Dissertation

Computational Guidance for Low-Thrust Spacecraft in Deep Space Based on Convex Optimization

Department of Aerospace Science and Technology

Politecnico di Milano

Christian Hofmann

2023

Advisor	Prof. Dr. Francesco Topputo
Chair of Doctoral Program	Prof. Dr. Pierangelo Masarati

To my grandmother and grandfather.

Abstract

A reliable, robust, and computationally efficient method to optimize interplanetary low-thrust trajectories is developed in this dissertation. Particular focus is on onboard applications where the reference trajectories are to be computed in real time. The nonlinear optimal control problem is convexified and transformed into a second-order cone program. A trust-region-based sequential convex programming method is presented to obtain solutions that satisfy the nonlinear and nonconvex constraints. As the choice of the discretization and trust-region method often has a significant impact on the performance, thorough assessments of various collocation and control interpolation methods are provided. Moreover, it is investigated how the update mechanism of the trust-region parameters influences the results. Standard and non-standard state vector representations that result in linear and nonlinear unperturbed dynamics are developed and compared. Their influence on the performance of the sequential convex programming method is studied, and two metrics are proposed to assess how the choice of the coordinate set affects the linearization accuracy within the successive convexification approach. A homotopic approach is developed to successively increase the fidelity of the dynamical model. Additional perturbations, a real-thruster model, and operational constraints are considered. Furthermore, a bang-bang mesh refinement procedure is presented to determine accurate control profiles for the fuel-optimal problem. The real-time implementation and performance on a single-board computer similar to real spacecraft hardware are addressed in a closed-loop guidance scenario. A processor-in-the-loop experiment is performed to simulate the deep-space cruise of a spacecraft where the trajectory is repeatedly optimized. It is demonstrated that low-thrust trajectory optimization based on convex programming can achieve sufficiently accurate solutions in an acceptable amount of time. Different global approximations of the nonlinear space-flight dynamics are developed using the Koopman operator theory. Methods to bilinearize and linearize nonlinear dynamical systems are provided. The proposed approach is an alternative way to the standard local Taylor linearization.

Zusammenfassung

In dieser Dissertation wird ein zuverlässiges, robustes und rechnerisch effizientes Verfahren zur Optimierung interplanetarer Niedrigschub-Trajektorien entwickelt. Besonderes Augenmerk liegt auf Onboard-Anwendungen, bei denen die Transferbahnen in Echtzeit berechnet werden müssen. Das nichtlineare Optimierungsproblem wird dabei in ein konvexes Programm umgewandelt. Es wird ein Trust-Region-Verfahren vorgestellt, um iterativ Lösungen zu bestimmen, die alle nichtlinearen und nichtkonvexen Nebenbedingungen erfüllen. Da die Wahl der Diskretisierungs- und Trust-Region-Methode oft einen signifikanten Einfluss auf die Konvergenz und Effizienz hat, werden verschiedene Kollokationsverfahren vorgestellt und verglichen. Außerdem wird untersucht, wie die Schrittweitenbestimmung des Optimierungsverfahrens die Ergebnisse beeinflusst. Weiterhin werden unterschiedliche Zustandsvektordarstellungen entwickelt und verglichen, die zu linearen und nichtlinearen Bewegungsgleichungen für das ungestörte Keplerproblem führen. Ihr Einfluss auf die Konvergenz des sequenziellen, konvexen Optimierungsverfahrens wird untersucht, und es werden zwei Metriken vorgeschlagen, um zu beurteilen, wie sich die Wahl der Koordinaten auf die Genauigkeit der Linearisierung auswirkt. Um ein realitätsnahes Modell zu erhalten, wird eine Homotopie-Methode entwickelt, welche sukzessive weitere Störungsterme, ein reales Antriebssystem, sowie Zeiträume berücksichtigt, in denen kein Schub verfügbar ist. Darüber hinaus wird ein Bang-Bang-Netzverfeinerungsverfahren vorgestellt, um akkurate Schubprofile bestimmen zu können. Die Echtzeit-Implementierung und die Leistung auf einem Einplatinencomputer, der der realen Raumfahrzeug-Hardware ähnelt, werden in einer Simulation mit geschlossenem Regelkreis untersucht. Ein Processor-in-the-Loop-Experiment wird durchgeführt, um den Satellitenflug zu simulieren, bei der die Flugbahn regelmäßig neu berechnet und optimiert wird. Es wird gezeigt, dass eine auf konvexer Optimierung basierende Bahnberechnung ausreichend genaue Lösungen in kurzer Zeit liefern kann. Mit Hilfe der Koopman-Operatorenmethode werden verschiedene globale Näherungen der nichtlinearen Bewegungsgleichungen entwickelt. Der vorgeschlagene Ansatz ist dabei eine Alternative zur lokalen Taylor-Linearisierung.

Sommario

In questa tesi viene sviluppato un metodo affidabile, robusto e computazionalmente efficiente per ottimizzare le traiettorie interplanetarie a bassa spinta. Particolare attenzione è rivolta alle applicazioni a bordo in cui le traiettorie di trasferimento devono essere calcolate in tempo reale. Il problema di ottimizzazione non lineare viene così trasformato in un problema convesso. Viene presentato un metodo di trust-region per determinare iterativamente soluzioni che soddisfino tutti i vincoli non lineari e non convessi. Poiché la scelta del metodo di trust-region ha spesso un impatto significativo sulla convergenza e sull'efficienza dell'algoritmo, vengono presentati e confrontati diversi metodi di aggiornamento della stessa. Inoltre, diversi metodi di trascrizione (detta anche collocazione) vengono analizzati. Vengono anche sviluppate e confrontate diverse rappresentazioni del vettore di stato, che portano a equazioni del moto lineari e non lineari per il problema di Keplero non perturbato. Viene analizzata la loro influenza sulla convergenza della procedura di ottimizzazione (detta Sequential Convex Programming) e vengono proposte due metriche per valutare come la scelta delle coordinate influisca sull'accuratezza della linearizzazione. Per ottenere un modello realistico, è stato sviluppato un metodo di omotopia che tiene conto progressivamente di termini di perturbazione aggiuntivi, di un sistema di propulsione reale e di periodi in cui non è disponibile la spinta. Inoltre, viene presentato un metodo di refinement dei profili di spinta per ottenere soluzioni che siano effettivamente bang-bang. L'implementazione in tempo reale e le prestazioni su un single-board computer simile all'hardware di un veicolo spaziale reale sono studiate in una simulazione closed loop. Viene eseguito un esperimento in cui un processore viene utilizzato per simulare il volo di un satellite, dove la traiettoria viene periodicamente ricalcolata e ottimizzata. Si dimostra che l'ottimizzazione della traiettoria basata sulla ottimizzazione convessa può fornire soluzioni sufficientemente accurate in tempi brevi. Utilizzando il metodo dell'operatore di Koopman vengono sviluppate diverse approssimazioni globali delle equazioni del moto non lineari. L'approccio proposto è un'alternativa alla linearizzazione locale standard di Taylor.

Acknowledgments

I am deeply grateful to everyone who has supported me throughout my doctoral journey. Without their unwavering encouragement, insightful feedback, and valuable contributions, this work would not have been possible.

First and foremost, I would like to express my sincere gratitude to my advisor, Prof. Francesco Topputo, for his exceptional guidance, mentorship, and support throughout my PhD studies. His expertise, patience, and encouragement have been invaluable, and I am truly grateful for his commitment to my academic success.

I would like to extend my sincerest thanks to Prof. Richard Linares at the Department of Aeronautics and Astronautics at the Massachusetts Institute of Technology, for his support and hospitality during my time as a visiting researcher in his group. I am deeply grateful for his expertise on the Koopman operator theory and generous assistance throughout my stay.

I owe a debt of gratitude to my colleagues and friends at the Department of Aerospace Science and Technology at Politecnico di Milano, and at the Department of Aeronautics and Astronautics at the Massachusetts Institute of Technology, who have provided me with invaluable support, encouragement, and friendship throughout my PhD studies.

I would like to extend a special thanks to Andrea C. Morelli and Simone Servadio for their invaluable contributions to the research presented in this thesis. Their expertise, perceptive remarks, and hard work were essential to the success of our collaborative efforts and the publication of our papers.

I would also like to thank Prof. Roberto Armellin and Prof. Behçet Açıkmeşe for their time and effort in reviewing my thesis. Their insightful comments and constructive feedback have significantly improved the quality of this work.

Finally, I would like to express my heartfelt gratitude to my family for their unwavering support and encouragement throughout my academic journey. In particular, I want to thank my wife Larissa for her endless patience, understanding, and love. Her belief in me has been a constant source of inspiration, and I am truly grateful for her presence in my life, which has made this journey all the more fulfilling.

Table of Contents

List of Figures

List of Tables

Nomenclature

Acronyms

GNC	guidance, navigation, and control
CGC	computational guidance and control
OCP	optimal control problem
SCP	sequential convex programming
PMP	Pontryagin's minimum principle
TPBVP	two-point boundary value problem
NLP	nonlinear program(ming)
SOCP	second-order cone program(ming)
DDP	differential dynamic programming
DNN	deep neural networks
RL	reinforcement learning
LG	Legendre–Gauss
LGL	Legendre–Gauss–Lobatto
LGR	Legendre–Gauss–Radau
ZOH	zero-order hold
FOH	first-order hold
LTO	low-thrust trajectory optimization
PDG	powered descent guidance
RPM	Radau pseudospectral method
FRPM	flipped Radau pseudospectral method
MEE	modified equinoctial elements
MOE	modified orbital elements

KS	Kustaanheimo–Stiefel
PIL	processor-in-the-loop
KOT	Koopman operator theory

Symbols

t	time
$\mathbf{f}$	dynamics of the problem
$\mathbf{x}$, $\mathbf{u}$	state and control vectors, respectively
n_x, n_u	number of states and controls, respectively
J	performance index
$\mathcal{H}$	Hamiltonian function
$\mathbf{r}$, $\mathbf{v}$	position and velocity vectors, respectively
m	mass
g_0	gravitational acceleration at sea level
μ	gravitational parameter
I_{sp}	specific impulse
T, $\mathbf{T}$	thrust magnitude, thrust components
$T_{\max}$	maximum thrust magnitude
$\boldsymbol{\lambda}_r$, $\boldsymbol{\lambda}_v$, $\boldsymbol{\lambda}_m$	costates of position, velocity, and mass
S	switching function
$(\cdot)^*$	optimal value of quantity $(\cdot)$
$\boldsymbol{\tau}$	acceleration vector due to thrust
Γ	magnitude of $\boldsymbol{\tau}$
w	modified mass
$\bar{(\cdot)}$	reference value of quantity $(\cdot)$
λ_g, λ_h, λ_l	penalty parameters
$\boldsymbol{\nu}$, $\boldsymbol{\eta}$, $\boldsymbol{\zeta}$	virtual controls
λ_ν, λ_η, λ_ζ	penalty parameters associated with $\boldsymbol{\nu}$, $\boldsymbol{\eta}$, $\boldsymbol{\zeta}$

$R, \alpha, \beta, \delta, \rho_0, \rho_1, \rho_2$	trust-region parameters
$\varepsilon_c, \varepsilon_J, \varepsilon_x$	tolerances for SCP algorithm
P_{in}	input power
$\mathbf{s}_\nu, \mathbf{s}_\eta, \mathbf{s}_\zeta, \mathbf{s}_{\text{TR}}$	slack variables

1 Introduction

Numerous new space missions attempt to satisfy the increasing need for exploration and exploitation of space. The space economy is rapidly increasing, and the interest in new technologies and applications has been growing tremendously in the past few years. Especially minor and major bodies in the deep space are important targets for space agencies and other institutions [1, 2]. The outer space is no longer restricted to space agencies and large companies because CubeSats are now a viable low-cost alternative to conventional spacecraft. They belong to the class of nanosatellites that are comprised of cubic modular units (10 cm edge length, mass of 1.3 kg) and granted universities and small companies access to space at relatively low cost. Their benefit for Earth observation is apparent: faster and cheaper design due to the modular concept, the possibility to launch CubeSats as a second payload, and the role as a low-cost technology demonstrator in space. Yet, their severe limitations regarding power, orbit control, propulsion, and communication prevented them from being utilized in interplanetary missions. The recent success of NASA's MarCO mission, however, has shown that interplanetary CubeSats are becoming reality [3]. Judging from the current pace, the number of small deep-space probes is expected to increase considerably in the next few years [4]. Interplanetary missions are on the verge of becoming profitable: lower development costs allow a shift away from the cautious operation of expensive conventional spacecraft to missions with higher risk. This will eventually result in new, precious knowledge of our solar system. Lowering the costs of interplanetary missions through the use of small satellites will therefore be of key importance for solar system science and exploration.

Given the advances in technology and the increasing number of new missions and launches, the current trend towards more autonomy aims at executing flight-related tasks such as guidance, navigation, and control (GNC) on board. NASA's technology roadmap foresees that autonomous GNC will be required for several future missions, including autonomous rendezvous, tours to multiple asteroids, and formation flying [5]. It is expected that an increased level of autonomy is of paramount importance for long-duration transfers in an unknown environment. Time-critical decisions, for example trajectory correction maneuvers, must be made on board using the data acquired on the fly instead of having to rely on the communication with ground stations. The reason is that decision making is required frequently, and time delays, the limited bandwidth and communication windows are potential risks for the success of a mission [6]. Especially quick responses to unexpected behavior such as fault conditions require immediate action.

Although the computational capability of onboard computers has increased continuously in the past years, only minor advances in the guidance and control systems have taken place. It is therefore not surprising that a paradigm shift is currently happening. Rather than calculating and updating the guidance and control[1] actions on ground, these tasks shall be performed on board without human intervention [7]. As already demonstrated in flight tests of the Masten Space Systems Xombie suborbital rocket, real-time guidance seems to become reality [8]. Instead of aiming at a closed-form solution, numerical algorithms are sought to iteratively compute feasible trajectories on spacecraft hardware. In this context, the phrase *computational guidance and control* (CGC) has recently emerged to highlight the computational aspect of the current trend [7]. As opposed to traditional guidance and control (GC) methods, CGC involves some iterative process where commands are computed on board. In particular, traditional closed-form GC laws (that define a mapping from states to controls) are replaced by numerical algorithms that make use of the problem data acquired during the orbital transfer to find the control actions at each time instant. Often, such approaches are more flexible as more constraints can be taken into account.

Yet, such a paradigm shift towards autonomous guidance and control requires tremendous effort. Even though the field of CGC has lead to some promising results, it is just the beginning of a new era where self-driving spacecraft become reality.

1.1 Motivations

The state of the art is to operate all spacecraft from ground. This includes the determination of the reference trajectory (guidance), the current position (navigation), and correction maneuvers (control). These flight-related tasks are of particular importance as they are essential during the whole lifetime of the satellite. Judging from the current pace of new space missions, ground facilities will soon saturate; the costs and manpower to control the spacecraft would be exorbitantly high as interplanetary transfers can last months or even years. Moreover, operating spacecraft from ground limits the mission design and poses great risks as a communication link between spacecraft and Earth is always required. A paradigm shift is needed towards more autonomous spacecraft.

In this dissertation, we focus on the guidance design. Due to the additional uncertainties and deviations resulting from autonomous navigation, following a predefined nominal trajectory does not seem appropriate anymore. Instead, recomputing the trajectory in real time is desirable in complex and uncertain environments. This, however, requires solving an optimal control problem (OCP), which is

[1]Even though guidance and control are often used interchangeably, we refer to *guidance* as the determination of the control actions that are required to steer the vehicle towards a desired state while satisfying certain constraints. *Control*, in contrast, aims at finding the required forces and torques for the actuators while preserving stability.

already a demanding task. Even though shifting it on board is an enormous step, it is of paramount importance to enable a sustainable exploration and exploitation of the deep space. Computing the reference trajectory on the fly poses risks as the algorithm must repeatedly solve an optimization problem in real time. Reliability (a feasible solution must be obtained at any instant), optimality (a cost function is to be minimized), accuracy (the solution must meet accuracy requirements on states and controls), robustness (the algorithm must be robust against disturbances), and computational efficiency (the algorithm must be compatible with available onboard hardware) are essential criteria for autonomous guidance algorithms. As the guidance design has always been performed on ground, none of the current techniques fulfills all requirements.

In contrast to conventional chemical propulsion systems that produce high thrust, spacecraft can also be equipped with low-thrust engines that provide only little thrust. As the state of the spacecraft changes slowly due to the small control actions, transfer times increase because the thruster has to operate over a significantly larger portion of the flight time. This causes new challenges and requires new trajectory design techniques.

A computationally efficient method is therefore desirable that allows for a reliable, rapid computation of the reference trajectory, potentially on board.

1.2 Research Question

Using convex optimization techniques, how can the optimization of interplanetary low-thrust trajectories be enhanced in terms of reliability, accuracy, robustness, and computational efficiency, with particular focus on onboard applications?

In this context, we propose to combine trajectory optimization and guidance. By making trajectory optimization more reliable, accurate, robust, and computationally efficient, we intend to use it as a guidance approach on board where the trajectory is recomputed when needed. The following objectives are defined:

1) Development of an algorithm with improved reliability, accuracy, robustness, and computational efficiency compared to the state-of-the-art low-thrust trajectory optimization methods.

2) Demonstration that the high-level requirements for onboard guidance are fulfilled.

1.3 Contributions

We summarize the main contributions of this dissertation:

1) A reliable, rapid, and computationally efficient low-thrust trajectory optimization method based on convex programming is developed for interplanetary transfers [9–11]. The output is a software that we call Convex Low-Thrust Trajectory Optimizer (COLTO) where high-fidelity models, a real-thruster model, no-thrust constraints, and mesh refinement are considered.

2) Thorough assessments of discretization and trust-region methods [12, 13], and also of different state vector representations are provided [14]. As the choice of the methods and coordinates often has a significant impact on the performance, we expect the results to be helpful for other researchers.

3) The implementation and performance on a single-board computer are addressed in a closed-loop guidance scenario [15]. A processor-in-the-loop experiment is performed to simulate the deep-space cruise of a spacecraft where the trajectory is repeatedly reoptimized. It is demonstrated that low-thrust trajectory optimization based on convex programming can achieve sufficiently accurate solutions in an acceptable amount of time. This serves as a proof of concept and first step towards autonomous guidance in real space missions.

4) Different global approximations of the nonlinear space-flight dynamics are developed using the Koopman operator theory [16]. As reducing the complexity of nonlinear systems is a desirable goal regardless of the application, the proposed approach is an alternative way to linearize a nonlinear dynamical system by lifting it into a higher-dimensional space.

1.4 Outline of Dissertation

This dissertation is structured as follows. First, some theoretical background of optimal control theory and numerical optimization is presented in Chapter 2. The necessary conditions of optimality are introduced, and relevant equations for solving OCPs are presented.

Chapter 3 gives an overview of the most important guidance methods along with their advantages and disadvantages. Furthermore, the high-level requirements for onboard guidance design are introduced, and each method is assessed based on these requirements.

The state of the art is reviewed in Chapter 4. In particular, the most important developments of convex optimization methods for aerospace applications are summarized, with a focus on low-thrust trajectory optimization. In addition, an overview of different discretization and trust-region methods is presented, and past works that consider high-fidelity models are reviewed. The most common state vector representations for describing the equations of motion are given, and recent work on the linearization of nonlinear dynamical systems is discussed.

Chapter 5 discusses the sequential convex programming (SCP) method, and how the constraints are convexified. A flowchart of the algorithm is provided, and three different trust-region methods are presented.

Several discretization methods are developed in Chapter 6. Specifically, two pseudospectral methods, a method based on Hermite interpolation, and a control interpolation method are discussed. Their performance within SCP is assessed in several orbital transfers, and the influence of different trust-region methods, number of nodes, and initial guesses on the success rate, iterations, and final mass is investigated.

In Chapter 7, various standard and non-standard state vector representations are introduced. Their influence on the performance of SCP is studied, and two metrics are proposed to assess how the choice of the coordinate set affects the linearization approach within SCP.

A homotopic approach for high-fidelity convex optimization is developed in Chapter 8. Specifically, n-body dynamics, solar radiation pressure, no-thrust constraints, and variable specific impulse and maximum thrust are considered. A mesh refinement procedure is presented, and several case studies are provided to demonstrate the effectiveness of the approach.

The closed-loop guidance simulation is presented in Chapter 9. A processor-in-the-loop experiment is described where the reference trajectory is repeatedly reoptimized on a single-board computer comparable to real spacecraft hardware. Monte Carlo analyses of transfers to near-Earth asteroids are performed to assess the reliability, accuracy, and computational time. Moreover, the real-time implementation is discussed. The parsing of various constraints into standard form is provided, and the constant and changing elements of the optimization problem are highlighted.

