$$

This modification maximizes feasibility at the expense of allowing a potential error in the final horizontal landing position. The relaxation is motivated by the fact that, if the initial state error is larger than expected and resources are degraded due to the fault, it may be preferable to tolerate a bounded horizontal error rather than risk non-convergence. For this reason, the glideslope constraint is modified accordingly such that the origin of the cone shifts with the final lateral position.

$$\left\| \begin{bmatrix} y - y(t_f) & z - z(t_f) \end{bmatrix}^\top \right\|_2 \leq \frac{x}{\tan \gamma_{gs}} \tag{6.61}$$

6.3.3 Box Constraints & Final Boundary Condition Update

A complete thrust loss in one engine (F1) is handled by setting the minimum and maximum thrust of the affected engine to zero. In addition, the corresponding thrust rate, TVC deflection, and TVC rate constraints are also set to zero.

$$T^i_{\min} = T^i_{\max} = 0, \quad \dot{T}^i_{\min} = \dot{T}^i_{\max} = 0 \tag{6.62}$$

$$\beta^{\min}_{y,i} = \beta^{\max}_{y,i} = 0, \quad \beta^{\min}_{z,i} = \beta^{\max}_{z,i} = 0 \tag{6.63}$$

$$\dot{\beta}^{\min}_{y,i} = \dot{\beta}^{\max}_{y,i} = 0, \quad \dot{\beta}^{\min}_{z,i} = \dot{\beta}^{\max}_{z,i} = 0 \tag{6.64}$$

A thrust degradation failure (F2) is addressed by scaling the box constraint on the maximum thrust of the affected engine according to the degradation factor ξ_i. For instance, if engine 3 experiences a partial thrust fault with 50% degradation, then its maximum thrust is limited to 50% of the nominal value.

$$T^i_{\max} = \xi_i \, T^i_{\max,\mathrm{nom}}, \qquad \frac{T^i_{\min,\mathrm{nom}}}{T^i_{\max,\mathrm{nom}}} < \xi_i < 1 \tag{6.65}$$

In the event of a TVC actuator jamming fault (F3),with jammed values $\beta^i_{\mathrm{jam},y}$ and $\beta^i_{\mathrm{jam},z}$, the corresponding deflection angles are fixed at the values reported in the fault information bus, and the box constraints on the affected TVC angles and rates are updated accordingly to reflect the jammed condition.

$$\beta^{\min}_{y,i} = \beta^{\max}_{y,i} = \beta^{\mathrm{jam}}_{y,i}, \qquad \beta^{\min}_{z,i} = \beta^{\max}_{z,i} = \beta^{\mathrm{jam}}_{z,i} \tag{6.66}$$

$$\dot{\beta}^{\min}_{y,i} = \dot{\beta}^{\max}_{y,i} = 0, \qquad \dot{\beta}^{\min}_{z,i} = \dot{\beta}^{\max}_{z,i} = 0 \tag{6.67}$$

As a consequence, the final boundary condition on the TVC angles specified in Table 5.2 is removed, allowing the healthy actuators to land with nonzero deflections to compensate for the jammed actuator even at touchdown, as illustrated in Figure 6.7.

6.3.4 Initialization

At t_{fault}, the FT-SCP receives as input the current augmented state, which includes the vehicle position, velocity, and mass, together with the control inputs applied by the controller. These quantities are scaled according to Table 5.3 and are then used as initial boundary conditions (IBCs) for the SCP problem. To better reflect the actual system behavior, the formulation enforces not only the state of the rocket at the time of the fault but also constrains the augmented state variables to coincide with the commands being applied by the controller. This prevents the guidance solution from returning control profiles that differ significantly from the current commanded values.

In order to accelerate convergence, the standard line-initialization approach is replaced with a scheme that exploits the nominal trajectory. Specifically, the algorithm is initialized using slices of the nominal state and control trajectories from t_{fault} to the final time, modified to reflect the specific fault condition. The final time guess is also initialized with the corresponding nominal value.

6.3.5 Performance Under Fault Scenarios

The performance of FT-SCP is evaluated by comparing its solutions under the different fault scenarios with the nominal solution from Chapter 5. In this case, the objective is not to reproduce the nominal trajectory, but to analyze how the state and control profiles evolve under the imposed fault conditions.

F1 – Total Thrust Loss

At $t_{\text{fault}} = 15$ s, engine 4 shuts down completely. To compensate for the lost engine, the remaining healthy engines ignite approximately 10 s earlier than in the nominal case, reach maximum thrust, and sustain it for a longer duration, as shown in Figure 6.9a, which results in an additional propellant consumption of about 200 kg (Figure 6.8b). The resulting position and velocity profiles (Figure 6.8a) demonstrate that, despite the degraded propulsion system, the vehicle achieves a soft landing at the pad center, with vertical touchdown velocity of -1 m/s and negligible lateral velocity components.

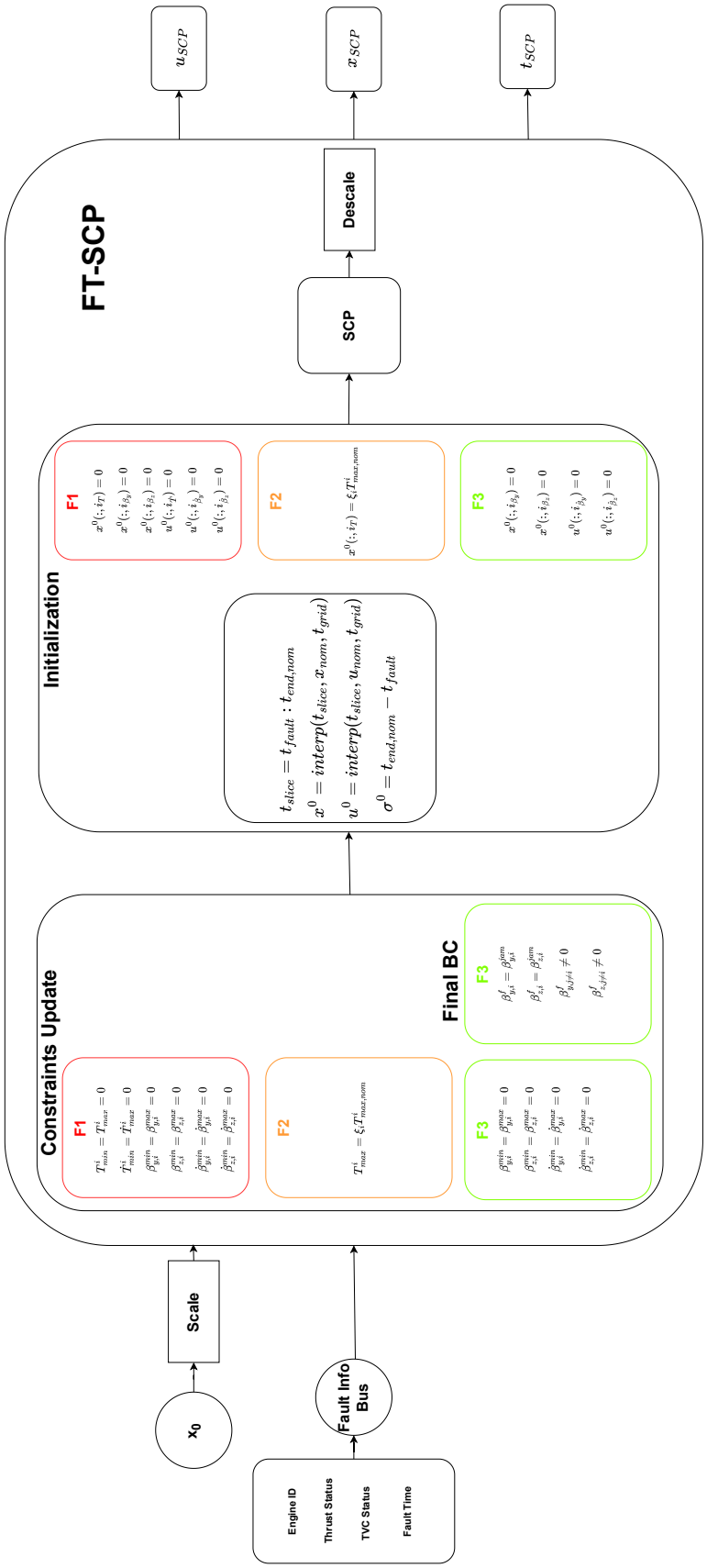
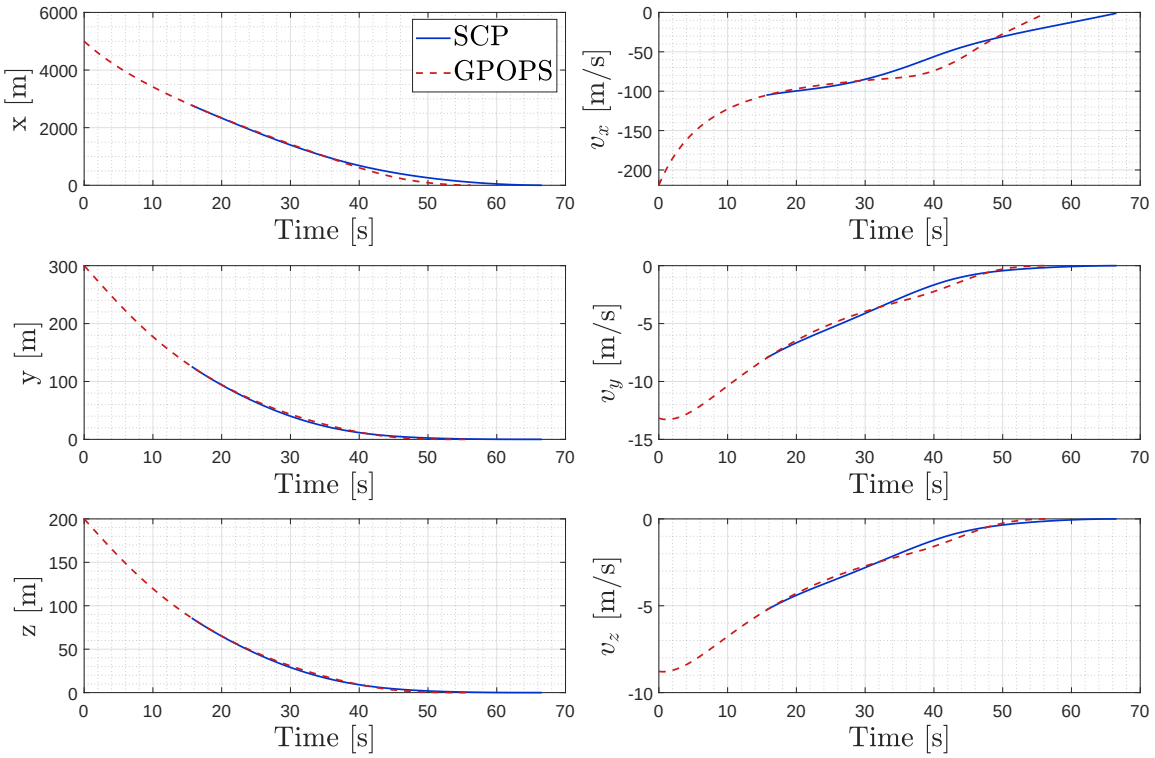
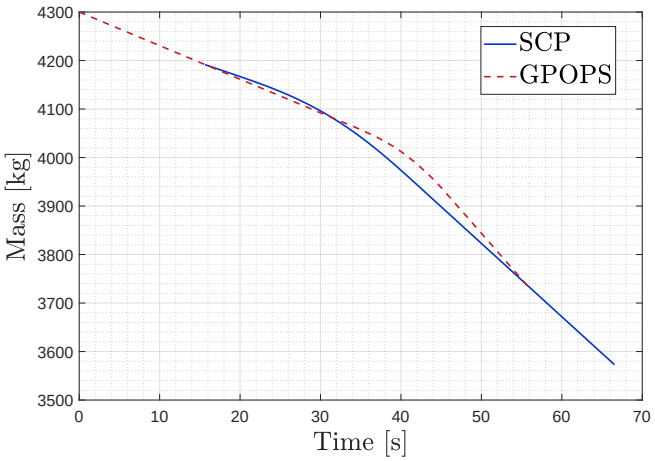


Figure 6.7: FT-SCP algorithm: integration of fault information.

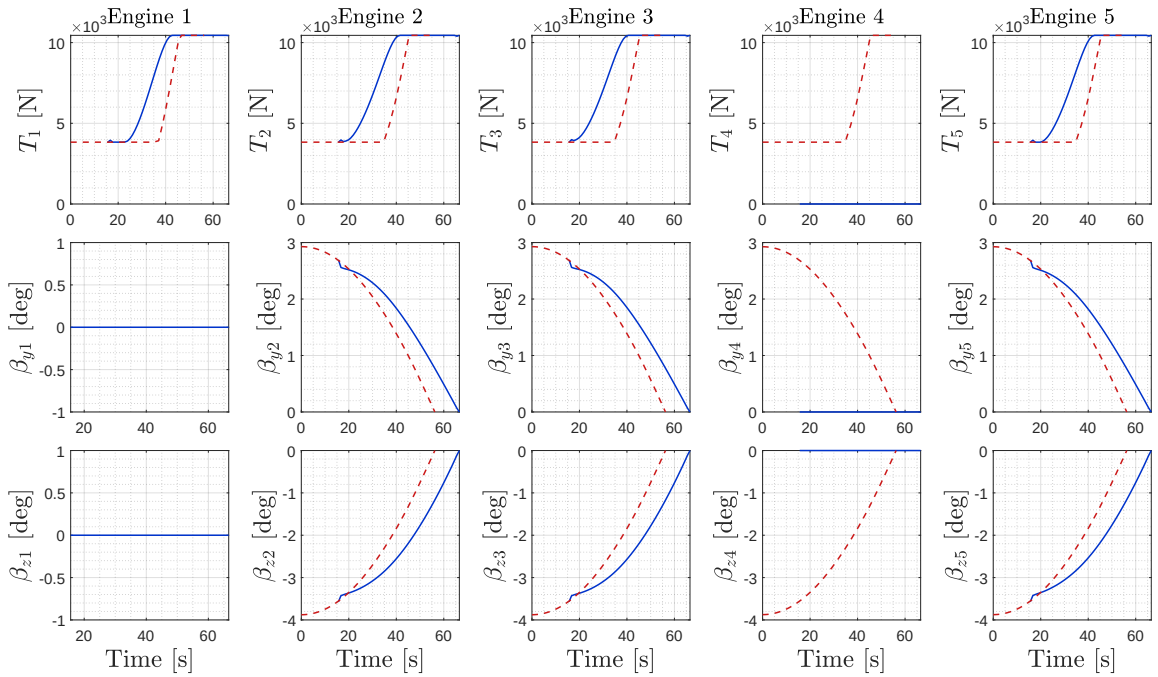


(a) Position and velocity.

