POLITECNICO MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE E DELL'INFORMAZIONE

Fast Desensitized Optimal Control for Powered Landing Applications

Tesi di Laurea Magistrale in Space Engineering - Ingegneria Spaziale

Author: **Tommaso Robbiani**

Student ID: 992846
Advisor: Prof. Francesco Topputo
Co-advisor: Dr. Marco Sagliano
Academic Year: 2022-23

Abstract

This research revisits the Desensitized Optimal Control Theory for its application to a computationally challenging numerical benchmark, specifically a descent and landing scenario involving a rocket propulsion system. The primary objective is to assess the efficacy of the proposed method in mitigating the impact of perturbations on the final state, thereby establishing a framework capable of simultaneously optimizing Guidance and Control for the specified case. Additionally, our focus is on formulating a rapid and computationally efficient approach to enhance speed without compromising precision. The investigation begins with a comprehensive analysis of the fundamental components of the method, particularly the sensitivity terms and the computation of feedback gains, with a comparison of alternative formulations to evaluate competitiveness. Subsequently, the application of this methodology to the target problem is thoroughly examined and characterized. Case dependent modifications are introduced to improve the method performances, and the results are critically compared against those obtained using conventional methods through an extensive Monte Carlo analysis campaign.

Keywords: rocket trajectories, powered landing, desensitized optimal control, NLP problems

Sommario

Questa ricerca considera la Teoria del Controllo Ottimale Desensitizzato per la sua applicazione ad un caso di studio computazionalmente oneroso, ovvero uno scenario di discesa e atterraggio per un veicolo con sistema di propulsione a razzo. L'obiettivo principale è valutare l'efficacia del metodo proposto nel mitigare l'impatto delle perturbazioni sullo stato finale, stabilendo così un approccio in grado di ottimizzare simultaneamente la Guida e il Controllo per il problema considerato. Inoltre, ci concentriamo sulla formulazione di un approccio rapido e computazionalmente efficiente per migliorare la velocità di convergenza del metodo del metodo senza comprometterne la precisione. L'indagine inizia con un'analisi approfondita degli elementi fondamentali del metodo, in particolare dei termini di sensitività e dei guadagni di retroazione, con confronto di diverse formulazioni per valutarne la competitività. Successivamente, viene esaminata l'applicazione di questa metodologia al problema specifico e caratterizzata in dettaglio, con variazioni per migliorarne le prestazioni. I risultati sono poi confrontati con quelli ottenuti mediante metodi tradizionali attraverso una approfondita campagna di analisi Monte Carlo.

Parole chiave: traiettorie di razzi, atterraggio con propulsione, controllo ottimo desensitizzato, problemi di programmazione non lineare

Contents

List of Figures

List of Tables

List of Symbols

Mathematical Symbols

$\boldsymbol{x}$	State Vector
$\boldsymbol{f}$	RHS Vector Field
J	Problem Cost Function
$\boldsymbol{c}$	Constraint
$\boldsymbol{h}$	Equality Constraints
$\boldsymbol{g}$	Inequality Constraints
$\boldsymbol{\Lambda}$	Lagrange Multipliers for Dynamics Constraint
$\boldsymbol{\omega}$	Lagrange Multipliers for Boundary Conditions
γ	Lagrange Multipliers for Constraints
$\boldsymbol{y}$	NLP Variables Vector
$\boldsymbol{X}$	Nodal State Values Vector
$\boldsymbol{U}$	Nodal Control Values Vector
Φ	Mayer Cost Term
$\mathcal{L}$	Lagrange Cost Term
L	Lagrangian Function
S	State Sensitivity Matrix
$\boldsymbol{\Lambda}$	Reduced State Sensitivity
$\boldsymbol{\Omega}$	Parameter Sensitivty

Physical Quantities

t	Physical Time	s
τ	Radau Time	s
x	Horizontal Position - 1	m
y	Horizontal Position - 2	m
h	Vertical Position	m
$\boldsymbol{r}$	Position Vector	m
v_x	Horizontal Velocity - 1	m/s
v_y	Horizontal Velocity - 2	m/s
v_h	Vertical Velocity	m/s
$\boldsymbol{v}$	Velocity Vector	m/s
m	Mass	kg
u	Control Norm	[-]
K	Feedback Gain Matrix	[-]
n_T	Number of Thrusters	[-]
γ	Thrusters Pointing Angle	rad
T	Thrust Value	N
I_{sp}	Thrusters Specific Impulse	s
g	Gravity Acceleration	m/s^2

1 | Introduction

1.1. Historical Evolution of Landing Missions

During the Cold War, the United States and the Soviet Union engaged in a space race to assert their dominance in space exploration. The Soviet Luna 9 mission achieved the first soft landing on the Moon in 1966, followed by successful landings on Venus and Mars [1]. In the same period, the United States focused on the Moon, with Surveyor 1 and the historic Apollo 11 mission in 1969. The American Viking 1 probe landed on Mars in 1976 [2].

After the dissolution of the USSR, the Russian program shifted to Venus, while Mars became a research focus for American Entry, Descent, and Landing (EDL) technologies. NASA's progress in EDL allowed the successful landing of the Curiosity rover on Mars with higher precision than in the past.

Space probes and prototypes have provided valuable testing opportunities to enhance landing performance, meeting rigorous mission requirements. The idea of reusable, cost-effective launchers had the potential to revolutionize the space industry, aligning with NASA Administrator Daniel Goldin's vision in 1992 of making missions "faster, better, cheaper" [3].

In 1993, McDonnell Douglas' DC-X achieved the first vertical landing on Earth after a 100-meter hop. NASA embraced the project and conducted further tests with the DC-XA, although it was ultimately dismantled in 1996, marking the end of the vertical landing concept under Goldin's leadership [4].

In 1999, the Japan Aerospace Exploration Agency initiated the Reusable Vehicle Testing campaign, inspired by the DC-XA model, with several flight series from 1999 to 2003 [5].

The first proper space vehicle to achieve vertical landing did not occur until 2015, with Blue Origin's New Shepard reaching 100.5 kilometers in altitude and precisely landing back on the launchpad [6]. A year and a half later, SpaceX achieved satellite orbit insertion using a launcher with a previously flown and refurbished first stage, marking a significant milestone in autonomous vertical landing with reused stages [7].

Rocket reusability gained traction beyond the United States, with various efforts initially focused on small demonstrators for GNC algorithm testing. Examples include EAGLE from DLR [8] and FROG [9] from CNES, both featuring tethered safety systems and turbojet engines for propulsion.

Several projects, including CALLISTO, and RETALT, are contributing to the advancement of reusable launchers. CALLISTO, a trilateral project involving DLR, JAXA, and CNES, aims to demonstrate Return-To-Landing-Site (RTLS) operations with a suborbital rocket [10–13].

The H2020-funded RETALT project, led by Elecnor Deimos, focuses on improving technologies essential for reusable launchers, particularly within the GNC subsystem [14].

1.2. Research Questions and Thesis Structure

In the context of landing strategies, the primary emphasis of the proposed work lies in the powered landing phase, where control is executed using a propulsion system. The main objective is to conduct a preliminary investigation into integrating the fundamental principles of Desensitized Optimal Control (DOC), originally introduced in [15], into the Powered Landing Scenario. Although a similar endeavor was previously undertaken in a study in 2010 [16], the proposed work seeks to expand upon it by applying it within a Nonlinear Programming (NLP) environment, specifically GPOPS [17]. This extension aims to provide a more comprehensive algorithmic characterization. Additionally, in alignment with the title of the work, the secondary goal is to develop a 'Fast' version of the algorithm in terms of CPU time.

As a result, the three key research questions are as follows:

1. What are the true capabilities of DOC techniques applied to the Rocket Landing scenario compared to classical separated (Guidance & Control) architectures?
2. Is it feasible to implement a computationally efficient version of the proposed algorithm in the GPOPS environment?
3. How can the method be characterized and improved to meet prescribed performance requirements?

Without delving into extensive details, Chapter 2 offers an overview of the Optimal Control Problem and the commonly employed numerical strategies. Chapter 3 delves deeply into the Desensitized Optimal Control strategy. Chapter 4 centers on the problem under consideration and how DOC can be applied to address it, while also equipping readers with the tools to assess the impact of various design choices on algorithm efficiency. Chapter 5 presents the obtained results, and Chapter 6 aims to address the research questions while offering insights into potential future developments.

2 | Optimal Control Problem

Optimal Control is a branch of applied mathematics focusing on determining the inputs to a dynamical system that optimize a specified performance index. The resulting solution has to fulfill the dynamical constraints within a certain threshold, while satisfying the prescribed boundary conditions. Various numerical methods have been created to cope with optimal control problems and they could be classified in two major categories, indirect methods and direct methods, that are presented in the latter part of the chapter.

2.1. Problem Formulation

Determine the state, $\boldsymbol{x}(t) \in R^n$, the control, $\boldsymbol{u}(t) \in R^m$, the initial time, $t_0 \in R$, the final time, $t_f \in R$, and the parameters $\boldsymbol{p} \in R^p$ that optimize the performance index:

$$J = \Phi(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f) + \int_{t_0}^{t_f} \mathcal{L}(\boldsymbol{x}(t), \boldsymbol{u}(t), t)\, dt \tag{2.1}$$

Subject to the n dynamic constraints,

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), t) \tag{2.2}$$

the q inequality constraints and the s equality ones, collected in the vector $\boldsymbol{c}$:

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

and the $r = r_i + r_f$ boundary conditions

$$\boldsymbol{\psi}_0(\boldsymbol{x}(t_0), t_0) = \boldsymbol{0} \quad \text{and} \quad \boldsymbol{\psi}_f(\boldsymbol{x}(t_f), t_f) = \boldsymbol{0} \tag{2.4}$$

The state, control, and static parameter can be written also component-wise as:

$$\boldsymbol{x}(t) = \begin{bmatrix} x_1(t) \\ \vdots \\ x_n(t) \end{bmatrix} \qquad \boldsymbol{u}(t) = \begin{bmatrix} u_1(t) \\ \vdots \\ u_m(t) \end{bmatrix}$$

Once defined the problem statement, the explicit dependency with respect to time can be dropped to simplify the notation, i.e. $\boldsymbol{x} = \boldsymbol{x}(t)$ and $\boldsymbol{x}_0 = \boldsymbol{x}(t_0)$.

Equation (2.1) outlines the index J that needs to be minimized along the optimization process, while Eq. (2.2) characterizes the dynamics of the system under analysis. The path constraints, as defined in Eq. (2.3), consist of $q + s$ equations imposing limitations to the system dynamics. These constraints may arise for various reasons (e.g., physical limitations of the system or safety measures). The control and states shall satisfy the boundary conditions, Eq. (2.4), which could be expressed in either vector or scalar form.

The Optimal Control Problem can be formalized as:

$$\min_{u} J \quad \text{s.t.} \begin{cases} \dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t) \\ \boldsymbol{c}(\boldsymbol{x}, \boldsymbol{u}, t) \leq \boldsymbol{0} \\ \boldsymbol{\psi}_0(\boldsymbol{x}_0, t_0) = \boldsymbol{0} \\ \boldsymbol{\psi}_f(\boldsymbol{x}_f, t_f) = \boldsymbol{0} \end{cases} \tag{2.5}$$

Remark 3.1 The formulation presented in Eq. (2.5) is intentionally the most general possible. Since the initial time is typically provided, t_0 can be excluded from the optimization problem variables. In closed final-time problem, the final t_f is also provided.

Remark 3.2 The performance index J of Eq. (2.1) is expressed in the most general Bolza formulation. Equivalently, it can be defined in either Lagrange or Mayer formulation, which, respectively, consider the integral term (path cost) and the term depending on boundary conditions only, respectively. In practical applications, the first term depends typically on the final conditions, so it can be referred to as the terminal cost.

In the upcoming sections, the methodologies for converting the OCP into a TPBVP and into an NLP are presented. Both formulations are useful when solving the OCP, as extensively discussed in Section 2.4.

2.2. OCP as Boundary Value Problem

The transformation of the Optimal Control Problem into a Two-Point Boundary Value Problem (TPBVP) is achieved through the variational approach reported in Section 2.2.1. Initially, the original problem undergoes dualization as shown in Section 2.2.2, by defining an augmented cost function and by incorporating Lagrange Multipliers. Consequently, the variational approach is applied to the cost function, given its functional nature.

2.2.1. Mathematical Background

This sections aims to provide the mathematical fundamentals needed to transform the Optimal control problem into a Boundary Value Problem. The first part focuses on the definition and the proprierties of a Boundary Value Problem, while the second introduces the main concepts of the Calculus of Variations.

Boundary Value Problems

A boundary value problem is a system of differential equations accompanied by conditions specifying the values of the solution and/or its derivatives at two or more points. The number of conditions is given by the product of the order of the differential equations and the number of such equations. Boundary value problems commonly have two relevant features: conditions are applied at two different points, the boundaries of the domain, and the solution is of interest between the two points, inclusive. These problems may have no solution, a unique solution or an infinite number of solutions, in contrast to initial value problems. Boundary value problems are denoted as BVP or TPBVP, two-point boundary value problem, if the boundary conditions are imposed at only two points.

The boundary conditions can be expressed in different mathematical forms leading to different problems. As reported in [18], they could be categorized as Dirichlet, whenever the value of the state is imposed at the boundary, or as Neumann, when the specified value is a derivative of the state. Therefore, a typical TPBVP, a second order system, with Dirichlet boundary conditions, can be defined as:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}(t), t) \qquad \text{s.t} \quad \begin{cases} \boldsymbol{x}(t_0) = \boldsymbol{x_0} \\ \boldsymbol{x}(t_f) = \boldsymbol{x_f} \end{cases}$$

where $\boldsymbol{x} \in R^n$ is the state vector, $t \in R$ is the time, $t_0 \in R$ and $t_f \in R$ are respectively the initial and the final time. $\boldsymbol{x}_0 \in R^n$ and $\boldsymbol{x}_f \in R^n$ are the boundary conditions, while the vector field is defined as $\boldsymbol{f} : R^n \to R^n$.

One key feature of BVPs is their state of posedness: if the problem admits at least one solution (existence), and, provided it has at most one solution (uniqueness) and that the solution depends continuously from the data, the problem is said well-posed. If one of the three reported conditions is not fulfilled, the problem falls into the category of ill-posed BVPs [19]. Given a generic problem, is advisable to reject any ill-posed formulation since the lack of one of the three conditions could respectively lead to the absence of a solution, multiple solutions or large sensitivity to perturbations. If such issue arise, mathematical analysis or regularization techniques should be employed for as intermediate steps before moving to the numerical solving phase.

Calculus of Variations

The calculus of variations is a branch of mathematics highly effective when seeking to minimize or maximize a functional. In order to provide a synthetic and complete overview of the topic, some key definitions from [20] are reported:

Definition 2.2.1: a functional J is a rule of correspondence that assigns to each vector function $\boldsymbol{\gamma}$ in a certain class $\boldsymbol{\Omega}$ a unique real number. $\boldsymbol{\Omega}$ is called the domain of the functional, and the set of real numbers associated with the functions in $\boldsymbol{\Omega}$ is called the range of the functional.

Definition 2.2.2: if $\boldsymbol{\gamma}$ and $\boldsymbol{\gamma} + \boldsymbol{\delta\gamma}$ are functions for which the functional J is defined, then the increment of J, denoted as ΔJ is:

$$\Delta J = J(\boldsymbol{\gamma} + \delta\boldsymbol{\gamma}) - J(\boldsymbol{\gamma})$$

where $\delta\boldsymbol{\gamma}$ is called the variation of the function $\boldsymbol{\gamma}$.

Definition 2.2.3: The increment of a functional can be written as:

$$\Delta J(\boldsymbol{\gamma}, \delta\boldsymbol{\gamma}) = \delta J(\boldsymbol{\gamma}, \delta\boldsymbol{\gamma}) + q(\boldsymbol{\gamma}, \delta\boldsymbol{\gamma}) \cdot ||\delta\boldsymbol{\gamma}||$$

where δJ is linear in $\delta\boldsymbol{\gamma}$. If

$$\lim_{||\delta\boldsymbol{\gamma}|| \to 0} q(\boldsymbol{\gamma}, \delta\boldsymbol{\gamma}) = 0$$

then J is said to be differentiable on $\boldsymbol{\gamma}$ and δJ is the variation of J evaluated for $\boldsymbol{\gamma}$.

The crucial concept is the variation of a functional, that can be interpreted as the first-order approximation of the functional's increment. Of course, if $||\delta\boldsymbol{\gamma}||$ is large, the approximation accuracy may be poor. Given all the building blocks, the Fundamental Theorem of Calculus of Variations can be enunciated.

Theorem 2.1. *Let $\boldsymbol{\gamma}(t)$ be a vector function of the time t and $J(\boldsymbol{\gamma})$ being a differentiable function of $\boldsymbol{\gamma}$. The Fundamental Theorem of Calculus of Variation states: if $\boldsymbol{\gamma}^*$ is an extremal, the first variation of J must vanish on $\boldsymbol{\gamma}^*$ for all admissible $\delta\boldsymbol{\gamma}$.*

$$\delta J\left(\boldsymbol{\gamma}^*, \delta\boldsymbol{\gamma}\right) = 0$$

The theorem presented is a necessary condition in determining the extreme values of functionals, and is analogous to the well known corresponding theorem for functions [20].

2.2.2. Problem Dualization

Once all the mathematical fundamentals are defined, the first step is the dualization process. It consists in augmenting the cost function with the problem's constraints, through Lagrange multipliers.

The augmented cost function of the optimal control problem can be defined as:

$$\begin{aligned} J_a = {} & \Phi(\boldsymbol{x}_0, t_0, \boldsymbol{x}_f, t_f) - \boldsymbol{\omega}_0^T \boldsymbol{\psi}_0(\boldsymbol{x}_0, t_0) - \boldsymbol{\omega}_f^T \boldsymbol{\psi}_f(\boldsymbol{x}_f, t_f) + \\ & \int_{t_0}^{t_f} \mathcal{L}(\boldsymbol{x}, \boldsymbol{u}, t) - \boldsymbol{\lambda}^T(t)\left(\dot{\boldsymbol{x}} - \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t)\right) + \boldsymbol{\gamma}^T(t)\boldsymbol{c}(\boldsymbol{x}, \boldsymbol{u}, t)\, dt \end{aligned} \tag{2.6}$$

where $\boldsymbol{\lambda}(t) \in R^n$, $\boldsymbol{\gamma}(t) \in R^p$, $\boldsymbol{\omega}_0 \in R^{r_i}$, $\boldsymbol{\omega}_f \in R^{r_f}$ are the Lagrange multipliers for Eq. (2.2), Eq. (2.3) and Eq. (2.4) respectively. As stated in the previous sections, the explicit dependence on time of these variables and the functional dependencies drop in favour of a lighter notation.

The first order variation with respect to the free variables can be computed as [21]:

$$\begin{aligned} \delta J_a = {} & \frac{\partial \Phi}{\partial \boldsymbol{x}_0}\delta \boldsymbol{x}_0 + \frac{\partial \Phi}{\partial t_0}\delta t_0 + \frac{\partial \Phi}{\partial \boldsymbol{x}_f}\delta \boldsymbol{x}_f + \frac{\partial \Phi}{\partial t_f}\delta t_f - \delta\boldsymbol{\omega}_0^T \psi_0 - \boldsymbol{\omega}_0^T \frac{\partial \psi_0}{\partial \boldsymbol{x}_0}\delta \boldsymbol{x}_0 - \boldsymbol{\omega}_0^T \frac{\partial \psi_0}{\partial t_0}\delta t_0 \\ & - \delta\boldsymbol{\omega}_f^T \psi_f - \boldsymbol{\omega}_f^T \frac{\partial \psi_f}{\partial \boldsymbol{x}_f}\delta \boldsymbol{x}_f - \boldsymbol{\omega}_f^T \frac{\partial \psi_f}{\partial t_f}\delta t_f + (\mathcal{L} - \boldsymbol{\lambda}^T(\dot{\boldsymbol{x}} - \boldsymbol{f}) - \boldsymbol{\gamma}^T\boldsymbol{c})_{t=t_f}\delta t_f \\ & - (\mathcal{L} - \boldsymbol{\lambda}^T(\dot{\boldsymbol{x}} - \boldsymbol{f}) - \boldsymbol{\gamma}^T\boldsymbol{c})_{t=t_0}\delta t_0 + \int_{t_0}^{t_f} \frac{\partial \mathcal{L}}{\partial \boldsymbol{x}}\delta \boldsymbol{x} + \frac{\partial \mathcal{L}}{\partial \boldsymbol{u}}\delta \boldsymbol{u} - \delta\boldsymbol{\lambda}^T(\dot{\boldsymbol{x}} - \boldsymbol{f}) + \\ & \boldsymbol{\lambda}^T \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\delta \boldsymbol{x} + \boldsymbol{\lambda}^T \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\delta \boldsymbol{x} - \boldsymbol{\lambda}^T \delta\dot{\boldsymbol{y}} - \delta\boldsymbol{\gamma}^T\boldsymbol{c} - \boldsymbol{\gamma}^T \frac{\partial \boldsymbol{c}}{\boldsymbol{x}}\delta \boldsymbol{x} - \boldsymbol{\gamma}^T \frac{\partial \boldsymbol{c}}{\boldsymbol{u}}\delta \boldsymbol{u}\, dt \end{aligned} \tag{2.7}$$

In order to derive the necessary conditions for optimality, all the partial derivatives in this equation with respect to the variables should be set equal to zero and then solved for the different unknowns. However, this is nontrivial due to the unhelpful notation.

2.2.3. Euler Lagrange Equations

In order to obtain a compact formulation, the augmented Hamiltonian can be defined as:

$$H(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\lambda}, \gamma, t) = \mathcal{L}(\boldsymbol{x}, \boldsymbol{u}, t) + \boldsymbol{\lambda}^T \cdot \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t) - \boldsymbol{\gamma}^T \cdot \boldsymbol{c}(\boldsymbol{x}, \boldsymbol{u}, t) \tag{2.8}$$

By merging Eq. (2.6) and Eq. (2.7), the first-order optimality condition, known as Euler-Lagrange equations, for the optimal control problem result in:

$$\begin{cases} \dot{\boldsymbol{x}}^T(t) = \dfrac{\partial H}{\partial \boldsymbol{\lambda}} \\ \dot{\boldsymbol{\lambda}}^T(t) = -\dfrac{\partial H}{\partial \boldsymbol{x}} \\ \boldsymbol{0} = \dfrac{\partial H}{\partial \boldsymbol{u}} \\ \boldsymbol{\lambda}^T(t_0) = -\dfrac{\partial \Phi}{\partial \boldsymbol{x}_0} + \boldsymbol{\omega}_o^T \dfrac{\partial \psi_0}{\partial \boldsymbol{x}_0} \\ \boldsymbol{\lambda}^T(t_f) = -\dfrac{\partial \Phi}{\partial \boldsymbol{x}_f} + \boldsymbol{\omega}_f^T \dfrac{\partial \psi_f}{\partial \boldsymbol{x}_f} \\ H|_{t=t_0} = -\boldsymbol{\omega}_0^T \dfrac{\partial \boldsymbol{\psi}_0}{\partial t_0} + \dfrac{\partial \Phi}{\partial t_0} \\ H|_{t=t_f} = \boldsymbol{\omega}_f^T \dfrac{\partial \boldsymbol{\psi}_f}{\partial t_f} - \dfrac{\partial \Phi}{\partial t_f} \\ \boldsymbol{\psi}_0 = \boldsymbol{0} \\ \boldsymbol{\psi}_f = \boldsymbol{0} \end{cases} \tag{2.9}$$

A further outcome of this procedure is that the $\boldsymbol{\gamma}$ components linked to the inequality constraints are equal to zero when the corresponding complementary condition is inactive. This means that inactive path constraints are effectively disregarded in the cost function.

Remark 3.3 The equations presented define the most general set of necessary conditions that must be satisfied for an extremal point of the optimal control.

Remark 3.4 Boundary conditions for states may be partially defined; any missing conditions are substituted by constraints on the corresponding costate. For example, if a state is set free at the final time, the corresponding costate will be fixed and equal to zero at final time.

Remark 3.5 If the cost function is expressed in Lagrange form, i.e., $\Phi = 0$, the initial time and final time are fixed, leading to $\delta t_i = 0$ and $\delta t_f = 0$, as well as the initial and final conditions on the state, $\boldsymbol{x}(t_0) = \boldsymbol{x}_0$ and $\boldsymbol{x}(t_f) = \boldsymbol{x}_f$, some terms vanish. The boundary conditions for $\boldsymbol{\lambda}(t)$ disappear as well as the ones for H. The initial and final state constraints are reformulated, since the boundary values for the state are fixed. The resulting system of equations reads as follows:

$$\begin{cases} \dot{\boldsymbol{x}}^T(t) = \dfrac{\partial H}{\partial \boldsymbol{\lambda}} \\ \dot{\boldsymbol{\lambda}}^T(t) = -\dfrac{\partial H}{\partial \boldsymbol{x}} \\ \boldsymbol{0} = \dfrac{\partial H}{\partial \boldsymbol{u}} \\ \boldsymbol{x}(t_0) = \boldsymbol{x}_0 \\ \boldsymbol{x}(t_f) = \boldsymbol{x}_f \end{cases} \tag{2.10}$$

Remark 3.6 The Euler-Lagrange conditions constitute a system of differential algebraic equations and also a TPBVP, with an augmented state comprising both $\boldsymbol{x}$ and $\boldsymbol{\lambda}$. An approach to treat the problem is the the Pontryagin's Maximum Principle, presented in the subsequent section, needed to express the control actions as functions of state and costate in order to obtain a TPBVP formulation, then solved.

2.2.4. Pontryagin's Maximum Principle

The Pontryagin's Maximum Principle (PMP) is employed to determine the optimal values of the control variables. Even if it has been developed for maximization problems [22] it can be extended to minimization ones, with the acronym PmP.

As reported in Remark 3.6, the third equation of Eq. (2.10) is used to calculate the external actions needed to fulfill the boundary conditions and the path constraints. The key issue is that the applicable control is not infinite and, therefore, it belongs to a finite feasible region, defined as U.

The weak form of the Pontryagin's principle is stated in [22] as: Find $\boldsymbol{u}^*$ such that:

$$\boldsymbol{u}^* = \min_{u \in U} H(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\lambda}, \boldsymbol{\gamma}, t)$$

Remark 3.7 In Euler-Lagrange equations, the partial derivative of H with respect to $\boldsymbol{u}$ is set to zero, therefore PmP can be interpreted as a constrained minimization over a set of the control actions.

2.3. OCP as Nonlinear Programming Problem

The alternative approach involves discretizing the continuous-time optimal problem, through a process named transcription. The resultant problem is then treated as a NLP problem, as discussed Section 2.3.1, wherein the optimization parameters include the components of the discretized state and control, as well as the initial and final times.

2.3.1. Mathematical Background

As already done for BVP, a brief discussion of necessary mathematical tools is presented in this section. The main goal is the definition of Nonlinear Programming, together with its optimality conditions, the KKT conditions.

Nonlinear Programming

The nonlinear programming (NLP) problem, also referred to as parametric (or static) optimization, involves finding a finite set of variables that minimizes a cost function, provided that the all the constraints are fulfilled. Multiple subsets of problems are included in the general NLP problem category, such as linear and quadratic programming [23].

The general nonlinear programming (NLP) problem can be stated as follows.
Find the vector $\boldsymbol{x} \in R^n$ that minimizes the scalar objective function $J(\boldsymbol{x})$, subject to equality constraints $\boldsymbol{h} : R^n \to R^s$ and inequality constraints $\boldsymbol{g} : R^n \to R^q$, such that:

$$\min_{\boldsymbol{x}} J(\boldsymbol{x}) \qquad \text{subject to} \begin{cases} \boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{0} \\ \boldsymbol{g}(\boldsymbol{x}) \leq \boldsymbol{0} \end{cases}$$

For every examined state, the active set of inequality constraints can be defined as:

$$\boldsymbol{c}_A(\boldsymbol{x}) : \{\boldsymbol{g}_i(\boldsymbol{x}) \mid \boldsymbol{g}_i(\boldsymbol{x}) = 0, \ \ i \in 1, \dots, n\}$$

The constraints belonging to $\boldsymbol{c}_A$ are treated as equality constraints. They can be collected with $\boldsymbol{h}(\boldsymbol{x})$ in the constraint vector, defined as $\boldsymbol{c}(\boldsymbol{x}) = \left[\boldsymbol{h}(\boldsymbol{x}); \ \ \boldsymbol{c}_A(\boldsymbol{x})\right]$

Numerous methods have been defined for solving NLP problems, broadly categorized in Gradient Methods and Heuristic Optimization Methods. Gradient methods aim to determine the best search direction and the optimal step length seeking for a local minimum of the problem, when an initial guess is provided. Commonly employed techniques include SQP (Sequential quadratic programming) or IP (Interior Point). In contrast, Heuristic

Methods research is performed in a stochastic manner, in contrast with the deterministic strategy of Gradient Methods. A frequently utilized method is the genetic algorithm [24].

Karush-Kuhn-Tucker Conditions

The necessary optimality conditions for an NLP problem are called Karush-Kuhn-Tucker conditions [25]. They must be satisfied for a solution to be optimal in a constrained optimization problem. In practice, solving optimal control problems often requires numerical techniques and optimization algorithms that take into account these conditions to find the best control strategies while respecting system dynamics and constraints.

They can be retrieved from the definition of the problem's Lagrangian. The Lagrangian is formed augmenting the cost function with the the vectors of Lagrangian multipliers, denoted as $\hat{\boldsymbol{\lambda}} \in R^{s+M\cdot n}$ and $\hat{\boldsymbol{\nu}} \in R^{q}$. Thus, the Lagrangian is expressed as:

$$L(\boldsymbol{y}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = \hat{J}(\boldsymbol{y}) + \hat{\boldsymbol{\lambda}}^T \hat{\boldsymbol{h}}_a(\boldsymbol{y}) + \hat{\boldsymbol{\nu}}^T \hat{\boldsymbol{g}}(\boldsymbol{y})$$

As stated in [26], if $\boldsymbol{y}^*$ is a local minimum of the problem, then exist unique vectors $\hat{\boldsymbol{\lambda}}^*$ and $\hat{\boldsymbol{\nu}}^*$ such that:

$$\begin{cases} \nabla_{\boldsymbol{y}} L = \boldsymbol{\nabla} \hat{\boldsymbol{J}}(\boldsymbol{y}^*) + \boldsymbol{G}(\boldsymbol{y}^*)^T \hat{\boldsymbol{\lambda}}^* + \boldsymbol{H}_a(\boldsymbol{y}^*)^T \hat{\boldsymbol{\nu}}^* = \boldsymbol{0} \\ \nabla_{\hat{\boldsymbol{\lambda}}^*} L = \hat{\boldsymbol{h}}(\boldsymbol{y}^*) = \boldsymbol{0} \\ \nabla_{\hat{\boldsymbol{\nu}}^*} L = \hat{\boldsymbol{g}}(\boldsymbol{y}^*) \leq \boldsymbol{0} \\ \hat{\boldsymbol{\nu}}^* L \geq \boldsymbol{0} \\ \hat{\boldsymbol{\nu}}^{*T} \hat{\boldsymbol{g}}(\boldsymbol{y}^*) = \boldsymbol{0} \end{cases}$$

where $\boldsymbol{G}$ and $\boldsymbol{H}_a$ are, respectively, the equality and inequality constraint Jacobians.

2.3.2. Problem Transcription

The process of transcription aims at transforming the continuous (or infinite-dimensional) optimal control problem into a discrete (or finite-dimensional) optimization problem. While various methodologies exist for this purpose, this work focuses on two specific approaches: the nodal approach and the modal approach. The nodal approach, presented in this section, employs nodal values of state and control as variables in the NLP problem. On the other hand, the modal approach approximates all the involved quantities as polynomials, whose coefficients are the NLP variables. The former strongly relies on the use of Lagrange polynomials, thanks to the isolation propriety, presented in Section 2.4.2.