In Chapter 10, the Koopman operator theory is introduced. Methods are presented to bilinearize and linearize a nonlinear dynamical system, and two examples are given to demonstrate the accuracy of the method.

Finally, Chapter 11 concludes this dissertation, and directions for future work are given.

2 Theoretical Background

The core part of this dissertation deals with the computation of reference trajectories for orbital transfers. That is, the control actions (and thus, the transfer trajectory) are to be determined to steer a spacecraft from an initial to a desired final state. Usually, additional constraints are to be satisfied, and the time of flight or fuel consumption are to be minimized. Such optimal control problems require some background in optimal control theory, which is reviewed in the following section. As numerical methods are used in this dissertation to solve OCPs, the basics of numerical optimization are presented. Additionally, the equations of motion are introduced, and the optimal control problem in space flight is stated.

2.1 Optimal Control Theory

We consider autonomous physical systems where the equations of motion are described by the function $\mathbf{f}\colon \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$,

$$\dot{\mathbf{x}}(t) := \frac{\mathrm{d}\mathbf{x}(t)}{\mathrm{d}t} = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \tag{2.1}$$

where $t \in \mathbb{R}$ is the time, $\mathbf{x}(t)\colon \mathbb{R} \to \mathbb{R}^{n_x}$ the state, and $\mathbf{u}(t)\colon \mathbb{R} \to \mathbb{R}^{n_u}$ the control. The goal of optimal control problems is to minimize a performance index $J\colon \mathbb{R}^{n_u} \times \mathbb{R} \to \mathbb{R}$ of the form

$$J(\mathbf{u}(t), t_f) := \phi(t_f, \mathbf{x}(t_f)) + \int_{t_0}^{t_f} L(\mathbf{x}(t), \mathbf{u}(t))\,\mathrm{d}t \tag{2.2}$$

$\phi\colon \mathbb{R} \times \mathbb{R}^{n_x} \to \mathbb{R}$ denotes the terminal cost at the final time t_f, and $L\colon \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}$ refers to the integral cost for $t \in [t_0, t_f]$, t_0 being the initial time. The general form of an optimal control problem is therefore given by

$$\begin{aligned}
&\underset{\mathbf{u}(t),\, t_f}{\text{minimize}} && J(\mathbf{u}(t), t_f) && \text{(2.3a)}\\
&\text{subject to:} && \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) && \text{(2.3b)}\\
& && \mathbf{g}(\mathbf{x}(t), \mathbf{u}(t)) \le \mathbf{0} && \text{(2.3c)}\\
& && \mathbf{h}(t_f, \mathbf{x}(t_f)) = \mathbf{0} && \text{(2.3d)}\\
& && \mathbf{x}(t_0) = \mathbf{x}_0 && \text{(2.3e)}
\end{aligned}$$

The functions $\mathbf{g}\colon \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_g}$ and $\mathbf{h}\colon \mathbb{R} \times \mathbb{R}^{n_x} \to \mathbb{R}^{n_h}$ refer to the path and final boundary constraints, respectively. The problem is to find $\mathbf{u}(t)$ such that the performance index in Eq. (2.3a) is minimized (including minimum-time problems with free final time t_f) while satisfying the dynamics in Eq. (2.3b), path constraints in Eq. (2.3c), and terminal and initial boundary constraints in Eqs. (2.3d) and (2.3e), respectively. In optimal control theory, this is addressed using the calculus of variations. We introduce an augmented performance index $\tilde{J}$ [17]

$$\begin{aligned}\tilde{J}(\mathbf{u}(t), t_f) &\coloneqq \phi(t_f, \mathbf{x}(t_f)) + \boldsymbol{\nu}^\top \mathbf{h}(t_f, \mathbf{x}(t_f)) \\ &\quad + \int_{t_0}^{t_f} \left[\tilde{\mathcal{H}}(\mathbf{x}(t), \mathbf{u}(t), \boldsymbol{\lambda}(t), \boldsymbol{\mu}(t)) - \boldsymbol{\lambda}(t)^\top \dot{\mathbf{x}}(t)\right] \mathrm{d}t\end{aligned} \tag{2.4}$$

with constant multipliers $\boldsymbol{\nu}$, and costate functions $\boldsymbol{\lambda}(t)$ and $\boldsymbol{\mu}(t)$. The augmented Hamiltonian $\tilde{\mathcal{H}}\colon \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_h} \to \mathbb{R}$ is defined as

$$\begin{aligned}\tilde{\mathcal{H}}(\mathbf{x}(t), \mathbf{u}(t), \boldsymbol{\lambda}(t), \boldsymbol{\mu}(t)) &\coloneqq L(\mathbf{x}(t), \mathbf{u}(t)) + \boldsymbol{\lambda}(t)^\top \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) + \boldsymbol{\mu}(t)^\top \mathbf{g}(\mathbf{x}(t), \mathbf{u}(t)) \\ &= \mathcal{H}(\mathbf{x}(t), \mathbf{u}(t), \boldsymbol{\lambda}(t)) + \boldsymbol{\mu}(t)^\top \mathbf{g}(\mathbf{x}(t), \mathbf{u}(t))\end{aligned} \tag{2.5}$$

with the regular Hamiltonian $\mathcal{H}$:

$$\mathcal{H}(\mathbf{x}(t), \mathbf{u}(t), \boldsymbol{\lambda}(t)) \coloneqq L(\mathbf{x}(t), \mathbf{u}(t)) + \boldsymbol{\lambda}(t)^\top \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \tag{2.6}$$

Each entry $\mu_i(t)$ of $\boldsymbol{\mu}(t)$ must satisfy

$$\mu_i(t) \begin{cases} > 0, & h_i = 0, \qquad i \in [1, n_h] \\ = 0, & h_i < 0, \qquad i \in [1, n_h] \end{cases} \tag{2.7}$$

where h_i is the ith element of $\mathbf{h}$. Omitting the dependent variables for better readability, the first-order necessary conditions for local optimality then read [18]

$$\dot{\mathbf{x}} = \mathcal{H}_{\boldsymbol{\lambda}}^\top \tag{2.8}$$

$$\dot{\lambda} = -\mathcal{H}_{\mathbf{x}}^\top \tag{2.9}$$

$$\left[\tilde{\mathcal{H}} + \phi_t + \boldsymbol{\nu}^\top \mathbf{h}_t\right]_{t_f} = 0 \quad \text{if } t_f \text{ is free} \tag{2.10}$$

$$\left[\boldsymbol{\lambda}^\top - \phi_{\mathbf{x}} - \boldsymbol{\nu}^\top \mathbf{h}_{\mathbf{x}}\right]_{t_f} = \mathbf{0}^\top \tag{2.11}$$

$$[\mathbf{h}]_{t_f} = \mathbf{0} \tag{2.12}$$

where the notation $(\cdot)_j \coloneqq \partial(\cdot)/\partial j$ refers to the partial derivative. Equations (2.8) and (2.9) are called Euler-Lagrange equations, whereas Eqs. (2.10)–(2.12) are called transversality conditions. According to Pontryagin's minimum principle (PMP) [17], an optimal trajectory minimizes the Hamiltonian, and therefore, the optimal control $\mathbf{u}^*$ is given by

$$\mathbf{u}^* = \underset{\mathbf{u} \in \mathcal{U}}{\arg\min}\, \mathcal{H} \tag{2.13}$$

with the set of admissible controls $\mathcal{U}$. The system of nonlinear equations (2.8)–(2.13) results in a two-point boundary value problem (TPBVP) that can be solved numerically.

2.2 Numerical Optimization

Instead of solving the TPBVP in Eqs. (2.8)–(2.13), the OCP can be transcribed into a parameter optimization problem. Rather than working in the continuous time domain, the time interval $[t_0, t_f]$ is divided into $N-1$ segments

$$t_0 = t_1 < t_2 < t_3 < ... < t_N = t_f \tag{2.14}$$

The state and control variables are also discretized and combined into one vector $\mathbf{y} \in \mathbb{R}^{n_y}$

$$\mathbf{y} = [\mathbf{x}_1^\top, \mathbf{x}_2^\top, \dots, \mathbf{x}_N^\top, \mathbf{u}_1^\top, \mathbf{u}_2^\top, \dots, \mathbf{u}_N^\top]^\top \tag{2.15}$$

where each entry $(\cdot)_k$ refers to the quantity at the kth discretization point. An important consequence is that the solution is only known at those discrete points $k = 1, 2, \dots, N$. The constraints in Eqs. (2.3b)–(2.3e) are transformed into a set of algebraic constraints, and the resulting parameter optimization problem then reads [19]

$$\underset{\mathbf{u}}{\text{minimize}} \quad F(\mathbf{y}) \tag{2.16a}$$

$$\text{subject to:} \quad \mathbf{g}(\mathbf{y}) \leq \mathbf{0} \tag{2.16b}$$

$$\mathbf{h}(\mathbf{y}) = \mathbf{0} \tag{2.16c}$$

$F\colon \mathbb{R}^{n_y} \to \mathbb{R}$ is the objective function that is obtained when rewriting the performance index in terms of $\mathbf{y}$, and $\mathbf{g}\colon \mathbb{R}^{n_y} \to \mathbb{R}^{n_g}$ and $\mathbf{h}\colon \mathbb{R}^{n_y} \to \mathbb{R}^{n_h}$ define the constraints. In general, $F(\mathbf{y})$, $\mathbf{g}(\mathbf{y})$, and $\mathbf{h}(\mathbf{y})$ are nonlinear functions, and the problem is a nonlinear programming (NLP) problem. It can be solved using state-of-the-art methods such as sequential quadratic programming or interior-point methods [19]. Analogous to defining the Hamiltonian function for OCPs, we introduce the Lagrangian $\mathcal{L}\colon \mathbb{R}^{n_y} \times \mathbb{R}^{n_h} \times \mathbb{R}^{n_g} \to \mathbb{R}$ [20]:

$$\mathcal{L}(\mathbf{y}, \boldsymbol{\Lambda}, \boldsymbol{\nu}) := F(\mathbf{y}) + \boldsymbol{\Lambda}^\top \mathbf{h}(\mathbf{y}) + \boldsymbol{\nu}^\top \mathbf{g}(\mathbf{y}) \tag{2.17}$$

with Lagrange multipliers $\mathbf{\Lambda} \in \mathbb{R}^{n_h}$ and $\boldsymbol{\nu} \in \mathbb{R}^{n_g}$. At the optimal solution $\mathbf{y}^*$, there exist Lagrange multipliers $\mathbf{\Lambda}^*, \nu^*$, and the following first-order necessary conditions for local optimality hold [20]:

$$\nabla_{\mathbf{y}} \mathcal{L}(\mathbf{y}^*, \mathbf{\Lambda}^*, \boldsymbol{\nu}^*) = \mathbf{0} \tag{2.18}$$

$$\nabla_{\mathbf{\Lambda}} \mathcal{L}(\mathbf{y}^*, \mathbf{\Lambda}^*, \boldsymbol{\nu}^*) = \mathbf{0} \tag{2.19}$$

$$\nabla_{\nu} \mathcal{L}(\mathbf{y}^*, \mathbf{\Lambda}^*, \boldsymbol{\nu}^*) \leq \mathbf{0} \tag{2.20}$$

$$\boldsymbol{\nu}^* \geq \mathbf{0} \tag{2.21}$$

$$\boldsymbol{\nu}^{*\top} \mathbf{g}(\mathbf{y}^*) = 0 \tag{2.22}$$

These conditions are also referred to as Karush–Kuhn–Tucker conditions. The basic idea is to use some numerical method to find the Karush–Kuhn–Tucker points that satisfy Eqs. (2.18) and (2.19).

In contrast to NLP, convex optimization deals with the minimization of a convex function over convex sets. A function $f\colon \mathcal{X} \to \mathbb{R}$ is convex if $\mathcal{X}$ is a convex set, and f satisfies [21, Chapter 3]

$$f(\alpha\, x + (1-\alpha)\, y) \leq \alpha\, f(x) + (1-\alpha)\, f(y) \tag{2.23}$$

for all $x, y \in \mathcal{X}$, and $\alpha \in [0,1]$. An important consequence is that optimal solutions are globally optimal compared to the local optimality for NLPs. The resulting parameter optimization problems are called convex programs. One important class is the second-order cone program (SOCP) that reads [21, Chapter 4]

$$\underset{\mathbf{u}}{\text{minimize}} \quad \mathbf{c}^\top \mathbf{y} \tag{2.24a}$$

$$\text{subject to:} \quad \mathbf{A}\,\mathbf{y} = \mathbf{b} \tag{2.24b}$$

$$\mathbf{G}\,\mathbf{y} + \mathbf{s} = \mathbf{h}, \quad \mathbf{s} \in \mathcal{K} \tag{2.24c}$$

where $\mathbf{y}$ is the decision vector similar to Eq. (2.15), and $\mathbf{c}$, $\mathbf{A}$, $\mathbf{b}$, $\mathbf{G}$, $\mathbf{s}$, and $\mathbf{h}$ are vectors and matrices that define the constraints. $\mathcal{K}$ is a set of convex cones. In case of nonconvex problems, the constraints are to be convexified, and a series of convex subproblems can be solved to obtain an approximate solution to the original problem. This approach is called sequential convex programming (SCP) [22].

2.3 Optimal Control Problem for Space Flight

The motion of a spacecraft around a primary body is governed by the dynamics

$$\dot{\mathbf{r}}(t) = \mathbf{v}(t) \tag{2.25}$$

$$\dot{\mathbf{v}}(t) = -\frac{\mu}{\|\mathbf{r}(t)\|_2^3}\,\mathbf{r}(t) + \sigma(t)\,\frac{T_{\text{max}}}{m(t)}\,\boldsymbol{\alpha}(t) \tag{2.26}$$

$$\dot{m}(t) = -\sigma(t)\,\frac{T_{\text{max}}}{g_0\,I_{\text{sp}}} \tag{2.27}$$

where $\mathbf{r}(t) \in \mathbb{R}^3$, $\mathbf{v}(t) \in \mathbb{R}^3$, and $m(t) \in \mathbb{R}$ denote the position, velocity, and mass of the spacecraft, respectively. $\sigma(t) \in [0, 1]$ is the throttle factor, and $\boldsymbol{\alpha}(t) \in \mathbb{R}^3$ the thrust direction unit vector. μ denotes the gravitational parameter of the primary body, g_0 the gravitational acceleration at sea level, T_{max} the maximum available thrust provided by the propulsion system, and I_{sp} the specific impulse. Note that T_{max} and I_{sp} may also depend on the instantaneous input power of the propulsion system.

For fuel-optimal problems, we seek to minimize fuel usage, which is equivalent to maximizing the mass at the final time t_f:

$$\underset{\sigma(t),\,\boldsymbol{\alpha}(t)}{\text{minimize}} \quad -m(t_f) \tag{2.28}$$

Time-optimal problems instead aim at minimizing the final time:

$$\underset{\sigma(t),\,\boldsymbol{\alpha}(t)}{\text{minimize}} \quad t_f \tag{2.29}$$

In this dissertation, we address the rendezvous problem, i.e., we intend to target a specific point $\left[\mathbf{r}_f^\top, \mathbf{v}_f^\top\right]^\top$ in space. Therefore, the boundary conditions at the initial and final times are

$$\mathbf{r}(t_0) = \mathbf{r}_0, \quad \mathbf{v}(t_0) = \mathbf{v}_0, \quad m(t_0) = m_0 \tag{2.30}$$

$$\mathbf{r}(t_f) = \mathbf{r}_f, \quad \mathbf{v}(t_f) = \mathbf{v}_f \tag{2.31}$$

where the value of the final mass is free. As the engine can provide only limited thrust, the following lower and upper bounds on the throttle factor need to be imposed:

$$0 \le \sigma_{\text{min}} \le \sigma \le \sigma_{\text{max}} \tag{2.32}$$

with the minimum $\sigma_{\text{min}} = 0$ and maximum $\sigma_{\text{max}} = 1$ values of the throttle factor. Defining the state vector as $\mathbf{x} := [\mathbf{r}^\top, \mathbf{v}^\top, m]^\top$, the optimal control problem can be stated as follows:

$$\underset{\sigma(t),\,\boldsymbol{\alpha}(t)}{\text{minimize}} \qquad -m(t_f) \quad \text{or} \quad t_f \tag{2.33a}$$

$$\text{subject to:} \qquad \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \sigma(t), \boldsymbol{\alpha}(t)) \tag{2.33b}$$

$$0 \le \sigma(t) \le 1 \tag{2.33c}$$

$$\|\boldsymbol{\alpha}(t)\|_2 = 1 \tag{2.33d}$$

$$\mathbf{r}(t_0) = \mathbf{r}_0,\ \mathbf{v}(t_0) = \mathbf{v}_0,\ m(t_0) = m_0 \tag{2.33e}$$

$$\mathbf{r}(t_f) = \mathbf{r}_f,\ \mathbf{v}(t_f) = \mathbf{v}_f \tag{2.33f}$$

The Hamiltonian of the fuel-optimal problem is given by [23]

$$\mathcal{H} = \frac{\sigma\, T_{\text{max}}}{g_0\, I_{\text{sp}}} + \boldsymbol{\lambda}_r^\top \mathbf{v} + \boldsymbol{\lambda}_v^\top \left[\mathbf{g}(\mathbf{r}) + \frac{\sigma\, T_{\text{max}}}{m}\, \boldsymbol{\alpha}\right] - \lambda_m \frac{\sigma\, T_{\text{max}}}{g_0\, I_{\text{sp}}} \tag{2.34}$$

where minimizing $\frac{T_{\text{max}}}{g_0\, I_{\text{sp}}} \int_{t_0}^{t_f} \sigma(t)\, \mathrm{d}t$ is equivalent to Eq. (2.28). Moreover, $\mathbf{g}(\mathbf{r}) = -\mu\, \mathbf{r}/r^3$ with $r = \|\mathbf{r}\|_2$. Applying PMP, substituting the optimal thrust direction $\boldsymbol{\alpha}^* = -\boldsymbol{\lambda}_v/\|\boldsymbol{\lambda}_v\| = -\boldsymbol{\lambda}_v/\lambda_v$ into Eq. (2.34), and rearranging terms yields

$$\mathcal{H} = \boldsymbol{\lambda}_r^\top \mathbf{v} + \boldsymbol{\lambda}_v^\top \mathbf{g}(\mathbf{r}) + \frac{u\, T_{\text{max}}}{g_0\, I_{\text{sp}}} \underbrace{\left(1 - \frac{g_0\, I_{\text{sp}}}{m}\, \lambda_v - \lambda_m\right)}_{=:S} \tag{2.35}$$

As an optimal trajectory minimizes the Hamiltonian, the characteristic bang-bang control profile in fuel-optimal problems solely depends on the sign of the switching function S. It is easy to verify that the optimal throttle factor σ^* is given by

$$\sigma^* = \begin{cases} 0, & S > 0 \\ 1, & S < 0 \end{cases} \tag{2.36}$$

If $S = 0$ for some finite duration, σ^* is not determined and can take any value in $[0, 1]$. This is called a *singular arc* [24, Chapter 2]. The Hamiltonian of the time-optimal problem is [23]

$$\mathcal{H} = 1 + \boldsymbol{\lambda}_r^\top \mathbf{v} + \boldsymbol{\lambda}_v^\top \left[\mathbf{g}(\mathbf{r}) + \frac{\sigma\, T_{\text{max}}}{m}\, \boldsymbol{\alpha}\right] - \lambda_m \frac{\sigma\, T_{\text{max}}}{g_0\, I_{\text{sp}}} \tag{2.37}$$

which yields the following switching function:

$$S = -\lambda_v \frac{g_0\, I_{\text{sp}}}{m} - \lambda_m \tag{2.38}$$

In this case, $\sigma^* = 1\ \forall t \in [t_0, t_f]$.

3 Overview and Assessment of Guidance Methods

There is a large variety of methods for the guidance design of spacecraft. An overview is given first, and the advantages and disadvantages of each method are assessed with particular emphasis on onboard applications and their requirements.

3.1 Overview of Methods

We briefly describe (semi-)analytical approaches, direct and indirect methods, dynamic programming, metaheuristics, machine learning, and a selection of other methods.