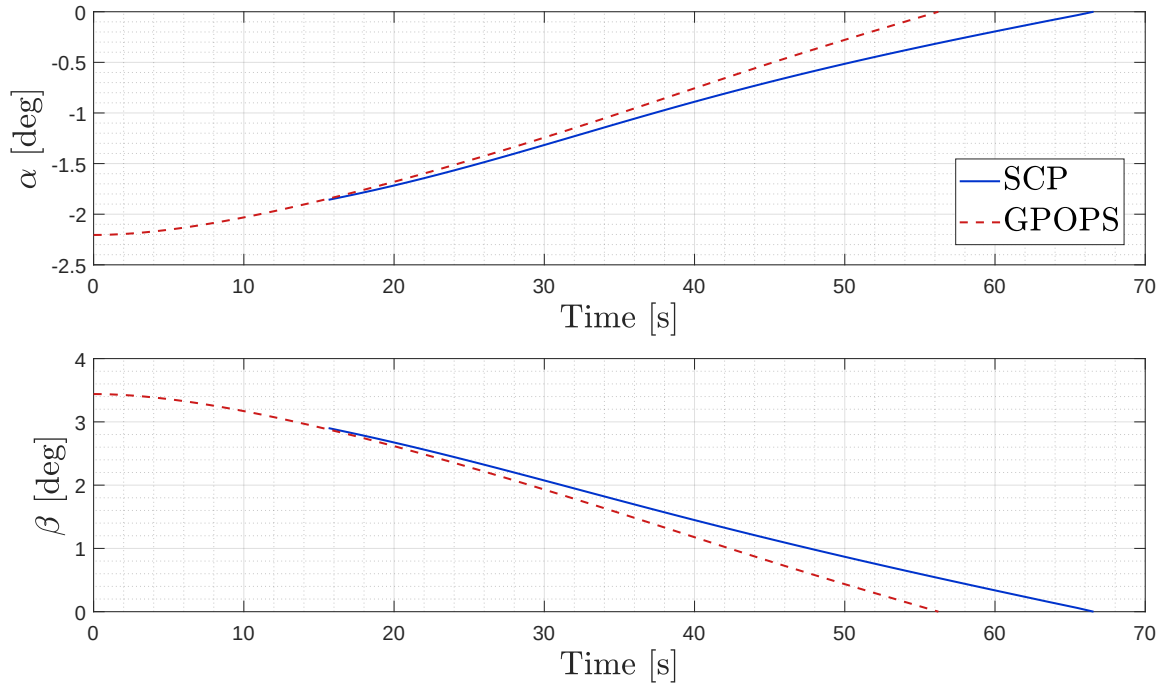


(b) Mass.

Figure 6.8: FT-SCP solution under F1 (total thrust loss): position, velocity, and mass evolution.



(a) Propulsive controls.



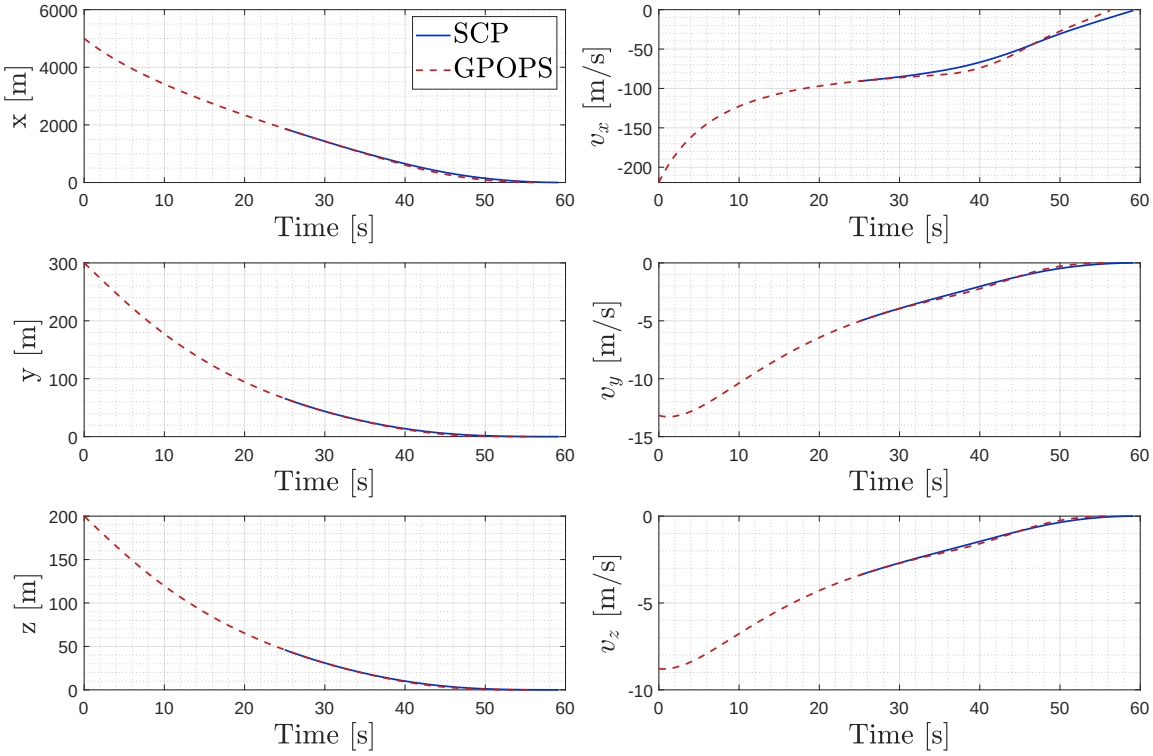
(b) Aerodynamic controls.

Figure 6.9: FT-SCP solution under F1 (total thrust loss): controls evolution.

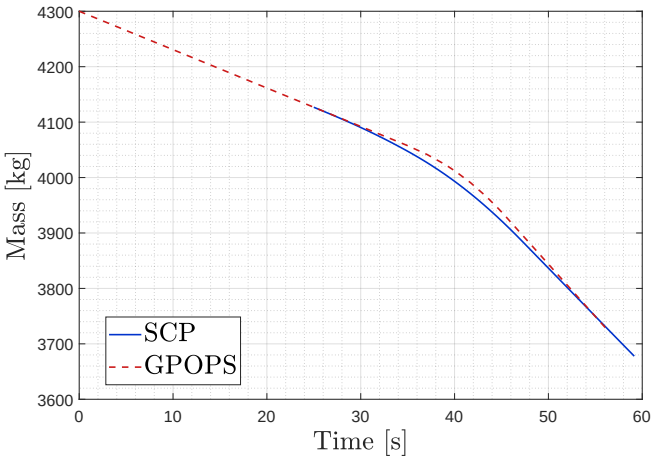
F2 – Partial Thrust Loss

A thrust degradation fault with degradation factor $\xi_1 = 60\%$ occurs in engine 1 at $t_{\text{fault}} = 25$ s. The healthy engines respond by transitioning to the maximum thrust immediately at the fault time, while the degraded engine approximately 10 s later, reach maximum thrust, and sustain it for a longer dura-

tion, as shown in Figure 6.11a, which results in an additional propellant consumption of about 30 kg (Figure 6.10b). The resulting position and velocity profiles are shown in Figure 6.10a.

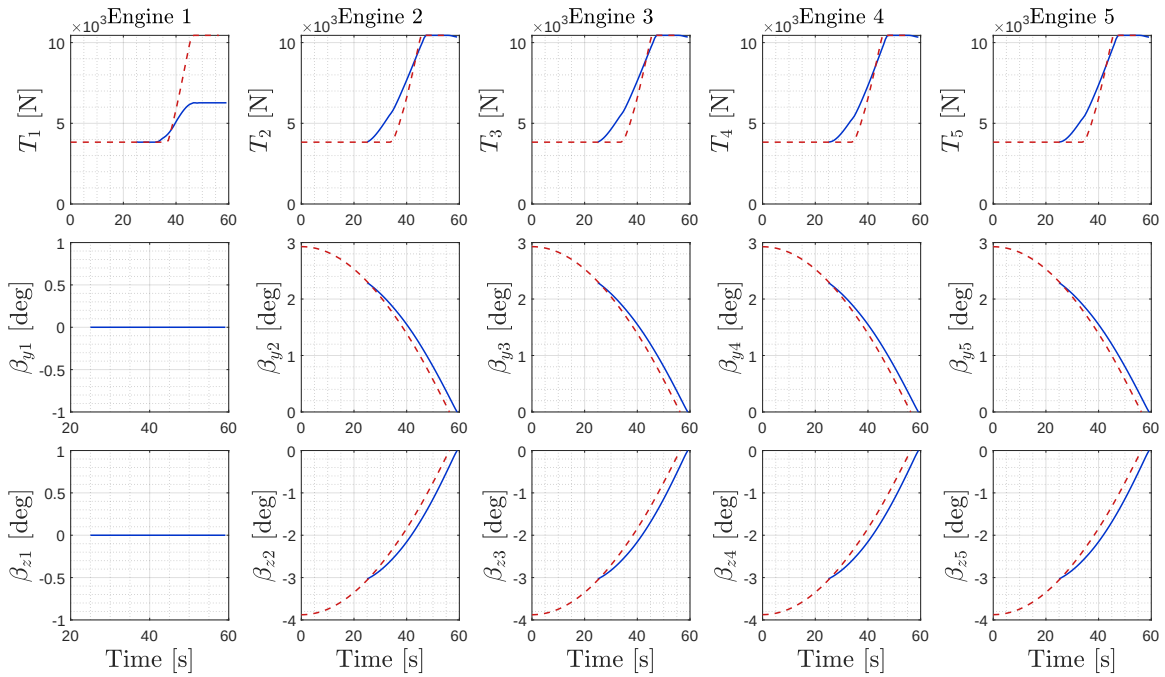


(a) Position and velocity.

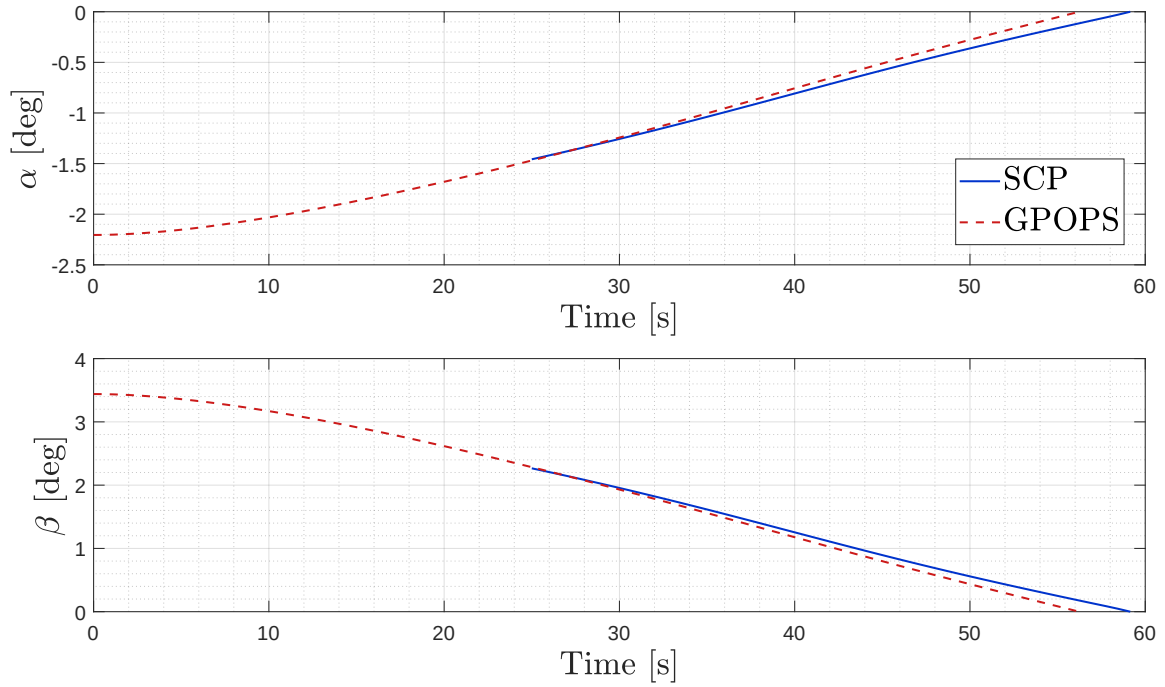


(b) Mass.

Figure 6.10: FT-SCP solution under F2 (thrust degradation): position, velocity, and mass evolution.



(a) Propulsive controls.



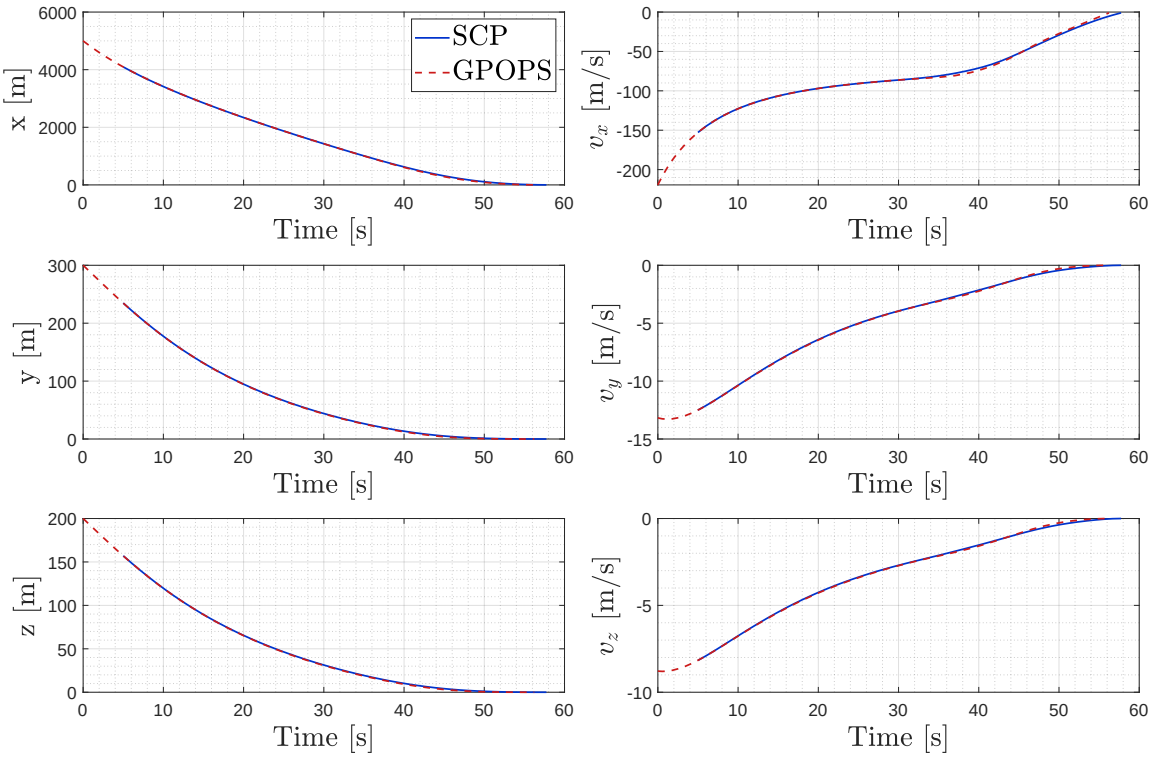
(b) Aerodynamic controls.