The first step of the nodal approach involves discretizing the state and the control variables, passing from the continuous domain to the discrete one. The time span of the problem, from t_0 to t_f, is divided in N nodes, t_i for $i = 1, \dots, N$. The states are evaluated in the same time instants, $\boldsymbol{x}_i = \boldsymbol{x}(t_i)$ for $i = 1, \dots, N$, while the controls can be evaluated at N_u other nodes, depending on the transcription method adopted. This operation leads to a NLP state vector defined as:

$$\boldsymbol{y} = \left[\boldsymbol{x}_1, \dots, \boldsymbol{x}_N, \boldsymbol{u}_1, \dots, \boldsymbol{u}_N, t_0, t_f\right]^T = \left[\boldsymbol{X}^T, \boldsymbol{U}^T, t_0, t_f\right]^T$$

in vector form to avoid the loss of generality and with length, $w = N \cdot n + N_u \cdot m + 2$.

The dynamics undergoing transcription become a set of algebraic nonlinear equalities, meaning that they are evaluated ad a fixed number of instants (M), not necessarily coincident with the state/control discretization vector. The result is a series of defect constraints, which are imposed on each interval in the discretization [27].

$$\boldsymbol{\xi}_j = \dot{\boldsymbol{x}}(t_j) - \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_j, t_j) = 0 \text{ for } j = 1, \dots, M \tag{2.11}$$

The objective function, Eq. (2.1), Mayer term is is evaluated at the initial and/or final nodes, while the path cost is approximated through a discrete integration scheme whose coefficients depend on the selected quadrature scheme. The approximated cost is $\hat{J}$.

$$\hat{J}(\boldsymbol{y}) = \Phi(\boldsymbol{x_1}, t_0, \boldsymbol{x_N}, t_N) + \sum_{i=1}^{N} w_i L(\boldsymbol{x}_i, \boldsymbol{u}_i, t)$$

The equality and inequality constraints must to be redefined, and they are represented as: $\hat{\boldsymbol{h}}_a : R^w \to R^{s+M \cdot n}$ and $\hat{\boldsymbol{g}} : R^w \to R^q$. The equality constraints vector is comprised by two different parts: the first part corresponds to the discretization of the optimal control problem's equality constraints augmented with the defect constraints, considered as equality additional equality constraints.

The transcripted NLP problem can be formulated as:

$$\min_{y \in R^w} \hat{J}(\boldsymbol{y}) \quad \text{s.t} \quad \begin{cases} \hat{\boldsymbol{h}}_a(\boldsymbol{y}) = \boldsymbol{0} \\ \hat{\boldsymbol{g}}(\boldsymbol{y}) \leq \boldsymbol{0} \end{cases}$$

The first-order necessary optimality conditions for the reported problem are already shown in Section 2.3.1, with a slightly different notation, but can be easily extended to the problem under analysis.

2.4. Numerical Methods

Upon formulating the OCP as TPBVP or as NLP problem, numerical methods are essential to solve it. With the exception of simple problems, optimal control problems must be solved numerically. In the first part of this section typical numerical methods are presented, while the second emphasizes on Pseudospectral methods (PSM), due to their significance in this work.

2.4.1. General Overview

The numerical methods can be divided into two categories: indirect methods and direct methods.

Indirect methods are based on the TPBVP formulation (refer to Section 2.2), since they exploit Euler-Lagrange necessary conditions Eq. (2.9). Examples of indirect methods include the Indirect Single-Shooting Method (ISSM) and Indirect Multiple-Shooting Method (IMSM) [27]. The fundamental idea behind these methods is to make an initial guess of the unknown boundary conditions at one end of the interval and then perform a numerical integration to the other end. Different schemes can be employed for propagation, classified as one-step methods and multi-step methods [23].

If the result of the integration deviates from the corresponding boundary conditions of more than a prescribed value, corrections to the guess are applied iteratively until the convergence is achieved. Various strategies, such as the State Transition Matrix technique, can be employed to facilitate convergence with the differential corrections method [28].

The primary distinction between ISSM and IMSM is that the latter discretize the trajectory in multiple segments. Since Hamiltonian dynamic evolution is typically ill-conditioned, reducing the time span mitigates the sensitivity of the problem to errors in initial conditions. However, this comes at the cost of an increased number of variables, and additional boundary conditions between segments are required to ensure the continuity of the state.

Another indirect method is the so called Indirect collocation method, where state and costate are parameterized using piece-wise polynomials. This procedure leads from a TPBVP to a root-finding problem, where the state is given by the coefficients of the polynomials [27].

Direct methods operate with a fundamental different Philosophy, compared to indirect methods. The core concept is to transform the OCP into a NLP, as illustrated in Section 2.3. These methods offer two significant advantages: they can be applied without explicitly deriving necessary conditions, and they do not require a priori specification of the arc sequence for problems with path inequalities [27]. The main distinction from indirect methods lies in the fact that direct methods build a series of points to minimize the objective function, whereas the indirect methods seek to find a root of the necessary conditions [23].

This category includes the Direct Single-Shooting Method (DSSM) and Direct Multiple-Shooting Method (DMSM). In these methods, the control is parameterized through a specified functional form, as expressed in:

$$\boldsymbol{u}(t) \approx \sum_{i=1}^{m} \boldsymbol{a}_i P_i(t) \tag{2.12}$$

where $P_i(t)$ are known functions and $\boldsymbol{a}_i$ are the parameters to be determined in the optimization process. The dynamic constraints are satisfied by integrating differential equations with a time-marching algorithm, and the cost function is approximated through a quadrature formula, as reported in Eq. (2.11). Although the sensitivity issue does not play a crucial role, it remains a challenge for direct methods [27].

In DMSM, the time interval is divided into $N + 1$ sub-intervals and the DSSM is applied in each interval, to minimize the sensitivity to initial conditions given the integration is performed over smaller time intervals [24]. However, both of these methods may encounter challenges when the control cannot be easily parameterized, due to the increment of NLP variables or the complexity of the discretization.

In line with Indirect Methods, a Direct collocation methods (DCM) can be formulated, and this family of methods holds significance within the broader panorama of Direct Methods. The fundamental idea is that the state and the control are approximated using a specified functional form [24]. Collocation can be performed locally or globally. The former is based on the discretization of the time interval into sub-intervals where local polynomial approximations is employed, while the latter aims to approximate the state and the control functions across the entire domain. Commonly used global collocation methods are the Pseudospectral Methods, that will be discussed in the next section.

A notable challenge associated with these methods is the potential largeness of the number of variables. In real applications, the relevant matrices of this NLP problem - specifically, the Jacobian and the Hessian - are typically sparse. Exploiting sparsity to reduce storage and computation time is a critical aspect for the numerical implementation. Details on addressing sparsity issues can be found in [27]. These problems are typically solved with commercial solvers, like SNOPT [29], SPRNLP [30] and KNITRO [31].

2.4.2. Pseudospectral Methods

Pseudospectral methods represent a category of indirect collocation methods based on orthogonal polynomials, with commonly used basis functions being Chebyshev or Lagrange polynomials. Their relevant advantage over traditional methods lies in their spectral convergence, i.e., they converge exponentially with respect to the number of collocation points. Therefore, even a coarse grid is sufficient to ensure a rapid convergence of the solution. The subsequent sections provide a brief description of the method, followed by an exploration of different types of Legendre nodes.

Introduction and Motivations

In practical applications, the state and/or the control could be approximated by different polynomial bases, in the form reported in Eq. (2.12). Two commonly employed bases are either Legendre polynomials or Chebyshev polynomials, both expressed in Lagrange form.

Lagrange polynomials offer a notable advantage, since the coefficients $\boldsymbol{a}_i$ in Eq. (2.12) are the nodal values of the original function. This property, known as the isolation property, implies that the parameters vector of the Nonlinear Programming (NLP) problem comprises the nodal values of the state and/or control from the original problem.

The Lagrange polynomial approximation is based on the fact that, given a set of N time nodes, $t_1, \ldots, t_N$ on a time interval $[t_0, t_f]$ and a continuous function of time $y(t)$, defined on the same interval, there exists a unique polynomial $Y(t)$, of degree $N-1$, such that:

$$Y(t_i) = y(t_i) = y_i \quad \text{for } i = 1, \ldots, N \tag{2.13}$$

The polynomial approximation formula is given by:

$$Y(t) = \sum_{i=1}^{N} a_i P_i(t)$$

where $P_i(t)$ are the Lagrange polynomials, defined as:

$$P_i(t) = \prod_{k=1, k\neq i}^{N} \frac{t - t_k}{t_i - t_k} \tag{2.14}$$

The already cited insulation problem states that:

$$P_i(t_k) = \delta_{ik} = \begin{cases} 1 & \text{if } i = k \\ 0 & \text{if } i \neq k \end{cases}$$

as a consequence, coherently with Eq. (2.13), $a_i = y_i$ for $i = 1, \dots, N$.

The Lagrange approximation is exact for functions of order up to $N-1$. The error between Lagrange approximation and the initial function can be computed as [32]:

$$y(t) - Y(t) = \frac{(t-t_1)\dots(t-t_N)}{N!} \frac{d^N}{dt^N} y(\xi)$$

where $\xi \in [t_0, t_f]$. Notably, if $t = t_i$, the numerator goes to zero, consistent with Eq. (2.13). Additionally, for functions of order up to $N-1$, the derivative vanishes, ensuring exact Lagrange approximation.

The central challenge is the potential occurrence of the Runge phenomenon for evenly distributed points. Intuitively, one might expect that increasing the number of nodes, denoted by N, would enhance the approximation quality, thereby reducing the error. However, in certain cases, the opposite effect takes place - increasing nodes may lead to a rise in error. High-degree polynomials can, in some instances, result in ill-conditioned problems, particularly towards the outer part of the interval [33].

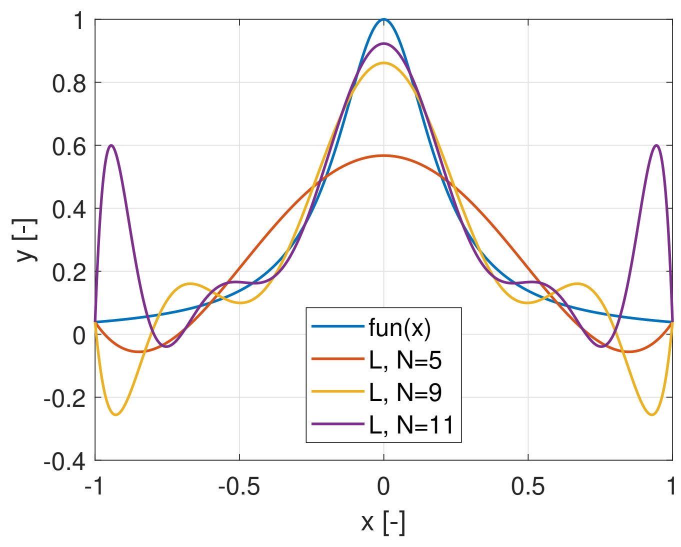


Figure 2.1: Runge Phenomenon for Lagrange Interpolation with N nodes

Consequently, oscillations emerge in the approximating functions between nodes, particularly at the ends of the time interval, as shown in Section 2.1, reporting the following functions and its polynomial interpolations:

$$\text{fun}(x) = \frac{1}{1+25x^2} \tag{2.15}$$

As N increases, the magnitude of these oscillations grows, exacerbating the discretization error.

In order to avoid the Runge phenomenon, discretization points must be properly chosen between the non-uniformly distributed nodes families. A non-uniform discretization can be obtained as roots of orthogonal polynomials, like Chebyshev or Legendre polynomials. Expansions of functions in terms of orthogonal polynomials are easy to manipulate, have good convergence proprieties and give a well-conditioned representation of the function [33].

Moreover, these specific collocation points guarantee the exponential convergence of the method as the number of points increases in case of smooth problems. This is the key advantage of using them as nodes, and it is the key feature of pseudospectral methods. Further analyses on the different types of points that could be employed are reported in Section 2.4.2.

Global collocation Since Legendre polynomials are orthogonal in the domain $\tau \in [-1,1]$, the pseudospectral methods refers to this domain as pseudospectral time. The mapping between the physical time $t \in [t_0, t_f]$ and the pseudospectral time is:

$$t = \frac{t_f - t_0}{2}\tau + \frac{t_f + t_0}{2}, \quad \tau \in [-1,1]$$

while the opposite one is:

$$\tau = \frac{2}{t_f - t_0}t - \frac{t_f + t_0}{t_f - t_0}, \quad t \in [t_0, t_f]$$

Assuming that the domain is discretized in N nodes, t_i for $i = 1, \dots, N$ the state $\boldsymbol{x}(t)$ with n elements, can be approximated by the sum of N Lagrange polynomials:

$$\boldsymbol{x}(t) \approx \sum_{i=1}^{N} \boldsymbol{X}_i P_i(t)$$

where L_i are the Lagrange Polynomials defined in Eq. (2.14) and $\boldsymbol{X}_i$ are the nodal values of the state. For pseudospectral methods, the number of nodes does not necessarily coincides with the number of collocation points (M), i.e., where the dynamics is evaluated. The collocation points are defined as τ_j for $j = 1, \dots, M$. The pseudospecral differential matrix $D \in R^{n \cdot N \times n \cdot M}$ can be defined as:

$$D_{j,i} = \frac{\partial \dot{\boldsymbol{X}}_j}{\partial \boldsymbol{X}_i} = \frac{dP_i}{d\tau}(\tau_j)\boldsymbol{I}_{n_s} = \frac{t_f - t_0}{2}\dot{P}(\tau_j)\boldsymbol{I}_{n_s} \tag{2.16}$$

If the control vector $\boldsymbol{U}_j = \boldsymbol{u}(t_j) \in R^u$ is considered, the problem is be formulated as:

$$D_j \boldsymbol{X} = \frac{t_f - t_0}{2}\boldsymbol{f}(\boldsymbol{X}_j, \boldsymbol{U}_j)$$

where D_j stands for the j$^{\text{th}}$ column of D matrix and $\boldsymbol{X} \in R^{n_s N_n}$ is the vector of all the states at all the nodes. In this way the dynamics is properly collocated at M points.

The Lagrange cost term $L(\boldsymbol{x}, \boldsymbol{u}, t)$ can be approximated as:

$$\int_{t_0}^{t_f} L(\boldsymbol{x}, \boldsymbol{u}, t)\, dt \approx \frac{t_f - t_0}{2}\sum_{i=1}^{N_n} \omega_i L(\boldsymbol{X}_i, \boldsymbol{U}_i, t_i) \tag{2.17}$$

where ω_i is the quadrature weight.

Quadrature formulas relying on evenly spaced points often fail to converge for many cases. Therefore, two alternative approaches can be considered: composite rules or Gaussian quadrature [34]. Gaussian quadrature coefficients, based on Legendre polynomials, offer high accuracy owing to their orthogonality properties. Thus, it is reasonable to adopt the integration scheme corresponding to the collocation points used. Each of these schemes has a corresponding set of quadrature coefficients.

An alternative perspective is to define the integral operator instead of the differential operator [35]. While in some methods, both procedures yield the same result, in others, they may differ. Further details are provided in the following sections.

Local Collocation Local collocation, also known as the *hp*-method, is as an extension of the global collocation method. According to [36], global collocation methods may demand larger CPU times compared to standard methods. This is attributed to the need for a large number of nodes to achieve precise function approximations, resulting in a non-sparse Jacobian matrix. To address this issue, the local collocation method divides the time interval into h segments. In each segment, an equivalent global collocation method, with order p, is employed.

The primary distinction from global methods lies in the introduction of linking conditions to ensure the continuity of the state, control, and time across the intervals. For a given interval $i \in 2, \ldots, h$, the linking conditions are:

$$\begin{cases} t_p^{i-1} = t_0^i \\ \boldsymbol{X}_p^{i-1} = \boldsymbol{X}_0^i \\ \boldsymbol{U}_p^{i-1} = \boldsymbol{U}_0^i \end{cases}$$

where the index p indicates the last node in each segment.

Legendre Points

Pseudospectral methods utilize the roots of Legendre polynomials or roots of linear combination of Legendre polynomials of different order as nodes. The Legendre polynomial of order N is:

$$L_N(\tau) = \frac{1}{2^N N!} \frac{d^N}{d\tau^N} (\tau^2 - 1)^N.$$

This works considers three different families of nodes: Legendre-Gauss (LG) points, also called Gauss points, Legendre-Gauss-Lobatto (LGL) points, also called Lobatto points, and Legendre-Gauss-Radau (LGR) points, also called Radau points [35]. A brief description is provided in the next paragraphs, while the rationale behind choosing Radau nodes is explained in the following section.

Gauss Points - LG These points exclude both endpoints. They are the roots of $P_n(\tau)$; as they do not include the ends, $\tau_1 > -1$ and $\tau_N < 1$. To obtain a state discretization, a node $\tau_0 = -1$ can be added, extending the range from $i = 0$ to $i = N$. Since the number of collocation points differs from the number of discretization points, the resulting differential matrix is non-square but is equivalent to the integral one.

Lobatto Points - LGL They include both endpoints and are the roots of $\dot{P}_{N-1}(\tau)$ plus the points $[-1, +1]$. Since they include both extremes, the first point is $\tau_1 = -1$, while the last is $\tau_N = +1$. In this case, the differentiation matrix is square since, but it is singular. The differential matrix and the integral one are not equivalent, and the linear system of equations for costate dynamics is under-determined, leading to the lack of correspondence in optimality conditions. The differential matrix has a non-empty null space, resulting in the existence of an infinite number of solutions for the costates values.

Radau Points - LGR They include one of the endpoints and are asymmetric with respect to the origin. They can be defined either using the initial point (LGR) or the final point in the flipped configuration (f-LGR points). They are the roots of $P_{N-1}(\tau)+P_N(\tau)$. In the nominal case, $\tau_1 = -1$ and $\tau_N < 1$, but the node $\tau_{N+1} = 1$ is added, not as collocation points. The differentiation matrix is non singular and, as for LG, is equivalent to the integral one.

Radau Method

The Lobatto pseudospectral method (LPM), based on Lobatto nodes, and the Gauss pseudospectral methods (GPM), based on the Gauss nodes, have been deeply investigated [34]. However, both of them are suboptimal selections.

The LPM strategy might seem optimal due to its architecture, with both endpoints collocated. The problem is that the optimality conditions of the NLP are not equivalent to the discretized form of the continuous-time optimality conditions. In particular, a poor estimation of LPM is obtained [34].

On the contrary, although the optimality conditions of the Gauss pseudospectral methods (GPM) in NLP are equivalent to the discretized form of the continuous-time optimality conditions, the Gauss points lack collocation at both the initial and final points. Consequently, the control at the initial point cannot be obtained from the NLP solution.

Moreover, when utilizing an *hp* pseudospectral method, where the terminal point of one sub-interval aligns with the initial point of the following one, the inclusion of both endpoints in LGL points results in a redundancy in the number of variables. In contrast, the GPM architecture faces challenges due to the absence of boundary conditions.

Considering these motivations, the LGR points result to be the optimal choice. Their NLP optimality conditions are equivalent to the discretized version of the continous-time ones and, since they have just one endpoint, no redundancy in the *hp* scheme is present. The only drawback is that the final/initial control is not obtained for nominal/flipped configuration [34], but can be retrieved by interpolation. Furthermore, the implementation of Radau pseudospectral method (RPM) is less complex than the GPM one.

Given a M collocation points, the number of nodes is $N = M + 1$. Therefore, the differential operator of Radau $D \in R^{n \cdot M \times n \cdot N}$ is a non-square matrix. In case of RPM, according to equation Eq. (2.16), $j \in [1, N]$ and $i \in [1, N+1]$ while for fRPM $\in [1, N]$ and $i \in [0, N]$. This is because in RPM the τ_{N+1} discretization points, while in fRPM the τ_0 [37].

The quadrature weights of Eq. (2.17) can be defined as:

$$\omega_i = \frac{1}{(1-\tau_i)[\dot{L}_{N-1}(\tau_i)]^2} \quad \text{for } i = 2, \dots, N \qquad \text{with } \omega_1 = \frac{2}{N^2}.$$

2.4.3. Numerical Implementation

Once the OCP problem has been reformulated as an NLP, the application of a numerical method is required. In recent years, some important features of the practical implementation have been considered and analyzed. This section aims to provide a brief overview as the most relevant aspects that need to be taken into account.

Differentiation Schemes Given that common solvers like SNOPT or IPOPT rely on gradient based methods, selecting an optimal differentiation scheme is crucial to minimize numerical errors. Analytical derivation is the most accurate but often impractical for complex problems [24]. However automatic differentiation [38] could produce a good alternative, as it aims to provide a quasi-analytical derivation tool. The ADiGator toolbox is properly integrated with the newest versions of GPOPS-II, the software used in this work. Another approach to address this problem is based on dual numbers theory. Although free from truncation and round-off errors this method is limited to analytical functions only and cannot handle tabular data. Moreover, the number of operations to be performed increases, introducing additional computational complexity due to the underlying mathematical rules [39].

Sparsity In direct collocation methods, the majority of the Jacobian terms are zero and the number of variables could be very large. Therefore, to efficiently handle this, matrices must be stored in sparse form [27]. As highlighted in [23], various solutions have been implemented to mitigate this issues. Many commercial software, like the aforementioned IPOPT, take advantage of the sparsity proprieties to minimize the needed memory.

Scaling In order to minimize the computational effort, the problem could be scaled. The normalization process could be based on linear or non-linear methods, and involves scaling the states, the costates, the dynamics, the constrains and the cost function. Various techniques for scaling are reported in [37]. The efficiency of the process can be evaluated by considering the ratio between the condition number of the non-scaled Jacobian and the condition number of the scaled one [40]. A higher value for this ratio indicates better performance of the scaling procedure.

3 | Desensitized Optimal Control

The theory of Desensitized Optimal Control (DOC) was initially introduced in [15]. This theory serves as a comprehensive approach for designing robust trajectories and controllers in a unified manner. It belongs to the category of sensitivity analysis methods, as it involves the definition of a sensitivity matrix denoted as S. The primary objective of DOC is to enhance the robustness of the optimal solution in the face of perturbations related to the state, control, or parameters. This approach provides a strategy for mitigating the effects of uncertainties and disturbances on an optimal trajectory while simultaneously ensuring the optimization of both guidance and state feedback control gains.

In most applications, Guidance and Control (G&C) are treated as separate components. First, the trajectory is optimized, and then an appropriate controller is synthesized offline in response to the optimized trajectory. However, simply combining two separately optimized sub-solutions does not always yield the optimal solution for the overall problem. DOC theory has the potential to simultaneously address both processes, achieving a synchronous optimization. The degree of correlation between these two processes can vary, as shown in subsequent sections.

The proposed solution represents a departure from the conventional paradigm, where robustness is an attribute expected from the controller. In DOC, robustness is inherently provided by the trajectory design, and the controller plays a role in supporting and enhancing this robustness.

3.1. Sensitivity Definition

Before delving into the core concept of DOC, it is essential to provide an overview of various sensitivity definitions. Sensitivity is essentially a means to measure how changes in one quantity can affect another. In this context, sensitivity is used to evaluate how perturbations in the state or parameters impact other states. These definitions correspond to the sensitivity matrix and sensitivity function, both of which will be examined in depth.

In general, sensitivity serves as a tool for predicting the effects of perturbations. However, this prediction is most accurate when working under a fundamental assumption: linearity. As demonstrated in the forthcoming sections, the dynamics of the sensitivity matrix inherently follow a linear pattern. In cases where the actual dynamical system is nonlinear, the prediction may become less precise. Furthermore, substantial perturbations applied to a nonlinear system could potentially worsen the accuracy of sensitivity predictions.

3.1.1. Sensitivity Matrix

Given a generic dynamic system in the form:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}(t), t) \tag{3.1}$$

and subject to the initial condition $\boldsymbol{x}(t_0) = \boldsymbol{x}_0$, the sensitivity matrix $S(t, t_0, \boldsymbol{x}_0)$ can be defined, according to [15], as:

$$S(t, t_0, \boldsymbol{x}_0) = \frac{\partial}{\partial \boldsymbol{x}_0} \boldsymbol{X}(t, t_0, \boldsymbol{x}_0) \tag{3.2}$$

where $\boldsymbol{X}(t, t_0, \boldsymbol{x}_0)$ is the state at time t obtained via propagation of Eq. (3.1) the from the initial conditions t_0 and $\boldsymbol{x}_0$.

S matrix can be seen as a state transition matrix, but, according to Eq. (3.2), it is able to map the sensitivity of the state with respect to perturbations on the initial conditions. Its dynamics is governed by the following differential equation:

$$\frac{\partial S(t, t_0, \boldsymbol{x}_0)}{\partial t} = \frac{\partial \boldsymbol{f}(\boldsymbol{x}, t)}{\partial \boldsymbol{x}}\bigg|_{\boldsymbol{x}=\boldsymbol{x}(t)} \cdot S(t, t_0, \boldsymbol{x}_0) = A \cdot S(t, t_0, \boldsymbol{x}_0) \tag{3.3}$$

having as initial condition $S(t_0, t_0, \boldsymbol{x}_0) = I$, with I the identity matrix of consistent size. To keep the notation coincise, from now on the matrix $S(t, t_0, \boldsymbol{x}_0)$ can be also addressed as $S(t)$.

Due to the mathematical properties of the sensitivity matrix, the following mathematical proprierties can be shown:

$$S(t, t_0, \boldsymbol{x}_0)^{-1} = S(t_0, t, \boldsymbol{X}(t, t_0, \boldsymbol{x}_0)) \tag{3.4}$$

$$S(t, t_0, \boldsymbol{x}_0) = S(t, t_1, \boldsymbol{X}(t_1, t_0, \boldsymbol{x}_0)) \cdot S(t_1, t_0, \boldsymbol{x}2_0) \tag{3.5}$$

This concept of sensitivity holds its own significance, but its importance becomes more relevant when considered within the framework of optimal control. When dealing with optimal control problems, one objective may involve minimizing the impact of perturbations on the state at a specific time, denoted as $S(\tilde{t})$. The matrix $S(\tilde{t})$ encapsulates this information in terms of sensitivities, and the problem can be formulated to penalize it.

Nevertheless, in many practical applications, the sensitivity of the final state can have seminal importance. For instance, in a landing scenario, the mission's success is determined more by the accuracy of the final state rather than that of a generic state at the generic time $\tilde{t}$. With the previously mentioned definition, the sensitivity of the final state can be expressed in two distinct manners.

The first one is trivial and relies on the evaluation of the sensitivity matrix defined in Eq. (3.2) at the final time:

$$S_1 = S(t_f, t_0, \boldsymbol{x}_0) \tag{3.6}$$

This is a proper measure of how perturbations arising at the initial time affect the state at the final time.

The alternative expression, reported in [15] and obtained from Eq. (3.4) and Eq. (3.5), measures how perturbations at any time affect the final state:

$$S_2(t) = S(t_f, t, \boldsymbol{x}) = S(t_f) \cdot S(t)^{-1} \tag{3.7}$$

The two definitions are inherently different; the first is a fixed quantity, while the second is time-dependent. Nevertheless, both embed information on the influence of uncertainties on the final state. Although notable distinctions already emerge, a more comprehensive comparison is furnished in Section 3.2.1.

The sensitivity matrices delineated in Eq. (3.6) and Eq. (3.7) can be perceived as black boxes containing valuable information. Nonetheless, some relevant insights can be offered. It is evident, through data inspection or problem analysis, that certain terms within the sensitivity matrix play major roles in specific scenarios, while they might be irrelevant in others. These components of the sensitivity matrix establish connections between various state variables; if the connected states are independent, the corresponding sensitivity term becomes zero. Consequently, it is reasonable to consider only specific portions of the sensitivity matrix rather than the entire S matrix, as some terms may be zero or irrelevant for the particular problem under consideration.

By the definition of the sensitivity matrix, it becomes apparent that the rows of the S matrices indicate how a perturbation in the state (at either the initial time or a specific time instant) influences a single component of the final state. Conversely, the columns reveal how a perturbation in a single state variable affects the entire state. Depending on the application and the analytical objective, different rows and columns may be of interest.

The extracted terms can be compiled into a vector denoted as $\boldsymbol{\Lambda}$, representing a linear combination of rows of the S matrices (superscript r) or columns (superscript c). This vector is formed by pre- or post-multiplying the S matrix by a constant vector of the appropriate dimension. An additional subscript can be added to indicate which matrix, either S_1 or $S_2(t)$, the components are derived from.

Although this approach may appear purely mathematical, it carries a clear physical significance, as illustrated in the following example. Given $b(\boldsymbol{x}(t_f))$, a generic linear function of the final state, its sensitivity with respect to a generic state is defined, as:

$$\boldsymbol{S}_b = \frac{\partial b(\boldsymbol{x}(t_f))}{\partial \boldsymbol{x}(t)}$$

However, through the chain rule, this sensitivity can be expressed as:

$$\boldsymbol{S}_b = \frac{\partial b(\boldsymbol{x}(t_f))}{\partial \boldsymbol{x}(t_f)} \cdot \frac{\partial \boldsymbol{x}(t_f)}{\partial \boldsymbol{x}(t)}$$

The second term of the multiplication is equivalent to the quantity defined in Eq. (3.7), therefore:

$$\boldsymbol{S}_b = \frac{\partial b(\boldsymbol{x}(t_f))}{\partial \boldsymbol{x}(t_f)} \cdot S_2(t) = b_{\boldsymbol{x}(t_f)} \cdot S_2(t) = \boldsymbol{\Lambda}_2^r \tag{3.8}$$

It is evident that this term is equivalent to $\boldsymbol{\Lambda}_2^r$. Hence, the $\boldsymbol{\Lambda}$ values can be construed with a well-defined physical interpretation. The same methodology can be extended to the other $\boldsymbol{\Lambda}$ definitions. Functions z_S and $b(\boldsymbol{x}(t_f))$ can be defined, and correspondingly, z_S $\boldsymbol{\Lambda}$ can be established.

The take home message is that any S matrix can be manipulated to suit the requirements of the analysis or of the problem itself. This choice is seminal in shaping the architecture of DOC, as it carries significant implications in terms of both physical outcomes and numerical implementation, as demonstrated in Section 3.2.1.

3.1.2. Sensitivity Function

In certain scenarios, parametric uncertainties assume a significant role. The approach outlined in Section 3.1.1 can accommodate these uncertainties, but only if the uncertain parameters are incorporated into the state, as discussed in [15]. However, this inclusion results in a substantial increase in the number of variables, particularly when dealing with numerous parameters. To mitigate this issue in variable count, an alternative formulation of sensitivity is presented and examined in [41]. Assuming an uncertain parameter p, the parameter sensitivity function can be defined as:

$$\boldsymbol{\Omega}(t) = \frac{\partial \boldsymbol{x}(p,t)}{\partial p}_{x=x(p_0,t)}$$

and its dynamics as:

$$\dot{\boldsymbol{\Omega}}(t) = A(t) \cdot \boldsymbol{\Omega}(t) + \boldsymbol{C}(t), \qquad \boldsymbol{\Omega}(t_0) = 0 \tag{3.9}$$

where $A(t)$ is defined equivalently to Eq. (3.3)and $C(t)$ is:

$$C(t) = \frac{\partial \boldsymbol{f}(\boldsymbol{x}, p, u(t), t)}{\partial p}_{x=x(p_0,t), p=p_0}$$

This leads to the first-order approximation of $\boldsymbol{x}(p,t)$ as:

$$\boldsymbol{x}(p,t) \approx \boldsymbol{x}(p_0,t) + \boldsymbol{\Omega}(t)(p - p_0) \tag{3.10}$$

The physical interpretation of this formulation is straightforward: when the variation of parameter p remains within a linear domain, Eq.3.10 holds and establishes a direct relationship between parametric perturbations and their impact on the current state.