Analytical and Semianalytical Methods

Analytical solutions exist only for special cases, for example the Hohmann transfer. Most approaches approximate the continuous thrust using simplified models, such as the Kepler (pure Kepler arcs followed by impulsive thrust maneuvers) or Stark model (constant thrust acceleration instead of impulses, [25]). Usually, certain assumptions on the perturbing acceleration or shape of the orbits are made. The results for computing the total velocity increment Δv are often of the form [26]

$$\Delta v = \sqrt{v_0^2 + v_f^2 - 2\, v_0\, v_f \cos\left(\pi/2\right) \Delta i} \tag{3.1}$$

This is an example for the required Δv for a desired inclination change Δi given two circular orbits with initial v_0 and final v_f velocity. In [27], low-thrust transfers between coplanar elliptic orbits are determined analytically using Hamiltonian mechanics. A closed-form solution for transfers between circular, coplanar orbits using elliptic functions is proposed in [28]. Moreover, transfers with constant tangential [29] or radial acceleration can be derived for circular and elliptic orbits using perturbation theory. The work in [30] assumes small eccentricities to calculate low-thrust transfers between coplanar orbits. Additional perturbations such as Earth shadow eclipses and J_2 perturbations are also considered in some works [31, 32]. Furthermore, semi-analytical approaches as in [33] where transfer charts are used to determine approximate solutions also avoid solving an optimal control problem numerically.

Analytical methods are fast and provide an explicit analytical solution. However, solutions exist only for special cases, and often, suboptimal solutions are obtained [24, Chapter 6].

Indirect Methods

In indirect methods, the first-order necessary conditions are derived as per Section 2.1, and the resulting TPBVP is solved. There are three main numerical methods [34, 35]:

Indirect single shooting method: The values of the costates at the initial time are guessed, and the differential equations of the states and costates in Eqs. (2.8) and (2.9) are integrated over $[t_0, t_f]$ (forward or backward in time). The unknown parameters are then adjusted in an iterative process until the boundary and transversality conditions are satisfied. As the integration is performed over the whole time interval, the single shooting method is quite sensitive to the initial guess. Minor changes in the initial conditions can considerably impact convergence.

Indirect multiple shooting method: These numerical sensitivities can be overcome by using the multiple shooting method. The time horizon is divided into subintervals, and the standard single shooting method is applied in each interval. This way, changes in the initial conditions are not as critical due to the shorter integration interval. However, additional continuity conditions on the states and costates are to be imposed between each subinterval.

Indirect collocation: States and costates are parameterized and approximated by piecewise polynomials. The goal is then to find the coefficients of the polynomials by solving the resulting system of nonlinear equations.

The main advantage of indirect methods is the analytical derivation of the optimality conditions, and therefore the high accuracy and local optimality of a converged solution. As the costates are to be included in the dynamical system, the dimension increases. Yet, they suffer from flexibility (e.g., constraints and perturbations may not be incorporated easily), the high sensitivity to the initial guess, and poor robustness [34, 35]. Although improvements have been proposed during the past years, for example through smoothing [36–39] and initialization techniques [40, 41], the convergence issues still remain.

Direct Methods

Direct methods transcribe the infinite-dimensional nonlinear OCP into a finite-dimensional parameter optimization problem as described in Section 2.2. In this context, we refer to gradient-based methods that directly solve the nonlinear program. In general, there are three main gradient-based methods to solve the resulting NLP [34, 35]:

Direct single shooting method: Only the control is parameterized, i.e., a certain function is assumed to approximate the control variables. Given the initial state, time-marching methods are used to integrate the dynamics forward from t_0 to t_f (or backward from t_f to t_0). This is an iterative process where the solver tries to find the controls such that the objective function is minimized and the constraints are satisfied.

Direct multiple shooting method: Again, only the control variables are parameterized. Similar to indirect multiple shooting methods, the time horizon is divided into segments, and the single shooting method is applied in each segment. As the state variables are continuous over the intervals, continuity conditions are required.

Direct collocation: Piecewise polynomials are used to approximate the states and controls. The solver tries to find the coefficients of the polynomials such that the objective function is minimized while satisfying all constraints at the collocation points. The major differences between collocation methods are the choice of the quadrature rule to approximate the differential equations, and the location of the discretization points.

Convex optimization: Convex programs are a special type of mathematical programs where the constraints and objective function are convex. In this case, there exists a globally optimal solution. Nonlinear programs can be convexified and successively solved to obtain an approximate solution [22].

Direct methods have a larger convergence domain than indirect methods, and complex constraints can easily be included [34]. Yet, the solution is only known at the discretization points, and a large number of points may be required to achieve high accuracy. Onboard computers lack in general the computational capability to solve the resulting large-scale nonlinear optimization problem in little time. Still, some studies refined the existing methods to further lower the computational effort and increase accuracy and reliability [42–44].

Dynamic Programming

Rather than finding open-loop solutions like in direct and indirect methods, dynamic programming aims at determining a feedback control law. The main advantage of feedback solutions is that they provide the optimal control trajectory given the current state. Based on Bellman's principle of optimality [45], the control law can be obtained by solving the Hamilton–Jacobi–Bellman equation. This system of partial differential equations, however, is often difficult to solve due to the curse of dimensionality [17]. As this would yield globally optimal solutions, some works use approximations to reduce the number of variables [46]. With regard to trajectory optimization, the probably most important method is differential dynamic programming (DDP) [47–49]. The optimization problem is divided into subproblems where the dynamics and objective function are approximated using a second-order Taylor series about a reference trajectory. As a consequence, the problem becomes computationally tractable, but only locally optimal solutions are obtained.

Similar to NLP-based methods, DDP is relatively robust against poor initial guesses, and it is straightforward to include complex constraints. Yet, it suffers from the same drawbacks, such as high computational effort [49].

Metaheuristics

Metaheuristics are global optimization algorithms that employ heuristic rules to find an optimal solution [50]. Therefore, gradients and an initial guess are not required. These methods are nature-inspired or use stochastic processes to iteratively find an optimal solution. Examples include evolutionary algorithms such as genetic algorithms [51, 52] and differential evolution [53, 54], and swarm algorithms such as ant colony optimization [55, 56] and particle swarm optimization [57, 58].

Although they are easy to implement and do not require an initial guess, the computational effort is high and the robustness is poor as there is mathematically no guarantee that an optimal solution will be obtained [50].

Machine Learning

Techniques based on machine learning follow a different approach, but become increasingly popular. A database of thousands of optimal trajectories is created first, and the optimal control structure is then learned using deep neural networks (DNN) [59–61]. Even though the generation of such a database is time consuming, this can be done offline. Once the control law is determined, the computational effort to compute the controls is negligible, making DNN a viable method for real-time applications [62]. Support Vector Machines use a different approach for the training, but also belong to the class of supervised learning [63]. Reinforcement learning (RL) was developed to account for disturbances or an uncertain environment. Based on some reward function, the control actions are selected such that the reward is maximized [64]. For example, this approach was applied to the hovering near asteroids [65]. Deep reinforced learning combines DNN and RL and is used in [66] and [67] for guidance and landing problems, respectively. More recently, physics-informed neural networks have been developed where the underlying physical laws of the dynamical system are incorporated [68].

Methods based on machine learning can easily be implemented and run on spacecraft hardware due to the low computational effort. Disadvantages are their lack of flexibility and optimality in case of deviations from the learned trajectories [62, 64].

Other Methods

Naturally, there are many more techniques that cannot be covered in detail in this dissertation. We briefly comment on a selection that has been used frequently for low-thrust trajectory optimization.

Feedback-driven methods are computationally simple and hence a popular choice for onboard guidance design. They are often based on Lyapunov stability theory, and the optimal control law is obtained by minimizing some Lyapunov function [69–72]. As such methods are not based on optimal control theory, they can only provide suboptimal solutions [73].

Shape-based methods assume a certain shape for the state trajectory, and the controls are obtained by imposing the dynamical constraints [74]. A higher accuracy can be achieved and additional constraints can be considered by optimizing the coefficients of the polynomial, for example within methods based on finite Fourier series [75, 76].

Hybrid methods combine two or more methods to exploit the advantages of each of them, for example by using the solution of evolutionary algorithms as an initial guess for an indirect or direct method [77]. Some researchers have combined machine learning techniques to improve the optimality of feedback-driven approaches [78].

3.2 Assessment and Selection

Shifting the guidance task on board poses a great challenge as the algorithm must repeatedly recompute the reference trajectory and guarantee a (near-optimal) solution in real time. This dissertation focuses on interplanetary transfers, in particular the deep-space cruise. The main characteristics to be considered are:

- Long time of flight: The time of flight in interplanetary transfers may span several years due to the low-thrust propulsion. This is relevant for the numerical integration and also discretization.
- Small number of revolutions: In contrast to Earth-bound transfers that may require hundreds of revolutions (e.g., geostationary transfer orbit to geostationary orbit), the interplanetary transfers considered in this work generally require fewer than ten revolutions. Especially the required number of discretization points to capture the trajectory accurately is affected.
- Recomputation of reference trajectory: In an onboard guidance scenario, the control actions (and thus, the reference trajectory) are to be recomputed in real time.

We define the following high-level requirements for onboard guidance:

1) Reliability: A solution shall be provided at all times, i.e., a high success rate and a large convergence domain are necessary.
2) Onboard capability: The algorithm shall be compatible with the limited hardware on board. Moreover, a solution shall be obtained within few minutes.

3) Optimality: The algorithm shall be able to find (at least locally) optimal solutions.

4) Robustness: The algorithm shall be able to deal with disturbances and exploit previous solutions when recomputing the trajectory. Moreover, it shall be robust against (small) changes in the input parameters.

5) Accuracy: The obtained solution shall be sufficiently accurate. We define the accuracy requirement to be $1000\,\mathrm{km}$ (position) and $1\,\mathrm{m\,s^{-1}}$ (velocity) for rendezvous problems. This corresponds to the accuracy that optical navigation can currently achieve [79, 80].

6) Flexibility: The method shall be capable of handling various scenarios with different constraints easily (e.g., high-fidelity models or operational constraints).

The advantages and disadvantages of each guidance method are summarized in Table 3.1. Our assessment of each method based on the high-level requirements for onboard guidance is given in Table 3.2. We use the ratings *poor*, *satisfactory*, *good*, and *very good*.

An algorithm that does not require an initial guess seems preferable in general. However, the first reference trajectory can be computed offline and stored on the flight processor. Even in case of disturbances and deviations from the reference, it is likely (and desirable) that the subsequent trajectories lie in the vicinity of the original reference trajectory. The reason is that fuel-optimal solutions are sought, and large deviations may result in a (significantly) higher propellant consumption. This is not acceptable, especially due to the limited amount of propellant.

Analytical solutions are desirable in general, but do not exist for complex interplanetary transfers without any limiting assumptions. Even though metaheuristics may be able to find a globally optimal solution, they are not suitable for onboard applications because no guarantees on convergence can be made due to the non-deterministic behavior.

Indirect methods determine highly accurate and locally optimal solutions. Due to the smaller number of variables compared to direct methods, they are an excellent choice for many-revolution transfers that would require a large number of discretization points. Yet, they suffer from poor flexibility (e.g., operational constraints and perturbations cannot be incorporated easily) and reliability. Especially the small convergence domain when more complex problems and constraints are considered is undesirable. Methods based on machine learning are often considered the most promising methods regarding onboard applications. The reason is that the computationally expensive generation of the training database can be carried out on ground, and the actual computation of the control actions can be done quickly in real time. The main disadvantage is the poor accuracy if unexpected deviations from the reference trajectory occur that were not modeled during the training process. As a consequence, the approximations of the control actions often result in large errors of the final boundary conditions [62]. It is expected that this becomes more significant for more complex dynamical models and constraints. Poor flexibility and suboptimality of the solutions are the major drawbacks of Lyapunov-based control laws and shape-based methods.

Although NLP and DDP are similar as both solve the full nonlinear program directly, the work in [49] suggests that DDP becomes beneficial for many-revolution transfers. If cases with few revolutions only are addressed, it seems that DDP requires a higher computational effort than state-of-the-art NLP solvers. Due to the relatively large convergence domain and high flexibility to include complex constraints, direct methods seem to be a good compromise. Especially SCP methods offer a good tradeoff. They have shown to work well in real-time applications due to their rapid calculation speed, large convergence domain, and high robustness [81–83]. Even though solving a large-scale convex optimization problem is computationally expensive, using only first-order derivatives and exploiting sparsity can reduce the required solving time considerably, thus alleviating the fact that a large number of variables is needed [9, 11]. Therefore, SCP is considered the most appropriate method for onboard guidance given the requirements and characteristics of the deep-space cruise.

Table 3.1: Advantages and disadvantages of each optimization method.

Method	Advantages	Disadvantages
(Semi-)Analytical [24]	Analytical solution Computationally efficient No initial guess required	Exist only for special cases Suboptimal solutions Low-fidelity models
Indirect [34, 35]	Locally optimal solutions Small number of variables	High sensitivity to initial guess Poor flexibility
Direct (NLP) [34]	Medium sensitivity to initial guess High flexibility	Large number of variables Computationally expensive Solution only at discrete points
Direct (SCP) [9, 11]	Low sensitivity to initial guess High flexibility Medium computational efficiency	Large number of variables Solution only at discrete points
DDP [49]	Medium sensitivity to initial guess High flexibility Provides feedback control law	Large number of variables Computationally expensive Solution only at discrete points
Metaheuristics [50]	No initial guess required Global technique	Poor reliability High computational effort Potentially suboptimal solution
Machine Learning [62, 64]	No initial guess required Computationally efficient	Poor accuracy Suboptimal solutions Poor flexibility
Feedback-driven [73]	No initial guess required Computationally efficient	Suboptimal solutions Poor flexibility
Shape-based [74, 76]	Medium sensitivity to initial guess Medium computational effort	Low flexibility Low Accuracy

Table 3.2: Assessment of optimization methods.

Criterion / Method	Reliability	Robustness	Flexibility	Optimality	Onboard capability	Accuracy
(Semi-)Analytical	Very good	Poor	Poor	Poor	Very good	Poor
Indirect	Poor	Good	Satisfactory	Very good	Satisfactory	Very good
Direct (NLP)	Satisfactory	Good	Good	Good	Poor	Good
Direct (SCP)	Good	Good	Good	Good	Good	Good
DDP	Satisfactory	Good	Good	Good	Poor	Good
Metaheuristics	Poor	Poor	Poor	Satisfactory	Poor	Satisfactory
Machine Learning	Very good	Poor	Poor	Satisfactory	Very good	Poor
Feedback-driven	Very good	Satisfactory	Poor	Poor	Good	Good
Shape-based	Good	Satisfactory	Poor	Satisfactory	Satisfactory	Satisfactory

4 State of the Art

Several different topics related to low-thrust trajectory optimization are addressed in this dissertation. A brief summary of the state of the art is therefore presented to familiarize the reader with past work in this field of research.

4.1 Sequential Convex Programming

As solving nonlinear programs directly often requires high computational power and a decent initial guess, sequential convex programming techniques have become a popular alternative in the past years [22]. Advances on lossless convexification techniques for certain nonconvex constraints have lead to a variety of new works [84, 85]. As the original nonlinear optimal control problem is approximated as a convex problem, it can be solved iteratively using sophisticated interior point methods [86]. Due to the rapid calculation speed and the fact that convex programs are shown to converge to the global minimum under certain conditions [87], such techniques are a popular choice for real-time applications, especially within the path planning of robots and quadrotors [82, 83], and collision avoidance for unmanned aerial vehicles [88, 89]. Because of the high demand for more and more autonomy in aerospace vehicles, tremendous effort was made to exploit the advantages of convex optimization in aerospace applications. Therefore, highly nonlinear spacecraft proximity and rendezvous [90, 91], power descent landing guidance [92–95], and entry trajectory optimization problems [96, 97] have recently been solved using SCP. In this context, an improved Radau pseudospectral discretization scheme has been applied in [98] to increase the sparsity of the powered descent and landing problem, and thus lower the computational effort.

In contrast to the majority of researchers that use a modeling language for convex programming to facilitate the SCP implementation [99], the work in [100] aims to improve the computational performance by tailoring the algorithm to the actual flight code requirements. A customized interior-point solver for real-time powered descent guidance was developed in [101], and general methods are presented in [102] to formulate a convexified optimal control problem. Furthermore, a first-order method to solve conic optimization problems is developed in [103] to improve the convergence rate.

With regard to low-thrust trajectory design, [104, 105] applied sequential convex programming to solve time- and fuel-optimal transfers for the first time. Their simple numerical examples show that the computational time can be reduced considerably compared to standard NLP solvers while still obtaining near-optimal solutions with rather poor initial guesses. Later, SCP was used to generate the initial guess for an indirect method [106].

In contrast to autonomous navigation where some promising results have been obtained recently [79], literature on real-time guidance is scarce. Model predictive control is a popular choice for tracking a reference trajectory [107, 108], for the autonomous landing within planetary missions [109], or for autonomous racing [110]. In the past few years, the real-time performance of convex optimization-based algorithms was assessed for the powered descend guidance problem on spaceflight hardware [8, 111]. Furthermore, quadrotor maneuvering problems were solved on board using convex optimization [112, 113]. An implicit trajectory generation method with convergence guarantees is developed in [114] for the powered descent guidance problem where feasible trajectories are computed using numerical integration. Yet, it is still to be investigated whether sufficiently accurate solutions can be obtained in real time for interplanetary, long-duration orbital transfers.

4.1.1 Discretization and Trust-Region Methods

Solving a series of simpler, convex subproblems makes SCP numerically tractable compared to solving a nonlinear program directly. Two key characteristics of SCP are the discretization and trust-region method because they strongly affect the results, especially the convergence properties. Yet, previous research activities lack a thorough assessment and comparison of relevant techniques for low-thrust trajectory optimization.

The most popular discretization techniques are collocation and control interpolation methods. The former parameterize both the states and controls using some basis functions, whereas the latter parameterize only the control history [34]. Due to its simplicity, the trapezoidal rule is one of the most important collocation methods [104, 105, 115]. Yet, its poor accuracy often prevents the solver to find solutions that satisfy the nonlinear dynamics for long-duration interplanetary transfers. Higher-order methods such as Hermite–Simpson collocation offer instead a good compromise between computational effort and accuracy [116]. The arbitrary-order Hermite–Legendre–Gauss–Lobatto discretization is a generalization of this method and a popular choice in nonlinear optimization because the states are approximated by higher-order Hermite interpolating polynomials [117]. In global pseudospectral methods, in contrast, single Lagrange interpolating polynomials are used to approximate the states and controls, respectively. As this results in dense matrices, adaptive methods were developed where piecewise polynomials are used to approximate the trajectories [43]. There are several categories of pseudospectral methods. The most important ones are based on Legendre–Gauss (LG) [118], Legendre–Gauss–Lobatto (LGL) [117], or Legendre–Gauss–Radau (LGR) points [119]. In an adaptive framework with several segments, methods using LG points do not include the initial and end point, and therefore, no control is obtained at the segment breaks. If LGL points are used, there is a redundancy of points at the end of each segment. Moreover, the optimality conditions of the optimal control problem are not equivalent to the ones of the discretized form, which may result in inaccurate costates [120].

Several works adapted collocation methods to convex programming and solved powered descent [94, 98, 121] and low-thrust guidance problems [105, 106]. With regard to control interpolation methods, the control is approximated using a zero-order-hold (ZOH) or first-order-hold (FOH) discretization. For ZOH, the control history is assumed piecewise constant, whereas for FOH, it is approximated by a piecewise affine function [122]. Both methods performed well for powered descent guidance problems [92, 122]. The ZOH discretization was also applied to low-thrust trajectory optimization problems in the circular restricted three-body problem [123]. However, it is yet to be investigated how FOH performs within low-thrust trajectory optimization (LTO).

The work in [124] compares different discretization methods for the powered descent guidance (PDG) problem. However, the results cannot be extended directly to the low-thrust trajectory optimization problem due to the different dynamics, constraints, and number of switching times for fuel-optimal problems. One major difference is that the time horizon is considerably shorter in PDG compared to the long-duration interplanetary transfers in LTO which can last several years. Therefore, many more nodes are required to capture the state and control profiles accurately, especially if the number of revolutions increases. Moreover, a global pseudospectral method using only one high-order polynomial as in [124] would result in dense matrices and a considerable higher solving time (if a solution is found at all).

Trust regions are imposed to keep the linearization close to a reference solution. Most of the SCP methods found in literature use some type of trust-region approach. For powered descent guidance problems, soft trust regions are often used [87, 122]. In low-thrust trajectory optimization, simple hard trust regions are more common [9, 105]. The choice of the trust-region approach is often crucial as this can decide whether a feasible solution is found or not. Furthermore, a poor choice of the parameters can deteriorate the convergence. Such a behavior is undesirable and unacceptable for onboard applications. Soft trust regions often work well in PDG problems [124], but they have not been used to determine low-thrust trajectories so far.

4.1.2 High-Fidelity Optimization

Low-thrust trajectory optimization problems that consider higher-fidelity models (for example, third-body perturbations, solar radiation pressure, realistic thruster model) have been solved with indirect [125, 126] and direct, NLP-based methods [127–129]. With regard to convex programming techniques, the accuracy and convergence can be poor for highly nonlinear applications due to the successive linearization.