Figure 6.11: FT-SCP solution under F2 (thrust degradation): controls evolution.

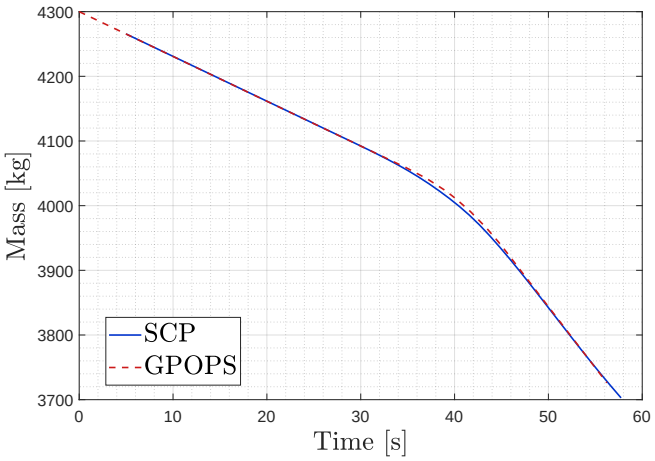
F3 – TVC Jamming

This case considers a fault occurring at $t_{\text{fault}} = 5$ s, where the TVC angles of engine 5 become stuck at 2.9° and −3.8° in the y and z axes, respectively. As a result, the thrust behavior of the remaining healthy engines remains essentially unchanged, leading to a final mass very close to the nominal case

(Figure 6.12b). The TVC angles of the healthy engines adapt accordingly: the β_y angles take slightly smaller values to counteract the positive $\beta_{y,5}$, while the β_z angles increase slightly to offset the negative $\beta_{z,5}$. In contrast, the thrust profile of the faulty engine deviates significantly from the nominal one. Its thrust never reaches or maintains the maximum value and gradually decreases, returning to the minimum at touchdown (Figure 6.13a).

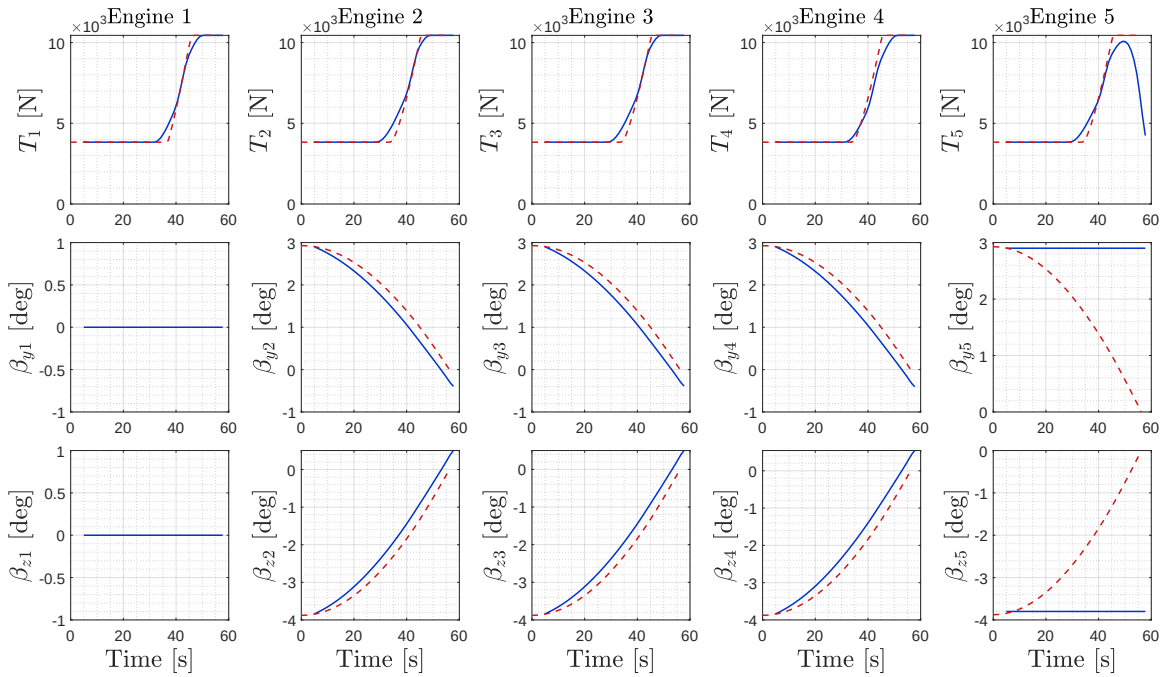


(a) Position and velocity.

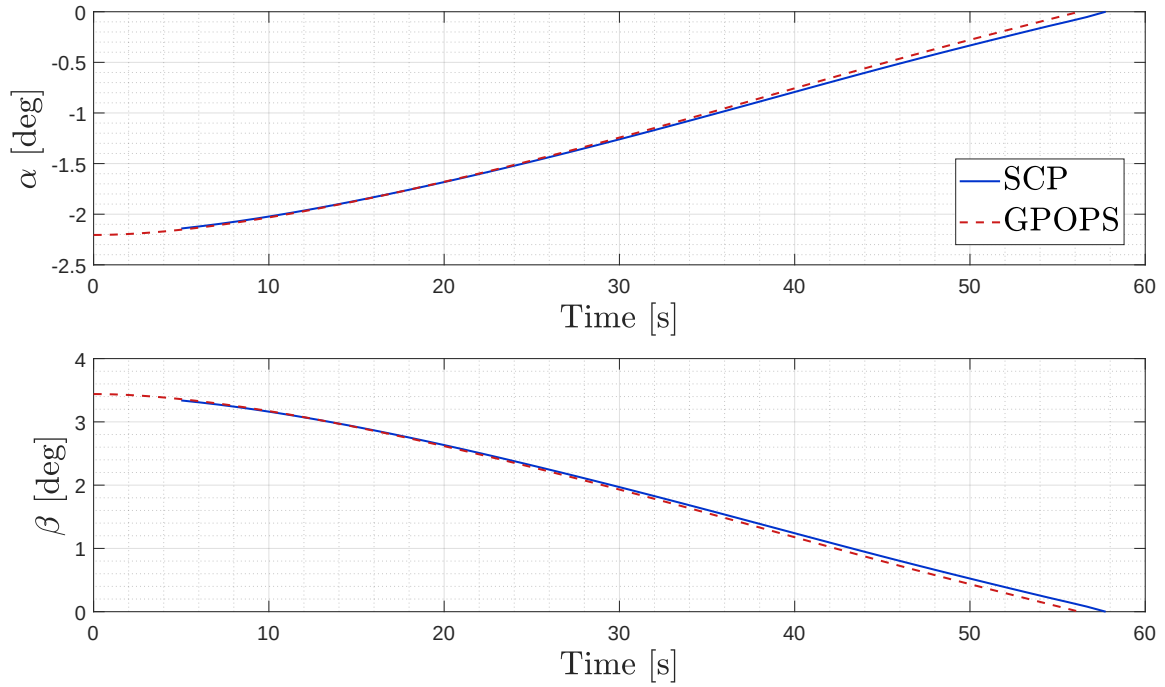


(b) Mass.

Figure 6.12: FT-SCP solution under F3 (TVC jamming): position, velocity, and mass evolution.



(a) Propulsive controls.



(b) Aerodynamic controls.

Figure 6.13: FT-SCP solution under F3 (TVC jamming): controls evolution.

7 Verification & Validation

The performance of the FT-SCP algorithm is evaluated in a closed-loop simulation environment. The plant model corresponds to the one described in Chapter 4, while the nominal controller and reference trajectory are those presented in Chapter 5. Uncertainty in aerodynamic parameters, atmospheric conditions, and initial states is introduced according to the dispersions listed in Table 5.4.

7.1 Implementation Details

The closed-loop simulations are implemented in *Simulink* and the FT-SCP algorithm is realized as a MATLAB [89] system block inside a triggered subsystem, with the trigger defined by the fault time. When the simulation time reaches t_{fault}, the subsystem is activated and the FT-SCP is executed. Problem 1 is assembled with *CVX*, a package for specifying and solving convex programs [24, 35] and the resulting SOCP is solved using the convex optimization solver *MOSEK* [8].

The plant dynamics evolve in continuous time, while guidance and control operate in discrete time at a frequency of 50 Hz. Each simulation trial begins with sampled aerodynamic uncertainties, initial dispersions, and fault. At $t = t_{\text{fault}}$, the plant model is updated to reflect the appropriate faulty dynamics. The FT-SCP is then triggered through the corresponding subsystem and computes a new reference trajectory, once it returns a feasible solution, the nominal reference trajectory used by the LQR controllers is switched to the FT-SCP generated one. Both the state and control trajectories are stored in lookup tables scheduled with respect to time. The scheduling parameter is reset to zero at $t = t_{\text{fault}}$ and synchronized with the time vector produced by FT-SCP.

7.2 Study 1: Single Faults

7.2.1 Goal

The objective of this study is to evaluate the impact of individual faults on mission success. Each fault type (F1, F2, F3) is analyzed separately in order to:

- Assess the robustness of the FT-SCP algorithm against initial dispersions.
- Gain insight into how specific fault modes influence the mission outcome.
- Investigate the effect of a realistic guidance computation delay on landing performance.

7.2.2 Setup

In each campaign, the fault time t_{fault} is varied as a percentage of the reference trajectory duration. For every fault type, three Monte Carlo campaigns are performed under the same aerodynamic uncertainties and initial condition dispersions:

Nominal: The FT-SCP is not executed; the closed-loop simulation runs with the constant nominal reference trajectory and controller.

Recovery: The FT-SCP is triggered at t_{fault}; once the guidance has converged, the reference trajectory is immediately switched to the recovery solution.

Recovery Delay: The FT-SCP is triggered at t_{fault}; once the guidance has converged, the reference trajectory is switched to the recovery solution only after a simulated delay of t_{delay}.

A mission is considered successful if it results in a soft pinpoint landing, defined by the requirements listed in Table 5.5.
The randomized parameters for each fault type are summarized in 7.1.

Table 7.1: Randomized parameters for single-fault cases F1–F3

Parameter	F1	F2	F3
Fault time	$\mathcal{U}(0,\, 80\%\, t_{\text{ref}})$	$\mathcal{U}(0,\, 80\%\, t_{\text{ref}})$	$\mathcal{U}(0,\, 80\%\, t_{\text{ref}})$
Engine number	$\mathcal{U}(1,\, 5)$	$\mathcal{U}(1,\, 5)$	$\mathcal{U}(2,\, 5)$
Fault-specific parameter	–	$\mathcal{U}\big(\frac{T_{\min,\text{nom}}}{T_{\max,\text{nom}}},\, 0.95\big)$	$\big[\mathcal{U}(-5°,\, 5°), \mathcal{U}(-5°,\, 5°)\big]$

F1: One Engine Loss

Figure 7.1 summarizes the results for fault type F1. As expected, no nominal cases achieve a successful landing, since tracking the original reference trajectory becomes impossible with only four engines. The nominal LQR controller therefore fails, as it continues attempting to follow a trajectory that cannot be realized after an engine loss. The FT-SCP guidance converges in approximately 78% of the runs, i.e. a feasible solution is found. However, only 65% of the closed-loop simulations satisfies the soft landing requirements. This discrepancy can be attributed to the limited authority of the nominal LQR controller, which is unaware of the engine failure. As a result, the controller continues issuing commands in terms of thrust increments and TVC deflections to the inoperative engine, and this, in combination with large aerodynamic uncertainties, leads to a fraction of feasible guidance solutions that cannot be successfully tracked, resulting in missed landings. The effect of guidance delay is relatively minor, reducing the overall success rate by only about 1.5%. Figures 7.2a and 7.2b show the 3D position trajectories of converged cases. The missed landings in the recovery cases can already be seen here, and even more clearly in Figure 7.3a, which shows the lateral positions at landing color-coded by touchdown vertical velocity. This highlights once again the difference with the guidance solutions in Figure 7.3b, where the relaxation of the final lateral position constraints introduced in (6.58) and (6.59) is in practice not exploited, as all converged FT-SCP solutions remain very close to the landing pad center. It can be observed from Figure 7.4, which shows the final mass at landing for all converged cases, that the average final mass is

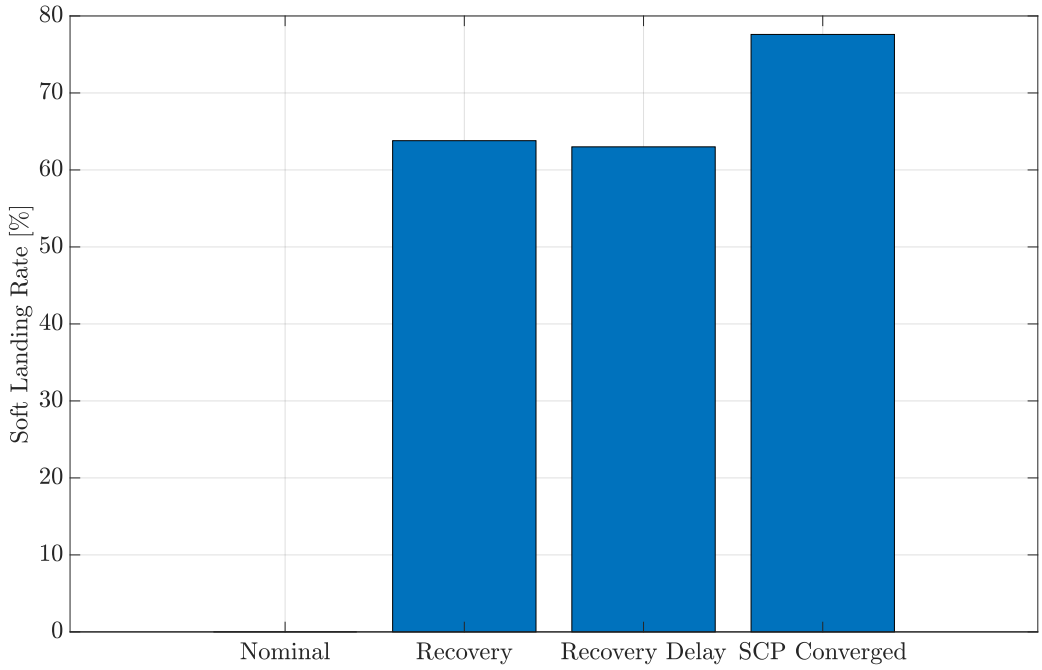
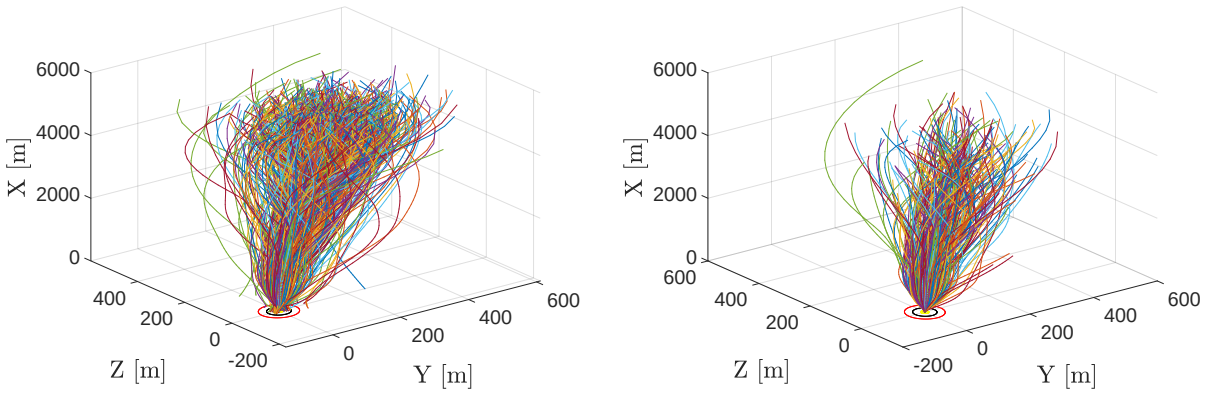