This formulation introduces a different framework for the DOC approach. In the case of having n states and l uncertain parameters, the original formulation by [15] requires $(n+l)^2+n+l$ states. However, in the formulation proposed by [41], this number is reduced to $n+n\cdot l$. This reduction offers a significant advantage in terms of computational efficiency and can be applied to a wide range of problems. However, this comes at the cost of reduced flexibility; the sensitivity matrix in this formulation is primarily designed for time-invariant parameters and is not able to capture the information regarding the relationship between $p(t)$ and $x(t)$ at various time instances. In this case as well, sensitivities with respect to multiple parameters can be defined, and their total number is denoted as $z_{\boldsymbol{\Omega}}$.

3.2. DOC Architecture

In the preceding section, the concept of sensitivity has been introduced and two specific forms of sensitivity discussed. This chapter is dedicated to elucidating the fundamental principles of DOC theory and how the new Optimal Control Problem (OCP) needs to be formulated. The primary contributions of this theory pertain to the modification of the cost function, which is detailed in the first section, and the definition of feedback control, which is presented in the second section of this chapter.

3.2.1. Cost Function

In the general context of an optimal control problem, the cost function is typically expressed as shown in Eq. 2.1. To simplify notation, the Lagrange term is omitted and merged with the Mayer term, resulting in the following expression:

$$J = \phi(\boldsymbol{x}(t_f), t_f) \tag{3.11}$$

As proposed in [15], the fundamental idea behind the DOC theory is to augment the cost function presented in Eq. 3.11 with some terms related to sensitivity. Instead of merely penalizing the original cost index, the sensitivities are also subject to penalization. This straightforward modification serves as the foundation for any application of DOC. By introducing a term that penalizes sensitivities, the trajectory is made more robust, meaning that certain variables become less sensitive to perturbations. Such variables are referred to as desensitized.

The cost function can be expanded with various types of sensitivities. However, for the purposes of this work, which aims to reduce the impact of perturbations on the final states, only sensitivities related to the final state are considered. Without loss of generality, a comprehensive DOC formulation combines the principles presented in [15] and [41], penalizing both a term related to the sensitivity matrix and a term related to the sensitivity function. Depending on the specific application, a design choice can be made to determine which sensitivity components to include. The DOC cost function results in:

$$J = \phi(\boldsymbol{x}(t_f), t_f) + \alpha J_S + \beta J_{\boldsymbol{\Omega}}$$

where J_S is the cost term related to the sensitivity matrix and $J_{\boldsymbol{\Omega}}$ the one referred to the sensitivity function, both suitably weighted by non-negative coefficients α and β.

Sensitivity Matrix DOC Cost First, the cost term related to the sensitivity matrix is defined. As explained in Section 3.1.1, various forms of sensitivity matrices, denoted as S matrices, can be defined, and corresponding $\boldsymbol{\Lambda}$ vectors can be derived. In theory, all of these sensitivities could be embedded into the cost function to penalize them, thus reducing the impact of perturbations on the final state. However, certain criteria should be considered.

S_1 from Eq. 3.7 and $S_2(t)$ from Eq. 3.6 should be penalized differently. Given that $S_2(t)$ is time-dependent and measures the sensitivity of the final state concerning states at any time, it requires integral penalization.

$$J_{S,1} = \|\boldsymbol{S}_1\|_\Xi \qquad J_{S,2} = \int_{t_0}^{t_f} \|\boldsymbol{S}_2(t)\|_\Xi \, dt \tag{3.12}$$

where W express a generic Ξ-matrix norm.

The second approach is somewhat conservative. In certain dynamic systems, sensitivity peaks can occur at specific time instants, making the system highly sensitive to perturbations at those moments. This effect is not penalized when considering sensitivity S_1, while integral penalization of $S_2(t)$ can effectively reduce sensitivity peaks at various time points. Therefore, it is convenient to penalize $S_2(t)$ in integral form.

While the black-box approach for the S matrix is conceptually straightforward, the alternative $\boldsymbol{\Lambda}$ formulations introduced in Section 3.1.1 could offer valuable proprierties. The choice is whether to penalize specific norms of $\boldsymbol{\Lambda}_2^r$ or $\boldsymbol{\Lambda}_2^c$. Notably, these vectors are of size $1 \times n$, where n represents the number of states in the problem, and correspond to rows or columns of the original S matrix. By properly selecting n different $\boldsymbol{\Lambda}$ terms, the entire S matrix can be effectively reconstructed. Thus, $\boldsymbol{\Lambda}$ can be penalized without loss of generality, since all the information collected in S can be retrieved by penalizing multiple $\boldsymbol{\Lambda}$ terms.

Although hybrid solutions can be implemented (e.g., penalizing the generic i^{th} row of S through a specific $\boldsymbol{\Lambda}^r$ and the generic j^{th} column of S through $\boldsymbol{\Lambda}^c$), it is advisable to focus on terms related to the rows of the sensitivity matrix. This is mainly due to two factors: first, the physical interpretation of rows and columns of the sensitivity matrix. Since the primary focus is on the final state rather than the perturbations themselves, it is more meaningful from a physical point of view to consider a row of the sensitivity matrix, which measures how a component of the final state is affected by all perturbations. Second, the row extraction approach is prone to reduced order formulations of the DOC problem, as discussed in Section 3.4.1.

Not all criteria are clear at this stage, but this reasoning highlights a unique characteristic of any DOC formulation. Given the substantial increase in the number of variables required, the advantages in terms of performance offered by any reduced-order formulation are highly significant, as they lead to a reduction in the CPU time required to solve the problem. However, this computational efficiency advantage can affect the physical significance of the method itself, potentially eliminating relevant aspects or reducing the method flexibility. Therefore, careful consideration of these two factors is essential when making algorithm design choices, as certain desired formulations, such as penalizing sensitivity matrix columns, could lead to such a degradation of computational efficiency that they become impractical. As a matter of fact, the term $\boldsymbol{\Lambda}_1^c$ also allows for a convenient reduced-order DOC formulation. However, the physical meaning of $\boldsymbol{\Lambda}_2^r$ guides the preference for this latter term.

The cost term related to J_S is expressed as:

$$J_S = \int_{t_0}^{t_f} \sum_{k=1}^{z_S} \|\boldsymbol{\Lambda}(t)_{k,2}^r\|_2 \, dt = \int_{t_0}^{t_f} \sum_{k=1}^{z_S} \|\boldsymbol{\Lambda}(t)_k\|_2 \, dt$$

where z_S express the number of $\boldsymbol{\Lambda}(t)$ terms, in the same fashion of Eq. (3.8). The generic Ξ norm of Eq. (3.12) is replaced by the common 2-norm. The superscript and subscript indices are omitted in favor of a more concise notation.

Sensitivity Function DOC Cost The second cost term related to DOC, denoted as $J_{\boldsymbol{\Omega}}$, is the next focus. Given that the primary goal of desensitization is to minimize the impact of perturbations on the final state, it is straightforward to penalize the norm of the $\boldsymbol{\Omega}$ term, evaluated at the final time, to gauge how parametric uncertainties influence the final state. While other penalty terms, such as integral ones, have been formulated, this approach emerges as the most relevant from a physical perspective. A total of $z_{\boldsymbol{\Omega}}$ functions can be penalized, resulting in the cost term:

$$J_{\boldsymbol{\Omega}} = \sum_{l=1}^{z_{\boldsymbol{\Omega}}} \|\boldsymbol{\Omega}(t_f)_l\|_2$$

DOC Cost function Defined all the building blocks, they can be put in Eq. (3.12), forming the complete DOC cost function.

$$J = \phi(\boldsymbol{x}(t_f), t_f) + \alpha \int_{t_0}^{t_f} \sum_{k=1}^{z_S} \|\boldsymbol{\Lambda}(t)_k\|_2 \, dt + \beta \sum_{l=1}^{z_{\boldsymbol{\Omega}}} \|\boldsymbol{\Omega}(t_f)_l\|_2 \tag{3.13}$$

The original problem is effectively recovered by setting α and β to zero. When $\alpha = 0$, only the sensitivity to parameters is penalized, but this term disappears when $\beta = 0$.

The significance of the sensitivity terms increases with higher values of α or β, making the minimization algorithm prioritize the reduction of these sensitivity terms. This has a clear physical interpretation: the optimal solution at the final time becomes less sensitive to both state and parameter perturbations along the trajectory. However, this approach comes with the trade-off of increasing the value of the original cost function $\phi(\boldsymbol{x}(t_f), t_f)$ of the non-desensitized problem because two terms are now present in the augmented cost. A precise tuning measures the relative importance of the two terms, and is needed to guarantee the desired balance between performance and robustness of the solution.

3.2.2. Feedback Control

The feedback control component holds significant importance in the DOC theory, as it constitutes one of the two fundamental building blocks of the approach. In addition to enhancing robustness, DOC also establishes a connection between trajectory and control optimization. As explained in [15], the controller gains can be embedded into the optimization process. By means of $\boldsymbol{u}$, an affine feedback control law can be derived as:

$$\boldsymbol{u}(t) = \boldsymbol{u}^*(t) + K^*(t)\left(\boldsymbol{x} - \boldsymbol{x}^*(t)\right) \tag{3.14}$$

where the K matrix contains the feedback gains and the apex * represents the optimal solution of the problem. Therefore, the new state dynamics $\tilde{\boldsymbol{f}}(\boldsymbol{x}, t)$ can be defined as:

$$\tilde{\boldsymbol{f}}(\boldsymbol{x}(t), t) = \boldsymbol{f}\left(\boldsymbol{x}(t), \boldsymbol{u}^*(t) + K^*(t)\left(\boldsymbol{x} - \boldsymbol{x}^*(t)\right), t\right) \tag{3.15}$$

Therefore, sensitivity matrix equation referred to Eq. (3.15) reads as follows:

$$\frac{\partial S}{\partial t} = \left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}} \cdot K\right) S = (A + B \cdot K)S \tag{3.16}$$

and sensitivity function differential equation becomes:

$$\dot{\boldsymbol{\Omega}}(t) = (A + BK) \cdot \boldsymbol{\Omega}(t) + \boldsymbol{C}(t), \qquad \boldsymbol{\Omega}(t_0) = 0 \tag{3.17}$$

The feedback gains can be regarded as additional variables in the problem, therefore increasing the problem's dimension. Alternatively, there are other methods for defining them, as will be explored in Section 3.4.

3.3. DOC Interpretations

While the DOC method introduced in the previous sections is characterized by a straightforward and lucid formulation, its interpretation is not trivial. To gain a deeper understanding of the method, two analyses are presented: the first analysis is documented in [42], and the second analysis has been fully developed by the authors of this work.

Statistical Interpretation As detailed in [42], a direct correspondence between the sensitivity matrix of the deterministic problem and the covariance matrix of the resulting stochastic problem can be established. The initial state and the dynamics of Gaussian white noises are encompassed in the variables $\boldsymbol{w}_0$ and $\boldsymbol{w}(t)$, which exhibit the following properties:

$$\begin{cases} E[\boldsymbol{w}_0] = \mathbf{0} \\ E[\boldsymbol{w}_0 \boldsymbol{w}_0^T] = P_0 \\ E[\boldsymbol{w}(t)] = \mathbf{0} \quad \forall t \\ E[\boldsymbol{w}(t)\boldsymbol{w}(\tau)^T] = W(t) \cdot \delta(t-\tau) \quad \forall t,\ \tau \end{cases}$$

where $E[\cdot]$ is the expected value operator, P_0 denotes the initial covariance matrix and $W(t)$ the power spectral density. Through some mathematical steps, a direct relation between the covariance matrix $P(t)$ and the sensitivity matrix $S(t)$ can be expressed as:

$$P(t) = S(t)P_0S(t)^T + \int_{t_0}^{t} \left(S(t)S(\tau)^{-1}\right) W(\tau) \left(S(t)S(\tau)^{-1}\right)^T d\tau \tag{3.18}$$

The provided equation illustrates that, given $S(t)$, $P(t)$ can be derived. The inverse relationship, from $P(t)$ to $S(t)$, is documented in [42]. This holds significant implications, as the DOC formulation can also be understood in a statistical context: reducing sensitivity results in a decreased covariance matrix of the states. This insight yields two direct consequences.

First, instead of using the $S(t)$ matrix, the equivalent $P(t)$ matrix can be employed in the DOC formulation to define the overall equation. The primary advantage of this approach is that $P(t)$ is symmetric, enabling a substantial reduction in the number of propagated states. However, this solution is not explored in this work, since other reduced dimensionality solutions are presented in Section 3.4. Nonetheless, it may prove useful in future applications.

The second, and more significant, implication is that the DOC impact can be assessed in terms of the covariance of the final state. Given the direct link between $S(t)$ and $P(t)$, reducing $S(t_f)$ results in a decrease in $P(t_f)$. In a Monte Carlo analysis, it becomes feasible to compute the standard deviations of the final states, which can serve as a reference for evaluating the success of the DOC procedure.

Feedback Gain Effect It is well-known that exists a physical gap between the nominal solution of an optimal control problem and the actual trajectory followed in real applications. This discrepancy arises due to numerous factors collectively termed uncertainties, which are typically not considered during the design of the optimal trajectory. These uncertainties can include model variations, state fluctuations, parametric deviations, and their influence on the trajectory can be mitigated by implementing a feedback control law. This entire process can be interpreted in terms of sensitivity, since the feedback gains aim to reduce the impact of uncertainties on the trajectory.

As demonstrated in Eq. (3.16), the dynamic equations of the sensitivity matrix are linear and governed by three matrices: A, B, and K. These matrices respectively represent the linearized versions of the problem concerning the state, control, and gain matrix. It is important to note that if the linearized system is stable, the sensitivity matrix dynamics will also be stable. Conversely, if the linearized system is unstable, the dynamics of the S matrix will be unstable.

In typical applications, the matrix A possesses at least one eigenvalue that is non-stable, necessitating the use of a controller to guide the system toward a desired final state. In the absence of control, when the K matrix is set to zero, initial state perturbations have a profound impact on the system dynamics as they grow indefinitely. This phenomenon can be accurately described by the S matrix: if its dynamics are unstable, indicating an unstable system, the terms in the S matrix grow, starting from the initial identity matrix condition. As time progresses, the S matrix increases, in line with the growth of perturbations. Consequently, a direct relationship can be established between the effects of perturbations on the trajectory and the sensitivity matrix.

The open-loop DOC method, where feedback gains are absent, seeks to reduce sensitivity by shaping the trajectory in such a way that the DOC term in Eq. (3.13) is minimized. However, to achieve linear system stability and manage the incremental growth of sensitivity, a closed-loop DOC formulation becomes imperative. If the stability condition is met, the trends in initial state perturbations diminish, and their impact on system dynamics gradually wanes, in contrast to the exponential amplification observed in an unstable system. This is the essence of feedback control laws, which can be comprehensively explained through the dynamics of the S matrix.

3.4. Implementation Strategies

The numerical implementation of the DOC approach necessitates an additional computational effort compared to the nominal case. To achieve the complete closed-loop DOC formulation, it is essential to compute the single or multiple $\mathbf{\Lambda}(t)$ terms and $\mathbf{\Omega}(t_f)$ terms as described in Eq. (3.13). Concurrently, the feedback gains outlined in Eq. (3.14) must be synthesized. For this reason, an augmented state is defined accordingly, and extra states must be introduced based on the chosen procedures for computing the sensitivity terms and the components of the K matrix.

This section seeks to examine various implementation strategies that can be employed to formulate the DOC procedure. The first part delves into the sensitivity-related terms, while the second part explores the feedback gains.

3.4.1. Sensitivity Terms

To implement DOC, it is essential to compute $\mathbf{\Omega}$ and $\mathbf{\Lambda}$ terms for each time step, whenever β and α respectively differ from zero. Calculating $\mathbf{\Omega}$ is a straightforward process, as it involves augmenting the state with its components, which are then propagated in accordance with Eq. (3.9). However, when computing the $\mathbf{\Lambda}$ term, various strategies are available, and this section highlights the most relevant ones.

S Matrix The most straightforward and intuitive method is to directly compute $\mathbf{\Lambda}$ using Eq. (3.7) and Eq (3.8). For this approach, the $S(t)$ matrix is required at each time step, and its propagation is carried out in accordance with Eq. (3.16). The problem state is augmented with the components of $S(t)$, namely n^2 components. Since the dynamic equation of $S(t)$ is linear, its numerical implementation is relatively simple once the initial condition, which is an identity matrix, is established. However, this formulation presents a major challenge: the computation of the matrix inverse.

To evaluate the $\mathbf{\Lambda}$ term, the inverse matrix of S is necessary at each time instant. One potential solution to this problem is the analytical computation of the matrix inverse. However, this approach is not practical for problems with a large number of states, as the number of elements in the $S(t)$ matrix scales proportionally to the square of the number of states. Additionally, problems may arise when dealing with ill-conditioned matrices, as the computation of the inverse becomes highly inaccurate. Moreover, in some cases, the matrix $S(t_f)$ may not be accessible, depending on the solver used and how boundary conditions are enforced. As demonstrated in [15], this formulation necessitates the introduction of another set of variables: S_f components, which are constant values matching the final values of $S(t)$ components. These are used to compute the inverse matrix of $S(t)$ at the final time. In this formulation, the number of extra states increases dramatically, introducing $2 \cdot n^2$ additional states.

S Inverse Matrix One strategy to address the aforementioned issue is to propagate the inverse of the $S(t)$ matrix rather than the $S(t)$ matrix itself. Since the inverse of the $S(t)$ matrix is required at all times, and the $S(t)$ matrix is only needed at the final time, an effective solution is to augment the state with the components of the $S(t)^{-1}$ matrix. This approach is feasible because, as mentioned in Section 3.1.1, the $S(t)$ matrix is never singular.

In theory, this dramatically reduces the computational effort, as matrix inversion only needs to be performed once, at the final time. This idea is supported by the fact that the differential equation for the $S(t)$ matrix is linear (Eq. (3.3)). Therefore, reformulating it using the inverse of $S(t)$ also results in a linear matrix differential equation, simplifying the analytical derivation of the dynamic equation for $S(t)$.

Assuming the identity matrix as I, the following derivation demonstrates this approach:

$$\frac{\partial I}{\partial t} = \frac{\partial}{\partial t}(S(t) \cdot S(t)^{-1}) = \frac{\partial S(t)}{\partial t} S(t)^{-1} + S(t) \frac{\partial S^{-1}}{\partial t} \tag{3.19}$$

By rearranging Eq. (3.19), the following equation is obtained:

$$\frac{\partial S(t)^{-1}}{\partial t} = -S(t)^{-1} \frac{\partial S(t)}{\partial t} S(t)^{-1} = -S(t)^{-1} \left.\frac{\partial \boldsymbol{f}(\boldsymbol{x}, t)}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\boldsymbol{x}(t)} S(t) \cdot S(t)^{-1}$$

This leads to the elegant form:

$$\frac{\partial S(t)^{-1}}{\partial t} = -S(t)^{-1} \left.\frac{\partial \boldsymbol{f}(\boldsymbol{x}, t)}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\boldsymbol{x}(t)} = -S(t)^{-1}(A + BK) \tag{3.20}$$

It is worth noting that although this idea was introduced in [42], it has not been practically implemented so far.

To establish the complete formulation, the initial condition for propagation must be defined. As mentioned in Section 3.1.1, the initial condition for the S matrix is the identity matrix. The generic $S(t)$ matrix, defined as $S(t, t_0, \boldsymbol{x}(t_0))$ according to Eq. (3.7), describes the effect of perturbations at the initial time t_0 on the state at the generic time t. Since time t equals time t_0 at the initial time, the initial $S(t)$ matrix can be expressed as $S(t_0, t_0, \boldsymbol{x}(t_0))$. This matrix measures the impact of perturbations at the initial time on the initial state, thus it is equal to the identity matrix.

As reported in [15], a property holds for two generic time instants t_1 and t_2:

$$S(t_1, t_2, \boldsymbol{x}(t_2)) = S(t_2, t_1, \boldsymbol{x}(t_1))^{-1}$$

By extension, this property is valid for the initial $S(t)$ matrix:

$$S(t_0, t_0, (x)(t_0)) = S(t_0, t_0, (x)(t_0))^{-1}$$

Hence, the initial condition for $S(t)$ matrix propagation is equivalent to that of $S(t)^{-1}$, and it is the identity matrix.

Lambda Vector While the previously proposed architecture mitigates numerical issues related to matrix inversion, it still necessitates the use of extra states equal to $2 \cdot n^2$. However, in many practical applications, there is no need to desensitize the entire final state, therefore just a part of the sensitivity matrix can be propagated. The goal of this second reformulation is to provide an efficient implementation scheme for the DOC problem, minimizing the number of extra states and the inversion-related issues.

In Eq. (3.20), both sides of the equation can be pre-multiplied by the term $S(t_f)$, leading to the following equation:

$$\frac{\partial \left(S(t_f)S(t)^{-1}\right)}{\partial t} = -\left(S(t_f)S(t)^{-1}\right)\left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}} \cdot K\right) \tag{3.21}$$

This equation is subject to the final condition $(S(t_f)S(t_f)^{-1}) = I$. According to [42], the initial boundary value problem becomes a final boundary value problem because:

$$S(t_f) = S(t)t = t_f \rightarrow \left(S(t_f)S(t)^{-1}t = t_f\right) = \left(S(t_f)S(t_f)^{-1}\right) = I$$

By multiplying both sides by $\dfrac{\partial b(\boldsymbol{x}(t_f), t_f)}{\partial \boldsymbol{x}(t_f)}$ and transposing the equations, a new formulation is derived:

$$\frac{\partial \boldsymbol{\Lambda}(t)}{\partial t} = -\left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}} \cdot K\right)^T \cdot \boldsymbol{\Lambda}(t) = -\left(A + BK\right)^T \cdot \boldsymbol{\Lambda}(t) \tag{3.22}$$

This equation is subject to the final condition as expressed in Eq. (3.23):

$$\boldsymbol{\Lambda}(t_f) = \left(\frac{\partial b(\boldsymbol{x}, t)}{\partial x}\right)^T_{\boldsymbol{x}=\boldsymbol{x}(t_f), t=t_f} \tag{3.23}$$

This new formulation significantly reduces the number of extra states compared to the previous ones, where components of $S(t)$ or $S(t)^{-1}$ and S_f were incorporated into the augmented state. In this approach, multiple functions $b(\boldsymbol{x}(t), t)$ can be desensitized, but the advantage in reducing the number of extra states diminishes if the number of functions exceeds n, the number of states in the initial problem, as each $\boldsymbol{\Lambda}$ has a size of n elements.

One of the primary issues of DOC, as addressed in [41], is the increase in the size of the state space. However, this new formulation only requires n extra elements, the same as the state size of the sensitivity function proposed in [41], providing a more complete and flexible formulation. Additionally, the boundary conditions, which were enforced by definition in the previous formulations as $S(t_f) = S_f$, are now automatically met through Eq. (3.23). As a result, the number of extra states is reduced from $2 \cdot n^2$ to n.

This propagation method clarifies the choice made in Section 3.2.1: using $\boldsymbol{\Lambda}_2^r$ offers a highly convenient reduced-order propagation with distinct properties of S while maintaining the desired physical meaning.

Table 3.1 reports a summary of the methods proposed, where n_S is the number of $\boldsymbol{\Lambda}(t)$ functions considered.

Quantity	**States**	**Final Cond.**	**Generality**
$S(t)$	n^2	n^2	Maximum
$S(t)^{-1}$	n^2	n^2	Maximum
$S(t_f)S(t)^{-1}$	n^2	-	Maximum
$\boldsymbol{\Lambda}(t)$	$z_S \cdot n$	-	Reduced

Table 3.1: Summary of Sensitivity computation strategies

3.4.2. Feedback Gain Strategies

As mentioned in Section 3.2.2 and explored further in Section 3.3, the significance of feedback gains in the desensitization procedure is very relevant. This is because these gains shape the sensitivity matrix and play a seminal role in ensuring system stability. Users have the option to prescribe these gains themselves [43], or they can be treated as constant values to be determined by the solver or considered as additional time-varying variables [15]. Another approach involves obtaining these gains through structured methodologies [16]. In this section, a concise overview of the available methods is offered, with a specific focus on the advantages and disadvantages associated with each choice.

User Defined The initial and most straightforward method for determining the feedback gains involves using constant values provided by the user. This approach has the notable advantage of significantly reducing the computational effort required by the solver. This reduction in the number of unknowns results in a faster problem solution and a substantial decrease in CPU time. However, there are potential drawbacks to this approach. Firstly, selecting appropriate gain values demands a deep understanding of the problem, which is not always available. Moreover, employing fixed user-defined values can lead to sub-optimal solutions, as it restricts the exploration of the research state-space. As discussed in the subsequent section, allowing gains to vary can potentially yield more optimal results. Lastly, is not guaranteed that user-defined values are able to guarantee the system stability for each condition.

Free Gains To reduce the constraints on gain values and grant more flexibility to the optimizer, another strategies can be implemented, allowing for freedom in computing gain values. In contrast to the fixed gain values approach, these strategies introduce extra variables, which can increase the time required for problem solution.

Since the gains must vary, they are treated as additional state or control variables to be determined during the optimization process. Several implementation strategies are available, with one of the most effective approaches being as follows.

Given the cost function reported in Eq. (3.13), it can be augmented with two extra terms as shown:

$$J = \Phi(\boldsymbol{x}(t_f), t_f) + \alpha \int_{t_0}^{t_f} \|\boldsymbol{\Lambda}(t)\|_2 \, dt + \gamma \int_{t_0}^{t_f} \|K\|_{W_K(t)} dt + \theta \int_{t_0}^{t_f} \|U_K\|_{W_U(t)} dt \qquad (3.24)$$

where $W_K(t)$ and $W_U(t)$ are respectively the W_K and the W_U norms, K is the gain matrix and U_K is the gain rates matrix, defined as:

$$\dot{K} = U_K$$

The number of $\boldsymbol{\Lambda}(t)$ functions is set to one to simplify the problem, as well as β set to zero.

The K matrix terms are extra states, while the U_K elements augment the control variables vector. The size of the problem is increased and the number of adding state/control variables depend on the problem formulation. The K and U matrix are m x n, therefore extra $2 \cdot m \cdot n$ states are required. A brief discussion of Eq. (3.24) is needed to deeply understand the meaning of the two new cost terms.

The first term, weighted by the non-negative coefficient γ, serves as a constraint on feedback effort. Without this term, control gains would have unrestricted variability, potentially leading to undesirable outcomes. The innovative aspect of this work is the inclusion of the gain-rate penalization, which had not been previously considered in the DOC formulation. Without the gain-rate matrix U_K penalization term, the K terms would act as control variables rather than state variables.

After defining this general framework, the weights γ and θ need calibration. The tuning of the weight α is guided by other specific requirements, such as fuel consumption or final sensitivity, and is not covered in this section. The tuning process is complex and necessitates consideration of various factors, but two main strategies are under consideration:

Setting θ to high values significantly penalizes the gain-rate term, resulting in almost zero values for U_K terms. This choice leads to nearly constant K values, which are chosen by the solver. While this approach provides greater flexibility, it might not be suitable for all applications, as some scenarios require time-varying feedback effort. The values of nearly constant K gains are strongly influenced by the γ value, and precise tuning is required to ensure reasonable feedback control values without exceeding bounds. Unfortunately, there are no predefined solutions, and a trial-and-error procedure assisted by Monte Carlo simulations is necessary.

Adopting small γ values uses U_K as a regularization term for time-varying feedback gains. Among the proposed solutions, this is the least constrained one, allowing the solver to search for the optimal solution with more degrees of freedom. However, this flexibility comes at the cost of increased computational effort. This formulation provides substantial flexibility and can lead to the simultaneous optimization of the trajectory and feedback gains, resulting in the optimal guidance and control combination.

Despite its functionality, this approach still presents some challenges. When using traditional methods for regulator tuning (e.g., manual, LQR, H-inf), system stability is a basic requirement. With this formulation, system stability is not guaranteed. Additionally, the precise tuning of controller performance (e.g., noise rejection, steady-state error) is difficult. The design of γ and θ values has unpredictable effects on the optimality of the problem, as certain values can lead to non-optimal solutions, reducing the search space.

Structured Approaches None of the previously mentioned solutions meet the minimum requirement of stability, which is a fundamental criterion for any feedback controller. In the absence of alternative methods, the design choice has leaned toward well-established structured approaches, specifically LQR (Linear Quadratic Regulator). LQR is favored for its ease of implementation and straightforward tuning process, where the gains are computed by propagating Riccati Equations alongside trajectory generation, augmenting the state variables as necessary.

Given that the problem is finite-horizon (i.e., the final time is not infinite), the differential form of LQR is employed. In contrast, the algebraic form is used in infinite-horizon cases. Consequently, the state must be expanded with components of the Riccati matrix, adding $n \times n$ states. Due to the symmetries inherent in the LQR sweep-method formulation, the Riccati matrix is symmetric, which means that a reduced number of extra states is introduced into the problem. This not only results in a more compact and efficient formulation but also avoids symmetry loss issues associated with numerical errors that can arise in certain applications. Another advantageous aspect of this solution is that it reduces the number of variables with respect to the full Free Gains methodology, explained in the previous paragraph.

Given the state penalty matrix $Q(t)$ and the control penalty matrix $R(t)$, and the linearized dynamics matrices of the problem $A(t)$ and $B(t)$, the differential equation for the Riccati Matrix $P(t)$ is:

$$-\dot{P}(t) = P(t)A(t) + A^T(t)P(t) - P(t)BR^{-1}B^T P(t) + Q(t) \tag{3.25}$$

subject to the final condition:

$$P(t_f) = Q(t_f)$$

The reported differential equations comes from the minimization of the corresponding finite-horizon cost function, defined as:

$$J_{\text{LQR}} = \boldsymbol{x}(t_f)^T Q(t_f)\boldsymbol{x}(t_f) + \int_{t_0}^{t_f} \boldsymbol{x}(t)^T Q(t)\boldsymbol{x}(t) + \boldsymbol{u}(t)^T R(t)\boldsymbol{u}(t)\, dt \tag{3.26}$$

The LQR optimal gains are computed as:

$$K = -R^{-1}B^T P(t)$$

It is important to note that the $Q(t)$ matrix in the Riccati equation must be positive semi-definite. Assuming $Q(t)$ is zero along the trajectory leads to a loss of penalization on the current state error, giving more importance to the terminal state error, which is weighted by $Q(t_f)$. This approach was proposed by [16], but it differs from the methodology applied in this work, as it can lead to significant state errors along the trajectory.

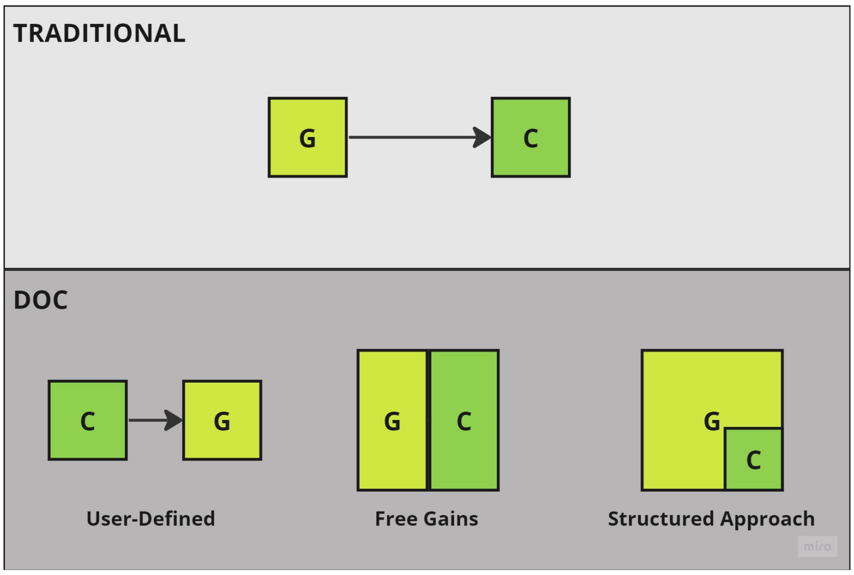


Figure 3.1: Summary of DOC Procedures

Since one of the aims of the DOC methodology is to integrate trajectory and control optimization into a single process, the consequences of selecting a structured approach need to be examined. One critical consideration is that a structured approach restricts the freedom of the solver. The gains and gain rate cannot be freely adjusted and used as optimization variables. Furthermore, LQR is already an optimal gain selection method, and the trajectory is shaped accordingly. In other words, each iteration of the guidance optimization process is based on an internally performed optimal gain selection through Riccati Matrix propagation. While guidance and control are not optimized simultaneously, as in the free gains approach, they are not entirely separate either. With the current state of DOC theory, this procedure finds a balance between fully decoupled guidance and control optimization (as in traditional methods) and fully coupled optimization (as in DOC with the free gains, using the cost function of Eq. (3.24)).