Therefore, most researchers considered only simple two-body dynamics. The literature on solving more complex models, for example the circular-restricted three-body problem using convex programming, is scarce. In [123], the SCP approach had to be combined with a nonlinear programming method to deal with the highly nonlinear dynamics and to achieve convergence. When additional perturbations are considered, convergence and accuracy deteriorate considerably, along with an increase in the number of iterations and computational time. No current SCP method can compute high-fidelity low-thrust trajectories reliably.

Indirect methods often make use of a homotopic approach to improve convergence: A series of easier problems is solved first where each solution is used as an initial guess for the next problem [130, 131]. The process continues until the original problem is solved. This can also be applied to direct methods where an easier problem (e.g., simple two-body dynamics) is considered first, and its solution is used to determine a solution in a higher-fidelity model (e.g., n-body dynamics). For example, it was shown that using a homotopy from the energy- to the fuel-optimal problem in a convex programming environment can improve convergence [132]. In previous works, each subproblem is solved to full optimality, and the step size is fixed and defined by the user [23, 39, 133]. This means that the algorithm requires at least a certain number of homotopic steps to achieve convergence. Depending on the problem structure, however, it may be beneficial to perform larger steps and therefore, achieve faster convergence. Still, current methods do not adjust the step size dynamically, but rely on a "solve to full optimality and then proceed to the next step" approach. Often, a conservative (i.e., small) value of the step size is chosen to avoid non-convergence. This procedure is not ideal, and it would be desirable to adjust the step size dynamically based on information that can be retrieved from the optimization problem. For example, a continuation method is embedded into SCP in [134] to model discrete logic constraints.

Only few works consider constraints where the thruster has to remain off during some periods. For example, the work in [135] requires a mesh refinement process to determine trajectories with shutdown constraints in the two-body environment. In [136], no-thrust periods are included in a model predictive control approach for small-body proximity operations using a zero-order-hold discretization method. The work in [137] uses a homotopic approach to successively obtain a solution with no-thrust constraints. Even though such duty cycles are of utmost importance for real space missions, most of the methods found in literature allow continuous thrust during the whole transfer. The resulting trajectories are therefore not mission-compliant.

Besides the discretization method, the number of discretization points also affects the performance of the optimization process [124]. A small number reduces the number of variables and thus, the dimension of the problem. However, this might yield poor accuracy or even result in non-convergence. For this reason, several works developed mesh refinement methods to adjust the number of nodes for nonlinear programs. In [44], a mesh refinement for the Legendre–Gauss–Radau pseudospectral method is developed where the number of collocation points and segments can be increased by comparing the optimized states with a solution that contains a larger number of nodes. The mesh in [138] is adjusted using the decay rates of Legendre polynomial coefficients, and interpolation errors are evaluated in [139] to adjust the mesh for the Mars atmospheric entry trajectory optimization problem. With regard to SCP, a method is presented in [140] where the mesh is updated after each iteration based on the linearization error. However, a major challenge are fuel-optimal problems because they require some mesh refinement to accurately capture the bang-bang control profile. A refined LGR pseudospectral method for nonsmooth problems is developed in [141]. Additional constraints and variables are included to account for the discontinuities. Other works determine the switching times in pseudospectral methods and then increase the number of nodes at these locations to have a more accurate representation of the controls [142, 143].

4.1.3 State Vector Representations

The equations of motion form a fundamental part in astrodynamics. Before solving an optimal control problem, the equations that govern the motion of the spacecraft (or any other object) are to be derived. Cartesian coordinates are probably the most common and popular representation of the dynamics. Yet, several different sets of coordinates have been developed in the past decades. The reason is that the choice of the coordinates can have a considerable impact on the performance of numerical methods that are used to solve problems related to astrodynamics [144]. Spherical coordinates (or polar coordinates in the planar case) seem to be a natural choice to describe the translational motion of a spacecraft around a primary body. Several works use these coordinates to solve the low-thrust trajectory optimization problem [104, 105, 116]. Cylindrical coordinates are often used in shape-based methods where the trajectory is approximated with certain functions [76]. Canonical elements such as Delaunay [145] and Poincaré [146] elements are advantageous for perturbation problems due to the simplifications that can be made when the perturbing acceleration is considered small. Equinoctial elements were developed to overcome the singularities of classical orbital elements [147]. They are a popular choice for many-revolution transfers in indirect methods because five out of six elements are constant for the unperturbed motion [39, 130, 148]. Alternative representations such as quaternion-like elements for the initial value problem [149], or coordinates that use the angular momentum and eccentricity vector for high-inclination orbital transfers were developed only recently [150]. Moreover, new coordinates based on Hill variables are introduced in [151], and compared with different element sets in the context of propagating perturbed Keplerian orbits.

The choice of the coordinates becomes even more crucial for SCP due to the successive linearization of the nonlinear dynamics. Even though many constraints can be relaxed and convexified, dynamics are usually approximated using a first-order Taylor series. Understanding the impact of the coordinate set on the performance of SCP is therefore of utmost practical interest. Yet, previous research activities lack a thorough assessment and comparison of relevant state vector choices for convex low-thrust trajectory optimization. As the work in [144] uses an indirect method to investigate the performance of several minimal coordinate sets for the low-thrust fuel-optimal trajectory optimization problem, their results cannot be applied directly to the SCP approach due to the fundamentally different characteristics of both methods. An indirect approach is compared with SCP in [152] where spherical and modified equinoctial elements are considered.

Besides an explicit comparison of the performance of different representations of the dynamics, a nonlinearity index was introduced in [153] to compare the nonlinearity of dynamical systems. This metric is intended to measure how nonlinear a dynamical system is for unperturbed initial value problems. This approach was later extended to include control terms [154]. More recently, an augmented nonlinearity index was introduced to account for state-costate dynamics in indirect methods [155]. This was applied to the spacecraft attitude control problem. Several other nonlinearity indices were proposed in the context of orbit uncertainty propagation, for example as a measure for automatic domain splitting [156].

4.2 Approximation of Nonlinear Dynamical Systems

Due to the highly nonlinear dynamics and often complex constraints, finding an optimal solution is still a challenging task as direct and indirect methods require a decent initial guess. For this reason, approximate solutions using simplified models (e.g., Kepler or Stark model [25]), predefined state or control profiles (e.g., shape-based methods [76] or constant radial and tangential thrust [29]) are often sought to generate the initial guess. However, these problems are still highly nonlinear and difficult to solve.

Although convex programming has become a promising technique in the past decade, it requires all constraints to be convex. Many constraints can be relaxed and convexified, but handling nonlinear dynamics properly is still a challenge [157]. As a consequence, dynamics are usually approximated using a first-order Taylor series [22]. This, however, is only a local approximation and can result in non-convergence if a poor initial guess is provided. How to deal with nonlinear and non-convex dynamics in engineering problems is therefore a challenging task.

Thus, linearization techniques have attracted a lot of attention in the last few years [158, 159]. Especially lifting nonlinear systems into a higher-dimensional space has become increasingly important as this allows us to represent a nonlinear system as a linear one, and sophisticated linear system theory can be applied. Yet, one drawback is the higher (often infinite) dimensional space of the transformed system. An important lifting technique is the linear, infinite-dimensional Koopman operator that describes the evolution of observable functions [160]. It has been applied to various problems in engineering, e.g., estimation [161], robotics [162], and fluid dynamics [163]. Despite the rising popularity, its application in astrodynamics is very limited because a high accuracy is required that data-driven approaches cannot achieve [164]. Recently, the zonal harmonics problem has been solved using the Koopman operator theory (KOT) [165]. In addition, it has been applied to attitude dynamics [166], the motion of satellites around libration points [167], and it was used to design control laws for the circular-restricted three-body problem [168]. However, it is to be investigated how problems with an external control can be approximated, and whether the accuracy is sufficient for problems in orbital mechanics.

5 Convexification and Sequential Convex Programming

Approaches based on convex optimization require all constraints to be convex. We refer to *convexification* as the process to transform the nonlinear constraints into convex expressions. There are two main classes of convexification techniques: lossless convexification and successive convexification [84, 87]. The former often uses relaxation techniques such that the nonconvex constraint is transformed into a convex one that yields the same optimal solution. Successive convexification, on the other hand, approximates the original constraint in each iteration, for example using a first-order Taylor series. In this chapter, the convexification of the nonlinear optimal control problem in space flight is addressed. Moreover, the sequential convex programming method is presented in detail. Parts of this chapter are taken from our work in [9, 12, 13, 169].

5.1 Convexification

Recalling the equations of motion in Eq. (2.25)–(2.27) and the minimum-fuel OCP in Eq. (2.33), it is evident that the differential equation of the velocity $\mathbf{v}$ and the constraint on the thrust directions $\boldsymbol{\alpha}$ are nonconvex. The nonlinear part

$$\sigma(t)\,\frac{T_{\max}}{m(t)}\,\boldsymbol{\alpha}(t) \tag{5.1}$$

of $\dot{\mathbf{v}}(t)$ is eliminated by a change of variables [92]:

$$\Gamma(t) := \frac{\sigma(t)\,T_{\max}}{m(t)} \tag{5.2}$$

$$\boldsymbol{\tau}(t) := \frac{\sigma(t)\,T_{\max}}{m(t)}\,\boldsymbol{\alpha}(t) \tag{5.3}$$

$$w(t) := \ln m(t) \tag{5.4}$$

Taking the derivative of Eq. (5.4) with respect to time yields the equation of motion for w:

$$\dot{w}(t) = \frac{1}{m}\,\dot{m} = -\frac{1}{m}\frac{\sigma(t)\,T_{\max}}{g_0\,I_{\mathrm{sp}}} = -\frac{\Gamma(t)}{g_0\,I_{\mathrm{sp}}} \tag{5.5}$$

Moreover, the constraint $\|\boldsymbol{\alpha}(t)\|_2 = 1$ changes to $\|\boldsymbol{\tau}(t)\|_2 = \Gamma(t)$. As this constraint is still nonconvex, it is relaxed to a second-order cone:

$$\|\boldsymbol{\tau}(t)\|_2 \leq \Gamma(t) \tag{5.6}$$

This convexification is lossless, and it can be shown that the solution of the relaxed problem is also an optimal solution of the original problem [92].

Approximation of Nonconvex Constraints

As only convex constraints are allowed within SCP, the remaining n_g nonconvex constraints

$$g_i(\mathbf{x}, \mathbf{u}) \leq 0, \qquad i = 1, \ldots, n_g \tag{5.7}$$

need to be approximated as affine or second-order cone constraints. In general, any approximation method can be used, for example first- and second-order Taylor series, or inner convex approximations [170]. Higher-fidelity approximations such as a second-order Taylor series may be preferable as they yield more accurate results. However, this requires the Hessian matrix $\nabla^2 g_i$ to be positive semidefinite. Moreover, the resulting second-order cone constraints increase the complexity and computational effort when solving the convex program. Therefore, we approximate the remaining nonconvex constraints using a first-order Taylor series due to the low computational effort, and the fact that the obtained expressions are guaranteed to be affine, i.e., convex. The first-order Taylor expansion of $g_i(\mathbf{x}, \mathbf{u})$ about the reference point $(\bar{\mathbf{x}}, \bar{\mathbf{u}})$ is given by

$$g_i(\bar{\mathbf{x}}, \bar{\mathbf{u}}) + \nabla g_i(\bar{\mathbf{x}}, \bar{\mathbf{u}}) \begin{bmatrix} \mathbf{x} - \bar{\mathbf{x}} \\ \mathbf{u} - \bar{\mathbf{u}} \end{bmatrix} \leq 0, \qquad i = 1, \ldots, n_g \tag{5.8}$$

Due to the change of variables, the constraint on the thrust magnitude in Eq. (2.33c) becomes

$$\Gamma(t) \leq T_{\max} \, \mathrm{e}^{-w(t)} \tag{5.9}$$

As the right-hand side of Eq. (5.9) is nonconvex, it is linearized about the reference $\bar{w}$:

$$0 \leq \Gamma(t) \leq T_{\max} \, \mathrm{e}^{-\bar{w}} \left[1 - w(t) + \bar{w}(t)\right] \tag{5.10}$$

The only remaining nonconvex constraint is the dynamics $\dot{\mathbf{x}}(t) = \mathbf{f}\left(\mathbf{x}(t), \mathbf{u}(t)\right)$ that read after the change of variables

$$\begin{aligned}\mathbf{f}\left(\mathbf{x}(t), \mathbf{u}(t)\right) &= \underbrace{\begin{bmatrix} \mathbf{v}(t) \\ -\mu\,\mathbf{r}(t)/[r(t)]^3 \\ 0 \end{bmatrix}}_{=:\,\mathbf{p}(\mathbf{x}(t))} + \underbrace{\begin{bmatrix} \multicolumn{2}{c}{\mathbf{0}_{3\times 4}} \\ \mathbf{1}_{3\times 3} & \mathbf{0}_{3\times 1} \\ \mathbf{0}_{1\times 3} & -1/(g_0\, I_{\text{sp}}) \end{bmatrix}}_{=:\,\mathbf{B}} \begin{bmatrix} \boldsymbol{\tau}(t) \\ \Gamma(t) \end{bmatrix} \\ &= \mathbf{p}(\mathbf{x}(t)) + \mathbf{B}\,\mathbf{u}(t)\end{aligned} \tag{5.11}$$

where the states and controls are defined as $\mathbf{x} \coloneqq \left[\mathbf{r}^\top, \mathbf{v}^\top, w\right]^\top$ and $\mathbf{u} \coloneqq \left[\boldsymbol{\tau}^\top, \Gamma\right]^\top$, respectively. Linearizing the dynamics according to Eq. (5.8) gives

$$\mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \approx \mathbf{f}(\bar{\mathbf{x}}(t), \bar{\mathbf{u}}(t)) + \nabla\mathbf{f}(\bar{\mathbf{x}}(t), \bar{\mathbf{u}}(t)) \begin{bmatrix} \mathbf{x}(t) - \bar{\mathbf{x}}(t) \\ \mathbf{u}(t) - \bar{\mathbf{u}}(t) \end{bmatrix} \tag{5.12}$$

As states and controls are decoupled if $I_{\text{sp}} = \text{const.}$ (i.e., the matrix $\mathbf{B}$ is constant and does not depend on $\mathbf{x}$), Eq. (5.12) simplifies to

$$\mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \approx \nabla\mathbf{p}(\bar{\mathbf{x}}(t))\,\mathbf{x}(t) + \mathbf{B}\,\mathbf{u}(t) + \mathbf{q}(\bar{\mathbf{x}}(t)) \tag{5.13}$$

with

$$\mathbf{q}(\bar{\mathbf{x}}(t)) = \mathbf{p}(\bar{\mathbf{x}}(t)) - \nabla\mathbf{p}(\bar{\mathbf{x}}(t))\,\bar{\mathbf{x}}(t) \tag{5.14}$$

As we will see in Section 8.3, it is often beneficial to directly work with the thrust magnitude instead of the acceleration as the control variable when applying a mesh refinement. In this case, it is possible to define different states $\hat{\mathbf{x}} \coloneqq \left[\mathbf{r}^\top, \mathbf{v}^\top, m\right]^\top$ and controls $\hat{\mathbf{u}} \coloneqq \left[\mathbf{T}^\top, T\right]^\top$, where $\mathbf{T} = \sigma\, T_{\max}\,\boldsymbol{\alpha}$ and $T = \|\mathbf{T}\|_2$. The corresponding dynamics read

$$\begin{aligned}\hat{\mathbf{f}}(\hat{\mathbf{x}}(t), \hat{\mathbf{u}}(t)) &= \underbrace{\begin{bmatrix} \mathbf{v}(t) \\ -\mu\,\mathbf{r}(t)/[r(t)]^3 \\ 0 \end{bmatrix}}_{=:\,\hat{\mathbf{p}}(\mathbf{x}(t))} + \underbrace{\begin{bmatrix} \multicolumn{2}{c}{\mathbf{0}_{3\times 4}} \\ \frac{1}{m}\,\mathbf{1}_{3\times 3} & \mathbf{0}_{3\times 1} \\ \mathbf{0}_{1\times 3} & -1/(g_0\, I_{\text{sp}}) \end{bmatrix}}_{=:\,\hat{\mathbf{B}}(\hat{\mathbf{x}}(t))} \begin{bmatrix} \mathbf{T}(t) \\ T(t) \end{bmatrix} \\ &= \hat{\mathbf{p}}(\hat{\mathbf{x}}(t)) + \hat{\mathbf{B}}(\hat{\mathbf{x}}(t))\,\hat{\mathbf{u}}(t)\end{aligned} \tag{5.15}$$

As $\mathbf{B}(\hat{\mathbf{x}})$ depends now on the state $\hat{\mathbf{x}}$, linearizing Eq. (5.15) fully about the reference $(\bar{\hat{\mathbf{x}}}, \bar{\hat{\mathbf{u}}})$ would result in an expression that depends on the reference control $\hat{\mathbf{u}}$. Previous work suggests that this deteriorates convergence due to the jittery behavior of the controls in subsequent iterations [171]. Therefore, we set

$$\hat{\mathbf{f}}(\hat{\mathbf{x}}(t), \hat{\mathbf{u}}(t)) \approx \hat{\mathbf{p}}(\hat{\mathbf{x}}(t)) + \hat{\mathbf{B}}(\bar{\hat{\mathbf{x}}}(t))\,\hat{\mathbf{u}}(t) \tag{5.16}$$

and linearize Eq. (5.15) only partially to obtain

$$\hat{\mathbf{f}}(\hat{\mathbf{x}}(t), \hat{\mathbf{u}}(t)) \approx \nabla \hat{\mathbf{p}}(\hat{\hat{\mathbf{x}}}(t))\, \hat{\mathbf{x}}(t) + \hat{\mathbf{B}}(\hat{\hat{\mathbf{x}}}(t))\, \hat{\mathbf{u}}(t) + \hat{\mathbf{q}}(\hat{\hat{\mathbf{x}}}(t)) \tag{5.17}$$

with

$$\hat{\mathbf{q}}(\hat{\hat{\mathbf{x}}}(t)) = \hat{\mathbf{p}}(\hat{\hat{\mathbf{x}}}(t)) - \nabla \hat{\mathbf{p}}(\hat{\hat{\mathbf{x}}}(t))\, \hat{\hat{\mathbf{x}}}(t) \tag{5.18}$$

This way, the current solution is independent of the previous control history $\hat{\mathbf{u}}$, and convergence is expected to enhance.

Artificial Infeasibility and Unboundedness

As the nonconvex constraints are approximated using a first-order Taylor series about some reference point, the solver may not be able to find a feasible solution anymore if the reference is poor. Therefore, it is important to restrict the search space to a region where the linearization is valid, i.e., sufficiently accurate. For this reason, a trust region of the form [87]

$$\|\mathbf{x} - \bar{\mathbf{x}}\|_1 \leq R \tag{5.19}$$

is imposed where the deviation of the solution vector $\mathbf{x}$ from the reference $\bar{\mathbf{x}}$ is limited by the (potentially varying) trust-region radius R. In addition, a linear approximation of nonconvex constraints may result in *artificial unboundedness* of a subproblem, i.e., an infinite decrease in the cost function when the solution trajectory is not enforced to lie in the neighborhood of the reference. This issue is resolved by adding a trust region [87].

When nonlinear constraints are linearized about a reference solution, we may encounter an infeasible convex subproblem even though the original problem is feasible. This phenomenon is called *artificial infeasibility*. An unconstrained virtual control $\boldsymbol{\nu} \in \mathbb{R}^{n_x}$ is added to the linearized dynamical constraints in Eq. (5.12) to prevent this [87]:

$$\dot{\mathbf{x}}(\mathbf{x}(t), \mathbf{u}(t)) = \mathbf{f}(\bar{\mathbf{x}}(t), \bar{\mathbf{u}}(t)) + \nabla \mathbf{f}(\bar{\mathbf{x}}(t), \bar{\mathbf{u}}(t)) \begin{bmatrix} \mathbf{x}(t) - \bar{\mathbf{x}}(t) \\ \mathbf{u}(t) - \bar{\mathbf{u}}(t) \end{bmatrix} + \boldsymbol{\nu}(t) \tag{5.20}$$

As $\boldsymbol{\nu}(t)$ is not constrained, the system can always reach a feasible point. The same problem can arise for the linearized control constraint in Eq. (5.10). Therefore, we relax this constraint by adding a slack variable $\eta(t) \geq 0$:

$$0 \leq \Gamma(t) \leq T_{\max}\, \mathrm{e}^{-\bar{w}} \left[1 - w(t) + \bar{w}(t)\right] + \eta(t) \tag{5.21}$$

Although these terms maintain feasibility, they also result in constraint violations when active. To ensure that $\boldsymbol{\nu}(t)$ and $\eta(t)$ are only used when infeasibility is detected, we incorporate them in our objective function with sufficiently large penalty parameters λ_h^{cvx} and λ_g^{cvx}:

$$\text{Minimize} \quad -w(t_f) + \lambda_h^{\text{cvx}} \sum_{i \in I_{\text{eq}}} \|\boldsymbol{\nu}_i\|_1 + \lambda_g^{\text{cvx}} \sum_{i \in I_{\text{ineq}}} \max(0, \eta_i) \tag{5.22}$$

where I_{eq} and I_{ineq} denote the set of equality and inequality constraints, respectively.