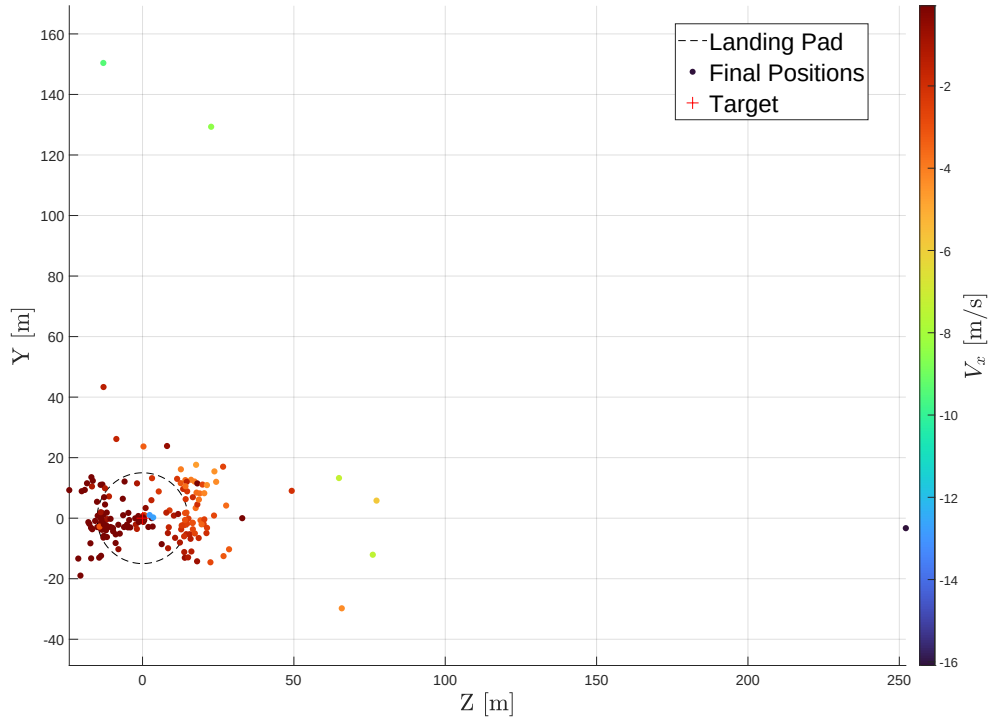
Figure 7.1: F1 (total thrust loss): Soft landing success rates for nominal, recovery, recovery-delay, and converged FT-SCP cases.



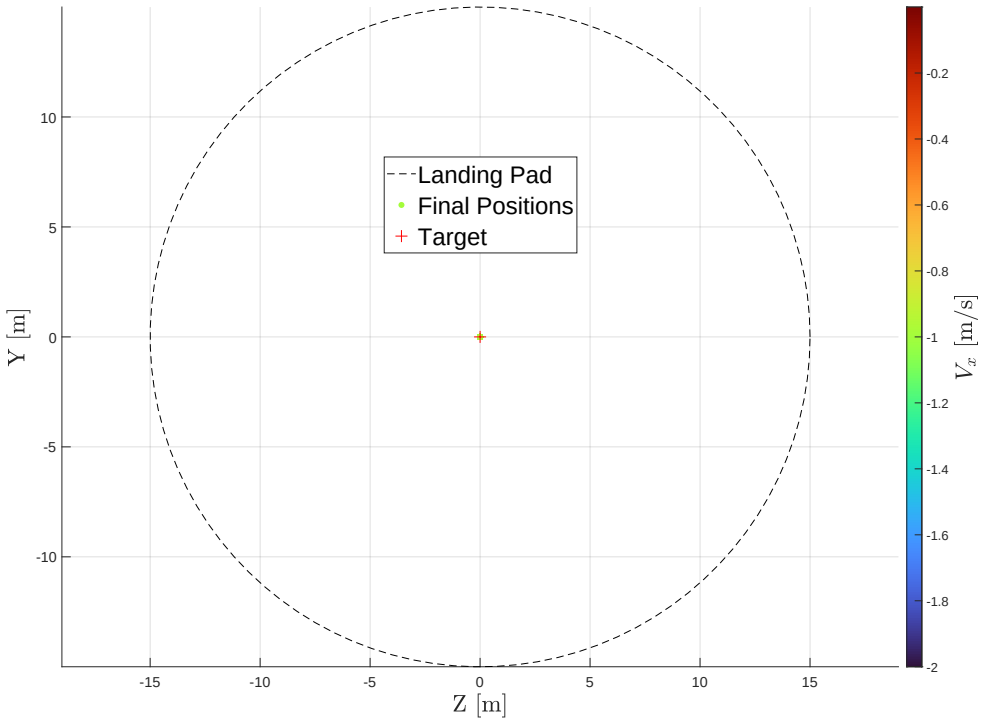
(a) Closed-loop recovery trajectories.

(b) FT-SCP reference solutions.

Figure 7.2: F1 (total thrust loss): 3D landing trajectories of all converged cases.



(a) Closed-loop recovery cases.



(b) FT-SCP solutions.

Figure 7.3: F1 (total thrust loss): Lateral touchdown positions at the landing pad color-coded by vertical touchdown velocity.

about 200 kg lower than in the nominal case. Nevertheless, sufficient propellant remains at touchdown, since the dry mass of the vehicle is 2750 kg.

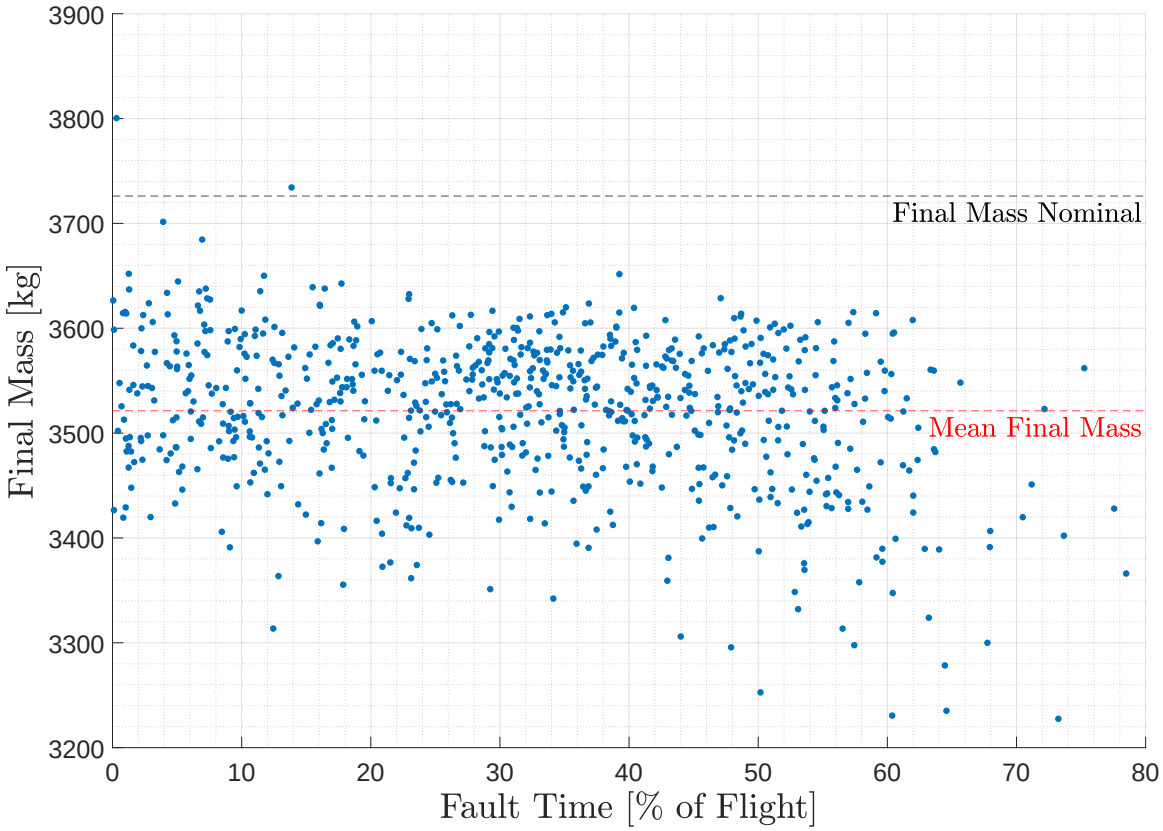


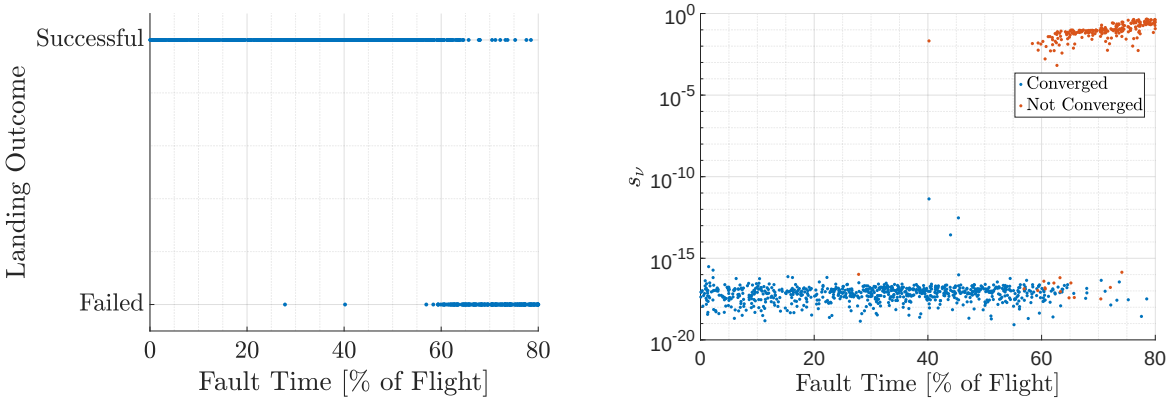
Figure 7.4: F1 (total thrust loss): Final mass at landing versus fault time for all converged cases.

Figure 7.5a highlights a clear dependency between guidance convergence and fault time: beyond 60% of the nominal flight duration, the algorithm fails to converge in most cases. This behavior does not indicate a weakness of the algorithm but reflects the physical infeasibility of the problem itself. As shown in Figure 6.9a, the total loss of one engine is compensated by firing the remaining healthy engines in advance (when compared to the nominal case) and for a longer duration. By the time 60% of the trajectory has elapsed, however, the nominal profile has already commanded the engines to ramp from minimum to maximum thrust (Figure 5.2a). Given the vehicle's altitude and vertical velocity at this stage, even firing all four remaining engines at full thrust cannot generate sufficient deceleration to ensure a safe landing. The lack of convergence therefore stems from the absence of a feasible solution rather than numerical difficulties. This interpretation is further supported by the behavior of the virtual control slack variables shown in Figure 7.5b. For the non-converged cases, the slacks remain large at the final iteration, preventing satisfaction of the convergence criterion (6.56). In other words, the discretized dynamics in (6.39) cannot be enforced without substantial artificial inputs, confirming that the underlying problem is infeasible.

It is important to note that the orange (non-converged) dots with values well below 0, similar to those of the blue converged cases, can be misleading. This does not mean that the algorithm failed to converge despite a feasible solution.

One must recall that in order to converge, the algorithm also includes the condition on the trust-region

difference between subsequent iterations (6.57). If this difference is not sufficiently small, the algorithm is considered to have not converged. This is precisely the case here: although a trajectory with negligible virtual controls is found, it is obtained through a large deviation in states and controls from the previous iteration rather than through a small step within the linearized region. For this reason, the solution is still considered infeasible and therefore the algorithm does not reach convergence.

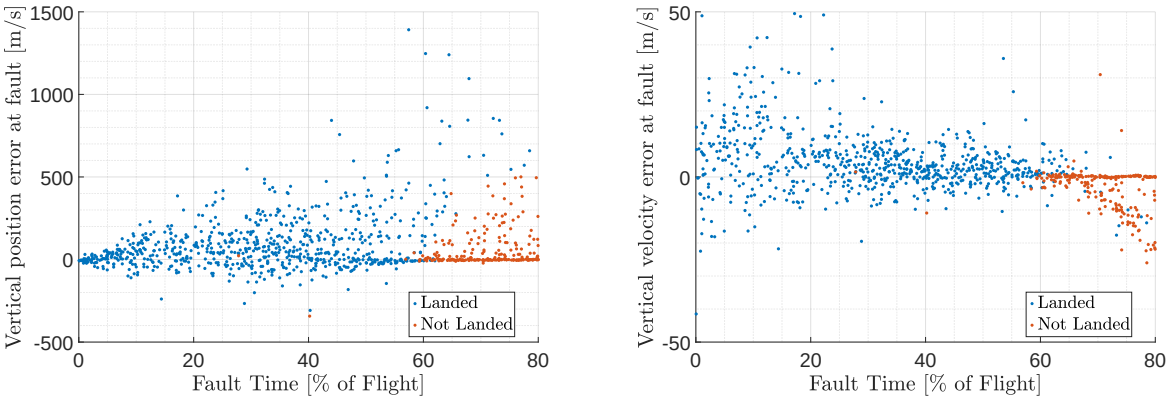


(a) Convergence versus fault time.

(b) Virtual controls slacks.

Figure 7.5: F1 (total thrust loss): FT-SCP convergence analysis.