3.4.3. NLP-based DOC

After presenting analyzing all available alternatives, the next step is to consider the most suitable options for implementing NLP-based DOC. As discussed in the preceding section, the software tools utilized include GPOPS [17] with Radau collocation and IPOPT as the solver to obtain solutions for the NLP problem. The general architecture of the code involves discovering the optimal control problem solution in a noDOC formulation and employing it as an initial guess for the DOC problems with increasing α values, for both open-loop (OL) and closed-loop (CL) scenarios.

The choice of strategies heavily depends on the software's properties, which, in turn, are influenced by the collocation method used. Given the computationally intensive nature of inverting the $S(t)$ matrix, the most valid and efficient formulation involves propagating the $\mathbf{\Lambda}(t)$ term, as described by the differential equation in Eq. (3.22). This term can be directly integrated into the cost function without the need for further processing, significantly reducing the number of states to be propagated (from $2 \cdot n^2$ to n).

The second design choice pertains to the method used for calculating feedback gains. As previously mentioned, a structured approach offers the best balance in terms of controller properties and solution optimality. The LQR method has been adopted, and the Q and R matrices have been configured to meet the desired performance criteria. However, it is important to note that a limitation with LQR-GPOPS integration is that Q and R must be chosen as constants to ensure manageable CPU time, which reduces the method's flexibility compared to time-variant weightings.

Due to the complexity of this formulation, the implementation must be carefully optimized to accelerate the entire search process. Whenever feasible, Matlab features are leveraged to their fullest extent. For instance, vector operators are used extensively instead of loops, and this can reduce the CPU time by a factor of 5 to 10.

4 | Powered Landing Scenario

Following the presentation of the theoretical foundation for Desensitized Optimal Control (DOC), the proposed methodology is applied in the context of the specific investigation scenario. This chapter's primary aim is to offer an overview of the Powered Landing Main Scenario, as described in Section 4.1. This involves detailing the equations governing the object's motion, reporting the fundamental assumptions, and introducing the solution for the nominal problem. The latter part of this chapter, found in Section 4.2, is dedicated to the integration of the Desensitized Optimal Control (DOC) approach with the case study. This section will dive into the definition of the algorithm. Subsequently, in Section 4.3, the focus shifts towards characterizing the tuning procedure and measuring the strategy performances.

4.1. Mission and Scenario

Powered descent refers to the phase during which the rocket's propulsion system provides thrust to control the trajectory, ensuring that specific final conditions are met. The primary objective of this phase is to guarantee that the landing site or a point slightly above it is reached with a predetermined final velocity. The control action for this process is achieved through the use of thrusters distributed around the landing vehicle in various configurations. This work adopts the configuration detailed in [44].

It is important to note that this study exclusively concentrates on the final stage of the landing process. Other design choices, such as determining the starting point of the powered phase, are beyond the scope of this work.

A crucial aspect to consider is that the ultimate goal of this problem is not just to derive any trajectory but to determine the optimal trajectory based on a specific cost index. Depending on different requirements, the problem can be approached as time-optimal, fuel-optimal, energy-optimal, or using mixed formulations, as discussed in [45]. However, this research focuses only on the mass-optimization aspect, and the cost function is defined accordingly.

4.1.1. Mathematical Model

Although this situation could be described with a high level of complexity, this work employs a simplified dynamical model. In this model, the landing body is treated as a point mass, and all rotational degrees of freedom are omitted to avoid dealing with the complexity of coupled translation-attitude guidance, which is beyond the scope of this preliminary analysis. Furthermore, aerodynamic forces are disregarded due to the low velocities values and the increased complexity introduced by nonlinear quadratic terms. Additionally, the rotation of the planet, as well as variations in the planet's gravity field, are excluded from consideration. This is justified by the mission's relatively short duration and the minor height changes involved compared to the scales associated with these phenomena. Consequently, the curvature of the planet's surface is also not accounted for.

Within this simplified model, the thrusters are idealized, with no transients or delays affecting their prescribed thrust values. They are depicted as a set of n_T identical engines, each canted at an angle γ with respect to the net thrust direction and providing the same nominal thrust value, T. The net thrust produced is calculated as n_T multiplied by T and the cosine of γ. For further details regarding the geometry and modeling, refer to [44]. It is important to note that all values are assumed to remain at their nominal levels throughout the entire mission, with no deviations in terms of pointing errors or performance variations. The only permissible perturbation considered is in the thrust value, represented as ΔT.

The reference frame used is a surface-fixed coordinate system, centered at the desired landing site. In this frame, the axes, denoted as $\hat{x}$ and $\hat{y}$, span the horizontal plane, which is tangential to the planet's surface at the landing point. The $\hat{h}$ axis points opposite to the direction of the gravity field and, even if this system is inherently non inertial, it is assumed to be inertial, due to the strong hypotheses considered. The state vector is defined as $\boldsymbol{x} = [x, y, h, v_x, v_y, v_h, m]$, where the first three components represent displacements in the coordinate axes, and the components starting with v the corresponding velocities. The quantity m represents the mass of the lander. The motion of the lander is controlled through thrust modulation using the vector $\boldsymbol{u} = [u_x, u_y, u_h]$, where each component corresponds to adjustments along $\hat{x}$, $\hat{y}$, and $\hat{h}$ axes, respectively. The magnitude of this vector, represented as u is constrained by the control constraint outlined in Eq. (4.2), ensuring that the magnitude remains within specific limits.

The dynamics of the system are described by the following set of equations:

$$\begin{cases} \dot{x} = v_x \\ \dot{y} = v_y \\ \dot{h} = v_h \\ \dot{v}_x = \dfrac{u_x \cdot n \cdot T \cdot \cos(\gamma)}{m} \\ \dot{v}_y = \dfrac{u_y \cdot n \cdot T \cdot \cos(\gamma)}{m} \\ \dot{v}_h = -g_m + \dfrac{u_h \cdot n \cdot T \cdot \cos(\gamma)}{m} \\ \dot{m} = -\dfrac{u \cdot n \cdot T}{I_{sp} \cdot g_0} \end{cases} \tag{4.1}$$

Subject to the control constraint:

$$0 < u_{min} \leq u = \sqrt{u_x^2 + u_y^2 + u_h^2} \leq u_{max} \tag{4.2}$$

and to the no-subsurface flight constraint:

$$h(t) \geq 0, \quad \forall t \leq t_f \tag{4.3}$$

where the lander's altitude h must remain above zero for all times less than the final time t_f. Even if in many applications the final time is fixed or obtained from multiple guesses, here it is left free to vary and considered as a variable of the problem.

In terms of additional constraints, factors like the maximum glide-slope, as discussed in [44], could be considered. However, they are not incorporated into this work due to its preliminary research nature. Similarly, the final conditions regarding position and velocity are set to zero, even though, in practical applications, the powered phase typically concludes with a non-zero final altitude and vertical velocity to achieve a precise vertical landing in the ending phase.

The boundary conditions are:

$$\begin{aligned} t_0 &= 0 & t_f &= \text{free} \\ \boldsymbol{r}(t_0) &= \boldsymbol{r}_0 & \boldsymbol{r}(t_f) &= \boldsymbol{0} \\ \boldsymbol{v}(t_0) &= \boldsymbol{v}_0 & \boldsymbol{v}(t_f) &= \boldsymbol{0} \\ m(t_0) &= m_0 & m(t_f) &> m_{dry} \end{aligned} \tag{4.4}$$

where $\boldsymbol{r} = [x, y, h]$ and $\boldsymbol{v} = [v_x, v_y, v_h]$, and m_{dry} is the vehicle dry mass.

It is important to note that while the same approach could potentially be applied to different bodies landing on various planets, the trajectory optimization in this work has been tailored for the Mars Landing Problem, ensuring consistency with reference numerical values. However, the same dynamical model can be readily adapted to different celestial bodies by simply altering the gravity numerical values.

Finally, the cost function is defined and it aligns with the fuel-optimal nature of the problem, aiming to minimize the final mass of the lander

$$J = \min_{u_x, u_y, u_h, t_f} -m(t_f) \tag{4.5}$$

Therefore, the problem can be formulated as: minimize the cost index Eq. (4.5) subject to Eq. (4.1), Eq. (4.2) and Eq. (4.3), given the boundary conditions Eq. (4.4)

4.1.2. Nominal Solution

The nominal solution for the problem under investigation has a well-known structure. As outlined in [45], the optimal trajectory possesses a control norm that exclusively adopts the minimum or maximum allowable values. Additionally, in the context of a three-dimensional scenario, there are, at most, two switching points for the thrust profile. Consequently, the optimal trajectory can be conceptualized as a piece-wise continuous function comprising three distinct segments. It can also be demonstrated that, without sacrificing generality, all solutions can be classified into the overarching categories of maximum-minimum-maximum or (*max-min-max*) trajectories. Hence, there exist max, min-max, and max-min-max profile possibilities. This theoretical insight is substantiated by the numerical solutions generated using an NLP solver, specifically GPOPS [17].

Before presenting the nominal solution, it is pertinent to provide the numerical values utilized in the analysis. The initial conditions draw inspiration from [16] but have been extended to represent a more general 3-D case. Notably, the initial y component is set to a non-zero value, as detailed in Table 4.1. The initial mass m is $m(t_0) = 1905\,kg$, while the dry mass of the vehicle is $m_{dry} = 1505\,kg$.

These initial conditions represent one of the most challenging scenarios: the vehicle is descending (with a negative vertical velocity) while also moving away from the final target due to positive in-plane velocity components. Consequently, this scenario puts the proposed strategy to be tested under harsh conditions. The parameter tuning process, as

	Unit	$\hat{x}$	$\hat{y}$	$\hat{h}$
Position	m	1900	800	3100
Velocity	m/s	40	20	-50

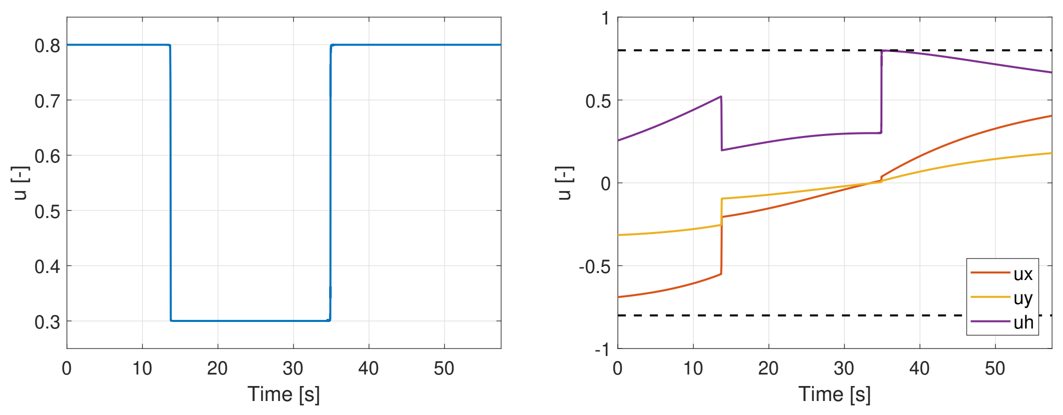
Table 4.1: Initial Conditions

characterized in Section 4.3, is applicable to conditions similar to those yielding a nominal maximum-minimum-maximum thrust profile. However, it is important to note that for entirely different initial conditions, such as when the vehicle is initially moving towards the target, the tuning process may vary.

All the other parameters of the problem, appearing in Eq. (4.1) are reported in the following table. They are typical values, adopted in [44] and in [16].

Quantity	**Symbol**	**Value**
Thrusters	n_T	6
Thrust Angle	γ	27 deg
Thrust	T	3100 N
Specific Imp.	I_{sp}	225 s
Min. control	u_{min}	0.3 [-]
Max. control	u_{max}	0.8 [-]
Mars gravity	g_m	3.7114 m/s^2
Earth gravity	g_0	9.8066 m/s^2

Table 4.2: Problem Parameters



(a) Control Profile (b) Control Components

Figure 4.1: Nominal Control

Even if polynomial approximation is not the best option to approximate constant piece wise functions, the resulting control profile have max-min-max shapes, as shown in Fig. 4.1a

An important observation to note is that, as the initial conditions become more and more challenging, the minimum thrust part is reduced, until it vanishes. This can be regarded as the last feasible point of the problem, which might become infeasible when dealing with even more extreme initial conditions. Under such circumstances, it may be necessary to relax certain constraints and adjust the final conditions, as demonstrated in [45].

4.2. DOC Architecture for Powered Landing Scenario

In this section, the integration of the Desensitized Optimal Control (DOC) theory with the scenario outlined above is presented. As discussed in Section 3.4, the optimal implementation strategy involves augmenting the state vector with the $\mathbf{\Lambda}$ vector, the $\mathbf{\Omega}(t)$ components, and the elements of the Riccati Matrix. This augmentation is essential for acquiring the optimal feedback gains required for solving the problem at hand.

The subsequent section provides an overview of the architecture, as the mathematical model chosen allows for variations in numerous variables. The general algorithmic implementation presented in this section aims to be as comprehensive as possible, consolidating contributions from various sources, including [42], [16], and [41]. The fundamental concept is to establish the most complete version of the DOC procedure and subsequently tune it after empirically characterizing the entire process. This section includes the comprehensive formulation of the DOC approach applied to the Powered Landing Problem, taking into account the outcomes of Section 3.4.3.

The cost function states as:

$$J = \min_{u_x, u_y, u_h} - m(t_f) + \alpha \left(\int_{t_0}^{t_f} \sum_{k=1}^{z_S} \alpha_k \|\mathbf{\Lambda}(t)_k\|_2 \, dt + \beta \cdot \mathbf{\Omega}_3(t_f) \right) \tag{4.6}$$

It is worth noting that, in contrast to Eq. (3.13), the two terms related to sensitivity are combined and pre-multiplied by a common factor, denoted as α, while the relative weights assigned to different terms are governed by non-negative coefficients, namely, α_i and β_l, respectively. This approach offers two advantages: firstly, it enhances the method's flexibility giving different weights to different terms, and secondly, once the values of α_i and β_l are determined, the influence of DOC on the cost function can be effectively controlled through the single parameter α.

The state dynamics is described by Eq. (4.1), subject to the constraints presented in Eq. (4.2) and Eq. (4.3), subject the boundary conditions in Eq. (4.4).

The evolution of the $\mathbf{\Lambda}(t)$ terms follows the differential equations outlined in Eq. (3.22), subject to the final conditions specified in Eq. (3.23). The matrices A and B are defined in Appendix A. There are no constraints on $\mathbf{\Lambda}(t)$ components; however, a precise definition is required. As mentioned in Section 3.4.1, it is asserted that each $\mathbf{\Lambda}(t)$ function corresponds to a linear function of the final state. Nevertheless, a linear combination of multiple sensitivities may result in cancellations within the cost function, potentially nullifying the sensitivity penalization. To address this, $b(\boldsymbol{x}(t_f))$ functions are defined as single components of the final state, for instance, setting $b(\boldsymbol{x}(t_f)) = h(t_f)$. This leads to $\boldsymbol{\Lambda}(t_f) =$ [0, 0, 1, 0, 0, 0, 0], aligning with the physical states of the system as used in Eq. (4.1). There are no mathematical constraints on the $\boldsymbol{\Lambda}(t)$ functions that need to be penalized, and the choice of which sensitivities to penalize, in conjunction with the relative weights α_k, forms a critical part of the tuning process. If no $\boldsymbol{\Lambda}(t)$ terms are penalized in the cost function, they are not included in the states of the problem.

In Section 3.1.2, $\boldsymbol{\Omega}(t)$ has been defined as the sensitivity of the state concerning variations in a specific variable denoted as p. However, since only the Thrust is considered as the uncertain parameter in the problem, the corresponding $\boldsymbol{\Omega}(t)$ term is defined. Therefore, $z_{\boldsymbol{\Omega}} = 1$. Furthermore, as better explained in Section 4.3.5, the implemented version of the algorithm just penalizes the third component of $\boldsymbol{\Omega}(t_f)$, the one related with the final altitude. Since it is just a numerical positive value, no norms or absolute values are needed. If β is set to zero, the components of $\boldsymbol{\Omega}(t)$ are removed from the problem state. The matrices A, B, and C for dynamics, in accordance with the differential equation in Eq. (3.17) and referencing the dynamics in Eq. (4.1), are provided in Appendix A.

After addressing some key aspects related to sensitivity terms, the strategy for computing feedback gains is discussed. As mentioned in Section 3.4.2, these gains are computed through the Riccati matrix, where its upper triangular components are added to the state. However, in the context of this application, a feedback correction for mass error is not applicable. Therefore, the Riccati Matrix components are reduced to 21, instead of the 28 components found in the 7x7 case. The propagation is carried out following Eq. (3.25), with the reduced matrices $\tilde{A}$ and $\tilde{B}$, which are sub-matrices of A and B. All propagated elements are collected in the $\boldsymbol{P}_u$ vector and added to the state, but only in the closed-loop scenario.

The augmented state is formed through the concatenation of different elements. Depending on the specific case, it can vary in size and composition. As an illustrative example, the augmented state for the problem with $z_S = 1$ is provided.

$$\boldsymbol{x}_{aug}(t) = \begin{bmatrix} \boldsymbol{x}(t) \\ \boldsymbol{P}_u(t) \\ \boldsymbol{\Lambda}(t) \\ \boldsymbol{\Omega}(t) \end{bmatrix}$$

4.3. Method Characterization

Once the architectural framework is established, meeting all specified requirements while aiming at optimizing numerical efficiency, it necessitates a precise characterization to achieve optimal performances. This process is complex, as it involves tuning various design variables by the user. This section aims to provide a general overview of the impact of these variables on performance, outlining the advantages and disadvantages of each choice. However, it is important to note that the outcomes of this section are based on the initial conditions presented in Table 4.1 and may not apply to different scenarios.

When characterizing a new procedure, different alternatives are evaluated, and the best one is chosen based on a predefined criterion. The more quantitative the criterion, the easier the tuning process becomes. However, due to the inherent presence of conflicting terms within the objective function of the optimization problem, as illustrated in Eq. (4.6), there is no single absolute criterion that can comprehensively address its complexity. Mass optimization and sensitivity reduction are competing objectives, and both must be considered simultaneously.

Additionally, it is important to account for uncertainties in the analysis, and the performance of any procedure is typically assessed through Monte Carlo (MC) analyses to measure the dispersion around the nominal landing point. While MC tools can provide reliable results, they often require extensive CPU time, making them challenging for a detailed characterization and tuning process. As explained in Section 3.3, sensitivity and covariance are interconnected, so reducing sensitivity results in a smaller final covariance hyper-ellipsoid. Given the expected perturbations in the analyzed scenario, it is reasonable to evaluate the solution's quality based on its sensitivity to these expected perturbations, better than using MC analysis.

For all the reasons mentioned above, since the primary source of uncertainties lies in the initial states, the most effective way to estimate the state's covariance in advance is by assessing the sensitivity of the state at time t to the initial state, as defined as:

$$S(t, t_0, \boldsymbol{x}_0) = \frac{\partial \boldsymbol{x}(t)}{\partial \boldsymbol{x}(t_0)}$$

As the primary focus of the optimization is the covariance of the final state, it is essential to evaluate this sensitivity matrix at the final time. Consequently, the appropriate measure of the final state's covariance is determined by:

$$S(t_f) = S(t_f, t_0, \boldsymbol{x}_0) \tag{4.7}$$

This choice might appear unconventional because this term does not align with the sensitivity-related term penalized in the cost function. While, for physical and convergence reasons, $\boldsymbol{\Lambda}(t)$ retains significance, it does not exclusively capture the impact of perturbations to the initial state on the final state, as it penalizes sensitivities at each time in an integral form.

Furthermore, when perturbations in the nominal thrust value are considered, the suitable measure of their effect on the final state is represented by $\boldsymbol{\Omega}(t_f)$.

The entire process can be characterized as perturbation-driven. Once the scenario and the primary sources of uncertainties are established, the design is guided by the minimization of sensitivities concerning these perturbations. The criteria presented here are validated in the subsequent chapter (Chapter 5), demonstrating the relationship between covariance prediction (i.e., sensitivity regarding a specified source of uncertainty) and the covariance derived from the Monte-Carlo (MC) analysis.

Additionally, another significant physical interpretation of sensitivity is provided: in the presence of initial state uncertainties, the best a priori estimate of the state covariance is determined by the expression detailed in Eq. (4.7). As $S(t_f, t_0, \boldsymbol{x}_0)$ is an $n \times n$ matrix, some synthetic performance indicators of solution efficiency should be offered. The element $S_{i,j}$ within this sensitivity matrix characterizes the sensitivity of the ith state component with respect to the jth component of the initial state. Furthermore, since the expected perturbations in position are approximately one order of magnitude larger than those in velocity, an appropriate measure of the standard deviation of the ith state is:

$$S_i(t) = \sqrt{\sum_{j=1}^{6} w_j \cdot S_{i,j}(t, t_0, \boldsymbol{x}_0)^2} \qquad w_j = \begin{cases} 100, & \text{for } j = 1, 2, 3 \\ 1, & \text{for } j = 4, 5, 6 \end{cases} \tag{4.8}$$

The maximum value for the summation index j is set to six, as mass perturbations are not taken into account and, therefore, they do not have any sensitivity impact. Consequently, according to Eq. (4.8), the accurate estimator for the final covariance of the ith state is represented by the $S_i(t_f)$ expression.

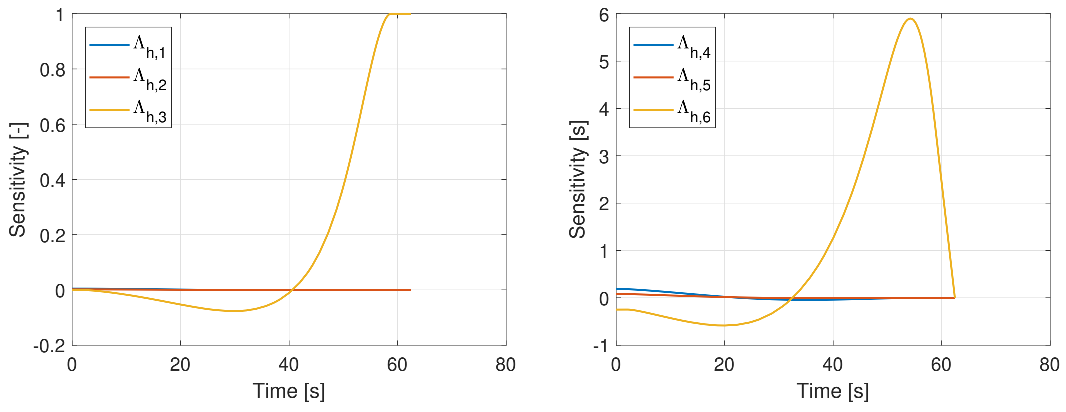
As the method's definition and characterization are not straightforward, the subsequent sections attempt to offer a broad overview of the method in the most clear manner possible. However, it is possible that some concepts are introduced before they are fully explained: the complete understanding of the method is achieved only at the end of the chapter.

4.3.1. Sensitivity Analysis

Before delving into the detailed algorithmic characterization, it is essential to offer some insights into sensitivities. These observations are valuable for gaining a profound understanding of the problem's dynamics and for presenting initial results.

The i$^{\text{th}}$ component of $\mathbf{\Lambda}(t_f)$, measures the sensitivity of $b(\boldsymbol{x}(t_f))$ with respect to the i$^{\text{th}}$ state in time. If $b(\boldsymbol{x}(t_f)) = h(t_f)$, the third component of $\mathbf{\Lambda}_h(t_f)$ is equal to one since:

$$\frac{\partial b(\boldsymbol{x}(t_f))}{\partial x_3(t_f)} = \frac{\partial h(t_f)}{\partial h(t_f)} = 1$$



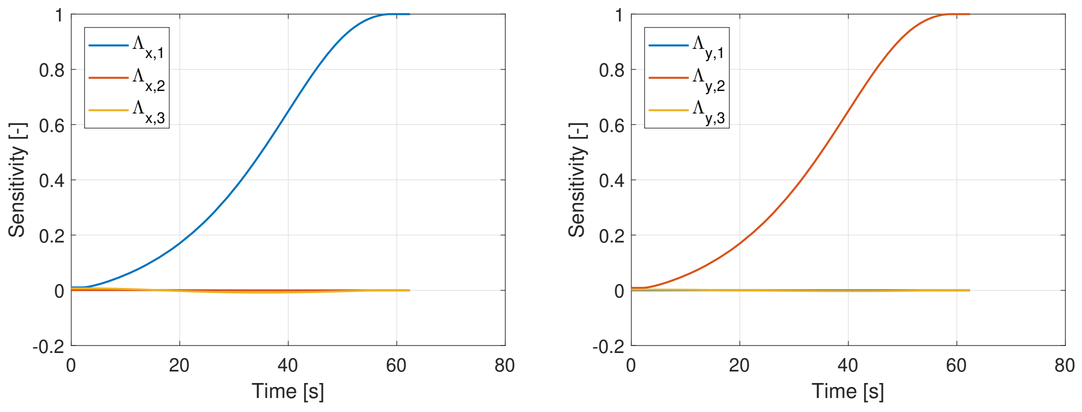
(a) Sensitivities with respect to positions (b) Sensitivities with respect to velocities

Figure 4.2: Sensitivities of the final altitude, $h(t_f)$

The plot provided illustrates the time evolution of $\mathbf{\Lambda}_h(t)$, the sensitivity of $b(\boldsymbol{x}(t_f)) = h(t_f)$. It is evident that the final condition is fulfilled since the third components to $\mathbf{\Lambda}(t)$ is equal to one at the final time, whereas the others are all null.

A further insight about sensitivity is the following: as clear from Eq. (4.1), the equations of motion in the three spatial directions are nearly independent, and cross sensitivities are minimal, with only a slight coupling due to the mass value. This is achieved through the problem dynamics and due to the design choice of having both LQR weighting matrices (Q and R) are chosen as diagonal. Therefore, in Fig. 4.2, h is sensitive to just h itself and v_h, linked with the last component of $\mathbf{\Lambda}(t)$, as reported in Fig. 4.2b.

Moreover, because of its symmetric formulation, if the same LQR weights are chosen for x and y components, the dynamic evolution of the sensitivities of various elements are nearly identical. The following graph displays the components of $\mathbf{\Lambda}_x(t)$, the sensitivity of the function $b(\boldsymbol{x}(t_f)) = x(t_f)$, and $\boldsymbol{\Lambda}_y(t)$, which represents the sensitivity of $b(\boldsymbol{x}(t_f)) = y(t_f)$. It is clear that they exhibit similar patterns, though they belong to different elements, as could be expected, since the sensitivity of a final position is just influenced by the corresponding position and velocity. For instance, the first component of $\boldsymbol{\Lambda}_x(t)$ significantly deviates from zero, since it impacts the sensitivity of $x(t_f)$ due to its connection with $x(t)$. The trends of $\boldsymbol{\Lambda}_x(t)$ and $\boldsymbol{\Lambda}_h(t)$, shown in Fig. 4.2a, differ when different Q weights are used.



(a) Sensitivities of $x(t_f)$ (b) Sensitivities of $y(t_f)$

Figure 4.3: Sensitivities with respect to the positions

4.3.2. Saturation Handling Strategy

A significant contribution from the work of Shen et al. in [16] is the development of a strategy to address the saturation issue. As discussed in Section 4.1.2, the nominal solution for the standard case, also referred to as noDOC, consists of a max-min-max control profile. However, in the presence of saturation, any feedback control is prone to be truncated, due to the saturation limits. The proposed approach involves selecting control gains that tend to decrease as the nominal control profile approaches these limits, ultimately reducing the feedback control to zero. This is achieved by defining a coefficient denoted as $\eta(t)$, which diminishes the feedback as the nominal solution approaches the saturation bounds. In this work, η is addressed to as the Feedback Capability Factor since it embeds information about the system's ability to guarantee feedback control.

$$\eta(t) = \frac{4 \cdot (u^*(t) - u_{min}) \cdot (u_{max} - u^*(t))}{(u_{max} - u_{min})^2}$$

The definition of $\eta(t)$ can take any functional form, provided that it fulfills two essential constraints: it must have zero values at the control bounds and must be non-negative in the entire interval. The chosen function fulfills these requirements, as it is the simplest possible and aligns with the existing literature. While it cannot guarantee that feedback control saturation will never occur, it effectively reduces the frequency of such events.

This design choice is robust, but it comes with a trade-off: to minimize sensitivity, no feedback control is allowed in specific segments of the trajectory when the control input u reaches the maximum or minimum bounds. In a real-world scenario, this is strategy is not implementable, as perturbations can arise throughout the entire trajectory, and the absence of feedback gains could lead to significant consequences. It is possible to mitigate this issue by modifying the $\eta(t)$ function, defining new bounds, or leaving some margin for feedback. However, these alternatives are not explored in this work, as the goal here is to provide a demonstration rather than a comprehensive implementation design. In practice, real design choices are influenced by various criteria beyond the scope of this work. This choice is also aided by the fact that, in the worst case, the considered system is marginally stable and not entirely unstable. Therefore, even in the absence of feedback gains during a short segment of the trajectory, perturbations do not have a catastrophic impact. Due to the introduction of the Feedback Capability Factor, Eq. (3.14) becomes:

$$\boldsymbol{u}(t) = \boldsymbol{u}^*(t) + \eta(t) \cdot K^*(t)\,(\boldsymbol{x} - \boldsymbol{x}^*(t)) \tag{4.9}$$

This design choice not only influences the feedback control but also has an impact on the differential equations governing $\boldsymbol{\Lambda}(t)$, as expressed in Eq. (3.22), which is modified as follows:

$$\frac{\partial \boldsymbol{\Lambda}(t)}{\partial t} = -\left(A + \eta \cdot B \cdot K\right)^T \cdot \boldsymbol{\Lambda}(t) \tag{4.10}$$

While Eq. (3.17) for $\boldsymbol{\Omega}(t)$ is:

$$\frac{\partial \boldsymbol{\Omega}(t)}{\partial t} = \left(A + \eta \cdot B \cdot K\right) \cdot \boldsymbol{\Omega}(t) + \boldsymbol{C}(t)$$

It becomes evident that the solution profile diverges from the max-min-max profile. This deviation can be explained by two main factors: one is mathematical, while the other is tied to the physical interpretation of the problem.

As a result of Eq. (4.9), the control no longer appears linearly in the cost function of the problem, and the solution cannot be of the max-min-max form [16]. Furthermore, as the nominal control approaches the control bounds, $\eta(t)$ factor tends to zero. Since one of the objectives in Eq. (4.6) is to reduce sensitivity through the feedback effect, the optimal solution is encouraged to stay away from the control bounds, thereby preserving some available feedback to diminish sensitivity. On the other hand, diverging too much from the max-min-max solution results in a significant increase in fuel consumption. This trade-off between fuel savings and sensitivity reduction is a critical aspect that must be considered throughout the characterization and tuning process.