Remark 5.1. *If the state vector representation* $\hat{\mathbf{x}}$ *is used, the term* $-w(t_f)$ *changes to* $-m(t_f)$ *in Eq.* (5.22). *Moreover, the slack variable* η *is not needed if the controls* $\hat{\mathbf{u}}$ *are chosen because the thrust magnitude constraint* $0 \leq T(t) \leq T_{\max}$ *is inherently convex provided that* $T_{\max}$ *is constant. Therefore, the last term in Eq.* (5.22) *can be omitted.*

Convex Optimization Problem

We define the convexified optimization problem as Problem 1.

Problem 1. Find the functions $\Gamma(t)$ and $\boldsymbol{\tau}(t)$ that solve the following second-order cone program:

$$\underset{\Gamma(t),\,\boldsymbol{\tau}(t)}{\text{minimize}} \quad -w(t_f) + \lambda_h^{\text{cvx}} \sum_{i \in I_{\text{eq}}} \|\boldsymbol{\nu}_i\|_1 + \lambda_g^{\text{cvx}} \sum_{i \in I_{\text{ineq}}} \max(0, \eta_i) \tag{5.23a}$$

$$\text{subject to:} \quad \dot{\mathbf{x}}(t) = \mathbf{f}\left(\mathbf{x}(t), \Gamma(t), \boldsymbol{\tau}(t)\right) + \boldsymbol{\nu}(t) \tag{5.23b}$$

$$\Gamma(t) \leq T_{\max} \, \mathrm{e}^{-\bar{w}} \left[1 - w(t) + \bar{w}(t)\right] + \eta(t) \tag{5.23c}$$

$$\|\boldsymbol{\tau}(t)\|_2 \leq \Gamma(t) \tag{5.23d}$$

$$\|\mathbf{x}(t) - \bar{\mathbf{x}}(t)\|_1 \leq R \tag{5.23e}$$

$$\mathbf{r}(t_0) = \mathbf{r}_0, \ \mathbf{v}(t_0) = \mathbf{v}_0, \ w(t_0) = w_0 \tag{5.23f}$$

$$\mathbf{r}(t_f) = \mathbf{r}_f, \ \mathbf{v}(t_f) = \mathbf{v}_f \tag{5.23g}$$

$$\mathbf{x}_{\text{lb}} \leq \mathbf{x}(t) \leq \mathbf{x}_{\text{ub}}, \ \mathbf{u}_{\text{lb}} \leq \mathbf{u}(t) \leq \mathbf{u}_{\text{ub}} \tag{5.23h}$$

where Eq. (5.23h) refers to the lower (subscript lb) and upper (subscript ub) bounds, respectively. Our simulations suggest that restricting the search space by imposing such bounds is often beneficial for the solver. The optimization problem for the modified states $\hat{\mathbf{x}}$ and controls $\hat{\mathbf{u}}$ can be defined accordingly.

Remark 5.2. *If some final position* $\Delta\mathbf{r}$ *or velocity* $\Delta\mathbf{v}$ *error is acceptable, the final boundary constraints in Eq.* (5.23g) *can be relaxed to*

$$|\mathbf{r}(t_f) - \mathbf{r}_f| \leq \Delta\mathbf{r} \tag{5.24a}$$

$$|\mathbf{v}(t_f) - \mathbf{v}_f| \leq \Delta\mathbf{v} \tag{5.24b}$$

5.2 Sequential Convex Programming Method

Trust-region methods use some kind of merit function to measure the progress in each SCP iteration k. We define $\rho^{(k)}$ as the ratio

$$\rho^{(k)} := \frac{\text{actual cost decrease}}{\text{predicted cost decrease}} = \frac{\Delta\varphi}{\Delta\hat{\varphi}} \tag{5.25}$$

where the actual cost decrease $\Delta\varphi$ is calculated using the nonconvex constraints, and the predicted cost decrease $\Delta\hat{\varphi}$ is based on the convex constraint violations [87]. Depending on the value of $\rho^{(k)}$, the solution is accepted or rejected. Defining three parameters $0 < \rho_0 < \rho_1 < \rho_2 < 1$, a step at iteration k is rejected if $\rho^{(k)} < \rho_0$ because this indicates that there is no (sufficiently large) progress. When a solution is accepted, the trust-region radius R is updated as follows:

$$R^{(k+1)} = \begin{cases} R^{(k)}/\alpha & \text{if } \rho_0 \leq \rho^{(k)} < \rho_1 \\ R^{(k)} & \text{if } \rho_1 \leq \rho^{(k)} < \rho_2 \\ \beta\, R^{(k)} & \text{if } \rho^{(k)} \geq \rho_2 \end{cases} \tag{5.26}$$

where the trust-region shrinking rate $\alpha > 1$ and growing rate $\beta > 1$ are two constants.

The trust-region radius R is therefore actively changed based on the evaluation of a merit function φ. The idea is to compare the exact cost reduction $\Delta\varphi$ (using original nonlinear and nonconvex constraints) with the predicted one $\Delta\hat{\varphi}$ (using linearized and convex constraints). At each iteration, the exact φ and predicted cost $\hat{\varphi}$, respectively, are calculated:

$$\varphi := \lambda_h^{\text{noncvx}} \sum_{i\in I_{\text{eq}}} \|\mathbf{h}_i^{\text{noncvx}}\|_1 + \lambda_g^{\text{noncvx}} \sum_{i\in I_{\text{ineq}}} \max\left(0, g_i^{\text{noncvx}}\right) + \lambda_l^{\text{cvx}} \sum_{j\in J_{\text{ineq}}} \max\left(0, l_j^{\text{cvx}}\right) \tag{5.27}$$

$$\hat{\varphi} := \lambda_h^{\text{cvx}} \sum_{i\in I_{\text{eq}}} \|\mathbf{h}_i^{\text{cvx}}\|_1 + \lambda_g^{\text{cvx}} \sum_{i\in I_{\text{ineq}}} \max\left(0, g_i^{\text{cvx}}\right) + \lambda_l^{\text{cvx}} \sum_{j\in J_{\text{ineq}}} \max\left(0, l_j^{\text{cvx}}\right) \tag{5.28}$$

It is convenient to use the 1-norm as the corresponding ℓ_1 merit function is known to be exact [20]. $\mathbf{h}_i^{\text{noncvx}}$ and g_i^{noncvx} denote the concatenated equality and inequality constraints of the original nonlinear, nonconvex problem, and I_{eq} and I_{ineq} the corresponding sets of constraints. l_j^{cvx} refers to the convex inequality constraints of the problem, and $\lambda_h^{\text{noncvx}}$, $\lambda_g^{\text{noncvx}}$, and λ_l^{cvx} are positive penalty weights. The corresponding quantities for the convex constraints are denoted with the superscript cvx and defined accordingly. The weights depend on the problem, and can vary to increase the importance of individual terms.

The convex constraint violations c_k^{cvx} at each node for the constraints corresponding to dynamics, thrust, boundary conditions, and bounds can be calculated as follows:

Dynamics: $$c_{\text{dyn},k}^{\text{cvx}} = \|\boldsymbol{\nu}_k\|_1, \quad k = 1, \ldots, N-1 \tag{5.29a}$$

Thrust magnitude: $$c_{\text{thrust},k}^{\text{cvx}} = \max\left(0, \eta_k\right), \quad k = 1, \ldots, N \tag{5.29b}$$

Thrust components: $c^{\mathrm{cvx}}_{\mathrm{comp},k} = \max\left(0, \|\boldsymbol{\tau}_k\|_2 - \Gamma_k\right), \quad k = 1, \dots, N$ (5.29c)

Initial boundary cond.: $c^{\mathrm{cvx}}_{\mathrm{IBC}} = \left\| \left[\mathbf{r}^\top(t_0), \mathbf{v}^\top(t_0), w(t_0)\right]^\top - \left[\mathbf{r}_0^\top, \mathbf{v}_0^\top, w_0\right]^\top \right\|_1$ (5.29d)

Final boundary cond. position: $c^{\mathrm{cvx}}_{\mathrm{FBC,pos}} = \left\|\max\left(\mathbf{0}, |\mathbf{r}(t_f) - \mathbf{r}_f| - \Delta\mathbf{r}\right)\right\|_1$ (5.29e)

Final boundary cond. velocity: $c^{\mathrm{cvx}}_{\mathrm{FBC,vel}} = \left\|\max\left(\mathbf{0}, |\mathbf{v}(t_f) - \mathbf{v}_f| - \Delta\mathbf{v}\right)\right\|_1$ (5.29f)

Lower bounds states: $c^{\mathrm{cvx}}_{\mathrm{lb,x},k} = \left\|\max\left(\mathbf{0}, \mathbf{x}_{\mathrm{lb},k} - \mathbf{x}_k\right)\right\|_1, \quad k = 1, \dots, N$ (5.29g)

Upper bounds states: $c^{\mathrm{cvx}}_{\mathrm{ub,x},k} = \left\|\max\left(\mathbf{0}, \mathbf{x}_k - \mathbf{x}_{\mathrm{ub},k}\right)\right\|_1, \quad k = 1, \dots, N$ (5.29h)

Lower bounds controls: $c^{\mathrm{cvx}}_{\mathrm{lb,u},k} = \left\|\max\left(\mathbf{0}, \mathbf{u}_{\mathrm{lb},k} - \mathbf{u}_k\right)\right\|_1, \quad k = 1, \dots, N$ (5.29i)

Upper bounds controls: $c^{\mathrm{cvx}}_{\mathrm{ub,u},k} = \left\|\max\left(\mathbf{0}, \mathbf{u}_k - \mathbf{u}_{\mathrm{ub},k}\right)\right\|_1, \quad k = 1, \dots, N$ (5.29j)

where the $\max(\cdot)$ function in Eqs. (5.29e)–(5.29j) denotes the element-wise $\max$ operator. The predicted cost $\hat{\varphi}$ is therefore

$$\hat{\varphi} = \lambda_h^{\mathrm{cvx}} \sum_{k=1}^{N-1} c^{\mathrm{cvx}}_{\mathrm{dyn},k} + \lambda_g^{\mathrm{cvx}} \left(\sum_{k=1}^{N} c^{\mathrm{cvx}}_{\mathrm{thrust},k} + \sum_{k=1}^{N} c^{\mathrm{cvx}}_{\mathrm{comp},k} \right) + \lambda_l^{\mathrm{cvx}} \Bigg(c^{\mathrm{cvx}}_{\mathrm{IBC}} + c^{\mathrm{cvx}}_{\mathrm{FBC,pos}} + c^{\mathrm{cvx}}_{\mathrm{FBC,vel}} + \sum_{k=1}^{N} c^{\mathrm{cvx}}_{\mathrm{lb,x},k} + \sum_{k=1}^{N} c^{\mathrm{cvx}}_{\mathrm{ub,x},k} + \sum_{k=1}^{N} c^{\mathrm{cvx}}_{\mathrm{lb,u},k} + \sum_{k=1}^{N} c^{\mathrm{cvx}}_{\mathrm{ub,u},k} \Bigg) \tag{5.30}$$

With regard to the nonconvex constraint violations, we obtain:

Dynamics: $c^{\mathrm{noncvx}}_{\mathrm{dyn},k} = \left\|\dot{\mathbf{x}}_k - \mathbf{f}(\mathbf{x_k}, \mathbf{u_k})\right\|_1, \quad k = 1, \dots, N-1$ (5.31a)

Thrust magnitude: $c^{\mathrm{noncvx}}_{\mathrm{thrust},k} = \max\left(0, \Gamma_k - T_{\max}\,\mathrm{e}^{-w_k}\right), \quad k = 1, \dots, N$ (5.31b)

Thrust components: $c^{\mathrm{noncvx}}_{\mathrm{comp},k} = \left|\, \|\boldsymbol{\tau}_k\|_2 - \Gamma_k \,\right|, \quad k = 1, \dots, N$ (5.31c)

Initial boundary cond.: $c^{\mathrm{noncvx}}_{\mathrm{IBC}} = c^{\mathrm{cvx}}_{\mathrm{IBC}}$ (5.31d)

Final boundary cond. position: $c^{\mathrm{noncvx}}_{\mathrm{FBC,pos}} = c^{\mathrm{cvx}}_{\mathrm{FBC,pos}}$ (5.31e)

Final boundary cond. velocity: $c^{\mathrm{noncvx}}_{\mathrm{FBC,vel}} = c^{\mathrm{cvx}}_{\mathrm{FBC,vel}}$ (5.31f)

Lower bounds states: $c^{\mathrm{noncvx}}_{\mathrm{lb,x},k} = c^{\mathrm{cvx}}_{\mathrm{lb,x},k}$ (5.31g)

Upper bounds states: $c^{\mathrm{noncvx}}_{\mathrm{ub,x},k} = c^{\mathrm{cvx}}_{\mathrm{ub,x},k}$ (5.31h)

Lower bounds controls: $c^{\mathrm{noncvx}}_{\mathrm{lb,u},k} = c^{\mathrm{cvx}}_{\mathrm{lb,u},k}$ (5.31i)

Upper bounds controls: $c^{\mathrm{noncvx}}_{\mathrm{ub,u},k} = c^{\mathrm{cvx}}_{\mathrm{ub,u},k}$ (5.31j)

The expression on the right-hand side in Eq. (5.31a) refers to the general defects of the dynamical constraints that depends on the discretization method. The violations of the (convex) boundary constraints and bounds are identical to the linear case. The actual cost is then given by

$$\begin{aligned}\varphi = \lambda_h^{\text{noncvx}} \left(\sum_{k=1}^{N-1} c_{\text{dyn},k}^{\text{noncvx}} + \sum_{k=1}^{N} c_{\text{comp},k}^{\text{noncvx}} \right) + \lambda_g^{\text{noncvx}} \sum_{k=1}^{N} c_{\text{thrust},k}^{\text{noncvx}} + \lambda_l^{\text{cvx}} \Bigg(c_{\text{IBC}}^{\text{noncvx}} + c_{\text{FBC,pos}}^{\text{noncvx}} + c_{\text{FBC,vel}}^{\text{noncvx}} \\ + \sum_{k=1}^{N} c_{\text{lb,x},k}^{\text{noncvx}} + \sum_{k=1}^{N} c_{\text{ub,x},k}^{\text{noncvx}} + \sum_{k=1}^{N} c_{\text{lb,u},k}^{\text{noncvx}} + \sum_{k=1}^{N} c_{\text{ub,u},k}^{\text{noncvx}} \Bigg)\end{aligned} \tag{5.32}$$

The actual and predicted cost reductions are then

$$\Delta\varphi = \varphi^{(k-1)} - \hat{\varphi}^{(k)} \tag{5.33}$$

$$\Delta\hat{\varphi} = \varphi^{(k-1)} - \varphi^{(k)} \tag{5.34}$$

where $k - 1$ refers to the values from the previous iteration. It can be shown that $\Delta\hat{\varphi} \geq 0$ for all k [87].

The algorithm converges if the maximum constraint violation $c_{\max}$ and the relative change of the objective function are lower than thresholds ε_c and ε_J, respectively:

$$c_{\max} < \varepsilon_c \tag{5.35}$$

$$\frac{\left| J_0^{(k-1)} - J_0^{(k)} \right|}{\left| J_0^{(k-1)} \right|} < \varepsilon_J \tag{5.36}$$

where $J_0 := -w(t_f)$ for fuel-optimal problems. The maximum constraint violation $c_{\max}$ is calculated similar to Eqs. (5.29) and (5.31) using the ℓ_∞ norm. The algorithm also terminates without successful convergence to an optimal solution if there is no sufficient progress, that is, if the relative difference of the solution vector in two consecutive iterations $k - 1$ and k is smaller than ε_x:

$$\frac{\left\| \mathbf{x}^{(k-1)} - \mathbf{x}^{(k)} \right\|}{\left\| \mathbf{x}^{(k-1)} \right\|} < \varepsilon_x \tag{5.37}$$

The SCP algorithm is based on [87] and shown in Fig. 5.1. After solving the convex optimization problem, the actual and predicted cost reductions are computed according to Eqs. (5.27) and (5.28). The maximum constraint violation is calculated using the original nonconvex representations. The ratio ρ of the cost reductions is then used to determine whether the solution at iteration k is accepted or rejected, and the trust-region size R is updated accordingly. The algorithm proceeds with the next iterations until the stopping criteria are met.

Such an algorithm that uses variable trust regions, an exact penalty method, and virtual controls would eventually converge to a global minimum [87]. Yet, this solution might not be feasible with regard to the nonlinear dynamics. If the final virtual controls $\boldsymbol{\nu}$ and slack variables η are zero, however, the converged trajectory is a local optimum of the original, nonconvex problem. This means that the final solution also satisfies the nonlinear constraints at the nodes.

Hard And Soft Trust-Region Methods

There are two main classes of trust-region methods:

1) Hard trust regions: The trust-region constraint is explicitly imposed as a constraint [87].

2) Soft trust regions: The trust-region constraint is penalized in the objective function [172].

Both are presented in the following subsections.

Hard Trust Regions

The trust-region mechanism presented in the previous section implements a standard hard trust-region method where the shrinking rate $\alpha > 1$ and growing rate $\beta > 1$ are two constants. A more flexible approach is to not only update the trust-region size R, but also α and β. For example, we propose in [169] to adjust α and β based on the values of ρ of the current k and previous iteration $k-1$:

$$\alpha^{(k)} = \begin{cases} \alpha_0 & \text{if } \rho^{(k)} \geq \frac{\rho_1+\rho_2}{2} \wedge \rho^{(k-1)} \geq \rho_0 \\ \alpha_1 & \text{if } \rho^{(k)} \geq \rho_1 \wedge \rho^{(k-1)} \geq \rho_0 \\ \alpha_2 & \text{otherwise} \end{cases}, \; \beta^{(k)} = \begin{cases} \beta_0 & \text{if } \rho^{(k)} \geq \frac{\rho_1+\rho_2}{2} \wedge \rho^{(k-1)} \geq \rho_0 \\ \beta_1 & \text{if } \rho^{(k)} \geq \rho_1 \wedge \rho^{(k-1)} \geq \rho_0 \\ \beta_2 & \text{otherwise} \end{cases} \tag{5.38}$$

with parameters $1 < \alpha_0 < \alpha_1 < \alpha_2$ and $1 < \beta_0 < \beta_1 < \beta_2$. Our rationale is that if the previous step was accepted, the trust-region radius can be increased by a larger value (or decreased by a smaller one) for the next iteration. If instead $\rho^{(k)}$ and $\rho^{(k-1)}$ are small, it is likely that we need to stay closer to the reference solution.