All cases with a fault occurring at $t_{\text{fault}} \gtrsim 60\% \, t_{\text{ref}}$ for which the guidance converges can be explained by looking at Figure 7.6, which shows the state error at the fault time, i.e. the deviation between the simulation state (the FT-SCP initial condition) and the nominal reference trajectory. Figure 7.6a illustrates this error in terms of vertical position, while Figure 7.6b shows the corresponding vertical velocity error. From Figure 7.6a, it is clear that all converged cases share a common feature: the vertical position error is significant, on the order of several hundred meters. At the same time, Figure 7.6b indicates that these cases exhibit higher downward velocity, though not excessively large. In practice, this means that the vehicle is descending faster but still retains additional altitude margin compared to the nominal trajectory. This combination provides more space and time for the remaining healthy engines to be fired in order to decelerate, which explains why the algorithm converges in these scenarios.



(a) Vertical position error.

(b) Vertical velocity error.

Figure 7.6: F1 (total thrust loss): State errors at fault time for converged and non-converged cases.

F2: Degraded thrust

Figure 7.7 summarizes the results for fault type F2. In this case some nominal cases are still able to land, mainly those with a mild thrust degradation factor that can be compensated by the controller. Figure 7.8a

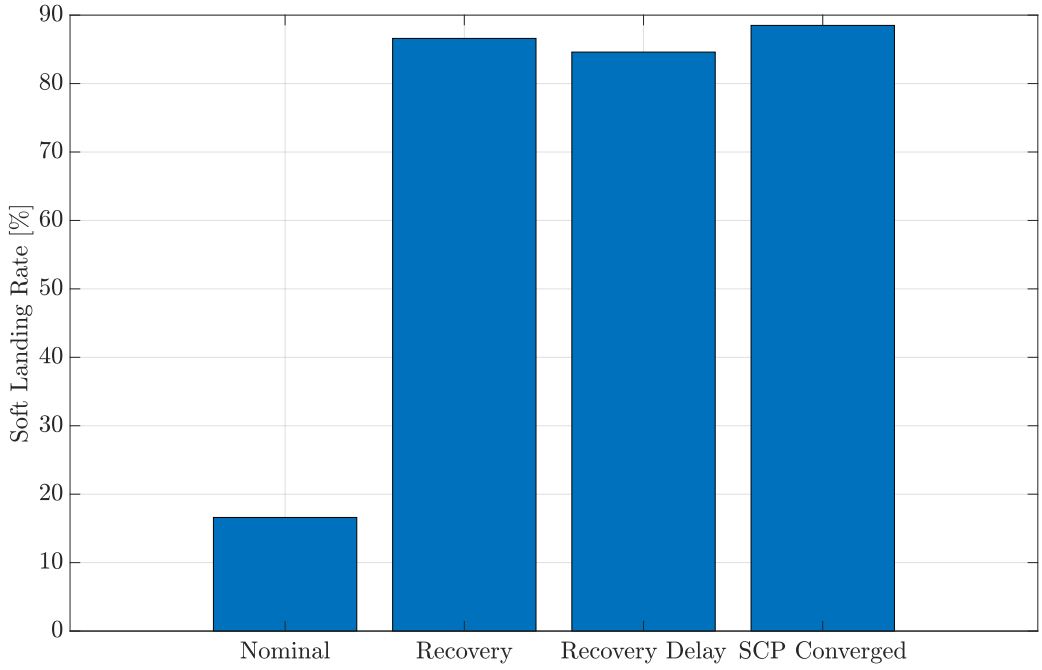
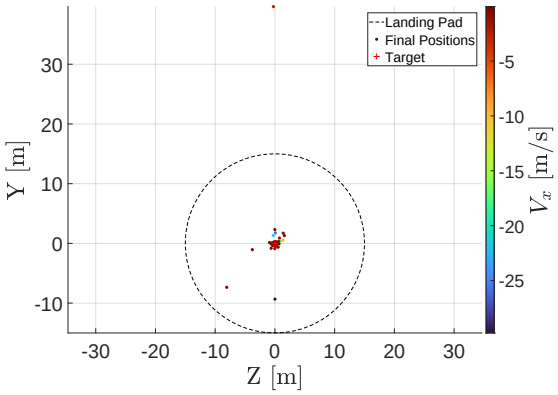
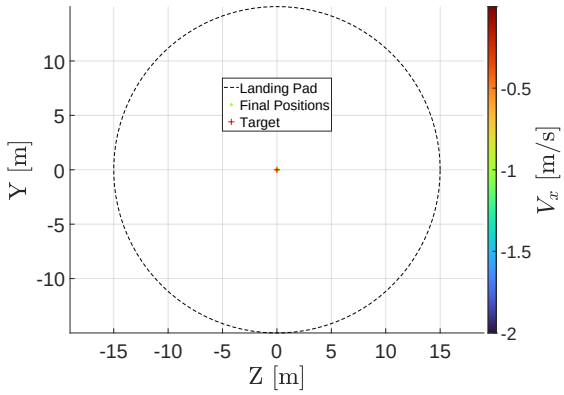


Figure 7.7: F2 (thrust degradation): Soft landing success rates for nominal, recovery, recovery-delay, and converged FT-SCP cases.

highlights how the dispersions at landing are much smaller than for F1 and track the guidance results (Figure 7.8b) much better. The reason is that in this failure mode the faulty engine is never completely inoperative. Even at maximum degradation (i.e., when the faulty engine can only deliver its minimum thrust), the corresponding TVC still provides some control authority, though with reduced effectiveness. This contrasts with F1, where the fault disables the engine entirely and removes the contribution of its TVCs.



(a) Closed-loop recovery cases.



(b) FT-SCP solutions.

Figure 7.8: F2 (thrust degradation): Lateral touchdown positions at the landing pad color-coded by vertical touchdown velocity.

F3: TVC jamming

Figure 7.9 summarizes the results for fault type F3. As in the case of F2, some nominal runs are still able to achieve a successful landing. These are mainly cases with low aerodynamic uncertainties, small initial dispersions, and mild TVC faults, conditions under which the feedback controller is still able to compensate effectively.

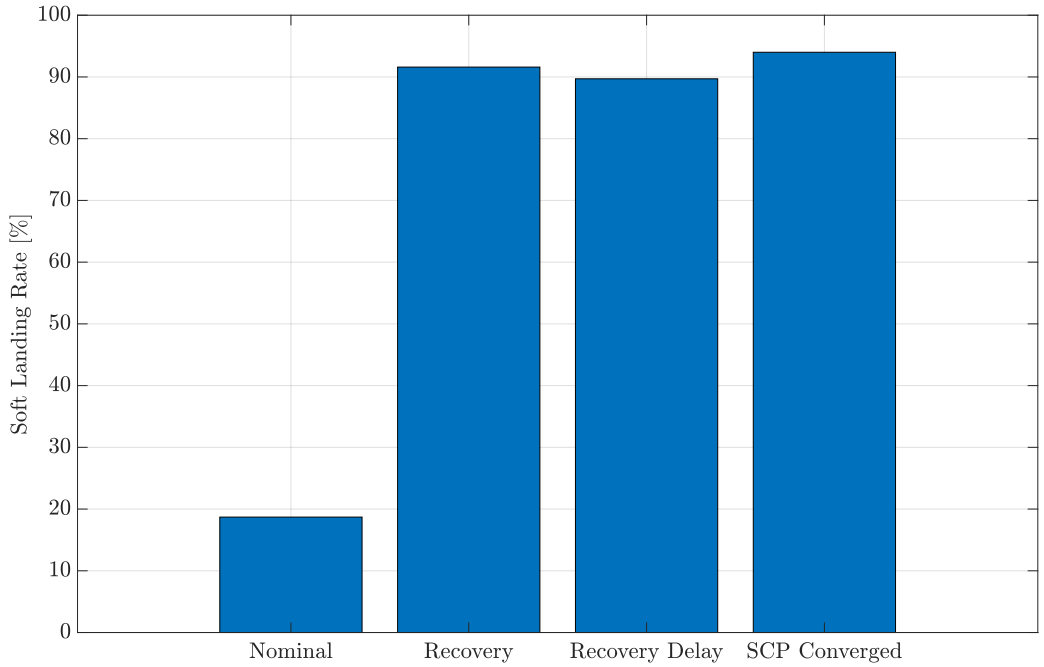
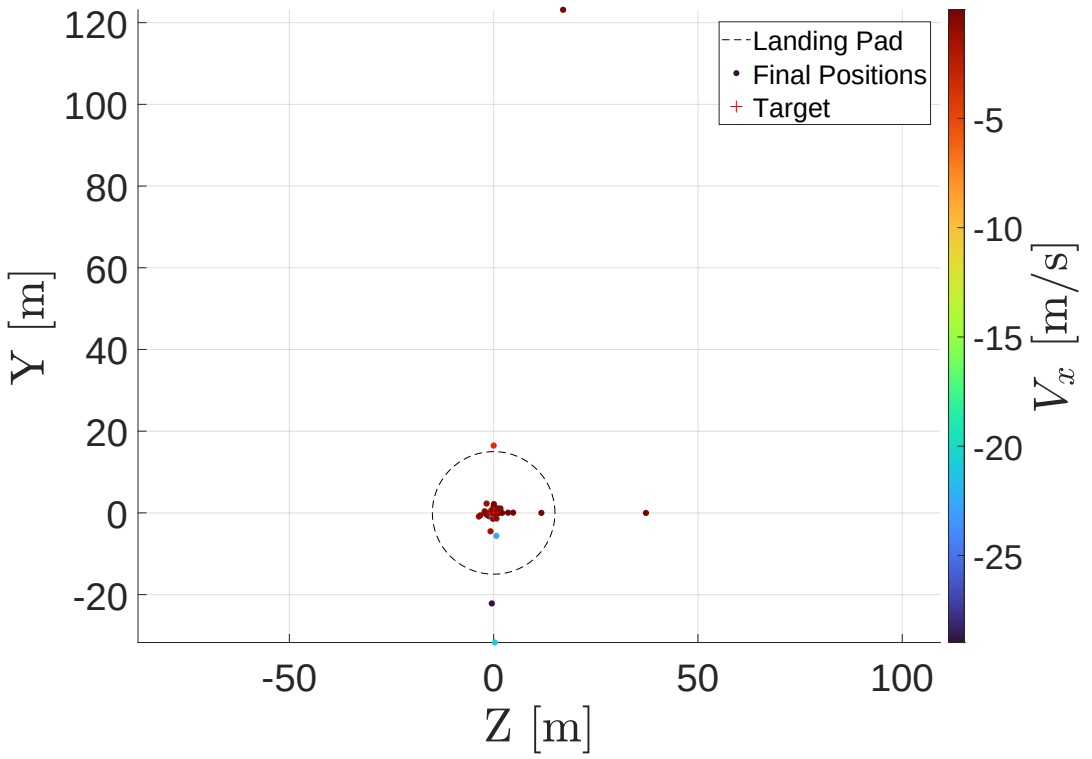


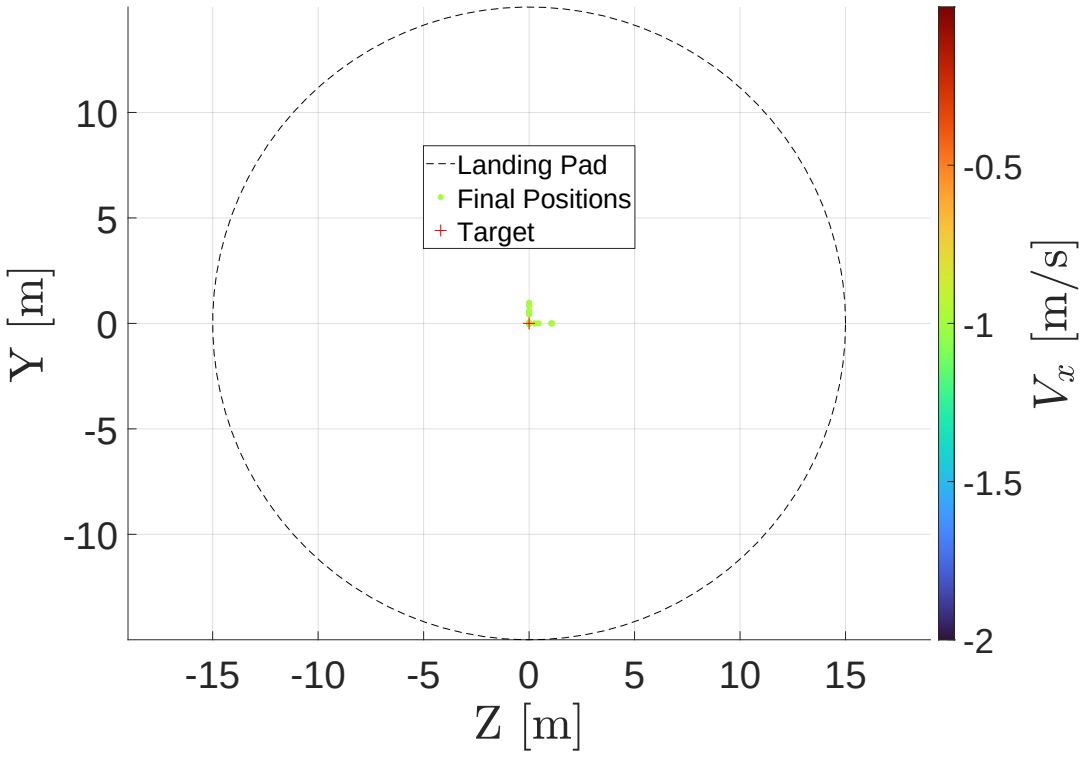
Figure 7.9: F3 (TVC jamming): Soft landing success rates for nominal, recovery, recovery-delay, and converged FT-SCP cases.

Figure 7.10b shows the FT-SCP solution's lateral positions at landing, in which it can be noticed that the final position, even if with limited deviation, is actually exploited for this case, with some solutions landing around 1-2 meters from the origin. Figure 7.11a shows that the sensitivity of convergence to the fault time is reduced compared to F1, with many runs still converging even towards the end of the interval, close to 80% of the flight time. At the same time, a higher number of early-fault cases (in terms of t_{fault}) fail to converge. Most of these can be attributed to situations where large TVC stuck angles are combined with significant lateral dispersions, preventing the algorithm from finding a feasible solution capable of steering the vehicle back to the landing pad.

As an example, a representative early non-converged case is summarized in Table 7.2. The combination of the initial conditions and the angles at which the TVC is stuck makes the problem of returning, or even approaching the landing pad, infeasible. Such a set of initial conditions can nonetheless produce a solution if the weight on the final lateral position relaxation is reduced. For instance, decreasing this weight from its original value of 50 to 1 leads to the results shown in Figure 7.12, where the algorithm is able to converge to a feasible solution with lateral touchdown positions of 100 m and 150 m in the y and z directions, respectively. The corresponding propulsive and aerodynamic control inputs are shown in Figure 7.13.



(a) Closed-loop recovery cases.