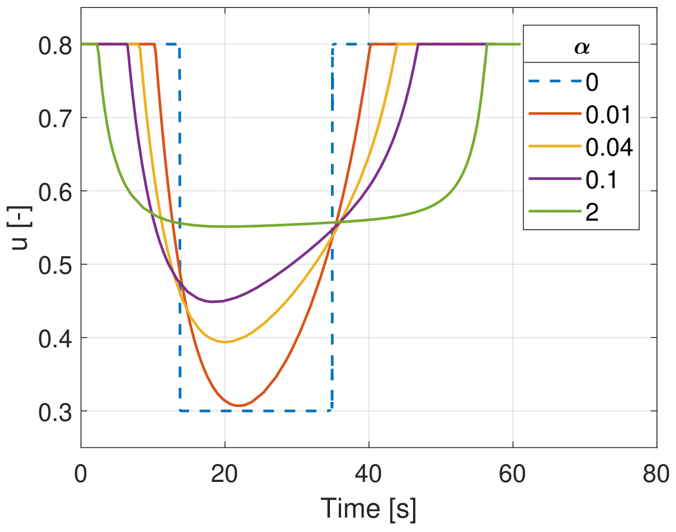


Figure 4.4: DOC Solutions for increasing α weights

Some control profiles are presented here as illustrative examples to demonstrate the behavior of the solution for increasing values of α, assuming $\beta = 0$ and $z_S = 1$, which correspond to the penalization of sensitivity for a single function of the final state according to the cost Eq. (4.6). In the presented case, it was chosen to penalize the sensitivity of the final y. The control profiles are shown, as well as the corresponding $\eta(t)$ trends.

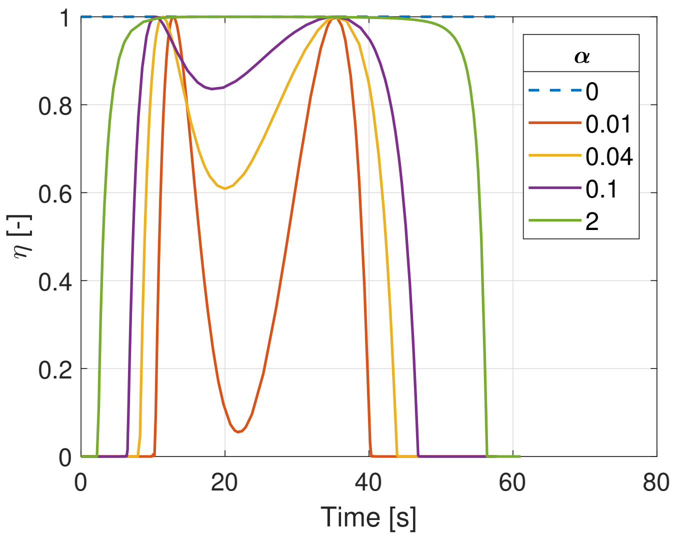


Figure 4.5: $\eta(t)$ for increasing α weights

As mentioned in the preceding section, the method's effective performance is measured by Eq. (4.7) and is accurately assessed on a component-wise basis by $S_i(t_f)$. However, it is equally important to appropriately measure the mass increase associated with the sensitivity penalization in order to have a comprehensive understanding of the method. To achieve this, a Pareto-like [46] plot is presented here: it is used in the subsequent sections as well to offer a clear graphical interpretation of the performance of DOC approach.

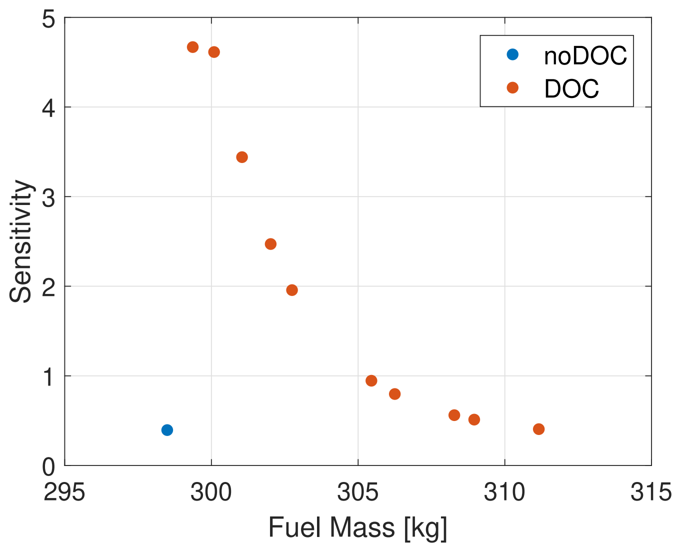


Figure 4.6: $S_3(t_f)$ for increasing α weights

This figure illustrates the sensitivity $S_3(t_f)$, which serves as an a-priori indicator of the algorithm's efficiency in terms of the final altitude sensitivity, along with the corresponding nominal fuel mass consumption for increasing values of α, growing moving towards the right. The different color refers to the noDOC case ($\alpha = 0$)

In comparison to the scenario with noDOC, we observe a notable increase in the sensitivity of the final state for low values of α. This might seem counter intuitive, as one would expect a decrease in final state sensitivity due to the penalization of the sensitivity-related term in the cost function with non-zero α values. However, the results presented here are entirely meaningful. As shown in Fig. 4.4, for low α values, the optimal solution closely resembles the max-min-max profile. Consequently, as illustrated in Fig. 4.5, the parameter $\eta(t)$ remains close to zero, resulting in a nearly open-loop solution.

In the case of noDOC, $\eta(t)$ is not explicitly defined, but it is effectively equivalent to one, representing a full-feedback scenario. The interpretation of the results is straightforward: for similar nominal control profiles, the full-feedback noDOC case is more effective in reducing sensitivities compared to the DOC quasi-open-loop case with $\alpha = 0.01$, mainly due to the strong influence of feedback gains. To make the DOC procedure effective, α

needs to be set sufficiently high to ensure that the solution deviates from the max-min-max profile. As α increases, $\eta(t)$ takes on values other than zero because sensitivities must be further reduced through feedback gains, allowing for more feedback control. This leads to the reduction in sensitivity depicted in Fig. 4.6. However, this improvement comes at the cost of increased fuel consumption. It is worth noting that these predictions do not consider the possibility of saturation. Consequently, the noDOC performances are considered ideal, as saturation only occurs in the MC, providing a proper measure. The issue is addressed in detail in Chapter 5.

In the absence of $\eta(t)$, the nominal control value remains closely aligned with the max-min-max solution, rendering the solutions impractical for real-world scenarios. Furthermore, they do not exhibit significant differences in sensitivity compared to the noDOC solution.

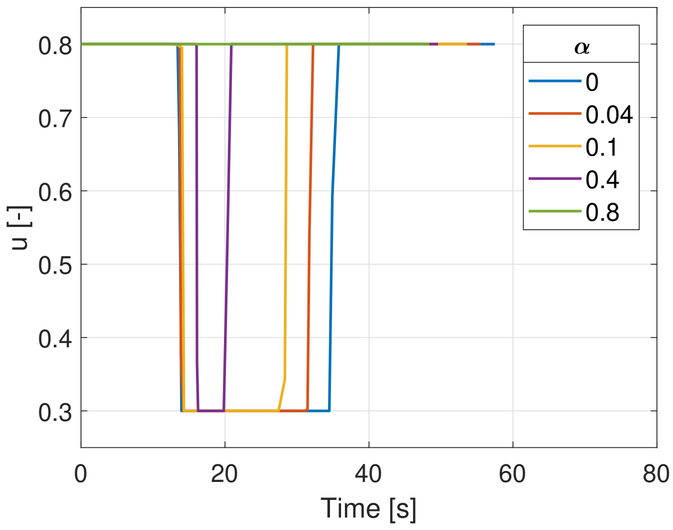


Figure 4.7: Control trends without η for increasing α

Figure 4.7 shows the different results when $\mathbf{\Lambda}(t)$ is propagated with Eq. (3.22) rather than Eq. (4.10). For increasing α values, the optimization attempts to minimize fuel consumption by reducing flight time through reduction of the minimum control part of the trajectory. This results in fully saturated control profiles and does not lead to major improvements in sensitivity.

In summary, the DOC strategy for the powered landing scenario can, in some cases, lead to an increase in sensitivity. The feedback capability factor plays a crucial role in ensuring the effective integration of Powered Landing and DOC, but it can also be the element that causes performance deterioration when α is small. It introduces additional degrees of freedom to the optimizer, giving the possibility of significantly impacting sensitivity, which is the core objective of the DOC approach.

4.3.3. Marginal DOC

In the Marginal DOC approach, a significant improvement over the implementation presented in [16] is introduced, involving the addition of an extra multiplicative coefficient called ν, which is referred to as the Marginal DOC coefficient.

The feedback capability factor, $\eta(t)$, is, by definition, constrained to the range of 0 to 1. The equation governing the trend of $\eta(t)$ over time is arbitrary, but its presence leads to a reduction in the feedback.

However, when the feedback capability factor is multiplied by a scalar value ν greater than one, a portion of the original feedback control level is restored. The new $\eta(t)$ function, denoted as $\tilde{\eta}(t) = \nu \cdot \eta(t)$, satisfies the two essential requirements reported in Section 4.3.2 and is a valid candidate for the problem under consideration. While the parabolic shape of $\eta(t)$ is retained, ν acts as an amplifier of the feedback gains. Although $\eta(t)$ alters the trend of feedback gains over time, the recovery of feedback can be quantified in terms of its norm across the entire domain.

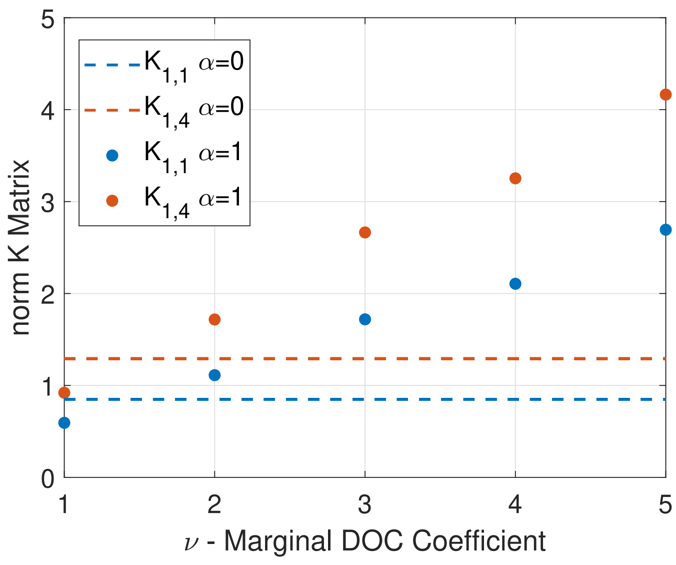
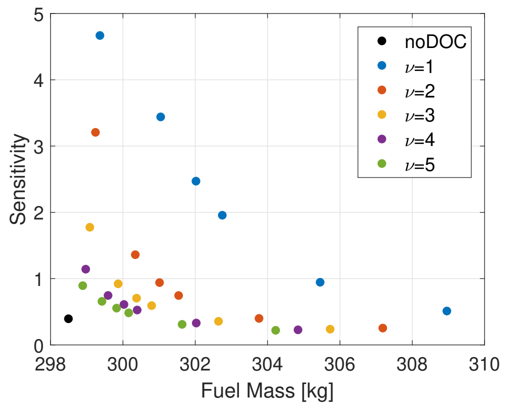


Figure 4.8: Feedback gains norms for different ν values

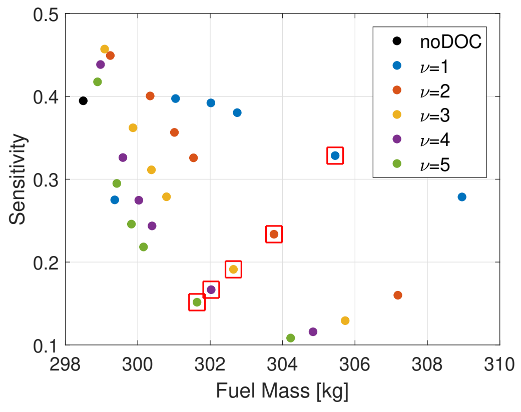
This illustration is intended to offer a consolidated measure of how ν impacts the feedback gains. It is important to bear in mind that the time trends of the gains are significantly influenced by $\tilde{\eta}$, and as a result, they may exhibit various behaviors over the entire trajectory. However, as evident from the diagram, when ν is set to one, the feedback gains without the DOC (where $\alpha = 0$) are greater than those with the DOC (where $\alpha = 1$). As the Marginal DOC Coefficient increases, the magnitude of the feedback gains increases in proportion to ν, fully restoring the total amount of feedback control and even surpassing the initial feedback values.

The effectiveness of the suggested approach is assessed by examining its impact on the sensitivity of the final state. Fig. 4.9 illustrate that, as the ν parameter increases, the final sensitivities decrease, resulting in an enhancement of the overall performance of the DOC procedure. This is attributed to the fact that an increase in feedback gains reduces the sensitivity of the final state to perturbations, as more substantial feedback control actions are applied. This design choice also has implications for the nominal control profile, as there is no longer a need for high η values to reduce sensitivity, since the impact of feedback gains on sensitivities is already amplified by ν. Consequently, for the same α value, the control profile remains further from the midpoint between the minimum and maximum control profiles, leading to a reduction in mass consumption. This relationship is depicted in Fig. 4.9b, where, for the same α value, marginal DOC with increasing ν manages to both decrease fuel consumption and improve performance. The points with constant $\alpha = 0.3$ are marked in red.

Since the proposed strategy only affects the DOC cases, the noDOC case is presented in black, as it is common to all the different scenarios. As usual, an increase in the fuel mass toward the right is associated with an increase in the α coefficient.



(a) Sensitivities of $h(t_f)$



(b) Sensitivities of $v_x(t_f)$

Figure 4.9: Sensitivities for different ν values

The final aspect to consider pertains to the maximum limit for the ν value. For positions, its increase becomes less significant after a certain threshold, as the curves become nearly identical for high ν values. In contrast, velocity performance steadily improves, and even for high ν values, various curves appear to converge. While theoretically, there may be no strict upper limit on the ν value, other factors must also be considered.

When determining the upper bound for the value of ν, two crucial criteria come into play. The first criterion is the gain margin, which is defined as the maximum multiplicative factor for the feedback gains that ensures system stability. Consequently, $\tilde{\eta}(t)$ should not exceed this value. The second criterion is based on the fact that, if ν becomes too large, $\tilde{\eta}(t)$ exhibits a steep variation near the control bounds, posing the risk of exceeding these bounds since the feedback gains are multiplied by high values. This choice would be counterproductive, as $\tilde{\eta}(t)$ is designed to address the saturation issue. Nevertheless, the second criterion can be assessed through Monte Carlo (MC) simulations. In practice, the first criterion tends to be more stringent due to the stability proprierties of the system and is thus considered a priority in determining the upper limit for ν.

4.3.4. LQR Tuning

As commonly recognized, the proper adjustment of feedback gains is crucial for achieving optimal performance for every kind of problems, especially in the presence of high disturbances. However, in the DOC formulation presented in this work, their impact is even more relevant. This is because they directly influence sensitivities, which are penalized in the cost function, and have also an impact the nominal control profile shape. This interplay arises due to the partial coupling of the Guidance and Control processes. Consequently, when selecting Q and R matrices, several additional criteria must be considered.

The process of tuning LQR controllers in itself is complex. When trying to improve disturbance rejection for one state, it often comes at the expense of feedback performance for other states, primarily due to constraints on total control effort, such as saturation or physical limits. In the case of DOC-LQR tuning, the complexity is further stressed, as some trends are less predictable. While some aspects align with established LQR properties, others are deeply related to the interplay between Guidance and Control.

In common applications, LQR tuning is a complex process. Many requirements can be considered, but in general the tuning process turns out to be a saturation-driven process, trying to use all the available control without exceeding the prescribed bounds. However, saturation is not an issue for the proposed DOC formulation, since feedback capability factor, defined in Section 4.3.2, offers a good handling strategy for saturation.

Contrary to the saturation-driven approach, DOC-LQR tuning can be seen as a sensitivity-driven process, since the goal is to reduce certain sensitivities. This method appears particularly suitable for the DOC procedure; however, measuring sensitivities to perturbations of the final state can serve as a valuable criterion for tuning controllers for other applications. Utilizing sensitivities as design criteria offers a more structured approach compared to the conventional trial-and-error method.

The integration of Guidance and Control introduces further complexities into the tuning process. The nominal trajectory is affected by the feedback gains, influencing both its shape and performance. Additionally, the CPU time required for obtaining the optimal trajectory is influenced by LQR design choices. In essence, characterizing the coupling between Guidance and Control is challenging, given the potential emergence of unmodeled behaviors. This section seeks to offer empirical results and practical insights for characterizing the tuning procedure.

The primary objective of this entire procedure is to minimize the final covariance, and therefore, $S(t_f)$, or a combination of its elements, is employed as the evaluation criterion for all design choices, as indicated in Eq. (4.8).

As a design guideline, it was decided to configure all the weighting matrices of the LQR as diagonal. Although this imposes a constraint on the allowable optimal solutions, it was done to maintain the independence of all components and enhance predictability in the system's general behavior. Moreover, this choice automatically satisfies the requirements of symmetric positive definiteness for the weighting matrices in the LQR architecture when the diagonal values are positive.

R Impact

The R matrix provides a indication on the maximum feedback control magnitude to be applied. According to Eq. (3.26), it penalizing the potential variations in u, the control variable. As a general guideline, a larger norm for the R matrix indicates more constrained feedback control efforts, often resulting in a reduction of the feedback gains. In classical LQR tuning, this matrix is of significant importance as it is defined based on control effort limitations. However, in the context of DOC-LQR tuning, where saturation is no longer a constraint, theoretically, the R matrix values could be reduced with no bounds to achieve optimal disturbance management. Strong feedback can effectively handle disturbances, as it leads to faster system responses. Nonetheless, following classical control theory, high gains may lead to a decrease in noise rejection performance. In practical applications, various noise sources, such as sensors and model parameters, can introduce uncertainties, and high gains might exacerbate the feedback effects. While these factors are not explicitly addressed in this work, they are essential to consider in real-world tuning scenarios.

In the specific scenario under consideration, the primary limitation on reducing the R values is associated with the inherent behavior of LQR feedback gains. As mentioned, reducing R results in an increase in the gains. This effect occurs throughout the entire trajectory but has a more pronounced impact in the final stages. Assuming that the R matrix is diagonal with uniform diagonal terms, its norm can be conveniently expressed in terms of the diagonal values. As illustrated in Fig. 4.10, the feedback gain matrix's norm increases as the R value is reduced. This has a further significant impact on the final stages. In a noDOC problem, this could lead to physical issues since a substantial amount of control is deferred to the end of the trajectory and not evenly distributed across the entire path. In the DOC framework, this effect becomes more pronounced, as high values of feedback gains at the end can introduce discontinuities throughout the problem, resulting in a substantial increase in CPU time or even the failure of NLP convergence. The following figure reports the feedback gain matrix norm, defined as:

$$K_{norm}(t) = \sum_{i=1}^{3}\sum_{j=1}^{6} \|K_{i,j}(t)\|_2$$

Even if the elements are not homogeneous, some interesting trends are shown in the picture.

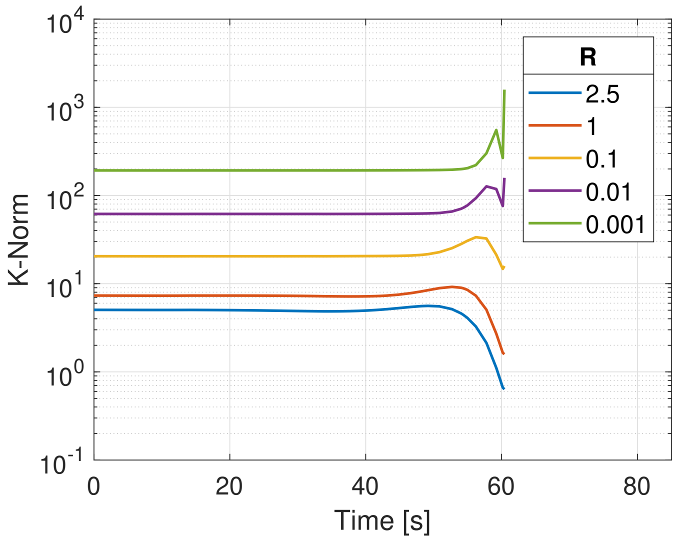


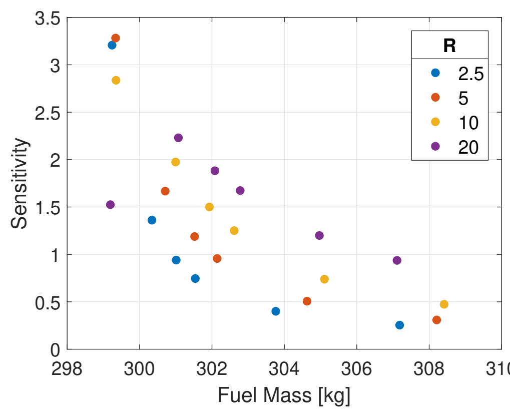
Figure 4.10: Gain Matrix Norm for increasing R values

Figure 4.10 shows that the magnitude of the feedback gains increases as the reduction of R progresses throughout the entire trajectory.

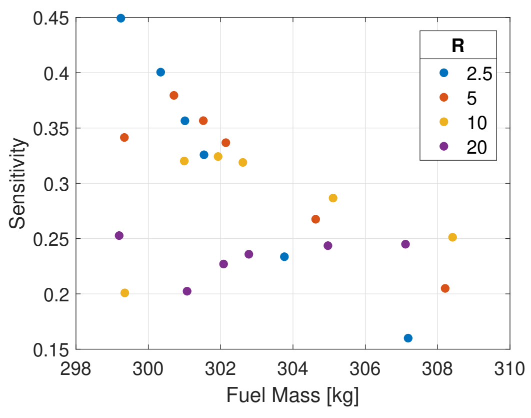
In order to mitigate the final stages issue, two design choices can be considered: either maintaining the R value within reasonable bounds or permitting the R value to increase along the trajectory for a more even distribution of feedback efforts, as demonstrated in [16]. While the latter strategy can enhance performance, it is not sustainable in GPOPS, as time-varying R values lead to a significant increase in CPU time. Therefore, the proposed design adopts constant R values, which need to be optimized.

As can be inferred from the previous section, the design optimization must account for competing factors: the sensitivities of the final states and the CPU times required to obtain the trajectories. Since this work focuses on obtaining a computationally efficient version of DOC, the second criterion plays a pivotal role. The reduction of the values of R leads to an increment of CPU time; simultaneously, if the absolute values of R are reasonably low, further reductions do not yield significant improvements in performance. Consequently, the priority is placed on maintaining CPU time below a reasonable threshold.

The following figures depict the sensitivities of two final states for various R values. A reduction of R, in a reasonable interval, leads to general performances improvements. However this can not take place for any component, due to the complex coupling between DOC and LQR. This is quite clear from the following figures: all the position sensitivities are reduced as R decreases, as well as the vertical velocity one, not shown here, whereas the v_x and v_y sensitivities have not clearly predictable behaviours are R grows. However, is worth mentioning that the reported plot are influenced by the Q values and that the trends should not be intended as general. In this scenario the worsening of performances on v_x and v_y linked with the reduction of R is way less relevant that the one associated with the reduction of the sensitivity of the position, in absolute sense and with respect to the expected perturbations, leading to the choice of keeping low R values.



(a) Sensitivities of $h(t_f)$



(b) Sensitivities of $v_x(t_f)$

Figure 4.11: Sensitivities for different R values

Q Impact

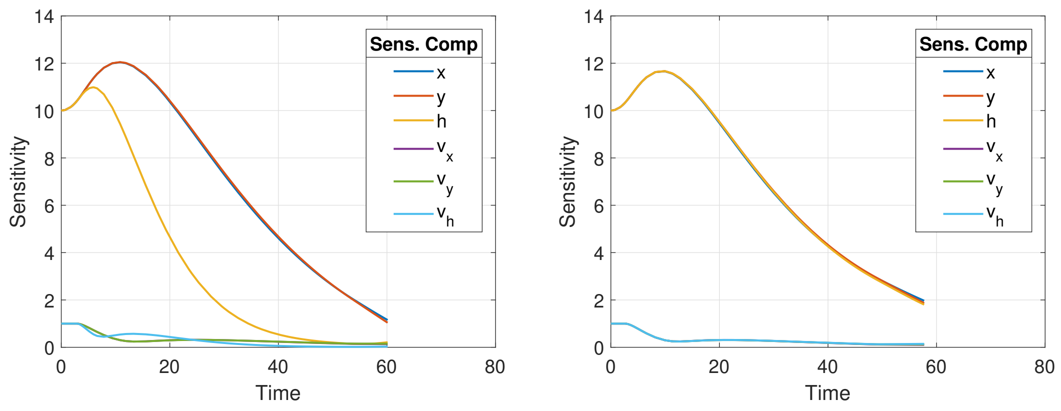
The next element to be addressed in tuning is the Q matrix, which is a well-known component typically adjusted based on the maximum allowable errors for each state. The higher the weight assigned to a state, the more feedback effort will be allocated to the associated gains. However, it is important to consider the weight on an individual component relative to all the other elements of the Q matrix, rather than in absolute terms. Since Q matrix is set to be diagonal, there is no difference between considering the full matrix itself or its diagonal vector, $\boldsymbol{Q}_d$. A good measure of the effective weight on a specific component, denoted as i, is given by the ratio $\boldsymbol{Q}_{d,i}/||\boldsymbol{Q}_d||_2$.

Two additional factors need to be considered. First, due to the dynamics of the problem, some states are time derivatives of others, such as velocities being derivatives of the position states. Therefore, proper tuning accounts for this coupling. Larger feedback gains on velocities can reduce the final errors of the corresponding positions since a significant portion of the position error is linked to the integral of velocity errors. However, the opposite relation, i.e., larger gains on position reducing the sensitivities of corresponding velocities, is not always valid. In the case under consideration, this implication is valid only for the vertical direction and will be utilized in the tuning procedure in Chapter 5.

The second important factor to consider is that LQR weights should be adjusted in accordance with the R value. In summary, the R value measures the total control effort, and the Q matrix determines how control is distributed among the states. However, to achieve GPOPS convergence and avoid the high-gain effects discussed in the previous section, it is advisable to select reasonable, not excessively large, values for the Q weights. The process is self-regulating in terms of performance because a strong penalty on one state can deteriorate the performance of all other states in terms of disturbance management.

Figure 4.12 demonstrates the effect of choosing different weights: individual sensitivities are penalized in varying degrees, leading to different sensitivity time behaviors. Since the problem displays a significant level of symmetry in three directions, using the same weights for different components leads to similar sensitivity patterns for both respectively positions and velocities. These patterns exhibit minor variations attributed to distinct initial conditions. The reason behind this behavior can be easily explained mathematically. The sensitivity differential equation chosen, as described in Eq. (3.22), stems from a series of analytical transformations applied to Eq. (3.21). Since sensitivities for various final states correspond to different rows within the complete sensitivity matrix, and all rows within a matrix follow the same dynamics, the dynamics of diverse sensitivities remain identical, except for the feedback gain values. Consequently, assigning the same

Q weights to two symmetric states (e.g., x and y) results in nearly identical sensitivity trends over time. This is clearly illustrated in both figures where the sensitivities of x and y, as well as v_x and v_y, closely overlap.



(a) Sensitivities with $\boldsymbol{Q}_d = 2,\ 2,\ 30,\ 1,\ 1,\ 0.5$ (b) Sensitivities with $\boldsymbol{Q}_d = 3,\ 3,\ 3,\ 3,\ 3,\ 3$

Figure 4.12: Sensitivities for different Q diagonals

As seen in Figure 4.12b, it is evident that the sensitivity of h closely resembles that of the other positions due to their similar dynamics. In the linearized version of the system, the gravitational terms are eliminated, resulting in the same differential equations for all three directions. Similar behaviors are observed for the position. However, when individual weights change, as shown in Figure 4.12a, different behaviors emerge. The sensitivity of h, with a weight of 30, dramatically diminishes along the trajectory, leading to smaller final values. It is worth noting that even though the weight on v_h is 0.5, the final sensitivity in Fig. 4.12a is smaller than that in Fig. 4.12b. This discrepancy can be primarily attributed to the increase in the weight of h in the Q matrix, as previously mentioned. While various speculations involving different Q matrices can be explored, it is important to remember that the ultimate outcome of interest is the final sensitivity value. Consequently, the tuning of the Q matrix should prioritize the final sensitivities as design criteria over their temporal trends.

4.3.5. Desensitization Strategies

Once an architecture is well-established and tested, a central aspect of its usage is the selection of elements to be desensitized in order to achieve the desired performance. As described in Section 3.4, the formulation allows for the desensitization of one or multiple $b(\boldsymbol{x}(t_f))$ functions. These functions must be linear and scalar in nature, and their unique form to prevent cancellations is:

$$b(\boldsymbol{x}(t_f), t_f) = \boldsymbol{x}(t_f)_i$$

where, $\boldsymbol{x}_i$ represents a component of the final state. Due to this definition of the penalized function, the penalization of $b(\boldsymbol{x}(t_f), t_f) = \boldsymbol{x}(t_f)_i$ can also be referred to as the "penalization of $\boldsymbol{x}(t_f)_i$".

For each $b(\boldsymbol{x}(t_f))$ function to be desensitized, seven states are introduced for each penalized function. It is worth noting that the most straightforward solution, desensitizing the entire final state, would add 49 extra state variables, resulting in a significant increase in CPU time. However, desensitizing some states might be irrelevant or even lead to a loss of performance. Therefore, the optimization of the architecture aims to achieve the best performance with the fewest additional variables. The goal is to identify the component(s) that can provide the best results in terms of sensitivity of the final states when desensitized while minimizing the number of required state variables.

Moreover, not all design choices lead to well-behaved or converging solutions, as this depends on the problem's dynamics. Incorrect design choices or badly tuned weighting factors can lead to numerical issues. Another crucial point to emphasize is that the performance of the desensitization strategy is influenced by the LQR tuning. With the same regulator, the desensitization of one component might smoothly converge to achieve good performance, while desensitizing another component could lead to sub-optimal or non-converging solutions.

A final consideration is that both desensitization and feedback gains contribute to the minimization of final state sensitivities in a coupled manner. This is not ideal from an engineering perspective because there is a high risk of ending up with sub-optimal solutions due to an improper coupling of the two contributors. For this reason, an extensive testing campaign has been implemented.

Three design choices are available: desensitizing a single final state, desensitizing multiple final states, or desensitizing the sensitivity of the final state with respect to Thrust. These design choices are thoroughly analyzed in the following sections.

Single State Desensitization

Single-state desensitization is the most straightforward and computationally efficient solution to implement. It is particularly simple because there is a single $\mathbf{\Lambda}(t)$ function, eliminating the need to tune relative weights between multiple cost terms. Consequently, a detailed analysis of this solution has been carried out, leading to the design choice presented in Chapter 5.

As discussed in Section 4.1.1, the equations of motion in each direction are almost completely decoupled from the others. However, position sensitivities and velocity sensitivities are respectively governed by similar dynamic equations. Therefore, it is reasonable to assume that penalizing one sensitivity could reduce or at least influence the others.

This assumption has been empirically validated through a series of tests. When a state is desensitized, all sensitivities, including both positions and velocities, are altered. Additionally, penalizing a position sensitivity positively affects other positions, and the same holds true for velocities. As a result, two categories of strategies can be defined: final position desensitizations (PD) and final velocity desensitizations (VD).

It has been observed that solutions falling into the PD category exhibit smooth convergence properties and are in line with optimal controller design. Conversely, VD solutions do not perform well in terms of CPU time and require sub-optimal LQR design to yield satisfactory results.