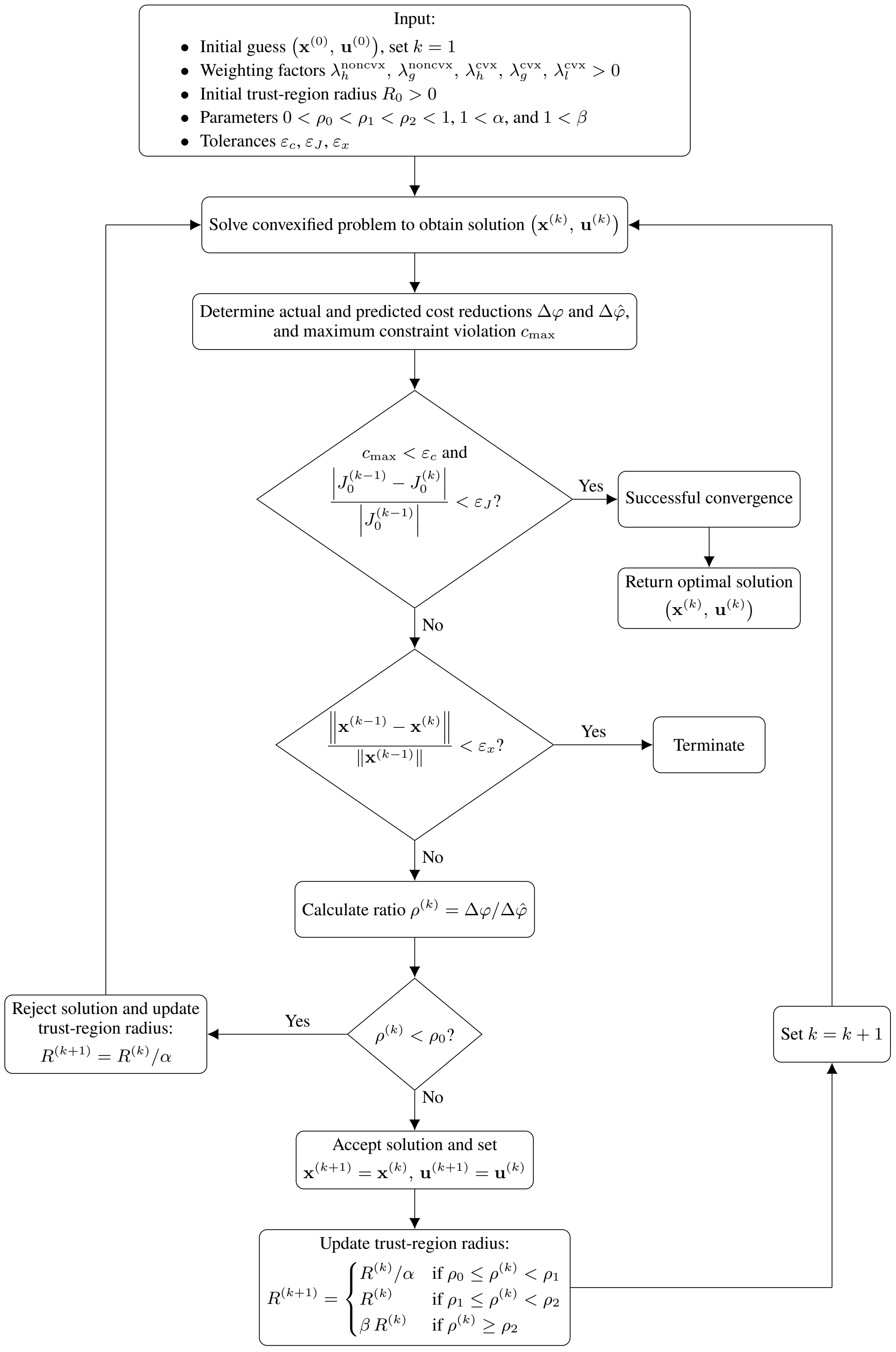
We propose a generalization of this approach in [132] where a constant $\delta > 1$ is introduced and the growing and shrinking rates are updated as follows:

1) If $\rho^{(k)} \geq \rho_0$ and $\rho^{(k-1)} \geq \rho_0$, then $\beta^{(k)} = \delta\,\beta^{(k-1)}$ and $\alpha^{(k)} = \alpha^{(k-1)}/\delta$. The growing rate is increased and the shrinking rate decreased if the previous and current iterations are accepted.

2) If $\rho^{(k)} \geq \rho_0$ and $\rho^{(k-1)} < \rho_0$, then $\beta^{(k)} = \beta^{(k-1)}/\delta$ and $\alpha^{(k)} = \delta\,\alpha^{(k-1)}$. The growing rate is decreased and the shrinking rate increased if only the current step is accepted.

3) If $\rho^{(k)} < \rho_0$ and $\rho^{(k-1)} \geq \rho_0$, then $\alpha^{(k)} = \alpha^{(k-1)}$ and $\beta^{(k)} = \beta^{(k-1)}$. The rates remain constant if the previous step was accepted and the current one is rejected.

4) If $\rho^{(k)} < \rho_0$ and $\rho^{(k-1)} < \rho_0$, then $\alpha^{(k)} = \delta\,\alpha^{(k-1)}$. The shrinking rate is increased if both steps are rejected.



Input:
• Initial guess $\left(\mathbf{x}^{(0)},\, \mathbf{u}^{(0)}\right)$, set $k = 1$
• Weighting factors $\lambda_h^{\text{noncvx}},\, \lambda_g^{\text{noncvx}},\, \lambda_h^{\text{cvx}},\, \lambda_g^{\text{cvx}},\, \lambda_l^{\text{cvx}} > 0$
• Initial trust-region radius $R_0 > 0$
• Parameters $0 < \rho_0 < \rho_1 < \rho_2 < 1$, $1 < \alpha$, and $1 < \beta$
• Tolerances ε_c, ε_J, ε_x
Solve convexified problem to obtain solution $\left(\mathbf{x}^{(k)},\, \mathbf{u}^{(k)}\right)$
Determine actual and predicted cost reductions $\Delta\varphi$ and $\Delta\hat{\varphi}$, and maximum constraint violation $c_{\max}$
$c_{\max} < \varepsilon_c$ and $\dfrac{\left|J_0^{(k-1)} - J_0^{(k)}\right|}{\left|J_0^{(k-1)}\right|} < \varepsilon_J$?
Yes
Successful convergence
Return optimal solution $\left(\mathbf{x}^{(k)},\, \mathbf{u}^{(k)}\right)$
No
$\dfrac{\left\|\mathbf{x}^{(k-1)} - \mathbf{x}^{(k)}\right\|}{\left\|\mathbf{x}^{(k-1)}\right\|} < \varepsilon_x$?
Yes
Terminate
No
Calculate ratio $\rho^{(k)} = \Delta\varphi / \Delta\hat{\varphi}$
$\rho^{(k)} < \rho_0$?
Yes
Reject solution and update trust-region radius: $R^{(k+1)} = R^{(k)}/\alpha$
No
Accept solution and set $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)},\, \mathbf{u}^{(k+1)} = \mathbf{u}^{(k)}$
Update trust-region radius:
$R^{(k+1)} = \begin{cases} R^{(k)}/\alpha & \text{if } \rho_0 \le \rho^{(k)} < \rho_1 \\ R^{(k)} & \text{if } \rho_1 \le \rho^{(k)} < \rho_2 \\ \beta\, R^{(k)} & \text{if } \rho^{(k)} \ge \rho_2 \end{cases}$
Set $k = k + 1$

Figure 5.1: Flowchart of the SCP algorithm.

In this case, bounds are imposed on α and β such that $\alpha_{\text{min}} \leq \alpha \leq \alpha_{\text{max}}$ and $\beta_{\text{min}} \leq \beta \leq \beta_{\text{max}}$.

Soft Trust Regions

Instead of enforcing the trust-region constraint in Eq. (5.19) directly, the performance index in Eq. (5.23a) is augmented with a penalty function $p(\mathbf{x})$ in soft trust-region methods:

$$\underset{\Gamma(t),\,\boldsymbol{\tau}(t)}{\text{minimize}} \quad -w(t_f) + \lambda_h^{\text{cvx}} \sum_{i \in I_{\text{eq}}} \|\boldsymbol{\nu}_i\|_1 + \lambda_g^{\text{cvx}} \sum_{i \in I_{\text{ineq}}} \max(0, \eta_i) + p(\mathbf{x}) \tag{5.39}$$

where $p(\mathbf{x})$ penalizes any violation of the trust-region constraint $g_{\text{TR}} \coloneqq \|\mathbf{x}(t) - \bar{\mathbf{x}}(t)\|_1 - R \leq 0$. In particular, we choose the differentiable and nondecreasing function

$$p(\mathbf{x}) \coloneqq \lambda_{\text{TR}} \left[\max\left(0, \|\mathbf{x}(t) - \bar{\mathbf{x}}(t)\|_1 - R\right)\right]^2 \tag{5.40}$$

with the penalty parameter $\lambda_{\text{TR}} > 0$. The update mechanism is defined as follows and based on [173]:

1) If $g_{\text{TR}} > 0$: Reject the step and set $\lambda_{\text{TR}}^{(k+1)} = \zeta\, \lambda_{\text{TR}}^{(k)}$ for $\zeta > 0$.

2) If $g_{\text{TR}} \leq 0$ and $\rho^{(k)} < \rho_0$: Reject the step and set $R^{(k+1)} = R^{(k)}/\alpha$.

3) If $g_{\text{TR}} \leq 0$ and $\rho_1 \leq \rho^{(k)} < \rho_2$: Accept the step and set $R^{(k+1)} = R^{(k)}/\alpha$ and $\lambda_{\text{TR}}^{(k+1)} = \lambda_{\text{TR},0}$.

4) If $g_{\text{TR}} \leq 0$ and $\rho^{(k)} > \rho_2$: Accept the step and set $R^{(k+1)} = \beta\, R^{(k)}$ and $\lambda_{\text{TR}}^{(k+1)} = \lambda_{\text{TR},0}$.

with some parameter $\lambda_{\text{TR},0} > 0$.

The performance of different trust-region methods is assessed in the next chapter.

6 Performance of Discretization and Trust-Region Methods

None of the existing works has investigated how different discretization and trust-region methods and their parameters affect the performance of the SCP algorithm for complex interplanetary low-thrust fuel-optimal transfers. Moreover, no conclusion can be drawn on how the choice might affect the real-time capability of the algorithm. For this reason, we compare the performance of the following discretization methods:

1) An adaptive Legendre–Gauss–Radau pseudospectral method.

2) An arbitrary-order Legendre–Gauss–Lobatto method based on Hermite interpolation.

3) A first-order-hold discretization.

The Radau pseudospectral method (RPM) has demonstrated to perform well for nonlinear programs [43]. Moreover, as either the initial or final point is not collocated, it is an appropriate choice for an adaptive framework as there is no redundancy or even lack of nodes between consecutive segments compared to other pseudospectral methods that are based on Legendre–Gauss or Legendre–Gauss–Lobatto points [120]. Therefore, convex formulations of the differential and integral forms of the adaptive Radau pseudospectral method are developed. In addition, an arbitrary-order Legendre–Gauss–Lobatto method based on Hermite interpolation is considered as it is a generalization of the well-known Hermite-Simpson collocation. It therefore covers a wide range of methods that have proven effective to solve nonlinear programs [117]. FOH is chosen as it belongs to the class of control interpolation techniques. Note that the zero-order-hold discretization is not considered due to the poor approximation of the thrust profile if accelerations are used as controls.

With regard to the trust regions, we compare the performance of two different hard trust-region methods, and a modified soft trust-region method presented in Section 5.2.

To the best of the author's knowledge, this is the first time that such an extensive assessment is carried out where the influence of discretization and trust-region methods, different orders of the interpolating polynomial, number of nodes, and initial guesses on the performance is analyzed at the same time.

Additionally, general requirements for onboard guidance applications are discussed. This chapter is based on our work in [12, 13].

6.1 Discretization Methods

The adaptive Radau pseudospectral method is presented in detail, and the first-order-hold discretization is described in this section. Moreover, a brief summary of the arbitrary-order Legendre–Gauss–Lobatto method is given.

6.1.1 Adaptive Radau Pseudospectral Method

Pseudospectral methods are a popular choice for solving NLPs because they show spectral convergence, do not suffer from Runge's phenomenon, and allow to retrieve the costates from the Lagrange multipliers [174]. As interplanetary trajectories often last several years, many discretization nodes might be needed to accurately approximate the states and controls. As a consequence, using a single interpolating polynomial of high degree would result in a dense problem that requires high computational effort. In this section, we apply an adaptive method where the trajectory is divided into several segments and the states and controls are approximated with polynomials of different degrees (and therefore, arbitrary number of nodes). In contrast to other optimization tools that use pseudospectral methods to solve NLPs (for example, the General Purpose Optimal Control Software (GPOPS-II) [175], Shefex-3 Pseudospectral Algorithm for Reentry Trajectory Analysis (SPARTAN) [176] and Direct and Indirect Dynamic Optimization (DIDO) [177]), we adapt the Radau pseudospectral method to convex programs.

In an adaptive pseudospectral method, the trajectory is divided into K segments, and the states and controls are approximated using Lagrange interpolating polynomials of arbitrary degrees. The collocation points are defined as the roots of the polynomial $P_{N-1}(\xi)+P_N(\xi)$ where P_N is the Nth degree Legendre polynomial. These points are defined in the pseudospectral time domain $\xi \in [-1, 1]$. Given N_k LGR points $(\xi_1, \xi_2, \dots, \xi_{N_k})$ where $\xi_1 = -1$ and $\xi_{N_k} < 1$, we define an additional (non-collocated) point that satisfies $\xi_{N_k+1} = 1$. The affine transformation between the physical t and pseudospectral time ξ is then given by [43]

$$t_i^{(k)} = \frac{t_{N_k+1}^{(k)} - t_1^{(k)}}{2}\xi_i^{(k)} + \frac{t_{N_k+1}^{(k)} + t_1^{(k)}}{2} \qquad i = 1, 2, ..., N_k + 1 \tag{6.1}$$

where $t_0 = t_1^{(1)}$ and $t_f = t_{N_K+1}^{(K)}$. Throughout this section, the number of collocation points per segment (and hence, the degree of the interpolating polynomial) is denoted as N_k, and $\mathbf{x}_i^{(k)}$, $\mathbf{u}_i^{(k)}$ refer to the ith point of the kth segment of states and controls at time $t_i^{(k)}$, with $i = 1, 2, \ldots, N_k + 1$ and $k = 1, \ldots, K$. The state is approximated in the interval $[-1, 1]$ using $N_k + 1$ points as follows:

$$\mathbf{x}^{(k)}(\xi) = \sum_{i=1}^{N_k+1} \mathbf{x}_i^{(k)}\, l_i^{(k)}(\xi), \qquad l_i^{(k)}(\xi) = \prod_{\substack{j=1\\ j\neq i}}^{N_k+1} \frac{\xi - \xi_j}{\xi_i - \xi_j}, \qquad k = 1, \ldots, K \tag{6.2}$$

where $l_i^{(k)}(\xi)$ denotes the Lagrange basis polynomial. Note that there are only N_k collocation points in a segment, but the state is approximated using $N_k + 1$ points in each segment. The last point of the last segment, $\mathbf{x}_{N_k+1}^{(K)}$, is thus added to the optimization problem so that final boundary constraints on the state can easily be included. In segments $k = 1, \ldots, K - 1$, the control is approximated in a similar way by

$$\mathbf{u}^{(k)}(\xi) = \sum_{i=1}^{N_k+1} \mathbf{u}_i^{(k)}\, l_i^{(k)}(\xi), \qquad l_i^{(k)}(\xi) = \prod_{\substack{j=1\\ j\neq i}}^{N_k+1} \frac{\xi - \xi_j}{\xi_i - \xi_j}, \qquad k = 1, \ldots, K-1 \tag{6.3}$$

In the last segment K, however, the last point at $\xi = 1$ is not included in the interpolation. Therefore, the continuous control in the last segment is

$$\mathbf{u}^{(K)}(\xi) = \sum_{i=1}^{N_K} \mathbf{u}_i^{(K)}\, l_i^{(K)}(\xi), \qquad l_i^{(K)}(\xi) = \prod_{\substack{j=1\\ j\neq i}}^{N_K} \frac{\xi - \xi_j}{\xi_i - \xi_j} \tag{6.4}$$

This means that $\mathbf{u}_{N_k+1}^{(K)}$ is not obtained in the solution process, and must be determined by extrapolation. The following linking conditions must therefore hold for segments $k < K$:

$$t_{N_k+1}^{(k)} = t_1^{(k+1)}, \qquad k = 1, \ldots, K-1 \tag{6.5}$$

$$\mathbf{x}_{N_k+1}^{(k)} = \mathbf{x}_1^{(k+1)}, \qquad k = 1, \ldots, K-1 \tag{6.6}$$

$$\mathbf{u}_{N_k+1}^{(k)} = \mathbf{u}_1^{(k+1)}, \qquad k = 1, \ldots, K-1 \tag{6.7}$$

The location of discretization and collocation points is illustrated in Fig. 6.1. Using

$$\frac{\mathrm{d}}{\mathrm{d}\xi} = \frac{t_{N_k+1} - t_1}{2} \frac{\mathrm{d}}{\mathrm{d}t} \tag{6.8}$$

an integral constraint is then discretized as

$$\int_{t_0}^{t_f} L(\mathbf{x}(t), \mathbf{u}(t))\, \mathrm{d}t \quad \longrightarrow \quad \sum_{k=1}^{K} \frac{t_{N_k+1}^{(k)} - t_1^{(k)}}{2} \sum_{i=1}^{N_k} w_i^{(k)} L(\mathbf{x}_i^{(k)}, \mathbf{u}_i^{(k)}) \tag{6.9}$$

where $w_i^{(k)}$ are the Radau quadrature weights given by [178]

$$w_1^{(k)} = \frac{2}{N_k^2} \tag{6.10a}$$

$$w_i^{(k)} = \frac{1}{N_k^2} \frac{1 - \xi_i}{[P_{N_k-1}(\xi_i)]^2}, \qquad i = 2, \ldots, N_k \tag{6.10b}$$

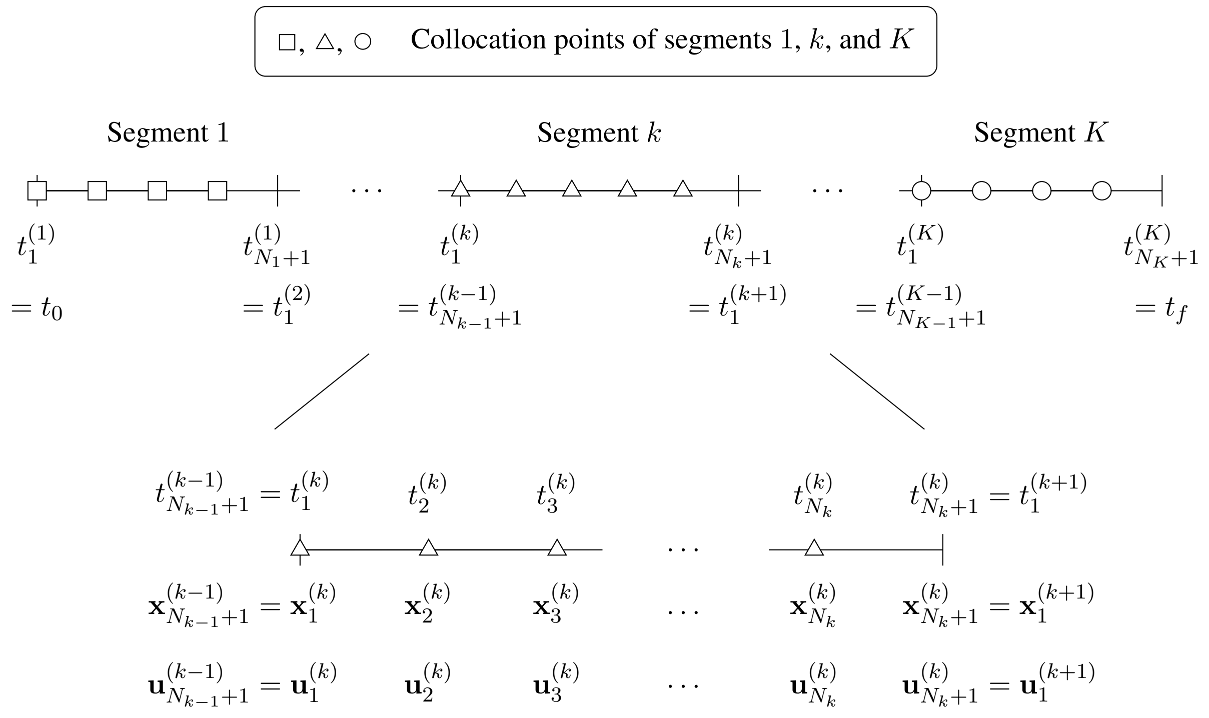

□, △, ○ Collocation points of segments 1, k, and K
Segment 1
Segment k
Segment K
$t_1^{(1)} = t_0$
$t_{N_1+1}^{(1)} = t_1^{(2)}$
$t_1^{(k)} = t_{N_{k-1}+1}^{(k-1)}$
$t_{N_k+1}^{(k)} = t_1^{(k+1)}$
$t_1^{(K)} = t_{N_{K-1}+1}^{(K-1)}$
$t_{N_K+1}^{(K)} = t_f$
$t_{N_{k-1}+1}^{(k-1)} = t_1^{(k)}$
$t_2^{(k)}$
$t_3^{(k)}$
$t_{N_k}^{(k)}$
$t_{N_k+1}^{(k)} = t_1^{(k+1)}$
$\mathbf{x}_{N_{k-1}+1}^{(k-1)} = \mathbf{x}_1^{(k)}$
$\mathbf{x}_2^{(k)}$
$\mathbf{x}_3^{(k)}$
$\mathbf{x}_{N_k}^{(k)}$
$\mathbf{x}_{N_k+1}^{(k)} = \mathbf{x}_1^{(k+1)}$
$\mathbf{u}_{N_{k-1}+1}^{(k-1)} = \mathbf{u}_1^{(k)}$
$\mathbf{u}_2^{(k)}$
$\mathbf{u}_3^{(k)}$
$\mathbf{u}_{N_k}^{(k)}$
$\mathbf{u}_{N_k+1}^{(k)} = \mathbf{u}_1^{(k+1)}$

Figure 6.1: Discretization and collocation points for RPM.