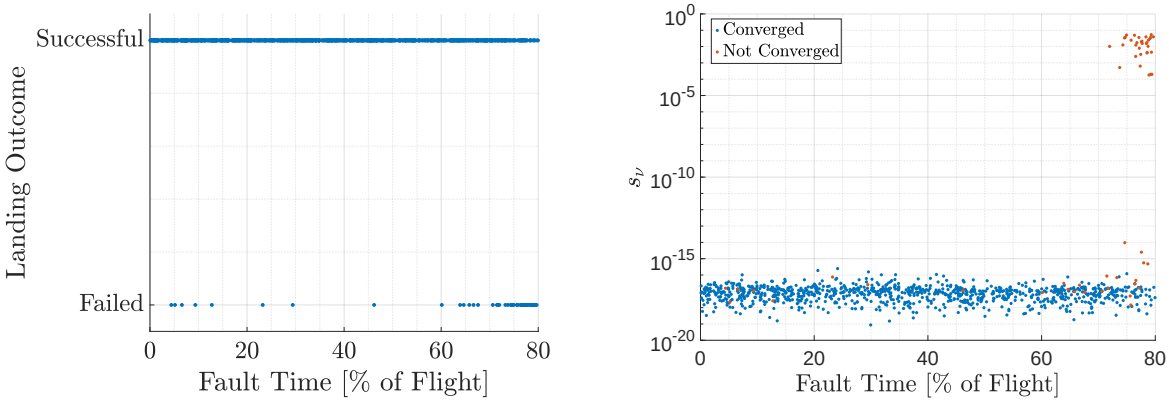


(b) FT-SCP solutions.

Figure 7.10: F3 (TVC jamming): Lateral touchdown positions at the landing pad color-coded by vertical touchdown velocity.

Table 7.2: Representative failed case (F3): Initial conditions and fault parameters

Parameter	Value	Unit
r_0	$[43459,\ 489.2,\ 485.6]^T$	[m]
v_0	$[-165.5,\ 1.2,\ 10.2]^T$	[m/s]
m_0	4233	[kg]
TVC motor ID	4	[–]
TVC stuck angle	$[2.9,\ -3.8]$	[deg]
Fault time	3.49	[s]



(a) Convergence versus fault time.

(b) Virtual controls slacks.

Figure 7.11: F3 (TVC jamming): FT-SCP convergence analysis.

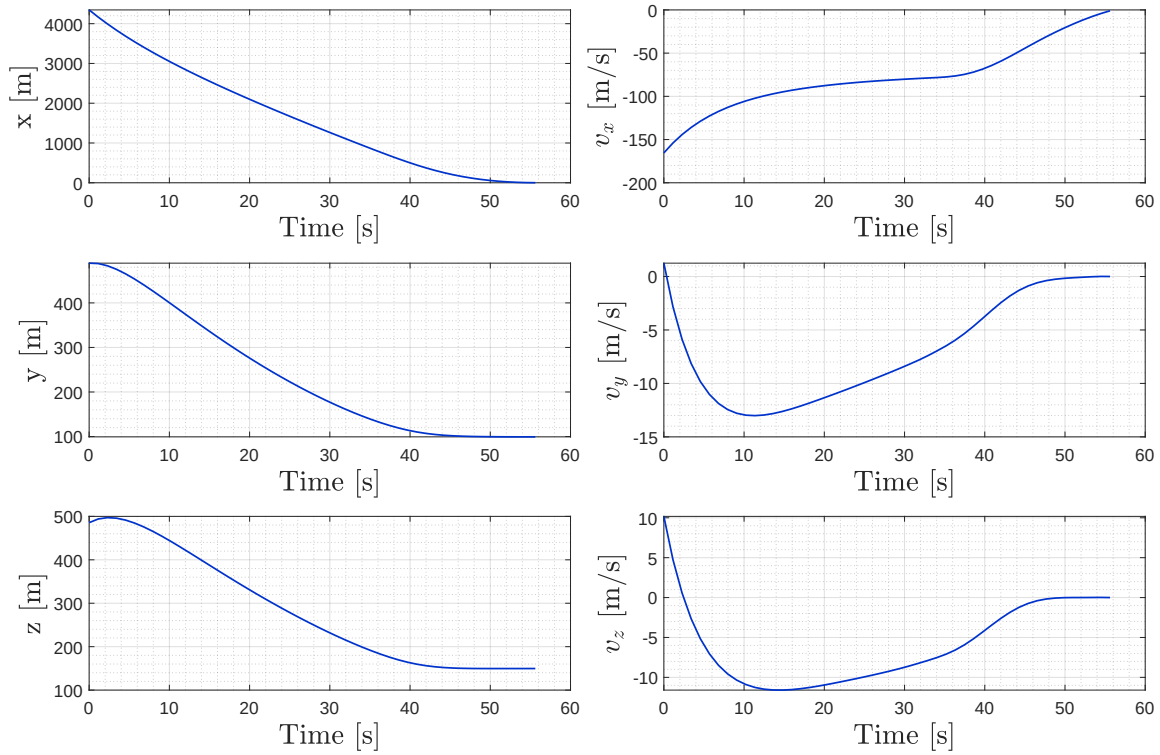
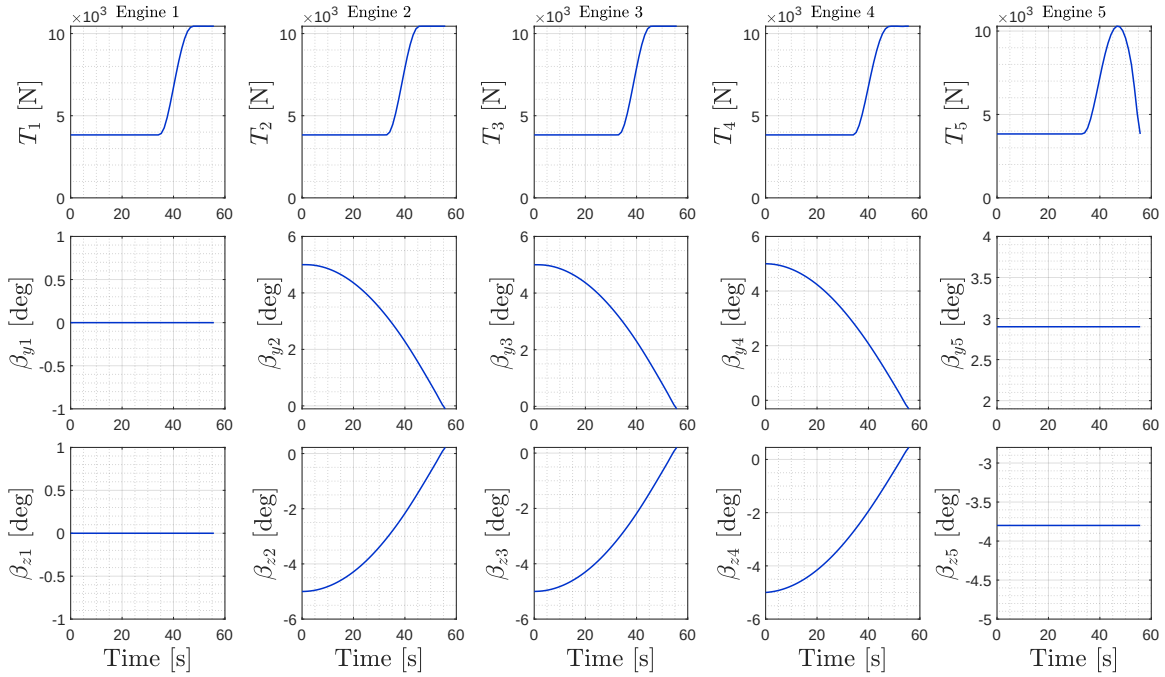
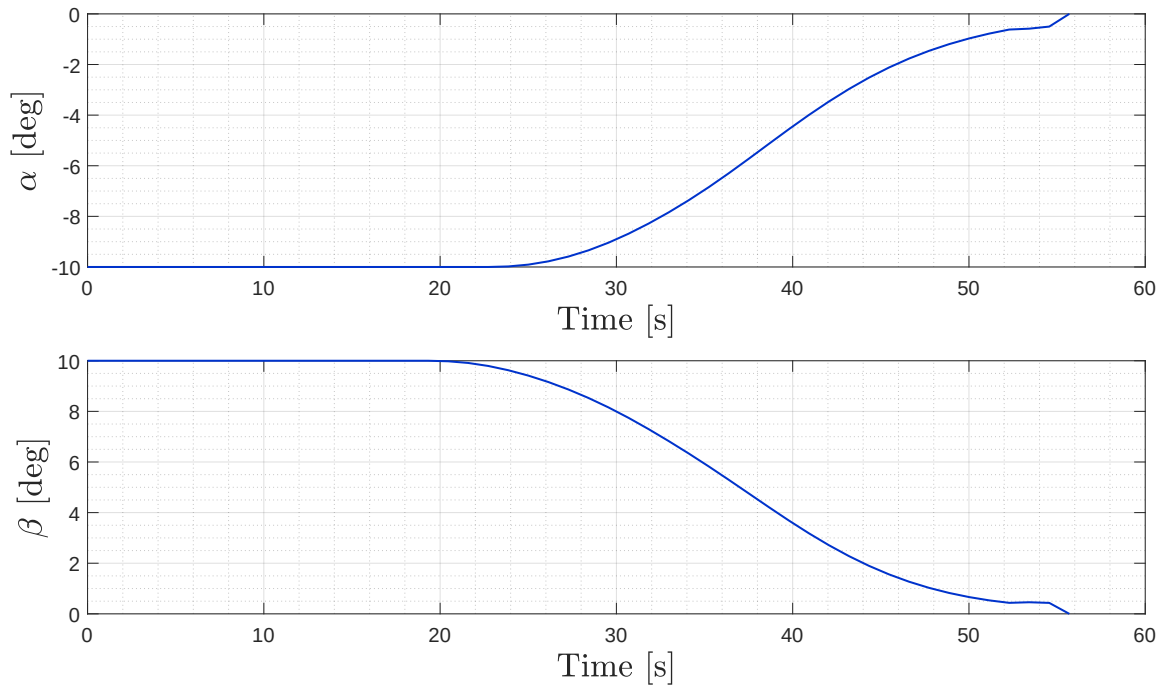


Figure 7.12: F3 (TVC jamming) relaxed solution: position and velocity.



(a) Propulsive controls.



(b) Aerodynamic controls.

Figure 7.13: F3 (TVC jamming) relaxed solution: controls.

7.3 Study 2: Multiple Faults

The goal of this study is to characterize the impact of simultaneous thrust loss from multiple engines on the mission outcome. To this end, a dedicated Monte Carlo campaign is executed in which, at the fault time, three different thrust degradation factors are applied to three distinct engines, as summarized in Table 7.3. Apart from the fault model, the simulation setup remains identical to the single-fault cases.

Table 7.3: Randomized parameters for multiple-fault case F4

Parameter	Distribution
Fault time	$\mathcal{U}(0,\ 80\%\, t_{\text{ref}})$
Engine numbers	distinct samples from $\mathcal{U}(1,\ 5)$
Thrust degradation factors ξ_1, ξ_2, ξ_3	$\mathcal{U}\left(\frac{T_{\text{min,nom}}}{T_{\text{max,nom}}},\ 0.95\right)$

Figure 7.14 shows that the FT-SCP converged in about 55% of the runs, with the recovery and recovery-delay cases displaying similar results.

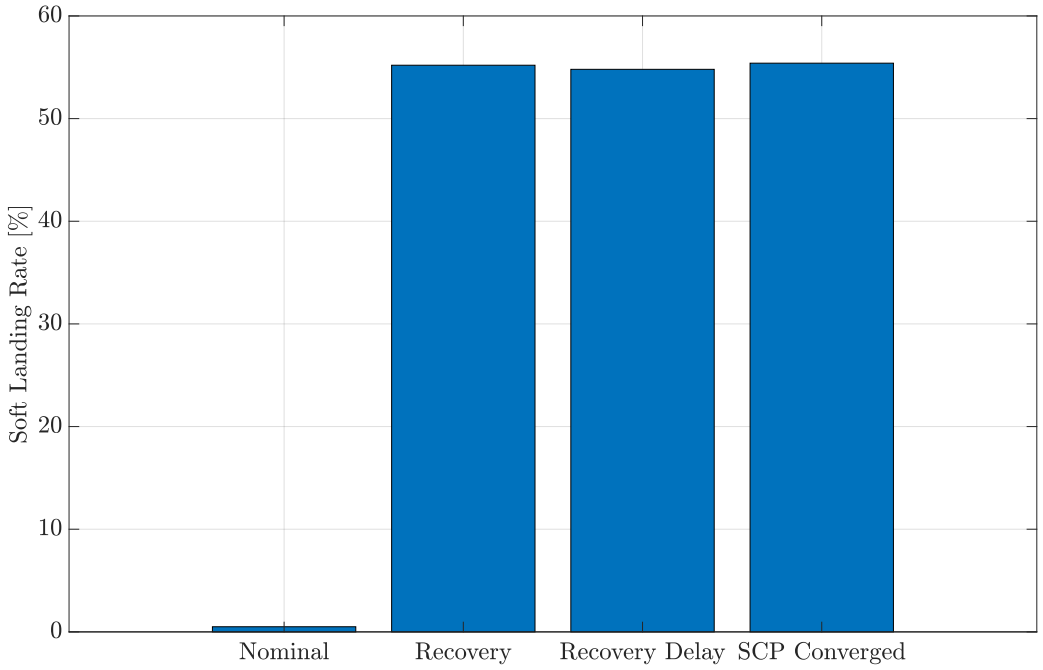


Figure 7.14: F2 (thrust degradation on multiple engines): Soft landing success rates for nominal, recovery, recovery-delay, and converged FT-SCP cases.