Given that the architecture under consideration allows for the desensitization of a single component, it is reasonable to select a design from the PD category and explore the impact of velocity desensitization in the next section, Section 4.3.5. Because positions can be considered as integrals of their corresponding velocities, minimizing the sensitivity of a position naturally leads to a reduction in the sensitivity of the corresponding velocity, as they are linked. However, when velocity is desensitized, there is no guarantee that position desensitization will be effective since final velocity is entirely independent of the evolution of the corresponding position.

In the class of PD architecture, three different options arise: desensitizing the final x, the final y or the final h. Even if the most critical component of to achieve a proper landing is the final altitude and even if it could seem reasonable to minimize its sensitivity, after performing some tests it becomes evident that the desensitization of $y(t_f)$ has the best effect in global terms.

Due to this results, another relevant concept of this work can be reported. The concept is the idea of *dominant sensitivity*, that can be seen as the single component of the final state to be desensitized to achieve the best performances possible. In the considered problem, it turns out to be the one referred to $y(t_f)$ but it is not guaranteed that it is always the same, if the initial conditions vary.

Multiple State Desensitization

The concept of dominant sensitivity suggest that further desensitization can be considered, such as other final states or the Thrust. However, they are treated as additional features of the problem where the sensitivity of the final y component is penalized. This decision is motivated by two primary reasons.

The first reason is that penalizing other sensitivities, like velocity sensitivities alone, leads to convergence issues, as demonstrated in previous sections. Therefore, if one intends to penalize the sensitivity of certain final velocities, it must be done in conjunction with a position desensitization. The second motivation is explained in this section and pertains to the combination of multiple desensitizations.

As discussed earlier, when multiple $b(\boldsymbol{x}(t_f))$ functions are desensitized, additional states are introduced into the problem. Consequently, since y desensitization can already lead to good performances, careful consideration is required when desensitizing other components of the final state. There is always a trade-off between improving performance and potentially increasing CPU time. In general, the penalization of the dominant sensitivity shapes the problem formulation by adding the DOC cost term, while other desensitizations can play a role in enhancing the already achieved performance.

The following two paragraphs aim to analyze two distinct strategies for multiple desensitizations. The first one deals with the desensitization of another final position component jointly with the dominant one, while the second focuses on penalizing the sensitivity of a final velocity and the y component one.

Multiple Positions Desensitization Given that the dominant sensitivity in the problem is $y(t_f)$, the other sensitivities that can be penalized are $x(t_f)$ and $h(t_f)$. When multiple sensitivities are penalized, the DOC term in the cost function increases in absolute value, necessitating proper tuning of the weights. To ensure a fair comparison, the solutions are evaluated in terms of sensitivity and the extra mass required, as the α weight is a design parameter with no clear physical interpretation.

The following plot illustrates four different desensitization strategies: y (Case 1), y and x (Case 2), y and h (Case 3), and h (Case 4).

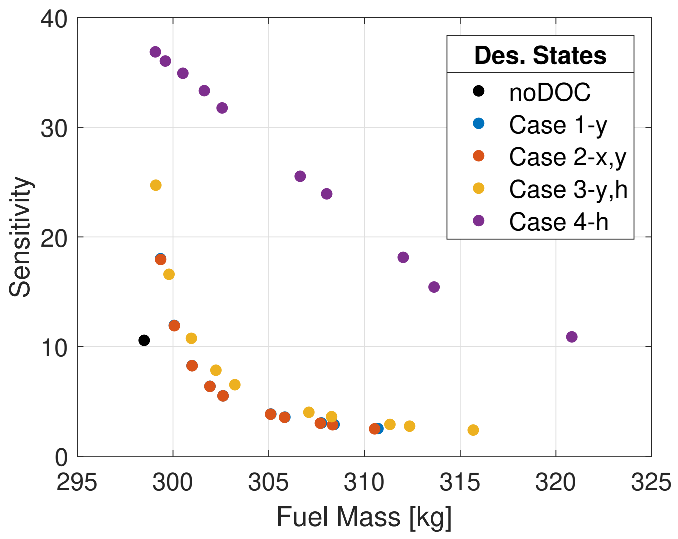


Figure 4.13: Final y sensitivities with different DOC strategies

It is evident that desensitizing only h results in the worst performance, while the other strategies yield comparable outcomes. Notably, Case 1 and Case 2 exhibit very similar behavior. Additionally, Case 3 desensitization leads to intermediate results between the desensitization of y and h. Therefore, a general conclusion can be drawn: when two or more final positions are desensitized, the overall sensitivity falls between the sensitivities of the individual positions when desensitized separately. As x and y dynamics are very similar, desensitizing x, y, or x and y produces the same results.

For this reason, once the dominant sensitivity has been chosen, further position desensitizations do not enhance performances. As a result, multiple positions desensitizations is both unnecessary and computationally inefficient.

Position and Velocity Desensitization Another potential extension of the architecture involves simultaneously desensitizing the dominant position and a velocity component. Given that the dynamics of velocities and positions are not identical, this approach could offer further performance improvements. Various combinations of relative weights and different components have been tested. However, desensitizing multiple velocities, as with positions, proved to be ineffective. Therefore, a single velocity is desensitized in conjunction with the dominant position. The configuration that yields the best results is the desensitization of v_h with y, even if the performances are not largely improved. This is mainly related to the dynamics of the problem: a position desensitization already influences the other positions' sensitivities, impacting on the corresponding velocities. Therefore, no large enhancements arise.

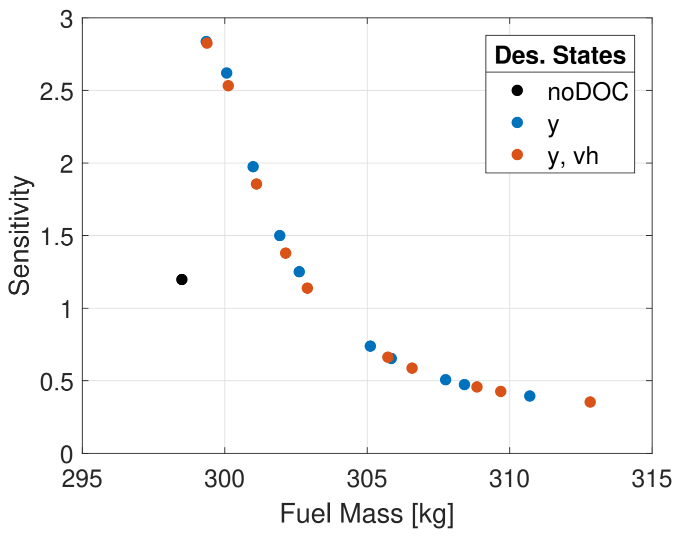


Figure 4.14: Final h sensitivities with different DOC strategies

Thrust Uncertainties Desensitization

The previous analyses have primarily focused on the sensitivity of the final state with respect to other states. However, it is also possible to penalize the sensitivity with respect to thrust by adding the seven states corresponding to the n components of the $\mathbf{\Omega}(t)$ vector, as defined in Section 3.1.2.

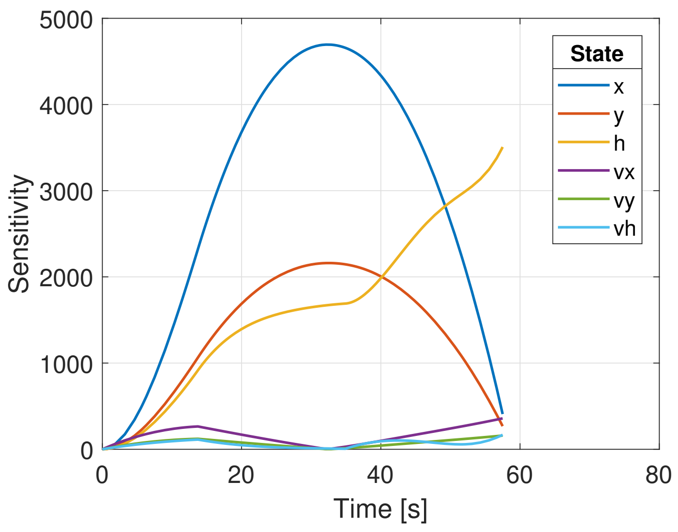


Figure 4.15: Sensitivities of the state with respect to Thrust

In the specific application under consideration, thrust uncertainties have a significant impact on the accuracy of the final state, with a particular influence on the vertical direction, where thrust should counteract the force of gravity. As expected, these uncertainties notably affect the error in the final altitude, as errors in velocity accumulate over time. It is essential to note that the critical values for this work are the final values, and the temporal evolution is provided to give a complete picture of sensitivity changes over time.

The penalty on $\mathbf{\Omega}(t_f)$ is not applied in norm but only the sensitivity of the final altitude has been considered, as it has the major impact on the final state, as shown in Fig. 4.15, and its desensitization has an impact on both the final vertical position and velocity. This is why no norms of $\mathbf{\Omega}(t)$ appear in Eq. (4.6). Since the value of the final altitude sensitivity with respect to thrust is always greater than zero, no absolute values are needed.

Similarly to multiple desensitization, thrust desensitization is considered as an additive feature of the problem, always penalized jointly with the dominant sensitivity. Additionally, it can be noted that the dynamics of $\mathbf{\Lambda}(t)$ and $\mathbf{\Omega}(t)$ share some common features. Thus, a penalty on $\mathbf{\Lambda}(t)$ already reduces the sensitivity with respect to thrust, as indicated by the typical sensitivity trend shown in Fig. 4.15, with $\beta = 0$. However, as β increases, more importance is given to thrust sensitivity in the cost function, leading to a reduction in sensitivity with respect to thrust, as shown in the cases with $\beta = 10$ and $\beta = 50$.

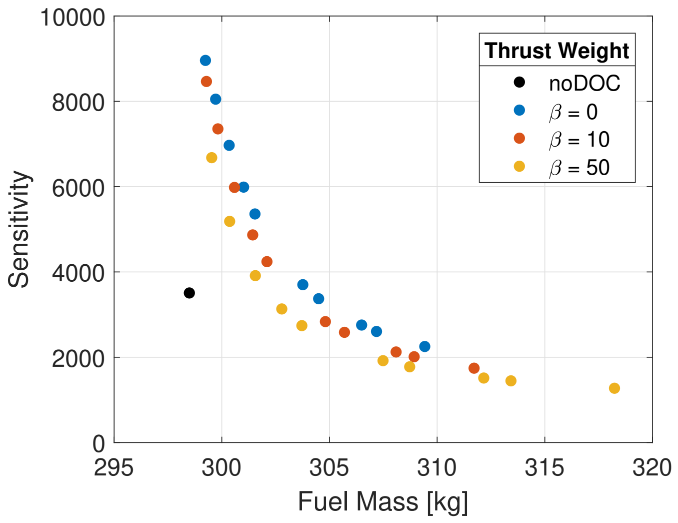
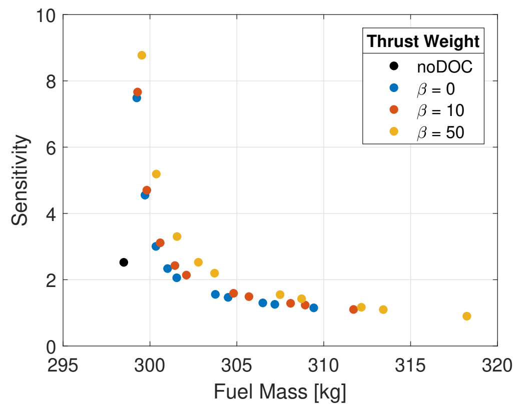
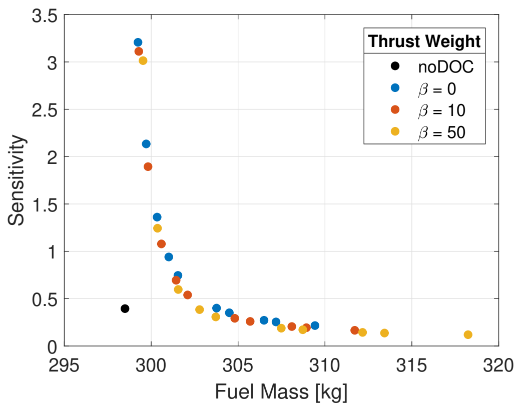


Figure 4.16: Thrust Sensitivities of different states in time

However, this improvement comes at a cost: since the thrust sensitivity has a greater importance in the cost function, the $\mathbf{\Lambda}(t)$ term is less penalized, resulting in a deterioration of the final sensitivities with respect to state sensitivities for higher β values, as shown in Fig. 4.17a. Nevertheless, as shown in Fig. 4.17b, penalizing $\mathbf{\Omega}_3(t_f)$ also reduces $\mathbf{\Lambda}_3(t_f)$, thanks to the mathematical coupling between their dynamic equations.



(a) Sensitivity of $x(t_f)$



(b) Sensitivity of $h(t_f)$

Figure 4.17: Sensitivities with respect to state perturbations for different β weights

A significant point to emphasize is that thrust desensitization requires the addition of seven extra variables to the state, representing the $\mathbf{\Omega}(t)$ components. This naturally results in an increase in CPU time. This increase in computational cost should ideally be balanced by improved performance. However, due to the distinct nature of perturbations (Thrust and initial conditions), precise predictions about the final standard deviation cannot be made. This is because the coupling between initial condition uncertainties and thrust perturbations is not precisely analytically predictable by any combination of sensitivities. Therefore, the performance enhancement resulting from this design choice can only be measured through a Monte Carlo analysis, considering expected perturbations in both initial state and thrust.

5 | Numerical Simulations and Results

This chapter is dedicated to the in-depth analysis of the numerical outcomes stemming from the DOC formulation. Its primary objective is to validate the effectiveness and characteristics of this method as it pertains to addressing the Powered Rocket Landing Problem. For each of the proposed implementation strategies, a preliminary examination of the nominal solution is conducted, which is subsequently reinforced and validated through the results obtained from Monte-Carlo (MC) simulations. This deliberate choice aims at emphasizing the connections between the G&C shaping process and its real-world performance within a perturbed environment.

The chapter is structured into four main sections. The first one deals with the definition of the problem setting both in terms of solver and MC. The second section places its focus on the Open-Loop case, shedding light on the dynamics and implications of this control approach, while the third section delves deeply into the Closed-Loop case, which serves as the central focus of this study. Here, the intricate interplay between control strategies and their impact on the Powered Rocket Landing Problem is thoroughly explored and dissected.

To conclude this chapter, an essential aspect is the comparison of CPU Times across the various proposed architectures. This comparative analysis highlights the computational efficiency of the different methods, providing valuable insights into their practical applicability and performance under real-world conditions.

5.1. Problem Settings

Before delving into the presentation of the results, some essential background information about the problem are provided. This information is organized into two distinct paragraphs: the first one details the specific parameters and configurations used in the numerical framework for problem-solving, while the second paragraph outlines the settings and the perturbations considered in the Monte Carlo (MC) analysis, elucidating their modeling process.

Numerical Settings The success of the strategy hinges significantly on the numerical implementation, given the large number of variables involved, which could lead to a considerable increase in CPU time. As a result, various implementation techniques have been developed and tested, as relying solely on a priori estimations proves unreliable. Two distinct versions of GPOPS, namely GPOPS-II, Version 1.0 (referred to as GPOPS-v1.0) and GPOPS-II, Version 2.5 (referred to as GPOPS-v2.5)[1], have been used. GPOPS-v2.5 is equipped with the added capability of automatic differentiation through the Adigator tool [47], enhancing the convergence process.

The first critical step involves problem scaling. While GPOPS is equipped with automatic scaling, this process is generic and not tailored to the specific problem at hand. Therefore, a deliberate decision was made to rescale the entire system, setting the initial altitude as the distance unit, the initial mass as the mass unit, and the Earth's gravity acceleration as unitary distance per squared time unit. Although GPOPS allows additional scaling, this chosen approach leads to a well-conditioned problem. Another significant code design choice is the minimization of for loops. Utilizing vector quantities and exploiting point-wise Matlab operators not only reduces CPU time but also ensures compatibility with automatic differentiation provided by Adigator. Further details are shown in Section 5.4.

All GPOPS settings were determined through a trial-and-error process aimed at minimizing CPU times. Each version necessitates unique configurations, which are not detailed here. The overarching code design philosophy aims to grant significant flexibility to the solver, facilitating fair comparisons between the proposed architectural approaches.

After performing collocation using GPOPS, the problem is subsequently solved using IPOPT [48], an interior-point filter line-search algorithm designed for large-scale nonlinear programming. The solver requires the Jacobian of the problem and can function with or without the Hessian . In this context, providing the Hessian to the algorithm was chosen to reduce the CPU time. Although GPOPS is compatible with the SNOPT [49]

[1]Available by kind courtesy of prof. Anil V. Rao

solver as well, it is found to be less efficient than IPOPT for the specific problem under consideration.

MC Settings The Monte-Carlo (MC) campaign involves the use of 1500 samples, and its convergence is assessed by examining the convergence of both the means and standard deviations of the final state variables. It is important to note that all the results are generated as zero-mean processes, with negligible biases arising from numerical integration errors or sampling errors, due to the sampling of the control and feedback gains to align with different time vectors. While it could be tried to adjust the final time to ensure a successful landing for each perturbed case, each sample in the MC analysis employs the same time of flight to maintain result consistency and measure the perturbation's impact on the final state.

Considering the initial conditions presented in Table 4.1, the state perturbations are characterized as deviations from the nominal conditions and follow Gaussian distributions. All state variables are perturbed with reasonable values, and the corresponding 3σ standard deviations are provided in the following table:

	Unit	$\hat{x}$	$\hat{y}$	$\hat{h}$
Position	m	100	100	100
Velocity	m/s	10	10	10

Table 5.1: Initial State Uncertainties, 3σ

To align with the actual physical behavior of the system, the thrust perturbation is represented as a combination of two components. This includes a constant bias term and a time-varying random noise component. Both of these components follow Gaussian distributions with zero mean. The 3σ values for these perturbations are set at 2% of the nominal thrust value, which is detailed in Table 4.2.

5.2. Open-Loop Case

In the Open-Loop (OL) configuration, the formulation is essentially equivalent to the Closed-Loop (CL) setup. However, in the OL case, the feedback gain matrix, denoted as K is set to zero due to the absence of feedback control. Consequently, the definitions of the state variables $\eta(t)$ and ν become irrelevant, and the components of the Riccati matrix are not included in the state representation.

While the OL case may not find practical application in real scenarios, given its inability to handle any form of perturbations without feedback control, this section provides an interpretation of the OL strategy. This is done to shed light on some aspects of the DOC strategy applied to the Rocket Landing Scenario. The reduced number of variables and parameters to be tuned in the OL case makes it easier to establish a direct link between design choices and outcomes, which is not as straightforward for the closed-loop cases with feedback capability factor.

In the absence of feedback gains, the desensitization of the OL case has limited degrees of freedom to act upon. Both the initial and final states are fixed, and control inputs are bounded by saturation limits. Simultaneously, the linearized version of the system is marginally stable, as evidenced by the upper triangular structure of the A matrix with zero values on the diagonal, as detailed in Appendix A. Although the code implementation penalizes the sensitivity of the final state y through the term $\mathbf{\Lambda}(t)$, some features of the problem can be obtained from the analysis of the standard sensitivity matrix, denoted as $S(t)$. In the absence of feedback gains, the dynamics of $S(t)$ follow Eq. (3.3).

The sensitivity matrix grows over time, starting from the initial condition represented by the identity matrix. Upon closer examination, it becomes apparent that the upper diagonal of the matrix $\dot{S}(t)$ is filled with ones. As a consequence, the corresponding terms in the $S(t_f)$ matrix are directly related to the time of flight. These two aspects are equivalent and illustrate that the sensitivity matrix, or some of its components, experience linear growth over time.

When solving the optimal control problem, minimizing the cost function leads to the minimization of the flight time. This is achieved by progressively increasing the control effort, essentially keeping the control input at its maximum saturation value and reducing the duration of the minimum control phase. This trend is clearly demonstrated in the accompanying figure: as the DOC term increases, the flight time decreases, and the control action shifts from a max-min-max solution towards a continuous max solution, where the control is maintained at its maximum value throughout the trajectory. A similar DOC behavior occurs in a CL architecture in the absence of $\eta(t)$, as evident from the similarities between Fig. 4.7 and Fig. 5.1.

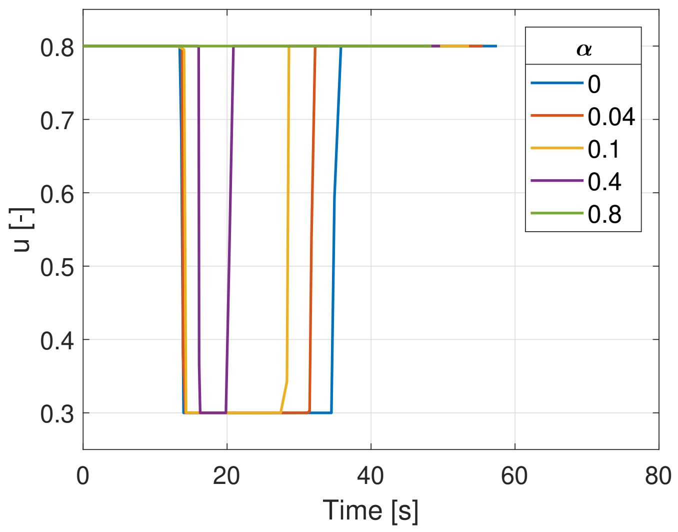


Figure 5.1: Control trends of OL Case for increasing α

As discussed in Section 4.3.2, in the closed-loop case, the presence of the $\eta(t)$ term is responsible for an increase in sensitivity, particularly for low α values. However, this is not observed in the Open-Loop (OL) case, as the sensitivity exhibits a monotonically decreasing trend. Nevertheless, the actual sensitivity values in the OL configuration do not vary significantly, rendering the OL version of the DOC strategy notably inefficient.

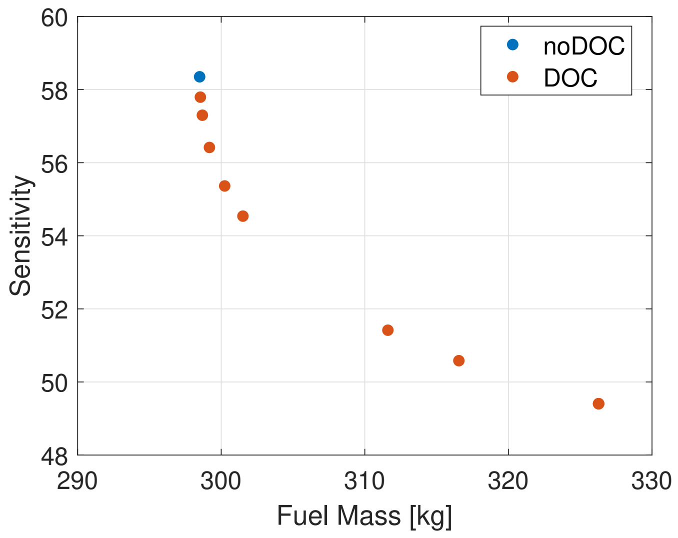
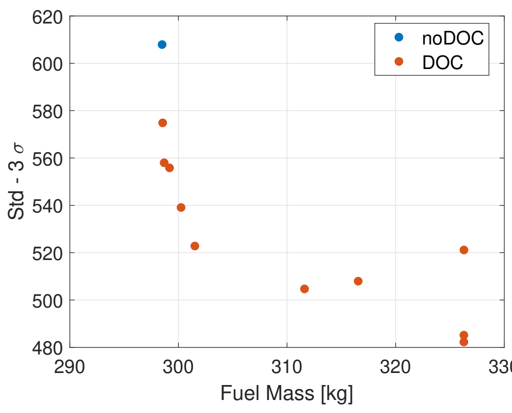


Figure 5.2: Sensitivity of $h(t_f)$ for OL case with α

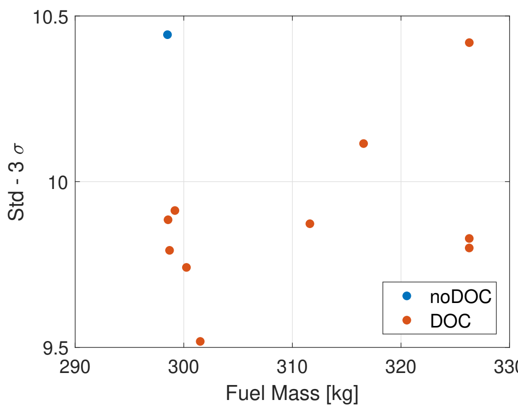
It is worth noting that when the solution reaches the full max configuration, no further reductions in sensitivity can be achieved. Therefore, the rightmost point on the plot, corresponding to the maximum fuel mass, represents the minimum sensitivity value for the system. If α were to increase beyond this point, subsequent values would be similar, as no further improvements in sensitivity are attainable.

In the absence of feedback control, initial position errors cannot be compensated for, nor can initial velocity errors be mitigated. These errors remain constant throughout the entire trajectory due to the quasi-linear dynamics of the system. However, velocity errors can lead to the growth of position errors since they are integrated over time. Therefore, desensitization can be interpreted as reducing the time of flight to minimize the time integrals of velocity errors, thereby mitigating the impact of velocity perturbations on position errors.

Performance can be quantified through Monte-Carlo analysis. When considering all perturbations, the 3σ standard deviation of the final altitude is depicted in the following plot. As anticipated in Fig. 5.2, the DOC strategy does not yield a substantial improvement in terms of the standard deviation of the final state, and the resulting values of the standard deviations are too high to be considered reasonable. Nevertheless, a reduction in standard deviation is evident, even though it is not consistently decreasing but exhibits a general decreasing trend. This behavior is influenced by velocity errors. Since the number of samples in the Monte-Carlo analysis is limited, perturbed velocity samples are not evenly distributed around the nominal value. Consequently, this leads to significant variations in the final position due to the integration of this discrepancy throughout the entire trajectory.



(a) 3σ standard deviation of $h(t_f)$



(b) 3σ standard deviation of $v_h(t_f)$

Figure 5.3: MC analysis for OL Case

This is evident in Fig. 5.3b. As the DOC strategy aims to minimize sensitivities by reducing the time of flight, it does not significantly impact the final velocity errors, which remain equal to the perturbations applied to the initial state, leading to sensitivity values of one for all the velocity components. Consequently, any variation in the final velocity standard deviation is linked to non-uniform sample distribution around the initial point and impact the final altitude errors.

An important observation relates to the role of the last three points to the right. As previously discussed in the context of Fig. 5.2, the point corresponding to the maximum mass-consumption trajectory represents the physical performance limit of the system. Consequently, in Fig. 5.3a, three vertically aligned points on the right side of the plot can be observed. These points correspond to max profiles with different α weights. In theory, their standard deviations should be the same, given that they represent the same maximum performance configuration. However, due to the limited number of samples for initial velocity errors, their standard deviations assume different values, leading to the observed vertical trend.

5.3. Closed Loop Case

This section presents the results of the thesis work, showcasing the outcomes of the architecture optimization. Before delving into the most significant findings, it is important to make a brief but significant observation.

As discussed in Section 4.3.2 the noDOC max-min-max is not capable of addressing the saturation issue, resulting in a control system with reduced feedback. In practical applications, this limitation is unacceptable as it severely restricts the management of perturbations. Therefore, a common approach is to compute the nominal trajectory with control bounds set narrower than the maximum and minimum nominal values, allowing room for feedback. The tuning becomes a saturation-driven process aiming at ensuring that the actuators do not operate beyond their physical limits while fulfilling the requirements in terms of prescribed performance.

In the context under examination, the choice of this approach plays a pivotal role. Since the DOC strategy includes a saturation handling mechanism through the coefficient $\eta(t)$, comparing it to a solution with only half the feedback would not be fair. Consequently, it was decided to adjust the bounds for the noDOC control from the range of 0.3-0.8 to 0.4-0.7 when defining the guidance and selecting the gains. This adjustment aimed to avoid exceeding the 0.3-0.8 bounds with the nominal control plus feedback effort. While the choice of 0.4-0.7 results in an increase in mass compared to the 0.3-0.8 case, it provides a reasonable benchmark for the DOC strategy since control saturation is rarely encountered.

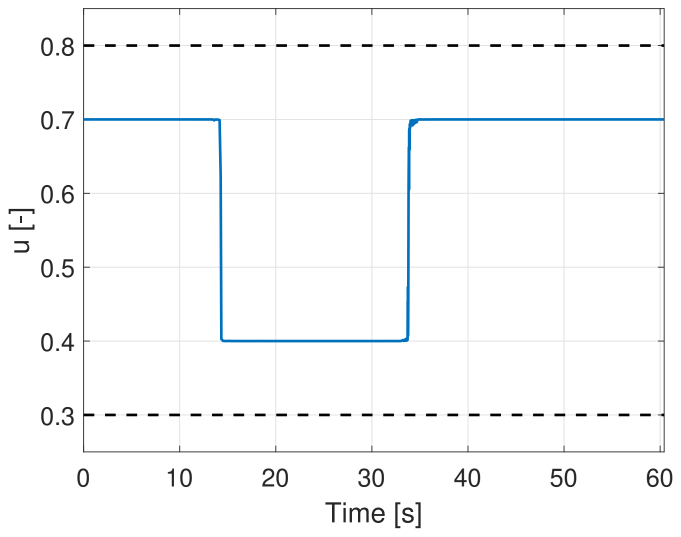


Figure 5.4: NoDOC solution with new bounds

The specific values of 0.4-0.7 were selected to ensure sufficient leeway in both directions, with particular emphasis on the upper value due to the initial phase of the nominal trajectory characterized by a control value $u = u_{max}$ and by the presence of significant perturbations. Therefore, careful tuning was applied to the upper bound of 0.7, while less attention was given to the lower bound of 0.4. It is worth noting that using 0.4 or other values, such as 0.35, has minimal impact on both fuel mass consumption and sensitivity values, since the most relevant role is played by the upper bounds of the interval. Fig. 5.4 displays the 0.3-0.8 boundaries and illustrates the noDOC solution achieved using the adjusted boundaries of 0.4-0.7.

5.3.1. Linearity and Decoupling

Before discussing the best design solutions, a brief overview of some key properties of the method and the application scenario is provided. These properties are related to the linearity of the model and serve to justify certain statements made in previous chapters.

A pivotal aspect of the entire model is its quasi-linearity. Despite being a result of strong assumptions, such as perfect vehicle modeling, the absence of drag, and ideal attitude control, which are not typically applicable in real-world scenarios, these assumptions simplify the preliminary characterization of the DOC, since they allow for some simplifications and predictions. Another important aspect linked to the aforementioned assumptions is the quasi-decoupled dynamics between different directions. These dynamics are not entirely independent or linear due to the presence of a mass term in the equations of motion's denominator and the norm constraint coupling the individual thrust components.

The implications of these two assumptions are significant in characterizing the method. Firstly, perturbations on the initial state in the i^{th} direction, both in terms of position and velocities, only affect the i^{th} direction. As a result, the impact of individual perturbations can be considered and analyzed separately, and the resulting covariance matrix is quasi-diagonal, except for the coupling between the position component and its corresponding velocity.

However, the most substantial consequence is the nearly exact match between the sensitivity of the final state with respect to initial state perturbations and the final state covariance. This alignment is straightforward because the propagation of initial state uncertainties follows the same differential equation as the sensitivity matrix, given that the original problem (governing perturbation propagation) and the linearized version (governing sensitivity propagation) are almost identical. In Section Section 4.3, the sensitivity of the final state was chosen as a criterion for comparing different design choices, as indicated in Equation (4.8). This criterion can be applied generally, even in more complex models, but it is particularly suitable in this case due to the linear relation between final sensitivities and final covariance matrices. As the results of Monte-Carlo (MC) analysis serve as the real validation of the methodology, forecasting their results is essential to reduce the number of MC analyses required during the tuning process. Additionally, when faced with two competitive design options, the best one in terms of final covariance can be selected without conducting the computationally demanding MC analysis. As demonstrated in Fig. 5.5, there is a linear correlation between sensitivity (represented by $S_i(t_f)$) and the standard deviation of the $y(t_f)$ component.