Differential Formulation

Taking the derivative of Eq. (6.2) with respect to ξ gives

$$\frac{\mathrm{d}\mathbf{x}^{(k)}(\xi)}{\mathrm{d}\xi} = \sum_{i=1}^{N_k+1} \mathbf{x}_i^{(k)} \frac{\mathrm{d}l_i^{(k)}(\xi)}{\mathrm{d}\xi} \tag{6.11}$$

The basic idea of the differential formulation of pseudospectral methods is to approximate the differential operator as

$$\dot{\mathbf{x}}^{(k)} \approx \mathbf{D}^{(k)}\mathbf{x}^{(k)} \tag{6.12}$$

where $\mathbf{D}^{(k)} \in \mathbb{R}^{N_k \times (N_k+1)}$ is a non-square differentiation matrix whose elements are defined as [179]

$$D_{ji} := l_i'(\xi_j), \qquad j = 1, \ldots, N_k, \quad i = 1, \ldots, N_k + 1 \tag{6.13}$$

with

$$(\cdot)' := \frac{\mathrm{d}}{\mathrm{d}\xi} \tag{6.14}$$

The dynamical constraints

$$\sum_{i=1}^{N_k+1} \mathbf{x}_i^{(k)} \frac{\mathrm{d}l_i^{(k)}(\xi_j)}{\mathrm{d}\xi} = \frac{t_{N_k+1}^{(k)} - t_1^{(k)}}{2} \mathbf{f}\left(\mathbf{x}_j^{(k)}, \mathbf{u}_j^{(k)}\right), \qquad j = 1, \ldots, N_k \tag{6.15}$$

can then be written as

$$\begin{aligned}\sum_{i=1}^{N_k+1} D_{ji}^{(k)}\mathbf{x}_i^{(k)} &= \frac{t_{N_k+1}^{(k)} - t_1^{(k)}}{2}\mathbf{f}\left(\mathbf{x}_j^{(k)}, \mathbf{u}_j^{(k)}\right) \\ &= \frac{t_{N_k+1}^{(k)} - t_1^{(k)}}{2}\left[\mathbf{A}\left(\bar{\mathbf{x}}_{\mathbf{j}}^{(\mathbf{k})}, \bar{\mathbf{u}}_{\mathbf{j}}^{(\mathbf{k})}\right)\mathbf{x}_j^{(k)} + \mathbf{B}\left(\bar{\mathbf{x}}_j^{(k)}, \bar{\mathbf{u}}_{\mathbf{j}}^{(\mathbf{k})}\right)\mathbf{u}_j^{(k)} + \mathbf{q}\left(\bar{\mathbf{x}}_j^{(k)}, \bar{\mathbf{u}}_j^{(k)}\right) + \boldsymbol{\nu}_j^{(k)}\right]\end{aligned} \tag{6.16}$$

for $j = 1, \ldots, N_k$ and $k = 1, \ldots, K$. The dynamical function $\mathbf{f}(\mathbf{x}, \mathbf{u})$ has been linearized according to Section 5.1, and the Jacobian matrices are defined as follows:

$$\mathbf{A}\left(\bar{\mathbf{x}}_{\mathbf{j}}^{(\mathbf{k})}, \bar{\mathbf{u}}_{\mathbf{j}}^{(\mathbf{k})}\right) = \nabla_x \mathbf{f}\left(\bar{\mathbf{x}}_{\mathbf{j}}^{(\mathbf{k})}, \bar{\mathbf{u}}_{\mathbf{j}}^{(\mathbf{k})}\right) \tag{6.17}$$

$$\mathbf{B}\left(\bar{\mathbf{x}}_{\mathbf{j}}^{(\mathbf{k})}, \bar{\mathbf{u}}_{\mathbf{j}}^{(\mathbf{k})}\right) = \nabla_u \mathbf{f}\left(\bar{\mathbf{x}}_{\mathbf{j}}^{(\mathbf{k})}, \bar{\mathbf{u}}_{\mathbf{j}}^{(\mathbf{k})}\right) \tag{6.18}$$

$$\mathbf{q}\left(\bar{\mathbf{x}}_j^{(k)}, \bar{\mathbf{u}}_j^{(k)}\right) = \mathbf{f}\left(\bar{\mathbf{x}}_{\mathbf{j}}^{(\mathbf{k})}, \bar{\mathbf{u}}_{\mathbf{j}}^{(\mathbf{k})}\right) - \nabla_x \mathbf{f}\left(\bar{\mathbf{x}}_{\mathbf{j}}^{(\mathbf{k})}, \bar{\mathbf{u}}_{\mathbf{j}}^{(\mathbf{k})}\right)\bar{\mathbf{x}}_{\mathbf{j}}^{(\mathbf{k})} - \nabla_u \mathbf{f}\left(\bar{\mathbf{x}}_{\mathbf{j}}^{(\mathbf{k})}, \bar{\mathbf{u}}_{\mathbf{j}}^{(\mathbf{k})}\right)\bar{\mathbf{x}}_{\mathbf{j}}^{(\mathbf{k})} \tag{6.19}$$

Defining the concatenated states, controls, and virtual controls as

$$\begin{aligned}\mathbf{X} &= \left[\left(\mathbf{x}^{(1)}\right)^\top, \ldots, \left(\mathbf{x}^{(K)}\right)^\top\right]^\top = \left[\left(\mathbf{x}_1^{(1)}\right)^\top, \left(\mathbf{x}_2^{(1)}\right)^\top, \ldots, \left(\mathbf{x}_{N_1}^{(1)}\right)^\top, \left(\mathbf{x}_1^{(2)}\right)^\top \ldots, \left(\mathbf{x}_{N_K+1}^{(K)}\right)^\top\right]^\top \\ \mathbf{U} &= \left[\left(\mathbf{u}^{(1)}\right)^\top, \ldots, \left(\mathbf{u}^{(K)}\right)^\top\right]^\top = \left[\left(\mathbf{u}_1^{(1)}\right)^\top, \left(\mathbf{u}_2^{(1)}\right)^\top, \ldots, \left(\mathbf{u}_{N_1}^{(1)}\right)^\top, \left(\mathbf{u}_1^{(2)}\right)^\top \ldots, \left(\mathbf{u}_{N_K}^{(K)}\right)^\top\right]^\top \\ \boldsymbol{\nu} &= \left[\left(\boldsymbol{\nu}^{(1)}\right)^\top, \ldots, \left(\boldsymbol{\nu}^{(K)}\right)^\top\right]^\top = \left[\left(\boldsymbol{\nu}_1^{(1)}\right)^\top, \left(\boldsymbol{\nu}_2^{(1)}\right)^\top, \ldots, \left(\boldsymbol{\nu}_{N_1}^{(1)}\right)^\top, \left(\boldsymbol{\nu}_1^{(2)}\right)^\top \ldots, \left(\boldsymbol{\nu}_{N_K}^{(K)}\right)^\top\right]^\top\end{aligned} \tag{6.20}$$

where $(\cdot)^{(k)}$ denotes the column vector of concatenated states, controls, or virtual controls of segment k. The dynamics can then be written in standard form as a linear equality constraint:

$$\left[\begin{array}{ccc:ccc:ccc} \hat{\mathbf{A}}^{(1)} & & & \hat{\mathbf{B}}^{(1)} & & & \hat{\mathbf{1}}^{(1)} & & \\ & & \mathbf{0} & & & \mathbf{0} & & & \mathbf{0} \\ & \ddots & & & \ddots & & & \ddots & \\ \mathbf{0} & & & \mathbf{0} & & & \mathbf{0} & & \\ & & \hat{\mathbf{A}}^{(K)} & & & \hat{\mathbf{B}}^{(K)} & & & \hat{\mathbf{1}}^{(K)} \end{array}\right]\begin{bmatrix}\mathbf{X} \\ \mathbf{U} \\ \boldsymbol{\nu}\end{bmatrix} = \begin{bmatrix}\hat{\mathbf{q}}^{(1)} \\ \vdots \\ \hat{\mathbf{q}}^{(K)}\end{bmatrix} \tag{6.21}$$

where

$$\hat{\mathbf{A}}^{(k)} = \left[\begin{array}{cccccc:c} D_{11}^{(k)}\,\mathbf{1}_{n_x} - \Delta^{(k)}\,\mathbf{A}_1^{(k)} & D_{12}^{(k)}\,\mathbf{1}_{n_x} & \dots & \dots & D_{1,N_k}^{(k)}\,\mathbf{1}_{n_x} & & D_{1,N_k+1}^{(k)}\,\mathbf{1}_{n_x} \\ D_{21}^{(k)}\,\mathbf{1}_{n_x} & D_{22}^{(k)}\,\mathbf{1}_{n_x} - \Delta^{(k)}\,\mathbf{A}_2^{(k)} & \dots & \dots & D_{2,N_k}^{(k)}\,\mathbf{1}_{n_x} & & D_{2,N_k+1}^{(k)}\,\mathbf{1}_{n_x} \\ \vdots & \vdots & \ddots & \ddots & \vdots & & \vdots \\ D_{N_k,1}^{(k)}\,\mathbf{1}_{n_x} & D_{N_k,2}^{(k)}\,\mathbf{1}_{n_x} & \dots & \dots & D_{N_k,N_k}^{(k)}\,\mathbf{1}_{n_x} - \Delta^{(k)}\,\mathbf{A}_{N_k}^{(k)} & & D_{N_k,N_k+1}^{(k)}\,\mathbf{1}_{n_x} \end{array}\right] \tag{6.22}$$

$$\hat{\mathbf{B}}^{(k)} = \begin{bmatrix} -\Delta^{(k)}\,\mathbf{B}_1^{(k)} & & & \mathbf{0} \\ & -\Delta^{(k)}\,\mathbf{B}_2^{(k)} & & \\ & & \ddots & \\ \mathbf{0} & & & -\Delta^{(k)}\,\mathbf{B}_{N_k}^{(k)} \end{bmatrix} \tag{6.23}$$

$$\hat{\mathbf{q}}^{(k)} = \begin{bmatrix} \Delta^{(k)}\,\mathbf{q}_1^{(k)} \\ \Delta^{(k)}\,\mathbf{q}_2^{(k)} \\ \vdots \\ \Delta^{(k)}\,\mathbf{q}_{N_k}^{(k)} \end{bmatrix} \tag{6.24}$$

Moreover, $\mathbf{1}_{n_x} \in \mathbb{R}^{n_x \times n_x}$ is the $n_x \times n_x$ identity matrix, n_x being the number of states. Similarly, $\hat{\mathbf{1}}^{(k)} = -\Delta^{(k)}\,\mathbf{1}_{n_x\,N_k}$, $\mathbf{1}_{n_x\,N_k} \in \mathbb{R}^{n_x\,N_k \times n_x\,N_k}$. The dashed vertical line in Eq. (6.22) indicates the segment break, and the entries to the right refer to the last, non-collocated point that is also the first point of the next segment. The notation

$$\Delta^{(k)} := \frac{t_{N_k+1}^{(k)} - t_1^{(k)}}{2} \tag{6.25}$$

$$\mathbf{A}_i^{(k)} := \mathbf{A}\left(\bar{\mathbf{x}}_{\mathbf{i}}^{(\mathbf{k})}, \bar{\mathbf{u}}_{\mathbf{i}}^{(\mathbf{k})}\right) \tag{6.26}$$

$$\mathbf{B}_i^{(k)} := \mathbf{B}\left(\bar{\mathbf{x}}_{\mathbf{i}}^{(\mathbf{k})}, \bar{\mathbf{u}}_{\mathbf{i}}^{(\mathbf{k})}\right) \tag{6.27}$$

$$\mathbf{q}_i^{(k)} := \mathbf{q}\left(\bar{\mathbf{x}}_{\mathbf{i}}^{(\mathbf{k})}, \bar{\mathbf{u}}_{\mathbf{i}}^{(\mathbf{k})}\right) \tag{6.28}$$

was introduced for the sake of brevity, and $k = 1, \dots, K$, $i = 1, \dots, N_k$.

Remark 6.1. *If the specific impulse and maximum thrust are constant, and if states and controls can be decoupled and written as* $\mathbf{f}(\mathbf{x}, \mathbf{u}) = \mathbf{p}(\mathbf{x}) + \mathbf{B}\,\mathbf{u}$*, the previous control* $\bar{\mathbf{u}}$ *in Eqs.* (6.17)–(6.19) *cancels out. Thus, the expressions reduce to*

$$\mathbf{A}\left(\bar{\mathbf{x}}_{\mathbf{j}}^{(\mathbf{k})}, \bar{\mathbf{u}}_{\mathbf{j}}^{(\mathbf{k})}\right) \longrightarrow \mathbf{A}\left(\bar{\mathbf{x}}_{\mathbf{j}}^{(\mathbf{k})}\right) = \nabla_x \mathbf{p}\left(\bar{\mathbf{x}}_{\mathbf{j}}^{(\mathbf{k})}\right) \tag{6.29}$$

$$\mathbf{B}\left(\bar{\mathbf{x}}_{\mathbf{j}}^{(\mathbf{k})}, \bar{\mathbf{u}}_{\mathbf{j}}^{(\mathbf{k})}\right) \longrightarrow \mathbf{B} = \nabla_u \mathbf{f}\left(\bar{\mathbf{x}}_{\mathbf{j}}^{(\mathbf{k})}, \bar{\mathbf{u}}_{\mathbf{j}}^{(\mathbf{k})}\right) \tag{6.30}$$

$$\mathbf{q}\left(\bar{\mathbf{x}}_{j}^{(k)}, \bar{\mathbf{u}}_{j}^{(k)}\right) \longrightarrow \mathbf{q}\left(\bar{\mathbf{x}}_{j}^{(k)}\right) = \mathbf{p}\left(\bar{\mathbf{x}}_{\mathbf{j}}^{(\mathbf{k})}\right) - \nabla_x \mathbf{p}\left(\bar{\mathbf{x}}_{\mathbf{j}}^{(\mathbf{k})}\right) \bar{\mathbf{x}}_{\mathbf{j}}^{(\mathbf{k})} \tag{6.31}$$

according to Eqs. (5.13) *and* (5.14).

In each iteration, the nonconvex constraint violations in Eq. (5.31a) are calculated using Eq. (6.16) and the nonlinear dynamics $\mathbf{f}_{\text{nonlin}}$:

$$c_{\text{dyn},j}^{\text{noncvx},(k)} = \sum_{i=1}^{N_k+1} D_{ji}^{(k)} \mathbf{x}_i^{(k)} - \frac{t_{N_k+1}^{(k)} - t_1^{(k)}}{2} \mathbf{f}_{\text{nonlin}}\left(\mathbf{x}_j^{(k)}, \mathbf{u}_j^{(k)}\right), \quad j = 1, \ldots, N_k,\ k = 1, \ldots, K \tag{6.32}$$

where $\mathbf{x}$ and $\mathbf{u}$ are the states and controls obtained from the optimization.

Integral Formulation

It was observed that an equivalent integral formulation of Eq. (6.16) may yield more consistent results for solving nonlinear programs [180]. We extend this approach to our convex optimization framework. Defining the Radau integration matrix $\mathbf{I}^{(k)} \in \mathbb{R}^{N_k \times N_k}$ as

$$\mathbf{I}^{(k)} := \left[\mathbf{D}_{2:N_k+1}^{(k)}\right]^{-1} \tag{6.33}$$

where $\mathbf{D}_{2:N_k+1}^{(k)}$ is obtained by removing the first column of $\mathbf{D}^{(k)}$ [181], the dynamics are

$$\begin{aligned}\mathbf{x}_{i+1}^{(k)} &= \mathbf{x}_1^{(k)} + \frac{t_{N_k+1}^{(k)} - t_1^{(k)}}{2} \sum_{j=1}^{N_k} I_{ij}^{(k)}\, \mathbf{f}\left(\mathbf{x}_j^{(k)}, \mathbf{u}_j^{(k)}\right) \\ &= \mathbf{x}_1^{(k)} + \frac{t_{N_k+1}^{(k)} - t_1^{(k)}}{2} \sum_{j=1}^{N_k} I_{ij}^{(k)} \left[\mathbf{A}\left(\bar{\mathbf{x}}_{\mathbf{j}}^{(\mathbf{k})}, \bar{\mathbf{u}}_{\mathbf{j}}^{(\mathbf{k})}\right) \mathbf{x}_j^{(k)} + \mathbf{B}\left(\bar{\mathbf{x}}_j^{(k)}, \bar{\mathbf{u}}_{\mathbf{j}}^{(\mathbf{k})}\right) \mathbf{u}_j^{(k)} + \mathbf{q}\left(\bar{\mathbf{x}}_j^{(k)}, \bar{\mathbf{u}}_j^{(k)}\right) + \boldsymbol{\nu}_j^{(k)}\right]\end{aligned} \tag{6.34}$$

for $k = 1, \dots, K$, $i = 1, \dots, N_k$. The matrices $\hat{\mathbf{A}}_{\text{int}}^{(k)}$, $\hat{\mathbf{B}}_{\text{int}}^{(k)}$, $\hat{\mathbf{I}}_{\text{int}}^{(k)}$, and the vector $\hat{\mathbf{q}}_{\text{int}}^{(k)}$ for the integral form are now given by

$$
\hat{\mathbf{A}}_{\text{int}}^{(k)} = \left[\begin{array}{cccccc|c}
-\mathbf{1}_{n_x} - \Delta^{(k)} I_{11}^{(k)} \mathbf{A}_1^{(k)} & \mathbf{1}_{n_x} - \Delta^{(k)} I_{12}^{(k)} \mathbf{A}_2^{(k)} & -\Delta^{(k)} I_{13}^{(k)} \mathbf{A}_3^{(k)} & \dots & \dots & -\Delta^{(k)} I_{1,N_k}^{(k)} \mathbf{A}_{N_k}^{(k)} & \mathbf{0}_{n_x} \\
-\mathbf{1}_{n_x} - \Delta^{(k)} I_{21}^{(k)} \mathbf{A}_1^{(k)} & -\Delta^{(k)} I_{22}^{(k)} \mathbf{A}_2^{(k)} & \mathbf{1}_{n_x} - \Delta^{(k)} I_{23}^{(k)} \mathbf{A}_3^{(k)} & \dots & \dots & -\Delta^{(k)} I_{2,N_k}^{(k)} \mathbf{A}_{N_k}^{(k)} & \mathbf{0}_{n_x} \\
\vdots & \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\
-\mathbf{1}_{n_x} - \Delta^{(k)} I_{N_k-1,1}^{(k)} \mathbf{A}_1^{(k)} & -\Delta^{(k)} I_{N_k-1,2}^{(k)} \mathbf{A}_2^{(k)} & -\Delta^{(k)} I_{N_k-1,3}^{(k)} \mathbf{A}_3^{(k)} & \dots & \dots & \mathbf{1}_{n_x} - \Delta^{(k)} I_{N_k-1,N_k}^{(k)} \mathbf{A}_{N_k}^{(k)} & \mathbf{0}_{n_x} \\
-\mathbf{1}_{n_x} - \Delta^{(k)} I_{N_k,1}^{(k)} \mathbf{A}_1^{(k)} & -\Delta^{(k)} I_{N_k,2}^{(k)} \mathbf{A}_2^{(k)} & -\Delta^{(k)} I_{N_k,3}^{(k)} \mathbf{A}_3^{(k)} & \dots & \dots & -\Delta^{(k)} I_{N_k,N_k}^{(k)} \mathbf{A}_{N_k}^{(k)} & \mathbf{1}_{n_x}
\end{array}\right] \tag{6.35}
$$

$$
\hat{\mathbf{B}}_{\text{int}}^{(k)} = \begin{bmatrix}
-\Delta^{(k)} I_{11}^{(k)} \mathbf{B}_1^{(k)} & -\Delta\, I_{12}^{(k)} \mathbf{B}_2^{(k)} & \dots & -\Delta^{(k)} I_{1,N_k}^{(k)} \mathbf{B}_{N_k}^{(k)} \\
-\Delta^{(k)} I_{21}^{(k)} \mathbf{B}_1^{(k)} & -\Delta^{(k)} I_{22}^{(k)} \mathbf{B}_2^{(k)} & \dots & -\Delta^{(k)} I_{2,N_k}^{(k)} \mathbf{B}_{N_k}^{(k)} \\
\vdots & \vdots & \ddots & \vdots \\
-\Delta^{(k)} I_{N_k-1,1}^{(k)} \mathbf{B}_1^{(k)} & -\Delta^{(k)} I_{N_k-1,2}^{(k)} \mathbf{B}_2^{(k)} & \dots & -\Delta^{(k)} I_{N_k-1,N_k}^{(k)} \mathbf{B}_{N_k}^{(k)} \\
-\Delta^{(k)} I_{N_k,1}^{(k)} \mathbf{B}_1^{(k)} & -\Delta\, I_{N_k,2}^{(k)} \mathbf{B}_2^{(k)} & \dots & -\Delta^{(k)} I_{N_k,N_k}^{(k)} \mathbf{B}_{N_k}^{(k)}
\end{bmatrix} \tag{6.36}
$$

$$
\hat{\mathbf{I}}_{\text{int}}^{(k)} = \begin{bmatrix}
-\Delta^{(k)} I_{11}^{(k)} \mathbf{1}_{n_x} & -\Delta\, I_{12}^{(k)} \mathbf{1}_{n_x} & \dots & -\Delta^{(k)} I_{1,N_k}^{(k)} \mathbf{1}_{n_x} \\
-\Delta^{(k)} I_{21}^{(k)} \mathbf{1}_{n_x} & -\Delta^{(k)} I_{22}^{(k)} \mathbf{1}_{n_x} & \dots & -\Delta^{(k)} I_{2,N_k}^{(k)} \mathbf{1}_{n_x} \\
\vdots & \vdots & \ddots & \vdots \\
-\Delta^{(k)} I_{N_k-1,1}^{(k)} \mathbf{1}_{n_x} & -\Delta^{(k)} I_{N_k-1,2}^{(k)} \mathbf{1}_{n_x} & \dots & -\Delta^{(k)} I_{N_k-1,N_k}^{(k)} \mathbf{1}_{n_x} \\
-\Delta^{(k)} I_{N_k,1}^{(k)} \mathbf{1}_{n_x} & -\Delta\, I_{N_k,2}^{(k)} \mathbf{1}_{n_x} & \dots & -\Delta^{(k)} I_{N_k,N_k}^{(k)} \mathbf{1}_{n_x}
\end{bmatrix} \tag{6.37}
$$

$$
\hat{\mathbf{q}}_{\text{int}}^{(k)} = \begin{bmatrix}
\Delta^{(k)} \sum_{j=1}^{N_k} I_{1j}^{(k)} \mathbf{q}_j^{(k)} \\
\Delta^{(k)} \sum_{j=1}^{N_k} I_{2j}^{(k)} \mathbf{q}_j^{(k)} \\
\vdots \\
\Delta^{(k)} \sum_{j=1}^{N_k} I_{N_k,j}^{(k)} \mathbf{q}_j^{(k)}
\end{bmatrix} \tag{6.38}
$$

The linear equality constraint of the dynamics has the same form as in Eq. (6.21).