Figure 7.15 highlights the key difference compared to the single-fault cases: the convergence trend with respect to fault time is much weaker. This can be attributed to the fact that severe thrust degradations may be unrecoverable even from the very beginning of the mission. To better understand how these faults affect the outcome, Figure 7.16 shows, in blue bars, the minimum thrust (as a percentage of the maximum nominal one) with which the rocket is able to land, plotted as a function of fault time, together with the number of converged runs for each interval. It is clear that as the fault occurs later in the descent (i.e., closer to the landing pad), the minimum thrust required for a successful landing increases, in other words, the engines cannot be too degraded. At the very beginning of the mission, even severe thrust

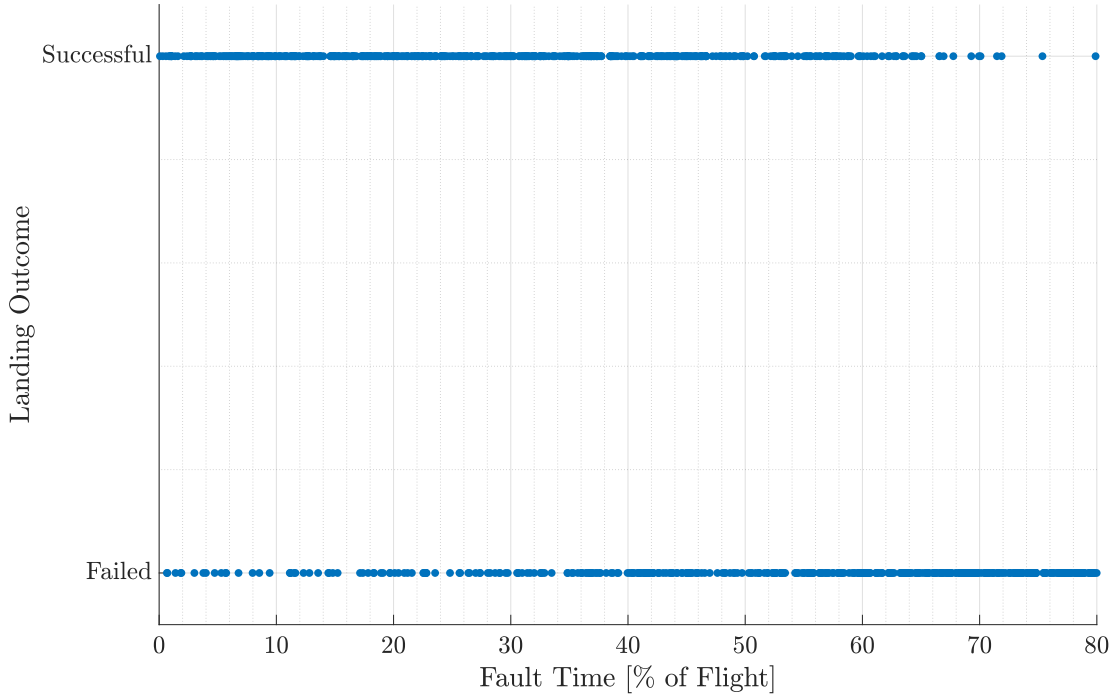


Figure 7.15: F4 (thrust degradation on multiple engines): Soft landing success rates for nominal, recovery, recovery-delay, and converged FT-SCP cases.

degradations remain recoverable. In fact, some cases in which three faulty engines can only fire at their nominal minimum thrust, thereby reducing the overall available thrust to about 60%, still result in a safe landing.

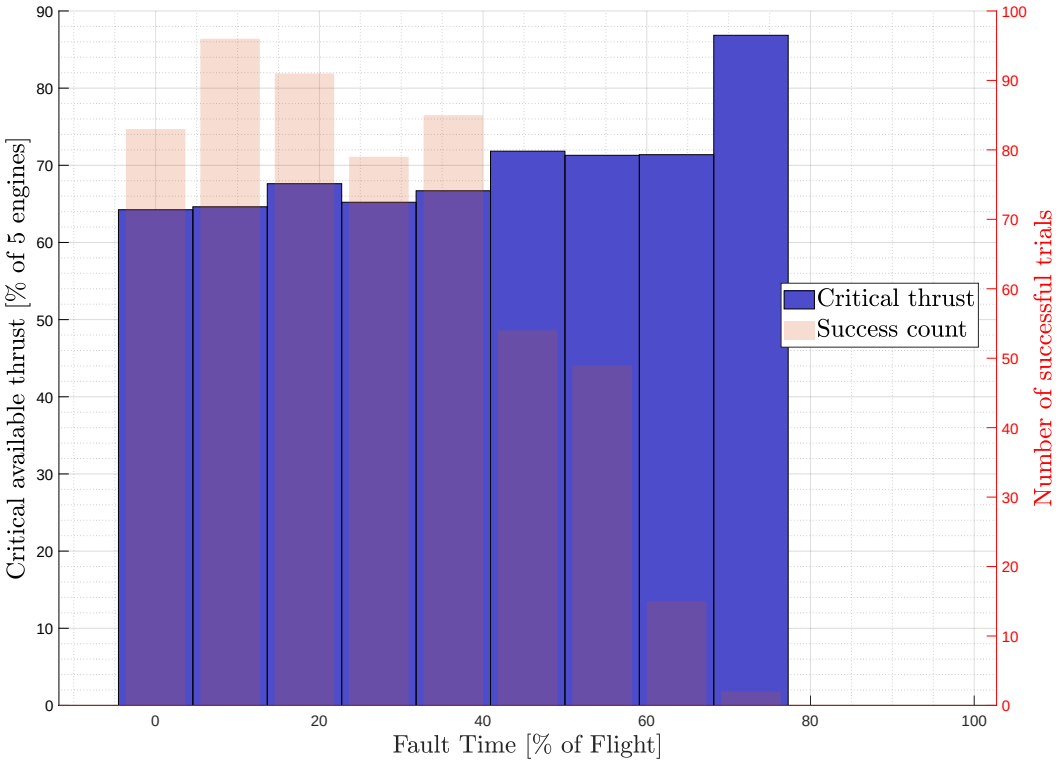


Figure 7.16: F4 (thrust degradation on multiple engines): Critical thrust as a function of fault time.

8 Conclusions & Future Directions

This thesis set out to investigate how fault-tolerant guidance can be achieved for reusable launch vehicles during the powered descent and landing phase. To this end, two research questions were posed:

RQ1: How can an online guidance provide fault tolerance in presence of faults affecting the engines in reusable launch vehicles during the landing burn?

RQ2: To what extent (i.e., until when across the descent) is the contingency guidance effective?

The following sections provide direct answers to these questions, summarize the main contributions of the thesis, and outline directions for future research as well as the limitations of the approach.

In the first half of Chapter 6, the foundation of the convex framework for trajectory optimization was established, leading to the standard SCP algorithm that provided the baseline for a general online guidance method. This formulation was then expanded in the second half of the same chapter in order to address the first research question. In response to the first research question, this thesis has shown that sequential convex programming, extended to a fault-tolerant SCP (FT-SCP) formulation, enables the recomputation of feasible landing trajectories even in the presence of faults. By appropriately updating the OCP formulation based on the fault information, the algorithm is able to compute a trajectory adapted to the reduced resources caused by the fault.

The second research question concerned the extent to which such contingency guidance remains effective across the descent. Extensive Monte Carlo campaigns revealed that the FT-SCP approach achieves safe landings for a wide range of fault scenarios, provided that faults occur with sufficient altitude and time margin. For faults that resulted in a complete engine loss, it was shown that without guidance recomputation none of the runs concluded with a successful landing. After the introduction of recovery guidance, however, many of the previously failed runs became successful, although recovery guidance alone was not always sufficient to guarantee a soft landing. In these cases, the baseline controller, designed under nominal conditions, was unable to track some of the feasible FT-SCP trajectories when aerodynamic uncertainties were significant. For other types of faults, such as partial thrust degradation and TVC jamming, the results demonstrated that recovery guidance was highly effective even when using the baseline controller.

The effectiveness of the approach decreases when faults occur later in the descent, as the combination of altitude, vertical velocity, and degraded conditions can make it physically impossible for the vehicle to decelerate sufficiently before touchdown. Overall, the method substantially extends the safe operating envelope of the vehicle, even though it cannot guarantee success in every possible scenario.

8.1 Limitations

Despite the promising results, several limitations of the present work must be acknowledged. First, the FT-SCP algorithm was only tested on a simulation model in which the plant dynamics closely matched those used in the OCP formulation. A more thorough verification against full 6-DoF simulations is required to ensure that the generated trajectories are physically feasible when all degrees of freedom are considered. Furthermore, results in Chapter 7 showed that guidance computation delay had a minor impact on the outcome of the mission. However, it remains unknown whether this conclusion would be altered with the inclusion of attitude dynamics.

Second, the algorithm is highly sensitive to the numerical values chosen for the cost function weights. Small variations in these values can drastically alter the solutions, and in some cases represent the difference between convergence and divergence. This sensitivity complicates the tuning process and raises questions about robustness.

Third, although convex optimization was selected for its predictable runtime and convergence guarantees, the current implementation is not yet online-capable. Convergence times remain too long for real-time applications, primarily due to (i) numerical computation of Jacobians via finite differences, and (ii) reliance on *CVX* as an intermediate modeling layer. Both aspects introduce significant computational overhead.

Finally, the entire work assumes instantaneous and error-free FDI. This assumption is clearly unrealistic and may lead to an overestimation of the practical effectiveness of the proposed approach.

8.2 Future Work

Several directions emerge naturally for future work. First, extending the algorithm to a full 6-DoF formulation of the dynamics would allow more realistic verification and potentially reveal new limitations of the current approach.

Second, reducing computational effort is critical for enabling online use. Promising directions include:

- Replacing aerodynamic lookup tables with polynomial approximations to allow for analytical expressions of the Jacobians, or applying automatic differentiation instead of computing Jacobians through finite differences.
- Implementing the algorithm in C and directly interfacing with the SOCP solver, removing *CVX* as a middle layer.

Third, combining trajectory reconfiguration with FTC strategies remains an open question. A systematic analysis is required to determine the range of faults for which a full guidance recomputation is beneficial, and the conditions under which a simpler FTC strategy would be sufficient, as well as how much the overall performance would improve from combining the two approaches. Simulations with an actual FDI module could provide further insight into how imperfect and delayed FDI would affect the mission outcome.

Finally, the mission scope considered in this thesis was restricted to the final PDL phase. Expanding the envelope to earlier mission stages, such as from the first descent burn, would provide more freedom and a

larger solution space for recovery. Likewise, allowing for adaptive mission objectives, such as diverting to an alternate landing pad in case the nominal site is no longer reachable, would significantly enhance the practical utility of the method. Along similar lines, the concepts explored here may also find applicability in the ascent phase, for example in connection with rescue-orbit strategies that have been proposed in the literature.

Bibliography

[1] (ESA), E. S. A.: Themis, https://www.esa.int/Enabling_Support/Space_Transportation/Themis.

[2] (ISRO), I. S. R. O.: Reusable Launch Vehicle – Technology Demonstrator (RLV-TD), %5Curl%7Bhttps://www.isro.gov.in/RLVTD.html%7D.

[3] Açıkmeşe, B.; Blackmore, L.: Lossless convexification of a class of optimal control problems with non-convex control constraints. Automatica, Vol. 47, No. 2, pp. 341–347, 2011.

[4] Açıkmeşe, B.; Carson, J. M.; Blackmore, L.: Lossless Convexification of Nonconvex Control Bound and Pointing Constraints of the Soft Landing Optimal Control Problem. IEEE Transactions on Control Systems Technology, Vol. 21, No. 6, pp. 2104–2113, 2013.

[5] Açıkmeşe, B.; Casoliva, J.; Carson, J.; Blackmore, L.: G-FOLD: A Real-Time Implementable Fuel Optimal Large Divert Guidance Algorithm for Planetary Pinpoint Landing. LPI Contributions, pp. 4193–, June 2012.

[6] Anderson, B. D. O.; Moore, J. B.: Optimal control: linear quadratic methods, USA: Prentice-Hall, Inc., ISBN 0136385605, 1990.

[7] Andersson, J. A. E.; Gillis, J.; Horn, G.; Rawlings, J. B.; Diehl, M.: CasADi – A software framework for nonlinear optimization and optimal control. Mathematical Programming Computation, Vol. 11, No. 1, pp. 1–36, 2019.

[8] ApS, M.: MOSEK Optimization Toolbox for MATLAB Manual. Version 11.0, 2025.

[9] Baiocco, P.: Overview of reusable space systems with a look to technology aspects. Acta Astronautica, Vol. 189, pp. 10–25, Dec. 2021.

[10] Bauer, W.; Rickmers, P.; Kallenbach, A.; Stappert, S.; Wartemann, V.; Hans-Joachim Merrem, C.; Schwarz, R.; Sagliano, M.; Grundmann, J. T.; Flock, A.; Thiele, T.; Kiehn, D.; Bierig, A.; Windelberg, J.; Ksenik, E.; Bruns, T.; Ruhe, T.; Elsäßer, H.: DLR Reusability Flight Experiment ReFEx. Acta Astronautica, Vol. 168, pp. 57–68, 2020.

[11] Bazaraa, M. S.: Nonlinear Programming: Theory and Algorithms, Wiley Publishing, ISBN 1118857569, 2013.

[12] Bedrossian, N.; Bhatt, S.; Lammers, M.; Nguyen, L.: Zero Propellant Maneuver: Flight Results for 180° ISS Rotation, NASA CP 2007-214158, Sept. 2007.

[13] Benson, D. A.; Huntington, G. T.; Thorvaldsen, T. P.; Rao, A. V.: Direct Trajectory Optimization and Costate Estimation via an Orthogonal Collocation Method. Journal of Guidance, Control, and Dynamics, Vol. 29, No. 6, pp. 1435–1440, 2006.

[14] Betts, J.: Practical methods for optimal control and estimation using nonlinear programming. 2nd ed, Vol. 19, Jan. 2010.

[15] Betts, J. T.: Survey of Numerical Methods for Trajectory Optimization. Journal of Guidance, Control, and Dynamics, Vol. 21, No. 2, pp. 193–207, 1998.

[16] Bittner, L.: L. S. Pontryagin, V. G. Boltyanskii, R. V. Gamkrelidze, E. F. Mishechenko, The Mathematical Theory of Optimal Processes. VIII + 360 S. New York/London 1962. John Wiley & Sons. Preis 90/–. ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik, Vol. 43, No. 10-11, pp. 514–515, 1963.

[17] Blue Origin: Blue Origin Makes Historic Rocket Landing, `https://www.blueorigin.com/news/blue-origin-makes-historic-rocket-landing`, 2015.

[18] Blue Origin: New Glenn Mission NG-1, `https://www.blueorigin.com/missions/ng-1`, 2025.

[19] Bonalli, R.; Cauligi, A.; Bylard, A.; Pavone, M.: GuSTO: Guaranteed Sequential Trajectory optimization via Sequential Convex Programming. In: 2019 International Conference on Robotics and Automation (ICRA), 2019.

[20] Boyd, S.; Vandenberghe, L.: Convex Optimization, Cambridge University Press, 2004.

[21] Bryson, A.: Applied Optimal Control: Optimization, Estimation and Control, CRC Press, ISBN 9781351465915, 2018.

[22] Clark, S.: SpaceX Successfully Fires Satellites into Orbit but Loses Booster on Landing, `https://spaceflightnow.com/2016/06/15/spacex-successfully-fires-satellites-into-orbit-but-loses-booster-on-landing/`, 2016.

[23] Cofield, C.: SpaceX Launches Falcon Heavy, World's Most Powerful Rocket, on Historic Test Flight, `https://www.space.com/39607-spacex-falcon-heavy-first-test-flight-launch.html`, 2018.

[24] CVX Research, I.: CVX: Matlab Software for Disciplined Convex Programming, version 2.0, Aug. 2012.

[25] Darby, C. L.; Hager, W. W.; Rao, A. V.: An hp-adaptive pseudospectral method for solving optimal control problems. Optimal Control Applications and Methods, Vol. 32, No. 4, pp. 476–502, 2011.