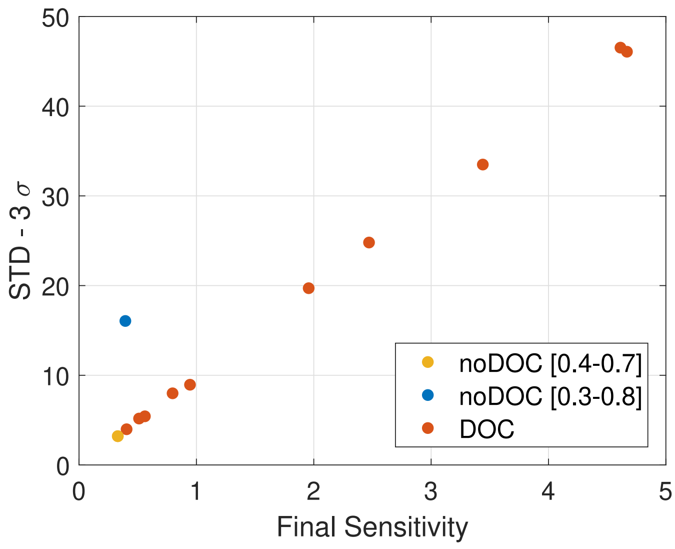


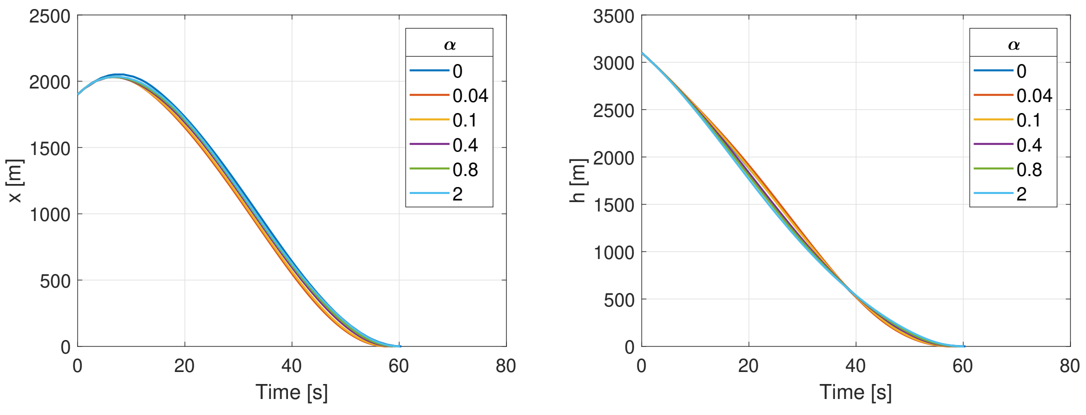
Figure 5.5: Correlation between $h(t_f)$ standard deviation and final sensitivity

Another noteworthy remark that can be made by observing Fig. 5.5 is that the blue point, corresponding to the noDOC solution with throttle bounds of 0.3-0.8, is the only point where the linearity between the final sensitivity and the final standard deviation of h is not respected. This deviation is a result of saturation occurring in the MC simulation, leading to worse actual performance, measured in terms of standard deviation, compared to the predicted values. In contrast, the yellow dot corresponds to the noDOC solution with the new throttle bounds of 0.4-0.7. This point aligns with the red ones, where the linear relationship between sensitivity and standard deviation is valid. Consequently, the connection between standard deviation and sensitivity remains linear in this case as well, suggesting that saturation either does not occur or has minimal impact on the final state accuracy. This provides a first verification of the shrinkage of the nominal control values as a valid saturation mitigation strategy for noDOC solution, both in terms of architecture and chosen values.

5.3.2. Final Design

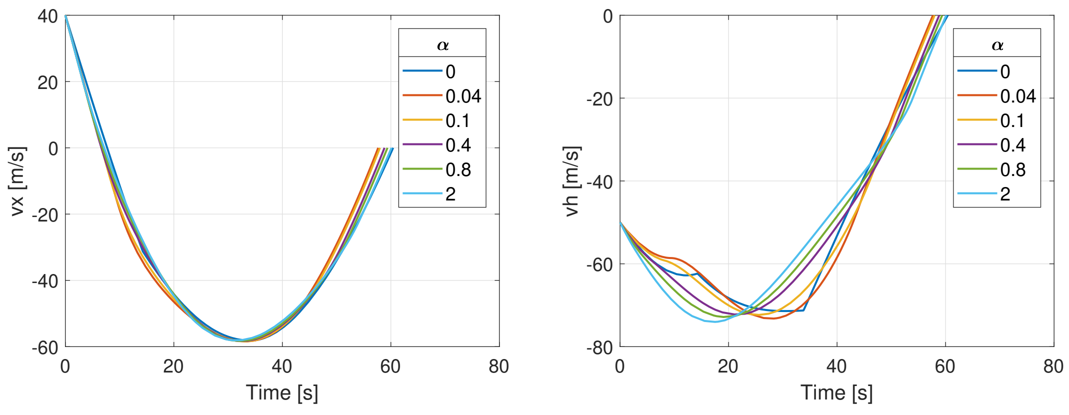
The design directly emanates from the characterization process of Section 4.3, taking into account all the elements reported. As a general guiding principle, the tuning process has been developed with the aim of ensuring specific performance criteria. This includes maintaining a final horizontal position error within the confines of 10 meters (with a 3σ deviation), limiting the vertical position error to 1 meter, constraining the final horizontal velocity error to 2 meters per second, and keeping the vertical velocity error at 1 meter per second.

These performance benchmarks are subjected to validation through a comprehensive suite of MC simulations, which are detailed in Section 5.3.4.



(a) Trend of $x(t)$ (b) Trend of $h(t)$

Figure 5.6: Positions trends for different α weights



(a) Trend of $v_x(t)$ (b) Trend of $v_h(t)$

Figure 5.7: Velocity trends for different α weights

The positions and velocities in the $\hat{x}$ and $\hat{h}$ directions can be observed in the graphical representations presented in Fig. 5.6 and Fig. 5.7. This choice was made for the sake of brevity and to ensure a concise and compact representation. It is worth noting that the motion in the $\hat{y}$ direction closely resembles that in the $\hat{x}$ direction.

It is important to highlight that, even when dealing with high α weights, there are no significant differences in terms of states. This outcome was expected since the DOC term has only a minimal impact on the problem's cost function.

In the $\hat{x}$ direction, the motion is profoundly affected by the challenging initial conditions: the vehicle initially moves away from the target with positive velocity and positive initial x. Nevertheless, within 10 seconds, the thrusters manage to alter the vehicle's trajectory, ultimately guiding it to the landing site in approximately 60 seconds. The time of flight is influenced by the value of α; as α increases, so does the time of flight.

Conversely, in the $\hat{h}$ direction, the motion follows a more predictable pattern. During the initial phases of the trajectory, as the control is primarily focused on changing the signs of v_x and v_y, the lander's velocity increases in the negative direction towards the target, and then it is reduced to ensure a final condition of soft touchdown. Notably, the velocity v_h, as shown in Fig. 5.7b, is more influenced by the increment of α compared to v_x, as depicted in Fig. 5.7a. This phenomenon can potentially be attributed to a greater portion of control being allocated to the $\hat{h}$ direction to counteract the effects of gravity, and since the DOC parameter influences control trends, v_h is more significantly affected.

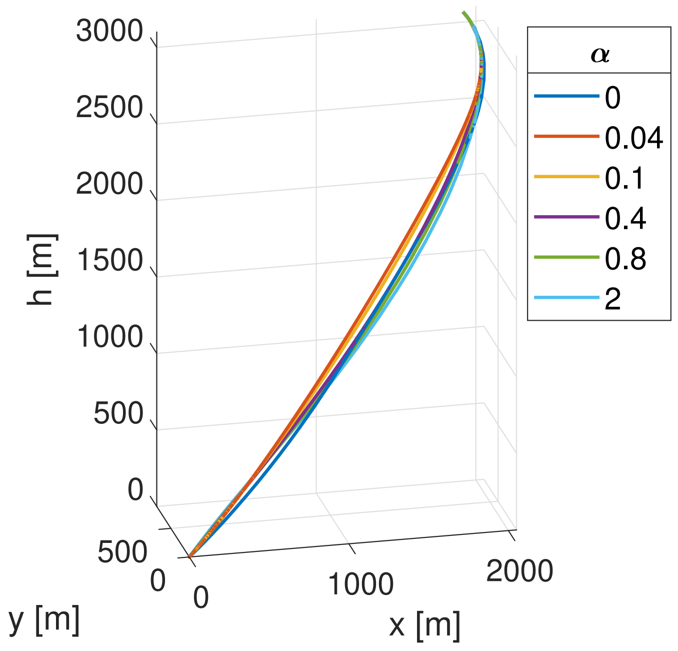
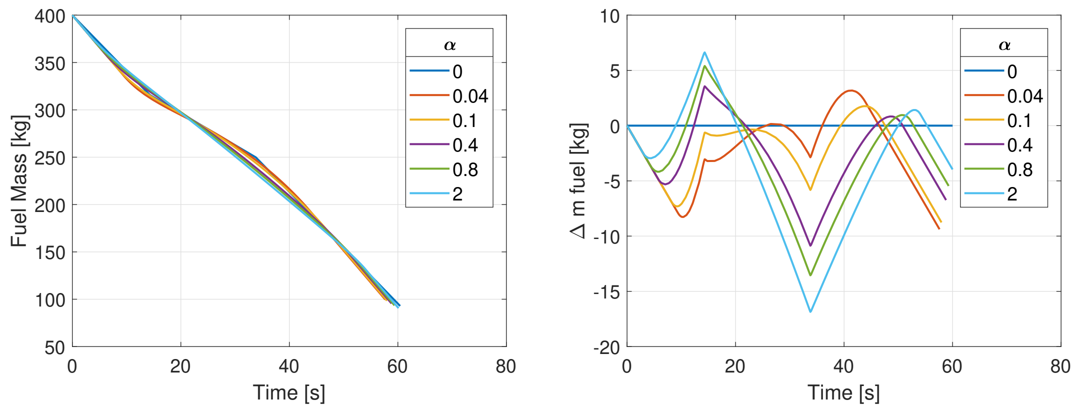


Figure 5.8: Trajectory for different α weights

The overall trajectory resulting from these conditions is visualized in Fig. 5.8. It is evident that for increasing values of α, the trajectories tend to descend more rapidly in altitude, as previously observed in Fig. 5.6b.



(a) Fuel Mass evolution (b) Δm with respect to $\alpha = 0$ case

Figure 5.9: Fuel Mass consumption for different α weights

Fig. 5.9, which displays the fuel mass, and Fig. 5.10, illustrating the trends in control u over time, exhibit a strong interconnection. This link arises from the fact that control is administered through a propulsive system, and the control value is directly proportional to the mass flow rate within the thrusters. Consequently, changes in control lead to fluctuations in the fuel mass. Therefore, the norm of control is directly proportional to the derivative of the fuel mass variation. This relationship becomes particularly evident when we examine the case with $\alpha = 0$. In the initial phase, a sharp decrease in fuel mass occurs as control is actively applied. However, when control is reduced, the rate of fuel mass reduction decreases, only to increase once again when the trajectory reaches its second maximum segment.

Other relevant trends become even more apparent in Fig. 5.9b, which shows the change in fuel mass, denoted as Δ fuel mass, between the noDOC case with $\alpha = 0$ and the other cases. In the DOC cases, a reduction in fuel mass occurs during the initial stages, as is evident from Fig. 5.10. Subsequently, the curves representing the DOC and noDOC cases intersect at various points, depending on the specific control pattern employed. By the end of the trajectory, Δm is negative, indicating that for lower α values, the DOC strategy consumes less fuel mass compared to the noDOC approach, which is computed with 0.4-0.7 bounds. However, as α increases, the differences in fuel consumption gradually diminish.

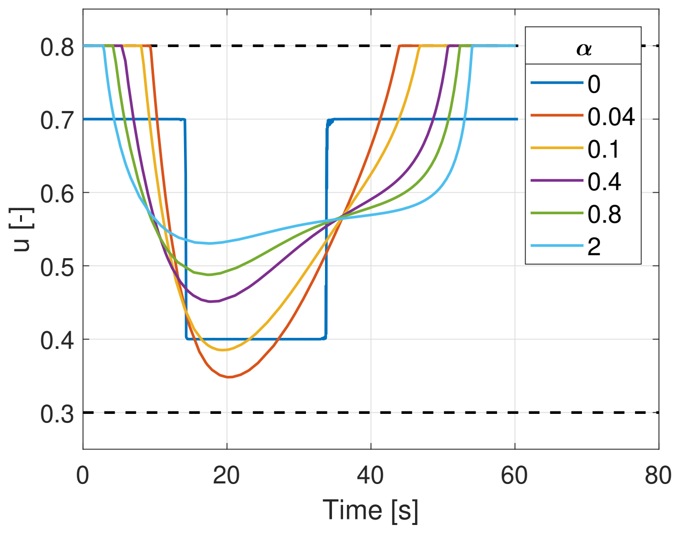


Figure 5.10: Control Norm in time for different α weights

Since the optimality conditions for optimal control problems are necessary conditions, it is essential to verify the Hamiltonian of the solution. The Hamiltonian is defined in accordance with Eq. (2.8), although the third term, which is associated with state constraints, vanishes. The optimality of the solutions is validated by examining the Hamiltonian profiles, which should remain constant and close to zero, given that the problem does not explicitly depend on time, and the final time is left unconstrained. The Hamiltonian, Fig. 5.11, deviate from zero, but they are in close proximity to it.

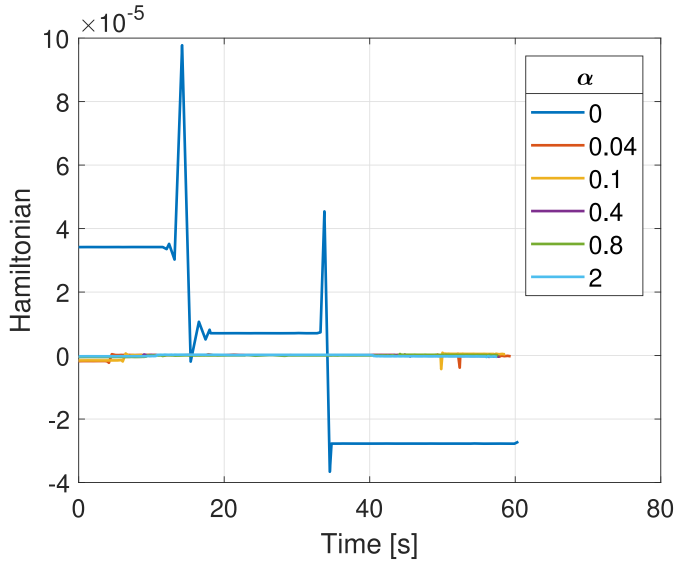


Figure 5.11: Hamiltonian for different α weights

It is worth emphasizing that spectral collocation methods encounter challenges when approximating constant piece-wise polynomials. This is evident from the curve with $\alpha = 0$, which displays values that are larger than the others and exhibits strong discontinuities at the switching points, transitioning from maximum to minimum values and vice versa.

Marginal DOC Coefficient As there are no constraints related to saturation on the value of ν, only two criteria warrant consideration, as detailed in Section 4.3.3: limitations on feedback gain effort and gain margin. It has been emphasized that greater attention is directed towards the latter. The system, being Multi-Input Multi-Output (MIMO) in nature, involves 7 states and 3 inputs, necessitating the evaluation of 21 gain margins. Nevertheless, the most critical one can be selected at each instance and used as a reference value. Fig. 5.12 provides a depiction of the gain margin in comparison to various ν values, represented in dB scale.

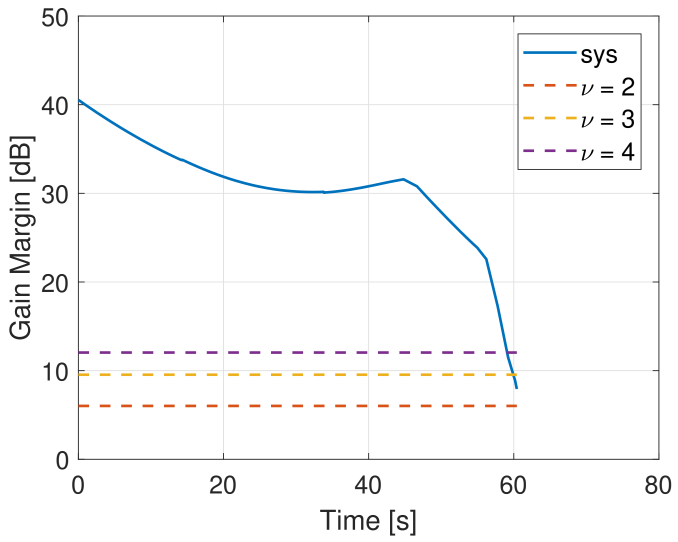


Figure 5.12: Gain Margin compared with different ν values

Despite the fact that the selection of $\nu = 3$ results in better overall performance compared to $\nu = 2$, as demonstrated in Section 4.3.3, the design choice leaned towards the latter option. This decision was primarily influenced by the fact that, in the later stages of the trajectory, the gain margin falls below the critical threshold of $\nu = 3$, leading to potential stability issues. It is well-established that in the final segment of the trajectory, $\tilde{\eta} = 3 \cdot \eta(t)$ becomes less than 3, mainly because the control input u is approaching its maximum value u_{max}. However, this behavior is not readily predictable in advance, and the more conservative choice of $\nu = 2$ has been made.

LQR Tuning The resulting R and Q matrices are presented below. The criteria and techniques used to determine these matrices are extensively detailed in Section 4.3.4. In this process, the R value is minimized to the greatest extent possible to enhance performance, taking into account the constraint on CPU time. Simultaneously, the Q matrix is tuned to distribute the total control effort across the states. It is important to note that the Q matrix corresponds to an unscaled version of the problem. Since these matrices are diagonal, only their diagonal components, denoted as $\boldsymbol{R}_d$ and $\boldsymbol{Q}_d$, are reported:

$$\boldsymbol{R}_d = \begin{bmatrix} 2.5 & 2.5 & 2.5 \end{bmatrix}$$

$$\boldsymbol{Q}_d = \begin{bmatrix} 0.63 & 0.63 & 9.49 & 100.00 & 100.00 & 50.00 \end{bmatrix} \tag{5.1}$$

Although the orders of magnitude of the components in the Q matrix are similar to the design choices made in [16], there have been specific adjustments. The weight assigned to the altitude (the 3$^{\text{rd}}$ component) has been increased to accommodate the more stringent requirements for this particular state. Conversely, the weight on vertical velocity has been reduced relative to the other velocity components because the substantial gain applied to altitude also indirectly impacts velocity errors. Given the integral relationship between altitude and vertical velocity, a significant reduction in altitude error has a positive cascading effect on velocity as well. Furthermore, due to comparable requirements for final error and symmetric dynamics, the weights assigned to x and y and to v_x and v_y have been set as equal pairs.

After tuning the Q and R matrix is was checked, through MC analyses, that the synthesized feedback gains did not lead to saturation issues when considered with the noDOC [0.4-0.7] case.

Desensitization Strategies As already stated in Section 4.3.5, the most convenient choice in terms of dominant sensitivity is the y one. Therefore, this choice has been operated, with several α values to obtain the different solutions. Multiple other solutions have been considered and tested but the results where not as good as the architecture desensitizing y component.

5.3.3. Alternative Design

Since multiple alternatives have been presented, it is worth mentioning two further design solutions, belonging to the Multiple states desensitization, Section 4.3.5, and to the Thrust Effect Desensitization one, Section 4.3.5. Since there are not relevant changes in terms of trajectories and control with respect to Fig. 5.8, the corresponding plots are omitted for brevity and the performances analysis are left to Section 5.3.4.

Multiple States Desensitization The most convenient choice for multiple states desensitization is the y and v_h couple, with a weight of $\alpha_1 = 100$ on y sensitivity and of $\alpha_2 = 10$ on v_h sensitivity, according to Eq. (4.6). Proper results in terms of performances are provided in Section 5.3.4.

Thrust Uncertainties Desensitization The design choice for the thrust sensitivity is to penalize the third component of $\mathbf{\Omega}(t_f)$ with a weight $\beta = 50$, while all the other weights and the LQR matrices are unchanged with respect to the final design case.

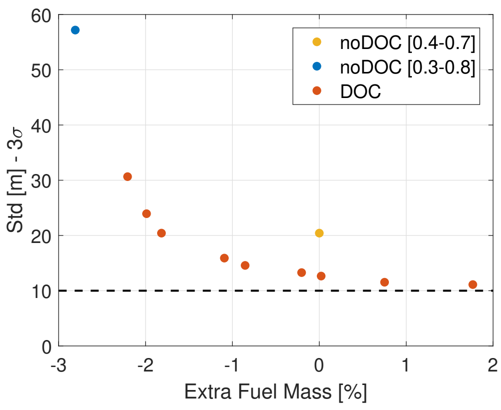
5.3.4. MC Analysis

This section presents the effective outcomes of the DOC Closed-Loop strategy for the powered landing problem, taking into account all perturbations affecting the initial state, as outlined in Section 5.1. To provide a comprehensive overview, we display the standard deviations related to DOC alongside those obtained from noDOC solutions within the range of $u \in [0.4 - 0.7]$, highlighted in yellow, as they serve as the established benchmark performances. Additionally, we include the MC performance for scenarios where $u \in [0.3 - 0.8]$ to validate the decision to narrow the interval. Each plot also features a dashed line denoting the landing site requirement to be met.

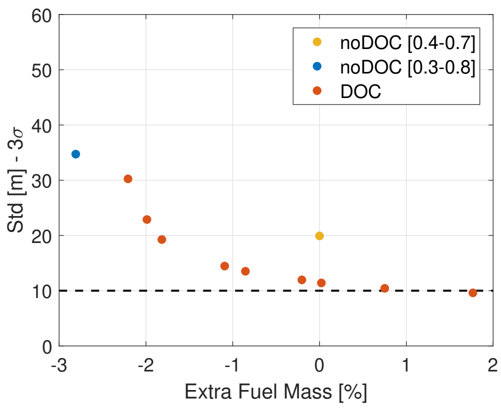
It is worth noting that both the noDOC solution and the DOC solution regulators are synthesized according to the same LQR matrices detailed in Section 5.3.2, and subjected to identical perturbations.

Final Design

To validate design decisions, we solely focus on perturbations to the initial state, as the nominal design embeds just penalties on state sensitivities. In the subsequent sections, when examining Thrust desensitization, we also take Thrust perturbations into account. In the design solution, various plots have been presented, illustrating the ultimate standard deviation across all states. Each image is accompanied by a concise description to ensure a more profound understanding of the results.



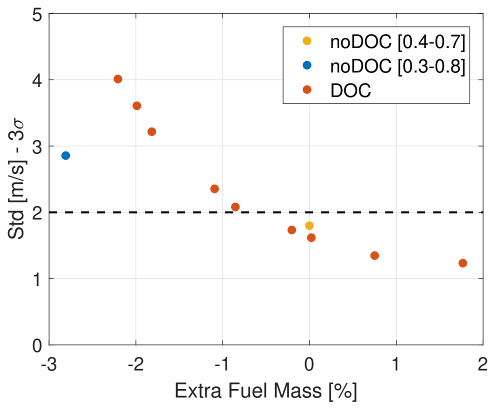
(a) 3σ standard deviation of $x(t_f)$



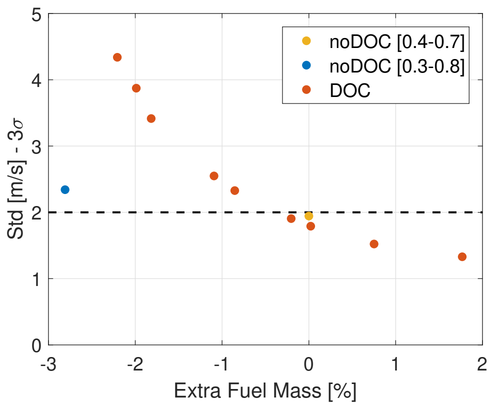
(b) 3σ standard deviation of $y(t_f)$

Figure 5.13: MC Analysis for $x(t_f)$ and $y(t_f)$

As emphasized on multiple occasions, the trends for both the x and y directions are nearly symmetrical, differing mainly in initial conditions and feedback gains. Consequently, it is quite evident that even the standard deviations of the final states, as depicted in Fig. 5.13, exhibit similar patterns, with slight values variations. In a general observation, it is clear that the DOC strategy effectively enhances performance compared to the noDOC baseline, represented in yellow, which serves as the reference for performance. With the same fuel consumption, the final uncertainty is reduced, and by slightly increasing the fuel mass, the requirements are met or nearly met.



(a) 3σ standard deviation of $v_x(t_f)$

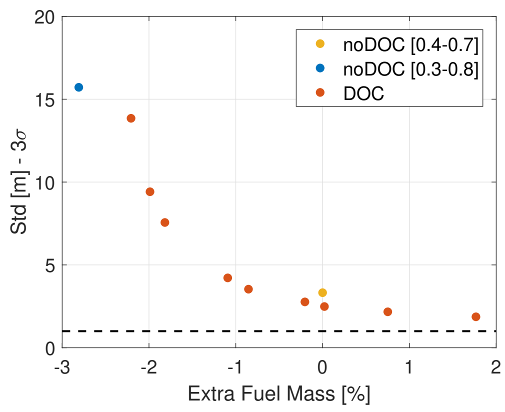


(b) 3σ standard deviation of $v_y(t_f)$

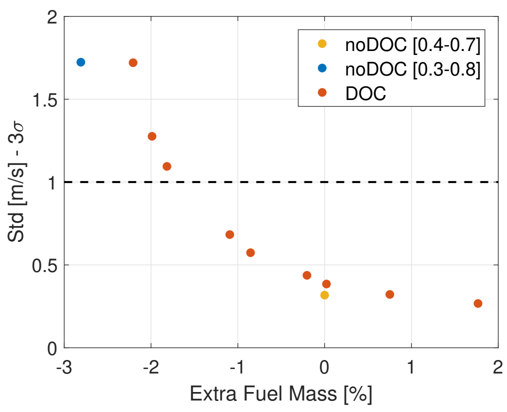
Figure 5.14: MC Analysis for $v_x(t_f)$ and $v_y(t_f)$

A noteworthy point is that the noDOC solution with $u \in [0.3 - 0.8]$, indicated in blue, exhibits worse behavior in the x direction compared to the y direction. This difference is evident in Fig. 5.14, which displays uncertainties in final velocities. This behavior can be explained with reference to Figure Fig. 4.1b. Due to challenging initial conditions in the x direction, significant nominal control effort is necessary in that direction. When the nominal control value is augmented by feedback control in the Monte-Carlo simulations, the total control in the x direction becomes more susceptible to saturation, leading to a deterioration in performance compared to the y direction.

Although the DOC strategy significantly enhances performance in terms of position, these improvements may not be extraordinarily remarkable in an absolute sense. As evident from Fig. 5.14, the standard deviation of final positions in the yellow case closely approaches those achieved with the DOC strategy, both in the x and y directions. The primary advantage of the DOC strategy lies in its ability to achieve further reductions in standard deviation with a slight increase in fuel consumption. Performance could potentially be improved by increasing the weight factor α. However, it is worth noting that the results are approaching a performance limit, and significant additional enhancement may be challenging to achieve.



(a) 3σ standard deviation of $h(t_f)$



(b) 3σ standard deviation of $v_h(t_f)$

Figure 5.15: MC Analysis for $h(t_f)$ and $v_h(t_f)$

In Fig. 5.15, the performance in the h direction, which is the most critical one due to stringent requirements on the final altitude uncertainty, is presented. The results in Figure Fig. 5.15a clearly show that the improvement in terms of final altitude uncertainty is not as significant compared to the noDOC case, and the 1-meter (3σ) requirement is not met, although the results are very close. On the other hand, the DOC procedure worsens the uncertainty in the vertical velocity, as demonstrated in Figure Fig. 5.15b, even though both strategies manage to meet the specified requirement. It is noteworthy that a small increase in fuel leads to a trajectory capable of restoring the initial performance and, at the same time, achieving better results in all other components.

Conversely, a remarkable outcome is that the DOC strategy enhances performance compared to the noDOC blue case. This result was unexpected based on the final sensitivity analysis. However, saturation has a dramatic impact on the h direction, leading to a significant degradation in performance for the blue case compared to the a-priori forecasts.

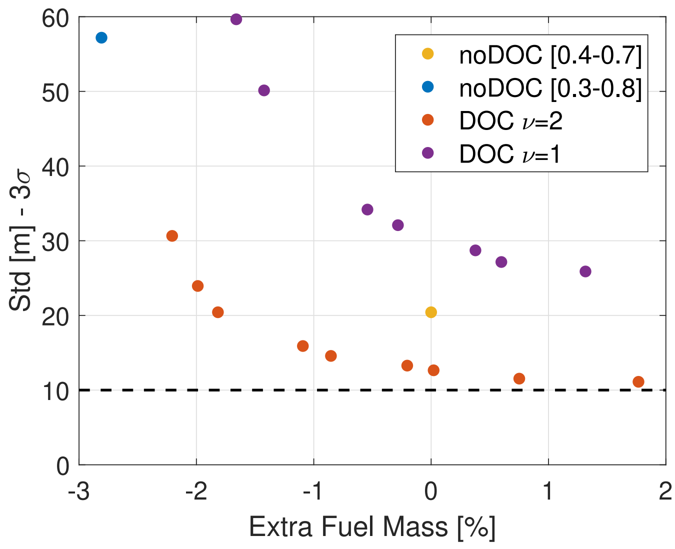


Figure 5.16: Comparison of $x(t_f)$ dispersion with $\nu = 1$ and $\nu = 2$

The final notable observation pertains to Fig. 5.16, which highlights the influence of the Marginal Coefficient Factor ν, as discussed in Section 4.3.3, on the performances. Without the incorporation of ν, the DOC strategy was unable to improve the noDOC solution or meet the requirements for final state accuracy. In the case with $\nu = 3$, the requirements are satisfied; however, it was decided to set $\nu = 2$ for the reasons elucidated in Fig. 5.12.

Saturation In Fig. 5.17, the percentage of time during the flight when the sum of nominal and feedback control exceeds the bounds is depicted. Two key observations emerge. Firstly, as indicated by the yellow dot, the reduction of the permissible control values for trajectory generation and the corresponding LQR tuning is effectively executed, with the saturation level decreasing from over 20% of the time to nearly zero. Additionally, the saturation handling strategy discussed in Section 4.3.2 proves capable of delivering optimal performance in terms of avoiding saturation, in line with the definition of $\eta(t)$ and as demonstrated by the DOC related values in the plot.