[26] Dietlein, I.; Bussler, L.; Stappert, S.; Wilken, J.; Sippel, M.: Overview of System Study on Recovery Methods for Reusable First Stages of Future European Launchers. CEAS Space Journal, Vol. 17, No. 1, pp. 71–88, 2025.

[27] Dormand, J.; Prince, P.: A family of embedded Runge-Kutta formulae. Journal of Computational and Applied Mathematics, Vol. 6, No. 1, pp. 19–26, 1980.

[28] Duchi, J.; Boyd, S.; Mattingley, J.: Sequential Convex Programming, `https://web.stanford.edu/class/ee364b/lectures/seq_notes.pdf`, 2018.

[29] Dumont, E.; Ecker, T.; Chavagnac, C.; Witte, L.; Windelberg, J.; Klevanski, J.; Giagkozoglou, S.: CALLISTO - Reusable VTVL launcher first stage demonstrator. In: May 2018.

[30] European Space Agency: History: Hermes Spaceplane (1987), `https://www.esa.int/About_Us/50_years_of_ESA/History_Hermes_spaceplane_1987`, 1987.

[31] Evaluation, S. S.; Team, A.: Surveyor I: Preliminary Results. Science, Vol. 152, No. 3730, pp. 1737–1750, 1966.

[32] Fari, S.; Seelbinder, D.; Theil, S.; Simplício, P.: Robust Fault Detection and Isolation algorithms for TVC systems: An experimental test. In: 75th International Astronautical Congress, 2024.

[33] Fernández, L. A.; Wiedemann, C.; Braun, V.: Analysis of Space Launch Vehicle Failures and Post-Mission Disposal Statistics. Aerotec. Missili Spaz. Vol. 101, pp. 243–256, Sept. 2022.

[34] Freeman, D. C.; Talay, T. A.; Austin, R. E.: Reusable Launch Vehicle Technology Program, NASA-TM-110473, 1996.

[35] Grant, M.; Boyd, S.: Graph implementations for nonsmooth convex programs. In: Recent Advances in Learning and Control. Springer-Verlag Limited, pp. 95–110, 2008.

[36] Greensite, A. L.: Analysis and Design of Space Vehicle Flight Control Systems. Volume VII: Attitude Control During Launch, NASA-CR-826, 1967.

[37] Hiriart-Urruty, J.; Lemarechal, C.: Convex Analysis and Minimization Algorithms I: Fundamentals. Grundlehren der mathematischen Wissenschaften, Springer Berlin Heidelberg, ISBN 9783540568506, 1996.

[38] Hussein, M.; De Weck, O.; Terakado, D.: Comprehensive Study of the International Space Launch Industry: Programmatic Analysis and Technical Failures. In: IAF Space Transportation Solutions and Innovations Symposium, 2024.

[39] Hwang, I.; Kim, S.; Kim, Y.; Seah, C. E.: A Survey of Fault Detection, Isolation, and Reconfiguration Methods. IEEE Transactions on Control Systems Technology, Vol. 18, No. 3, pp. 636–653, 2010.

[40] Isermann, R.; Ballé, P.: Trends in the application of model-based fault detection and diagnosis of technical processes. Control Engineering Practice, Vol. 5, No. 5, pp. 709–719, 1997.

[41] Jarre, F.: Interior-point methods for convex programming. Applied Mathematics and Optimization, Vol. 26, No. 3, pp. 287–311, 1992.

[42] Jones, A.: Long March 8 Debuts with Successful Launch of Nine Satellites, `https://www.nasaspaceflight.com/2020/12/long-march-8-debuts-nine-satellites/`, 2020.

[43] Josselyn, S.; Ross, I.: Rapid Verification Method for the Trajectory Optimization of Reentry Vehicles. Journal of Guidance Control and Dynamics - J GUID CONTROL DYNAM, Vol. 26, pp. 505–508, May 2003.

[44] Liu, X.: Fuel-Optimal Rocket Landing with Aerodynamic Controls. Journal of Guidance, Control, and Dynamics, Vol. 42, No. 1, pp. 65–77, 2019.

[45] Liu, X.; Lu, P.; Pan, B.: Survey of convex optimization for aerospace applications. Astrodynamics, Vol. 1, No. 1, pp. 23–40, 2017.

[46] Liu, X.; Shen, Z.; Lu, P.: Entry Trajectory Optimization by Second-Order Cone Programming. Journal of Guidance, Control, and Dynamics, Vol. 39, No. 2, pp. 227–241, 2016.

[47] Lobo, M. S.; Vandenberghe, L.; Boyd, S.; Lebret, H.: Applications of second-order cone programming. Linear Algebra and its Applications, Vol. 284, No. 1, International Linear Algebra Society (ILAS) Symposium on Fast Algorithms for Control, Signals and Image Processing, pp. 193–228, 1998.

[48] Lu, P.: Convex–Concave Decomposition of Nonlinear Equality Constraints in Optimal Control. Journal of Guidance, Control, and Dynamics, Vol. 44, pp. 1–11, Oct. 2020.

[49] Malyuta, D.; Reynolds, T.; Szmuk, M.; Lew, T.; Bonalli, R.; Pavone, M.; Açıkmeşe, B.: Convex Optimization for Trajectory Generation, June 2021.

[50] Mao, Y.; Szmuk, M.; Açıkmeşe, B.: Successive Convexification: A Superlinearly Convergent Algorithm for Non-convex Optimal Control Problems, Apr. 2018.

[51] Miao, X.; Song, Y.; Zhang, Z.; Gong, S.: Successive Convexification for Ascent Trajectory Replanning of a Multistage Launch Vehicle Experiencing Nonfatal Dynamic Faults. IEEE Transactions on Aerospace and Electronic Systems, Vol. 58, No. 3, pp. 2039–2052, 2022.

[52] Morio, V.: Design and development of an autonomous guidance law by flatness approach. Application to an atmospheric reentry mission. May 2009.

[53] NASA: NASA Tipping Point Partnership with Blue Origin to Test Precision Lunar Landing Technologies.

[54] National Aeronautics and Space Administration: Viking 1: Early Results, SP-406, 1976.

[55] Nocedal, J.; Wright, S.: Numerical optimization. Springer series in operations research and financial engineering, New York, NY: Springer, ISBN 978-0-387-30303-1, 2006.

[56] Orr, J.; Zwieten, T. V.: Robust, Practical Adaptive Control for Launch Vehicles. In: AIAA Guidance, Navigation, and Control Conference.

[57] Pascucci, C. A.; Fernandez, A.; Jerez, J.; Bennani, S.; Jean-PhilippePreaud, ‡.; Bru, J.; Carlo; Pascucci, A.: Design Simulation and Software Validation for On-board and Real-Time Optimal Guidance. In: 2019.

[58] Patterson, M. A.; 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, Oct. 2014.

[59] Paulino, N.; Roche Arroyos, C.; Ferreira, L.; Pascucci, M.; Cachim, P.; Lourenço, P.; García, J.; Navarro-Tapia, D.; Marcos, A.; Mohamed, L.; Alexandre, P.; Simplicio, P.; Bennani, S.: Fault Tolerant Control for a Cluster of Rocket Engines -Methods and outcomes for guidance and control recovery strategies in launchers. In: June 2023.

[60] Rao, A.: A Survey of Numerical Methods for Optimal Control. Advances in the Astronautical Sciences, Vol. 135, Jan. 2010.

[61] `http://www.falcon-m.com` Rieck, M.; Bittner, M.; Grüter, B.; Diepolder, J.; Piprek, P.; Göttlicher, C.; Schwaiger, F.; Hosseini, B.; Schweighofer, F.; Akman, T.; Holzapfel, F.: FALCON.m User Guide. version 1.32, 2024.

[62] Ross, I. M.; Fahroo, F.: User's Manual for DIDO 2001α: A MATLAB Application for Solving Optimal Control Problems, AAS-01-03, 2001.

[63] Runge–Kutta Methods. In: Numerical Methods for Ordinary Differential Equations. John Wiley Sons, Ltd, chap. 3, pp. 137–316, ISBN 9780470753767, 2008.

[64] Sagliano, M.: Generalized hp Pseudospectral-Convex Programming for Powered Descent and Landing. Journal of Guidance, Control, and Dynamics, Vol. 42, No. 7, pp. 1562–1570, 2019.

[65] Sagliano, M.: Pseudospectral Convex Optimization for Powered Descent and Landing. Journal of Guidance, Control, and Dynamics, Vol. 41, No. 2, pp. 320–334, 2018.

[66] Sagliano, M.; Heidecker, A.; Farì, S.; Alfredo, M. H. J.; Schlotterer, M.; Woicke, S.; Seelbinder, D.; Dumont, E.: Powered Atmospheric Landing Guidance for Reusable Rockets: the CALLISTO studies. In: AIAA SCITECH 2024 Forum.

[67] Sagliano, M.; Heidecker, A.; Hernández, J. M.; Farì, S.; Schlotterer, M.; Woicke, S.; Seelbinder, D.; Dumont, E.: Onboard Guidance for Reusable Rockets: Aerodynamic Descent and Powered Landing. In: AIAA Scitech 2021 Forum.

[68] Sagliano, M.; Hernández, J. A. M.; Farı, S.; Heidecker, A.; Schlotterer, M.; Woicke, S.; Seelbinder, D.; Krummen, S.; Dumont, E.: Unified-Loop Structured H-Infinity Control for Aerodynamic Steering of Reusable Rockets. Journal of Guidance, Control, and Dynamics, Vol. 46, No. 5, pp. 815–837, 2023.

[69] Sagliano, M.; Ishimoto, S.; Macés-Hernández, J. A.; Seelbinder, D.; Dumont, E.: Guidance and Control Strategy for the CALLISTO Flight Experiment. In: July 2019.

[70] Sagliano, M.; Lu, P.; Seelbinder, D.; Theil, S.: Analytical Treatise on Endo-Atmospheric Fuel-Optimal Rocket Landings. Journal of Guidance, Control, and Dynamics, Vol. 48, No. 3, pp. 450–469, 2025.

[71] Sagliano, M.; Seelbinder, D.; Theil, S.: SPARTAN: Rapid Trajectory Analysis via Pseudospectral Methods. In: June 2021.

[72] Sagliano, M.; Seelbinder, D.; Theil, S.; Im, S.; Lee, J.; Lee, K.: Booster Dispersion Area Management through Aerodynamic Guidance and Control. In: AIAA SCITECH 2022 Forum.

[73] Sagliano, M.; Seelbinder, D.; Theil, S.; Lu, P.: Six-Degree-of-Freedom Rocket Landing Optimization via Augmented Convex–Concave Decomposition. Journal of Guidance, Control, and Dynamics, Vol. 47, No. 1, pp. 20–35, 2024.

[74] Scharf, D. P.; Regehr, M. W.; Vaughan, G. M.; Benito, J.; Ansari, H.; Aung, M.; Johnson, A.; Casoliva, J.; Mohan, S.; Dueri, D.; Açikmeşe, B.; Masten, D.; Nietfeld, S.: ADAPT demonstrations of onboard large-divert Guidance with a VTVL rocket. In: 2014 IEEE Aerospace Conference, 2014.

[75] Shampine, L. F.; Reichelt, M. W.: The MATLAB ODE Suite. SIAM Journal on Scientific Computing, Vol. 18, No. 1, pp. 1–22, 1997.

[76] Siddiqi, A. A.: Beyond Earth: A Chronicle of Deep Space Exploration, 1958–2016, National Aeronautics and Space Administration, Office of Communications, NASA History Division, 2018.

[77] Simplicio, P.; Marcos, A.; Bennani, S.: A Reusable Launcher Benchmark with Advanced Recovery Guidance. In: 5th CEAS EuroGNC Conference, 2019.

[78] Sissenwine; Teweles, S.; Dubin, M.: U. S. Standard Atmosphere, 1976. In: National Aeronautics and Space Administration, October 1976, pp. 1 - 227, NASA-TM-X-74335, NOAA-S/T 76-1562.

[79] Slyuta, E.: CHAPTER 3 - The Luna program. In: Sample Return Missions. Elsevier, pp. 37–78, ISBN 978-0-12-818330-4, 2021.

[80] Smaili, H.; Breeman, J.; Lombaerts, T.; Stroosma, O.: A Benchmark for Fault Tolerant Flight Control Evaluation. IFAC Proceedings Volumes, Vol. 42, No. 8, 7th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes, pp. 241–246, 2009.

[81] Song, Z.-y.; Wang, C.; Theil, S.; Seelbinder, D.; Sagliano, M.; Liu, X.-f.; Shao, Z.-j.: Survey of autonomous guidance methods for powered planetary landing. Frontiers of Information Technology & Electronic Engineering, Vol. 21, No. 5, pp. 652–674, 2020.

[82] Song, Z.; Pan, H.; Zhao, Y.; Yao, W.; He, Y.; Wang, C.: Reviews and Challenges in Reliability Design of Long March Launcher Control Systems. AIAA Journal, Vol. 60, No. 2, pp. 537–550, 2022.

[83] Song, Z.; Wang, C.; Gong, Q.: Joint dynamic optimization of the target orbit and flight trajectory of a launch vehicle based on state-triggered indices. Acta Astronautica, Vol. 174, pp. 82–93, 2020.

[84] SpaceX: Starship Flight 5, https://www.spacex.com/launches/starship-flight-5, 2024.

[85] Stevens, B.; Lewis, F.; Johnson, E.: Aircraft Control and Simulation: Dynamics, Controls Design, and Autonomous Systems, Third Edition, ISBN 9781119174882, Oct. 2015.

[86] Sutton, G.; Biblarz, O.: Rocket Propulsion Elements. A Wiley Interscience publication, Wiley, ISBN 9780471326427, 2001.

[87] Szmuk, M.; Acikmese, B.: Successive Convexification for 6-DoF Mars Rocket Powered Landing with Free-Final-Time. In: 2018 AIAA Guidance, Navigation, and Control Conference.

[88] Szmuk, M.; Acikmese, B.; Berning, A. W.: Successive Convexification for Fuel-Optimal Powered Landing with Aerodynamic Drag and Non-Convex Constraints. In: AIAA Guidance, Navigation, and Control Conference.

[89] The MathWorks, Inc.: MATLAB, 2024.

[90] Wächter, A.; Biegler, L.; Lang, Y.-d.; Raghunathan, A.: IPOPT: An interior point algorithm for large-scale nonlinear optimization, Jan. 2002.

[91] Wall, M.: SpaceX Successfully Lands Falcon 9 Rocket After Launching Satellites, https://www.space.com/31420-spacex-rocket-landing-success.html, 2015.

[92] Wang, Z.; Lu, Y.: Improved Sequential Convex Programming Algorithms for Entry Trajectory Optimization. Journal of Spacecraft and Rockets, Vol. 57, No. 6, pp. 1373–1386, 2020.

[93] Zolghadri, A.: The challenge of advanced model-based FDIR techniques for aerospace systems: 2011 situation. EUCASS Proceedings Series, Dec. 2013.