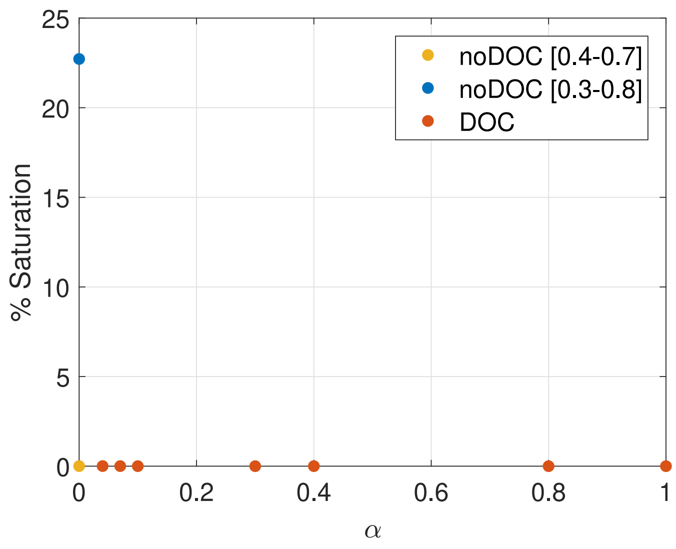


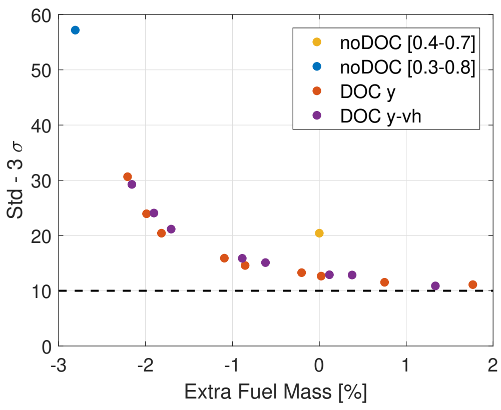
Figure 5.17: % extra saturation time over time of flight

Alternative Design

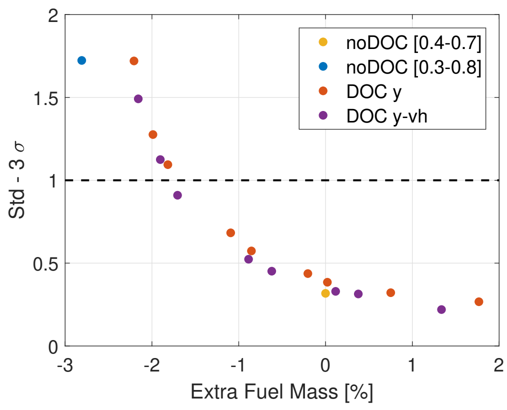
This section is dedicated to presenting the Monte-Carlo performances of the alternative design architectures outlined in Section 5.3.2. It is important to note that while various design architectures were considered, this section primarily concentrates on the most significant and relevant ones.

Multiple States Desensitization This sections reports the MC results of the Multiple States Desensitization illustrated in Section 4.3.5 and focusing on y and v_h. While this approach does lead to performance improvements, the degree of change is not significant. This is because the dominant sensitivity approach already delivers satisfactory results in terms of reducing dispersion in final states, and further desensitization does not yield substantial enhancements. Moreover, the reduction in sensitivity is influenced both by the feedback gains and the desensitization process. The significant weighting of the velocities in the Q matrix, as clear from Eq. (5.1), already maximizes the reduction of sensitivity in velocity components.

These findings clearly indicate that desensitizing multiple states with the considered Q and R weights does not offer a significant performance boost. The architecture turns out to be inefficient, given that it necessitates a larger number of state variables compared to the single desensitization case. Hence, the concept of dominant sensitivity becomes even more relevant, as it provides superior performance with minimal computational effort.



(a) 3σ standard deviation of $x(t_f)$

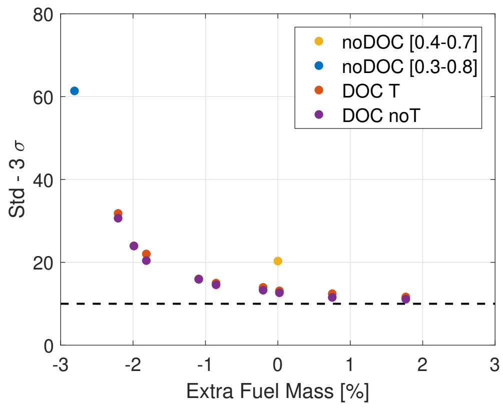


(b) 3σ standard deviation of $v_h(t_f)$

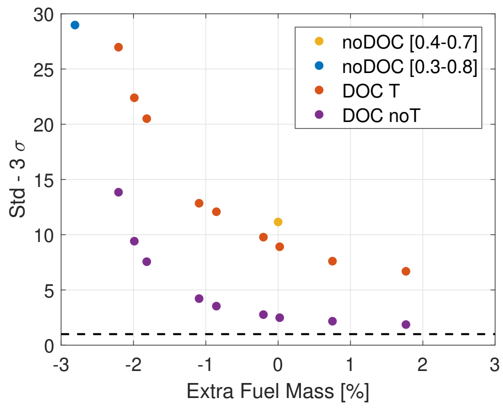
Figure 5.18: MC Analysis for $x(t_f)$ and $v_h(t_f)$ with y and v_h desensitization

Thrust Uncertainties Desensitization When desensitizing the thrust effect, the third element of $\mathbf{\Omega}(t_f)$ is penalized to mitigate the influence of thrust perturbations on the ultimate altitude. Before presenting the results, it is valuable to provide a clear quantification of the effect of thrust perturbations on the final state.

As mentioned in Section 4.3.5, thrust perturbations primarily impact the vertical direction. This is evident in Fig. 5.19, which depicts the final dispersion of the final x and v_h with both state and thrust perturbations ('T' case) and with state perturbations only ('noT'). As anticipated, thrust perturbations have a minimal effect on the x component, whereas the impact on h component is substantial, and it almost fails to meet the final error requirement.



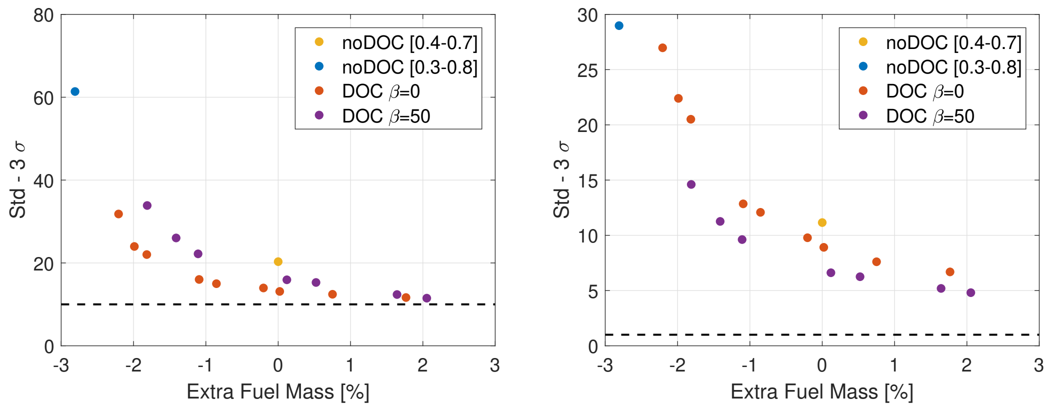
(a) 3σ standard deviation of $x(t_f)$



(b) 3σ standard deviation of $v_h(t_f)$

Figure 5.19: MC Analysis for $h(t_f)$ and $h(t_f)$ with Thrust desensitization

After demonstrating the influence of Thrust perturbations, the desensitization effect is illustrated in Fig. 5.20, where different solutions are exposed to Thrust perturbations. When $\beta = 0$, the Thrust is not penalized, while it is subject to a penalty with $\beta = 50$ in the other scenario. The results align with the explanation provided in Section 4.3.5. The worsening in performance in the x direction is a consequence of state perturbations. This is attributed to the cost function's focus on the vertical direction, which arises from the penalization of $\mathbf{\Omega}_3(t_f)$, that leads to the increment of sensitivity to state perturbations in the x direction. However, Fig. 5.20b clearly highlights the impact of Thrust desensitization on the final altitude. The solution with $\beta = 0$, which corresponds to the one labeled 'DOC T' in Fig. 5.19, performs worse than the $\beta = 50$ solution in vertical direction.



(a) 3σ standard deviation of $x(t_f)$ (b) 3σ standard deviation of $v_h(t_f)$

Figure 5.20: MC Analysis for $h(t_f)$ and $h(t_f)$ with Thrust desensitization

5.4. CPU Time Analysis

Considering that the single desensitization strategy proves to be the most suitable choice for the problem considered, the subsequent topic for discussion pertains to the computational time required to obtain such solutions. As mentioned in Section 5.1, various architectural designs and software tools have been employed and tested, and this section aims to provide insights into the most efficient implementation for each of them.

All the CPU times presented here are derived from a machine equipped with an Intel 12th Gen Core™ i5-1250P processor. These times are calculated as the mean values of 15 runs for each case, a practice adopted to minimize the influence of perturbations. Additionally, a tolerance of 10^{-6} was set for GPOPS, ensuring the accuracy of the results. An important note to consider is that while different tests are categorized into four main paragraphs for clarity, the results in one paragraph are interconnected with the others, and the complete context is provided at the end of the section.

As a general procedure, the implementation of the DOC was carried out in a consistent manner across all cases. Since IPOPT necessitates an initial guess for the optimal solution, the approach involved computing the noDOC trajectory and utilizing it as the initial guess for each DOC case. This was done by first obtaining the trajectory with $\alpha = 0$, and then gradually increasing the α weight to generate the nominal trajectories. Further details on how α was incremented are provided in the second paragraph of this section.

Sensitivity Term The first topic of discussion centers on the comparison of different implementation strategies employed for computing the sensitivity terms. While the presence of $\mathbf{\Omega}(t)$ deserves attention, its computation is relatively straightforward and does not allow for much flexibility in implementation. Therefore, this section focuses on $\mathbf{\Lambda}(t)$. The insights are based on the findings presented in Section 3.4.1. In that section, four distinct implementation alternatives are detailed, referred to as S Matrix (Case 1), S Inverse Matrix (Case 2), and Lambda Vector (Case 3). The latter is further divided into two subcases, Case 3a and Case 3b, corresponding to scenarios where the state is augmented with either $(S(t_f)S(t)^{-1})$ components, propagated as per Eq. (3.21), or $\mathbf{\Lambda}(t)$ components, propagated according to Eq. (3.22).

It is important to note that each of these solutions necessitates a different number of additional elements to be incorporated into the state vector. This factor plays a critical role in terms of computational efficiency. Furthermore, without delving into too much detail, it is relevant to mention that the components of the S_f matrix, which are required in Case 1 and 2 to satisfy the final condition, have been treated as constant parameters rather than as states. This choice was made to enhance the computational properties of the algorithm, as discussed in Section 3.4.3. Throughout this process, all available Matlab tools were utilized to optimize CPU times.

Table 5.2 reports the average CPU times required by GPOPS to obtain a solution with $\alpha = 0.01$ using the noDOC solution as the initial guess, and considering all the optimized parameters discussed in Section 4.1.2.

Case	**Extra Variables**	**CPU Time [s]**
1	49 + 49	811.44
2	49 + 49	997.90
3a	49	160.31
3b	7	25.24

Table 5.2: CPU Time - Sensitivity Term

As expected, the implementation that utilizes $\mathbf{\Lambda}(t)$ components to augment the state emerges as the most efficient choice, requiring the fewest additional variables. However, due to the limited number of cases provided, it is not possible to establish a direct relationship between the number of extra variables required and CPU time, even though the increase in CPU time is noticeable. It is important to note that Case 2 was expected to perform better than Case 1, as the S matrix is inverted only once at the final time. However, in the context of GPOPS implementation, the actual outcome contradicts this expectation, even though analytical results suggested the opposite behavior. While the exact reasons for this discrepancy are not entirely clear, it can be attributed to the conditioning number of the problem. The problem with Case 2 implementation exhibits a condition number of 166671, whereas the Case 1 implementation reduces it to 144350, providing evidence of the validity of the results.

Single jump or Homotopy Since the solutions that are comparable to the noDOC solution with $u \in [0.4 - 0.7]$ in terms of mass involve high α weights (around 2), it is essential to investigate the most effective strategy for transitioning from the noDOC solution to the DOC solution with $\alpha = 2$. This paragraph presents two alternatives, named the Single Jump and Homotopy strategies.

In the Single Jump approach, the desired solution is directly obtained from the noDOC initial guess by setting $\alpha = 2$. Conversely, the Homotopy strategy, which is a effectively a quasi-Homotopy approach, involves a prescribed series of intermediate steps to ensure a smooth and uniform variation in the α value. For simplicity, only one intermediate α value has been considered in this case, which is $\alpha = 0.01$.

Table 5.3 shows the CPU times required for these strategies, obtained using the most computationally efficient sensitivity implementation, Case 3b of Table 5.2.

Case	$\boldsymbol{\alpha = 0.01}$	$\boldsymbol{\alpha = 2}$
Homotopy	25.24 s	18.17 s
Single Jump	-	18.95 s

Table 5.3: CPU Time - α steps

The table clearly illustrates that the Single Jump strategy is the most efficient option, as the total CPU time for the Homotopy scheme is higher. This indicates that the primary challenge when transitioning from a noDOC solution to a DOC solution lies change of solution category, rather than in implementing a substantial change in the α value.

Two additional noteworthy points should be considered. The first is that the solver encounters difficulties in converging to an optimal solution with lower α values compared to higher ones, even though solutions with lower α values are closer to the noDOC solution. This distinction is evident when comparing the CPU time required to achieve a solution with $\alpha = 0.01$ (in the first row) and the time needed to reach $\alpha = 2$ directly from the noDOC case. However, it is important to note that the transition from $\alpha = 0.01$ to $\alpha = 2$ demands less time than the shift from $\alpha = 0$ to $\alpha = 2$, as one might expect.

Softwares Comparison In order to determine the most suitable software for the given problem, it is essential to perform a thorough comparison in terms of CPU time. The three solutions under evaluation are as follows: GPOPS-v1.0 (Case A), GPOPS-v2.5 (Case B), and GPOPS-v2.5 with Adigator (Case C). It is important to note that different GPOPS settings have been applied to each case to optimize their performance, although these specific settings are not detailed here. These configurations have been fine-tuned based on CPU time minimization. The comparison involves transitioning from a noDOC solution (with $\alpha = 0$) to a relevant DOC solution (with $\alpha = 2$) using the Single Jump approach with $\boldsymbol{\Lambda}(t)$ components to augment the state.

Before delving into the analysis of the results, it is important to clarify the meaning of the third column. When using the Adigator tool, automatic differentiation can be performed either during the minimization process of IPOPT or offline. This distinction arises because the problem dynamics are already known and well-defined. However, it is crucial to

Case	CPU Time	CPU Time & AD
Case A	16.13 s	-
Case B	18.95 s	-
Case C	16.55 s	22.40 s

Table 5.4: CPU Time - Softwares

recognize that performing online derivatives computation during the optimization process can impact algorithm efficiency, resulting in increased CPU time, as demonstrated in Table 5.4. In general, it is advantageous to compute the Jacobian and Hessian matrices beforehand, as it reduces CPU time. Nonetheless, this approach may not be feasible for online applications, such as real-time scenarios.

Overall, the performances of the two GPOPS versions are comparable, although Case A exhibits a slight reduction in CPU time. Since a detailed analysis of the two software versions is beyond the scope of this discussion, this result can be accepted as it stands, without further investigation.

noDOC comparison Once the most efficient version of the DOC implementation is identified, it is essential to compare it with an equivalent noDOC solution in terms of CPU time. While the performance improvements in final state dispersion are noteworthy, they come at a substantial cost in terms of CPU time. As demonstrated in Table 5.4, the minimum achievable CPU time is 16.13 seconds. In contrast, the corresponding noDOC case, with the same weights and tuning, only requires 0.33 seconds to compute, resulting in a CPU time that is 50 times smaller.

It is important to note that these CPU times have been measured on a single machine, so it is not appropriate to generalize that DOC invariably demands 50 times the CPU time of a noDOC solution. However, this factor allows to gauge the actual impact of DOC on the solution process, which appears to be extremely relevant.

6 | Conclusions and future developments

This chapter aims to highlight the most significant results and discussing the advantages and disadvantages of the solutions proposed. While specific design solutions were previously detailed in Chapter 5, a brief summary is reiterated in Section 6.1 to emphasize the key outcomes of this work. The latter part focuses on suggesting potential future developments for the application under consideration. Given the preliminary nature of this study, numerous avenues remain unexplored, but the most pertinent ones are presented in Section 6.2.

6.1. Conclusions

As demonstrated in this study, the integration of Desensitized Optimal Control (DOC) into GPOPS for the examined problem has proven to be highly successful, demonstrating both computational efficiency and performance improvements with respect to classical architectures.

Addressing the research questions outlined in Section 1.2, the initial focus is on the effective incorporation of the strategy within the GPOPS environment. Despite the inherent complexity of the process, successful implementation was achieved by utilizing a convenient form of sensitivity terms, denoted as $\mathbf{\Lambda}(t)$ and $\mathbf{\Omega}(t)$, and by identifying an optimal strategy for computing feedback gains through the Linear Quadratic Regulator (LQR) approach. Certain limitations persist, such as the constant setting of the LQR matrix R throughout the entire trajectory, but the results prove to be satisfactory. This achievement is closely tied to the second research question, as the successful implementation of the proposed methodology was made possible through an optimized form. Without this optimization, as indicated in Table 5.2, the computational times for finding a solution would have been prohibitively high, making the validation and characterization of the implemented scheme more challenging. Therefore, the implementation architecture and its optimal form are interconnected.

Once the implementation architecture is established, in accordance with the third research question, the method needs to be characterized and refined to meet the specified performance requirements. Despite the apparent simplicity of this process, numerous factors interact, requiring a careful tuning of each when possible. The fundamental steps are outlined in Section 4.3; however, they are case-dependent, and it remains unclear whether they are applicable to entirely different scenarios of the same problem (e.g., different initial conditions). The role of the Feedback Capability Factor $\eta(t)$ has been extensively analyzed and the innovative solution of Marginal DOC Coefficient introduced. Additionally, the concept of dominant sensitivity is shown and utilized extensively, along with clarifying the impact of LQR matrices on the optimal solution. While different desensitization strategies have been considered, they proved to be less efficient than the dominant sensitivity-based one, as discussed in Section 5.3.4.

Based on the results, the method's actual performance is measured in terms of the covariance of the final state. The comprehensive analysis suggests that the DOC strategy outperforms the noDOC approach, with some flexibility in trajectory design choices. Three equivalent interpretations are possible, highlighting different factors. The first suggests that DOC, with the marginal coefficient, achieves the same performance as the noDOC solution with reduced mass. Furthermore, the method can be seen as a means to reduce errors while maintaining the same fuel mass consumption or a way to enhance performance with a slight increase in fuel mass. These effects can be interchanged by modifying a single factor: α. Another significant outcome is that DOC performance can be interpreted and evaluated solely in light of this synthetic multiplying coefficient α, provided that the internal DOC weights α_i and β_i assume reasonable values.

It is important to note that, to achieve the most effective formulation of the DOC problem, various contributions have been amalgamated into a unified and comprehensive DOC form, as outlined in Section 3.2.1. This formulation is not case-dependent and can be considered for any kind of problem once the sensitivity terms are appropriately defined. While this was not the primary goal of the thesis, it naturally emerged in the quest to address the research questions.

Despite the seemingly straightforward nature of the concepts presented, the tuning and management of the algorithm pose challenges. The tuning process is demanding due to the interplay between different variables, even for a quasi-linear problem like the one analyzed in this study. The difficulty for non-linear problems or more complex dynamical systems is uncertain. A crucial point to consider is the validity and optimality of the numerical results. The thesis aimed to construct a framework and thoroughly characterize it to achieve the best possible performance. While this objective has been largely accomplished, the results should not be considered absolutely optimal, as there are no criteria to assess the optimality of the chosen design variables. Although a general overview has been provided, it cannot be ruled out that some relevant trends or relationships have not been explicitly mapped.

6.2. Future directions

Among the various alternatives available, some weaknesses of the method are identified as potential avenues for future research. Despite the satisfactory performance results and the well-defined desensitization strategy, there is room for improvement in the determination of feedback gains along two distinct directions.

The first aspect, as previously discussed in Section 3.2.2, involves the creation of a reliable method for treating gains as optimization variables. This aspect remains absent in the DOC theory but could serve as a crucial element in establishing a comprehensive theory capable of simultaneously optimizing trajectory and feedback gains. While structured approaches like the LQR employed in this study offer effective solutions, the exploration of a free gains approach, if properly developed, holds promise for further enhancements. Various methods could be explored to address this issue, ensuring the minimum requirement of the controller's stability. Additional enhancements might involve introducing constraints on gain margins or noise rejection performance.

Further improvements in feedback gains can be pursued without altering the DOC theory, as discussed in the first set of improvement directions. Simpler solutions, such as finding a reasonable way to have varying Q and R along the trajectory or defining feedback gains through alternative strategies like H-inf, can be explored. These suggestions serve as initial pointers, and numerous other modifications could be implemented to achieve superior performance.

The final noteworthy observation underscores that, given the preliminary stage of the work, certain robust assumptions were taken into account. Future advancements could center on minimizing the number of assumptions or adjusting them to better correspond with real-world scenarios.

Bibliography

[1] Evgeny Slyuta. Chapter 3 - the luna program. In Andrea Longobardo, editor, *Sample Return Missions*, pages 37–78. Elsevier, 2021.

[2] Gerald A. Soffen and Clayton W. Snyder. The first viking mission to mars. *Science*, 193(4255):759–766, 1976.

[3] Bruce I Larrimer. *Promise Denied - NASA's X-34 and the Quest for Cheap, Reusable Access to Space.* National Aeronautics and Space Administration, 1994.

[4] Ray O. Charette, Don A. Steinmeyer, and Ray R. Smiljanic. Delta clipper lessons learned for increased operability in reusable space vehicles. *AIP Conference Proceedings*, 420(1):969–978, 01 1998.

[5] Satoshi Nonaka, Hiroyuki Nishida, Hiroyuki Kato, Hiroyuki Ogawa, and Yoshifumi Inatani. Vertical landing aerodynamics of reusable rocket vehicle. *Transactions of the Japan Society for Aeronautical and Space Sciences, Aerospace Techonolgy Japan*, 10:1–4, 2012.

[6] Seth B Aaron, Dylan Conway, Daniel Clouse, Andrew E Johnson, Yang Cheng, Adnan I Ansar, Nikolas Trawny, Stefan R Bieniawski, Mark Castelluccio, and Sam Pedrotty. Performance analysis of terrain relative navigation using blue origin new shepard suborbital flight telemetry. In *AIAA SCITECH 2022 Forum*, page 1830, 2022.

[7] Lars Blackmore. Autonomous precision landing of space rockets. In *Frontiers of Engineering: Reports on Leading-Edge Engineering from the 2016 Symposium*, volume 46, pages 15–20. The Bridge Washington, DC, 2016.

[8] Marco Sagliano, Michael Dumke, and Stephan Theil. Simulations and flight tests of a new nonlinear controller for the eagle lander. *Journal of Spacecraft and Rockets*, 56(1):259–272, 2019.

[9] Massimo Ferlin, Badr Rmili, Sebastien LE Martelot, Anouk Schindler, Olivier Boisneau, Stéphane DUSSY, Stéphane Querry, Błażej Marciniak, Paweł Surmacz, Marek

Płochów, et al. Frog project: small demonstrator for big ambitions in rlv european objectives. status quo and results. *Eucass*, 2022.

[10] Etienne Dumont, Shinji Ishimoto, Pascal Tatiossian, Josef Klevanski, Bodo Reimann, Tobias Ecker, Lars Witte, Johannes Riehmer, Marco Sagliano, Sofia Giagkozoglou Vincenzino, et al. Callisto: a demonstrator for reusable launcher key technologies. *Transactions of the Japan Society for Aeronautical and Space Sciences, Aerospace Technology Japan*, 19(1):106–115, 2021.

[11] Marco Sagliano, Shinji Ishimoto, José A. Macés-Hernández, David Seelbinder, and Etienne Dumont. Guidance and control strategy for the callisto flight experiment. In *8th European Conference for Aeronautics and Aerospace Sciences*, 07 2019.

[12] Marco Sagliano, Tsukamoto Taro, Ansgar Heidecker, Josè A. Macés Hernández, Stefano Farì, Markus Schlotterer, Woicke Svenja, David Seelbinder, Shinji Ishimoto, and Etienne Dumont. Robust control for reusable rockets via structured h-infinity synthesis. In *11th International ESA Conference on Guidance, Navigation & Control Systems*, 2021.

[13] Etienne Dumont, Michel Illig, Shinji Ishimoto, Christophe Chavagnac, Yasuhiro Saito, Sven Krummen, Silas Eichel, Hauke Martens, Sofia Giagkozoglou, Janis S. Häseker, Tobias Ecker, Josef Klevanski, Felix Krziwanie, Rotärmel Waldemar, Silvio Schröder, Anton Schneider, Christian Grimm, Svenja Woicke, Marco Sagliano, Markus Schlotterer, Markus Markgraf, Benjamin Braun, Moritz Aicher, Lale E Briese, Ivaylo Petkov, Johannes Riehmer, and Bodo Reimann. Callisto: A prototype paving the way for reusable launch vehicles in europe and japan. In *73rd International Astronautical Congress (IAC)*, 2022.

[14] Gabriele De Zaiacomo, Gonzalo Blanco Arnao, Riccardo Bunt, and Davide Bonetti. Mission engineering for the retalt vtvl launcher. *CEAS Space Journal*, 14(3):533–549, 2022.

[15] Hans Seywald and R.R. Kumar. Desensitized optimal trajectories. *Advances in the Astronautical Sciences, light Mechanics*, pages 103–116, 01 1996.

[16] Haijun Shen, Hans Seywald, and Richard W. Powell. Desensitizing the minimum-fuel powered descent for mars pinpoint landing. *Journal of Guidance, Control, and Dynamics*, 33(1):108–115, 2010.

[17] Anil V. Rao, David A. Benson, Christopher Darby, Michael A. Patterson, Camila Francolin, Ilyssa Sanders, and Geoffrey T. Huntington. Algorithm 902: Gpops, a

matlab software for solving multiple-phase optimal control problems using the gauss pseudospectral method. *ACM Trans. Math. Softw.*, 37(2), 2010.

[18] Alfio Quateroni. *Numerical Models for Differential Problems.* Springer Milano, 2014.

[19] Ivar Stakgold and Michael Holst. *Green's Functions and Boundary Value Problems.* John Wiley & Sons, Inc., 2011.

[20] Donald E. Kirk. *Convex Optimization.* Dover Publications Inc., 2004.

[21] Artuhr E. Bryson and Yu-Chi Ho. *Applied Optimal Control.* Taylor & Francis, 1975.

[22] Lev S. Pontryagin, Vladimir G. Boltyanskii, Revaz V. Gamkrelidze, and Evgenii F. Mishchenko. *The Mathematical Theory of Optimal Processes.* CRC Press, Taylor & Francis Group, 1986.

[23] John T. Betts. *Practical Methods for Optimal Control Using Nonlinear Programming.* Siam, 2010.

[24] Anil Rao. A survey of numerical methods for optimal control. *Advances in the Astronautical Sciences*, 135, 01 2010.

[25] Harold W. Kuhn and Albert W. Tucker. Nonlinear programming. In *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 481–492. University of California Press, 1951.

[26] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization.* Cambridge University Press, 2015.

[27] John T. Betts. Survey of numerical methods for trajectory optimization. *Journal of Guidance, Control, and Dynamics*, 21(2):193–207, 1998.

[28] Ryan Russell. Global search for planar and three-dimensional periodic orbits near europa. *The Journal of the Astronautical Sciences*, 54, 06 2012.

[29] Philip E. Gill, Walter Murray, and Michael A. Saunders. Snopt: An sqp algorithm for large-scale constrained optimization. *SIAM Review*, page 99–131, 01 2002.

[30] Richard H. Byrd, Jorge Nocedal, and Richard A. Waltz. Knitro: An integrated package for nonlinear optimization. *Large Scale Nonlinear Optimization, Springer Verlag*, page 35–59, 01 2006.

[31] John T. Betts and Walter P. Huffman. A sparse nonlinear optimization algorithm. *Journal of Optimization Theory and Applications*, page 519–541, 09 1994.

[32] Josef Stoer and Roland Bulirsch. *Introduction to Numerical Analysis*. Springer, 2002.

[33] Germund Dahlquist and Ake Björck. *Numerical Methods.* John Wiley & Sons, Inc., 2011.

[34] Divya Garg. *Advances in global pseudospectral methods for optimal control.* PhD thesis, Graduate School, University of Florida, 2011.

[35] Marco Sagliano. *Development of a Novel Algorithm for High Performance Reentry Guidance.* PhD thesis, Bremen University, 2016.

[36] Marco Sagliano. Generalized hp pseudospectral-convex programming for powered descent and landing. *Journal of Guidance, Control, and Dynamics*, 42(7):1562–1570, 2019.

[37] Marco Sagliano, Stephan Theil, Michiel Bergsma, Vincenzo D'Onofrio, Lisa Whittle, and Giulia Viavattene. On the radau pseudospectral method: theoretical and implementation advances. *CEAS Space Journal*, 9(3):313–331, 2017.

[38] Matthew Weinstein and Anil Rao. Algorithm 984: Adigator, a toolbox for the algorithmic differentiation of mathematical functions in matlab using source transformation via operator overloading. *ACM Transactions on Mathematical Software*, 44:1–25, 08 2017.

[39] Vincenzo D'Onofrio, Marco Sagliano, and Yunus E Arslantas. Exact hybrid jacobian computation for optimal trajectories via dual number theory. In *AIAA Guidance, Navigation, and Control Conference*, page 0867, 2016.

[40] Divya Garg, Patterson Michael A., Camila Francolin, Christopher L. Darby, Geoffrey T. Huntington, William W. Hager, and Anil V. Rao. Direct trajectory optimization and costate estimation of finite-horizon and infinite-horizon optimal control problems using a radau pseudospectral method. *Computational Optimization and Applications*, pages 335–358, 09 2011.

[41] Venkata Ramana Makkapati, Mehregan Dor, and Panagiotis Tsiotras. Trajectory desensitization in optimal control problems. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 2478–2483, 2018.

[42] Kevin Seywald and Hans Seywald. Desensitized optimal control. In *AIAA SciTech 2019 forum*, page 0651, 2019.

[43] Hans Seywald. Desensitized optimal trajectories with control constraints. *Advances in the Astronautical Sciences*, 114:737–744, 2003.

[44] Behcet Acikmese and Scott R. Ploen. Convex programming approach to powered

descent guidance for mars landing. *Journal of Guidance, Control, and Dynamics*, 30(5):1353–1366, 2007.

[45] Ping Lu. Propellant-optimal powered descent guidance. *Journal of Guidance, Control, and Dynamics*, 41(4):813–826, 2018.

[46] V. Pareto. *Cours d'économie politique: professé à l'Universitè de Lausanne*. Number v. 2 in Cours d'économie politique. F. Rouge, 1897.

[47] Matthew J. Weinstein and Anil V. Rao. Algorithm 984: Adigator, a toolbox for the algorithmic differentiation of mathematical functions in matlab using source transformation via operator overloading. *ACM Trans. Math. Softw.*, 44(2), aug 2017.

[48] Andreas Wächter and Lorenz T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *ACM Trans. Math. Softw.*, 106, 2006.

[49] Philip E. Gill, Walter Murray, and Michael A. Saunders. Snopt: An sqp algorithm for large-scale constrained optimization. *SIAM Journal on Optimization*, 12(4):979–1006, 2002.

A | Appendix A

A, B and C matrices for Rocket Landing Problem are reported here. In order to simplify the notation, two preliminary quantities have to be defined:

$$\gamma = n \cdot T \cdot \cos(\phi) \qquad \beta = \frac{n_T \cdot T}{I_{sp} \cdot g_0}$$

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\gamma\frac{u_x}{m^2} \\ 0 & 0 & 0 & 0 & 0 & 0 & -\gamma\frac{u_y}{m^2} \\ 0 & 0 & 0 & 0 & 0 & 0 & -\gamma\frac{u_h}{m^2} \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \frac{\gamma}{m} & 0 & 0 \\ 0 & \frac{\gamma}{m} & 0 \\ 0 & 0 & \frac{\gamma}{m} \\ -\beta\frac{u_x}{u} & -\beta\frac{u_y}{u} & -\beta\frac{u_h}{u} \end{bmatrix} \qquad \boldsymbol{C} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{u_x \cdot n_T \cdot \cos(\phi)}{m} \\ \frac{u_y \cdot n_T \cdot \cos(\phi)}{m} \\ \frac{u_h \cdot n_T \cdot \cos(\phi)}{m} \\ \frac{u \cdot n_T}{I_{sp} \cdot g_0} \end{bmatrix}$$