As the initial and final states are included in the optimization process, the boundary conditions for the rendezvous problem are simply

$$\mathbf{r}(t_0) = \mathbf{r}_0,\ \mathbf{v}(t_0) = \mathbf{v}_0,\ w(t_0) = w_0 \tag{6.39}$$

$$\mathbf{r}(t_f) = \mathbf{r}_f,\ \mathbf{v}(t_f) = \mathbf{v}_f \tag{6.40}$$

The nonconvex constraint violations are determined in a similar way using Eq. (6.34):

$$c_{\mathrm{dyn},i}^{\mathrm{noncvx},(k)} = \mathbf{x}_{i+1}^{(k)} - \mathbf{x}_1^{(k)} - \frac{t_{N_k+1}^{(k)} - t_1^{(k)}}{2} \sum_{j=1}^{N_k} I_{ij}^{(k)} \, \mathbf{f}_{\mathrm{nonlin}} \left(\mathbf{x}_j^{(k)}, \mathbf{u}_j^{(k)} \right), \quad i = 1, \dots, N_k,\ k = 1, \dots, K \tag{6.41}$$

6.1.2 Adaptive Flipped Radau Pseudospectral Method

In [94, 98], a flipped Radau pseudospectral method (FRPM) was developed to solve convexified powered descent and landing problems without adding measures for artificial infeasibility. However, instead of keeping a constant number of nodes per segment, we extend this approach and allow this number to vary, hence resulting in the adaptive FRPM.

In the FRPM, the collocation points are defined on the interval $(-1, 1]$, i.e., the initial node of each segment is not collocated. Given the standard LGR points $\xi \in [-1, 1)$, the flipped values $\tilde{\xi} \in (-1, 1]$ can be computed using

$$\tilde{\xi} = \mathrm{sort}(-\xi) \tag{6.42}$$

where *sort* sorts the values in ascending order.

Considering now N_k flipped LGR points $(\tilde{\xi}_1, \tilde{\xi}_2, \dots, \tilde{\xi}_{N_k})$ where $\tilde{\xi}_1 > -1$ and $\tilde{\xi}_{N_k} = 1$, an additional (non-collocated) point is defined that satisfies $\tilde{\xi}_0 = -1$. The affine transformation between the physical and pseudospectral time is given by

$$t_i^{(k)} = \frac{t_{N_k}^{(k)} - t_0^{(k)}}{2} \tilde{\xi}_i^{(k)} + \frac{t_{N_k}^{(k)} + t_0^{(k)}}{2}, \qquad i = 0, 1, ..., N_k \tag{6.43}$$

where $t_0 = t_0^{(1)}$ and $t_f = t_{N_K}^{(K)}$. The same notation as in the previous section is used with $i = 0, 1, \dots, N_k$ and $k = 1, \dots, K$. The state is approximated in the interval $[-1, 1]$ using $N_k + 1$ points as follows:

$$\mathbf{x}^{(k)}(\tilde{\xi}) = \sum_{i=0}^{N_k} \mathbf{x}_i^{(k)} \, l_i^{(k)}(\tilde{\xi}), \qquad l_i^{(k)}(\tilde{\xi}) = \prod_{\substack{j=0 \\ j \neq i}}^{N_k} \frac{\tilde{\xi} - \tilde{\xi}_j}{\tilde{\xi}_i - \tilde{\xi}_j}, \qquad k = 1, \dots, K \tag{6.44}$$

In contrast to RPM where the last point of the last segment is added as a decision variable, we do not include the initial, non-collocated node this time. Rather, we make use of the fact that $\mathbf{x}_0^{(1)}$ is equal to the initial boundary condition $\mathbf{x}_0$. Moreover, the initial control $\mathbf{u}_0^{(1)}$ of the first segment is not part of the optimization process in FRPM. Therefore, the continuous control in the first segment is given by

$$\mathbf{u}^{(1)}(\tilde{\xi}) = \sum_{i=1}^{N_1} \mathbf{u}_i^{(1)}\, l_i^{(1)}(\tilde{\xi}), \qquad l_i^{(1)}(\tilde{\xi}) = \prod_{\substack{j=1\\ j\neq i}}^{N_1} \frac{\tilde{\xi}-\tilde{\xi}_j}{\tilde{\xi}_i-\tilde{\xi}_j} \tag{6.45}$$

For all other segments, the control interpolation is

$$\mathbf{u}^{(k)}(\tilde{\xi}) = \sum_{i=0}^{N_k} \mathbf{u}_i^{(k)}\, l_i^{(k)}(\tilde{\xi}), \qquad l_i^{(k)}(\tilde{\xi}) = \prod_{\substack{j=0\\ j\neq i}}^{N_k} \frac{\tilde{\xi}-\tilde{\xi}_j}{\tilde{\xi}_i-\tilde{\xi}_j}, \qquad k=2,\dots,K \tag{6.46}$$

The linking conditions for segments $k > 1$ are:

$$t_{N_k}^{(k)} = t_0^{(k+1)}, \qquad k=2,\dots,K \tag{6.47}$$

$$\mathbf{x}_{N_k}^{(k)} = \mathbf{x}_0^{(k+1)}, \qquad k=2,\dots,K \tag{6.48}$$

$$\mathbf{u}_{N_k}^{(k)} = \mathbf{u}_0^{(k+1)}, \qquad k=2,\dots,K \tag{6.49}$$

The location of discretization and collocation points is illustrated in Fig. 6.2. Using

$$\frac{\mathrm{d}}{\mathrm{d}\xi} = \frac{t_{N_k}-t_0}{2}\frac{\mathrm{d}}{\mathrm{d}t} \tag{6.50}$$

the discretized form of an integral constraint is

$$\int_{t_0}^{t_f} L(\mathbf{x}(t),\mathbf{u}(t))\,\mathrm{d}t \quad\longrightarrow\quad \sum_{k=1}^{K} \frac{t_{N_k}^{(k)}-t_0^{(k)}}{2} \sum_{i=1}^{N_k} \tilde{w}_i^{(k)} L(\mathbf{x}_i^{(k)},\mathbf{u}_i^{(k)}) \tag{6.51}$$

with weights

$$\tilde{w}_i^{(k)} = \frac{1}{N_k^2}\frac{1+\tilde{\xi}_i}{\left[P_{N_k-1}(\tilde{\xi}_i)\right]^2}, \qquad i=1,\dots,N_k \tag{6.52}$$

Differential Formulation

Following the same procedure as for RPM, the dynamical constraints read

$$\sum_{j=0}^{N_k} D_{ij}^{(k)}\mathbf{x}_j^{(k)} = \frac{t_{N_k}^{(k)}-t_0^{(k)}}{2}\mathbf{f}\left(\mathbf{x}_i^{(k)},\mathbf{u}_i^{(k)}\right), \qquad i=1,...,N_k \tag{6.53}$$

Substituting the linearized dynamics into Eq. (6.53) and rearranging terms yields

$$\begin{aligned} D_{i0}^{(k)}\mathbf{x}_0^{(k)} + \sum_{j=1}^{N_k} D_{ij}^{(k)}\mathbf{x}_j^{(k)} = \frac{t_{N_k}^{(k)}-t_0^{(k)}}{2} \Big[&\mathbf{A}\left(\bar{\mathbf{x}}_\mathbf{i}^{(\mathbf{k})},\bar{\mathbf{u}}_\mathbf{i}^{(\mathbf{k})}\right)\mathbf{x}_i^{(k)} + \mathbf{B}\left(\bar{\mathbf{x}}_i^{(k)},\bar{\mathbf{u}}_\mathbf{i}^{(\mathbf{k})}\right)\mathbf{u}_i^{(k)} \\ &+\mathbf{q}\left(\bar{\mathbf{x}}_i^{(k)},\bar{\mathbf{u}}_i^{(k)}\right) + \boldsymbol{\nu}_i^{(k)}\Big] \end{aligned} \tag{6.54}$$

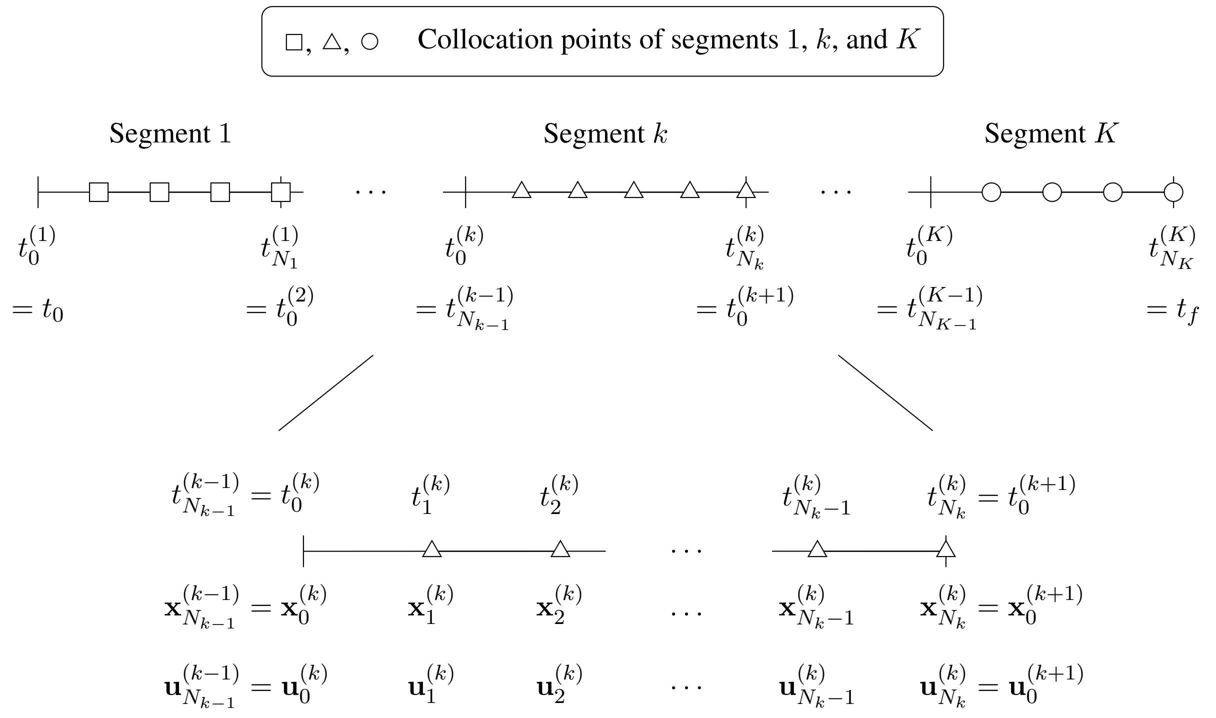

□, △, ○ Collocation points of segments 1, k, and K
Segment 1
Segment k
Segment K
$t_0^{(1)} = t_0$
$t_{N_1}^{(1)} = t_0^{(2)}$
$t_0^{(k)} = t_{N_{k-1}}^{(k-1)}$
$t_{N_k}^{(k)} = t_0^{(k+1)}$
$t_0^{(K)} = t_{N_{K-1}}^{(K-1)}$
$t_{N_K}^{(K)} = t_f$
$t_{N_{k-1}}^{(k-1)} = t_0^{(k)}$
$t_1^{(k)}$
$t_2^{(k)}$
$t_{N_k-1}^{(k)}$
$t_{N_k}^{(k)} = t_0^{(k+1)}$
$\mathbf{x}_{N_{k-1}}^{(k-1)} = \mathbf{x}_0^{(k)}$
$\mathbf{x}_1^{(k)}$
$\mathbf{x}_2^{(k)}$
$\mathbf{x}_{N_k-1}^{(k)}$
$\mathbf{x}_{N_k}^{(k)} = \mathbf{x}_0^{(k+1)}$
$\mathbf{u}_{N_{k-1}}^{(k-1)} = \mathbf{u}_0^{(k)}$
$\mathbf{u}_1^{(k)}$
$\mathbf{u}_2^{(k)}$
$\mathbf{u}_{N_k-1}^{(k)}$
$\mathbf{u}_{N_k}^{(k)} = \mathbf{u}_0^{(k+1)}$

Figure 6.2: Discretization and collocation points for FRPM.

for $i = 1, \ldots, N_k$. $\mathbf{x}_0^{(k)}$ is the initial state of the kth segment. For $k = 1$, this is the initial boundary condition $\mathbf{x}_0$. Similar to Eq. (6.21), the dynamical constraints can be written as a single linear equality constraint. The matrix $\hat{\mathbf{A}}^{(1)}$ for the first segment is given by

$$\hat{\mathbf{A}}^{(1)} = \begin{bmatrix} D_{11}^{(k)}\,\mathbf{1}_{n_x} - \Delta^{(k)}\,\mathbf{A}_1^{(k)} & D_{12}^{(k)}\,\mathbf{1}_{n_x} & \cdots & D_{1N_k}^{(k)}\,\mathbf{1}_{n_x} \\ D_{21}^{(k)}\,\mathbf{1}_{n_x} & D_{22}^{(k)}\,\mathbf{1}_{n_x} - \Delta^{(k)}\,\mathbf{A}_2^{(k)} & \cdots & D_{2N_k}^{(k)}\mathbf{1}_{n_x} \\ \vdots & \vdots & \ddots & \vdots \\ D_{N_k1}^{(k)}\,\mathbf{1}_{n_x} & D_{N_k2}^{(k)}\,\mathbf{1}_{n_x} & \cdots & D_{N_kN_k}^{(k)}\,\mathbf{1}_{n_x} - \Delta^{(k)}\,\mathbf{A}_{N_k}^{(k)} \end{bmatrix} \tag{6.55}$$

For subsequent segments $1 < k \leq K$, we obtain

$$\hat{\mathbf{A}}^{(k)} = \left[\begin{array}{c|ccccc} D_{10}^{(k)}\,\mathbf{1}_{n_x} & D_{11}^{(k)}\,\mathbf{1}_{n_x} - \Delta^{(k)}\,\mathbf{A}_1^{(k)} & D_{12}^{(k)}\,\mathbf{1}_{n_x} & \cdots & \cdots & D_{1,N_k}^{(k)}\,\mathbf{1}_{n_x} \\ D_{20}^{(k)}\,\mathbf{1}_{n_x} & D_{21}^{(k)}\,\mathbf{1}_{n_x} & D_{22}^{(k)}\,\mathbf{1}_{n_x} - \Delta^{(k)}\,\mathbf{A}_2^{(k)} & \cdots & \cdots & D_{2,N_k}^{(k)}\,\mathbf{1}_{n_x} \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ D_{N_k,0}^{(k)}\,\mathbf{1}_{n_x} & D_{N_k,1}^{(k)}\,\mathbf{1}_{n_x} & D_{N_k,2}^{(k)}\,\mathbf{1}_{n_x} & \cdots & \cdots & D_{N_k,N_k}^{(k)}\,\mathbf{1}_{n_x} - \Delta^{(k)}\,\mathbf{A}_{N_k}^{(k)} \end{array}\right], \quad k = 2,\ldots,K \tag{6.56}$$

$\hat{\mathbf{B}}^{(k)}$ takes the same form as in Eq. (6.23), and $\hat{\mathbf{q}}^{(k)}$ is

$$\hat{\mathbf{q}}^{(1)} = \begin{bmatrix} \Delta^{(1)}\,\mathbf{q}_1^{(1)} - D_{10}^{(1)}\,\mathbf{x}_0 \\ \Delta^{(1)}\,\mathbf{q}_2^{(1)} - D_{20}^{(1)}\,\mathbf{x}_0 \\ \vdots \\ \Delta^{(1)}\,\mathbf{q}_{N_1}^{(1)} - D_{N_1 0}^{(1)}\,\mathbf{x}_0 \end{bmatrix}, \qquad \hat{\mathbf{q}}^{(k)} = \begin{bmatrix} \Delta^{(k)}\,\mathbf{q}_1^{(k)} \\ \Delta^{(k)}\,\mathbf{q}_2^{(k)} \\ \vdots \\ \Delta^{(k)}\,\mathbf{q}_{N_k}^{(k)} \end{bmatrix}, \quad k = 2,\ldots,K \tag{6.57}$$

Note that the initial condition $\mathbf{x}_0$ is included in $\hat{\mathbf{q}}^{(1)}$ in Eq. (6.57).

The nonconvex constraint violations are computed using Eqs. (5.31a) and (6.53).

Integral Formulation

The integral form is

$$\begin{aligned} \mathbf{x}_{i+1}^{(k)} &= \mathbf{x}_0^{(k)} + \frac{t_{N_k}^{(k)} - t_0^{(k)}}{2} \sum_{j=1}^{N_k} I_{ij}^{(k)}\,\mathbf{f}\left(\mathbf{x}_j^{(k)}, \mathbf{u}_j^{(k)}\right) \\ &= \mathbf{x}_0^{(k)} + \frac{t_{N_k}^{(k)} - t_0^{(k)}}{2} \sum_{j=1}^{N_k} I_{ij}^{(k)} \left[\mathbf{A}\left(\bar{\mathbf{x}}_{\mathbf{j}}^{(\mathbf{k})}, \bar{\mathbf{u}}_{\mathbf{j}}^{(\mathbf{k})}\right)\mathbf{x}_j^{(k)} + \mathbf{B}\left(\bar{\mathbf{x}}_j^{(k)}, \bar{\mathbf{u}}_{\mathbf{j}}^{(\mathbf{k})}\right)\mathbf{u}_j^{(k)} + \mathbf{q}\left(\bar{\mathbf{x}}_j^{(k)}, \bar{\mathbf{u}}_j^{(k)}\right) + \boldsymbol{\nu}_j^{(k)}\right] \end{aligned} \tag{6.58}$$

where $i = 1,\ldots,N_k$. The $N_k \times N_k$ integration matrix $\mathbf{I}^{(k)}$ is again obtained by removing the first column of $\mathbf{D}^{(k)}$ and then taking the inverse:

$$\mathbf{I}^{(k)} := \left[\mathbf{D}_{1:N_k}^{(k)}\right]^{-1} \tag{6.59}$$

The concatenated quantities are defined in a similar manner to Eq. (6.20). The only difference is that the initial state $\mathbf{x}_0 = \mathbf{x}_0^{(1)}$ and control $\mathbf{u}_0^{(1)}$ are not part of the solution vector. The matrix $\hat{\mathbf{A}}_{\text{int}}^{(1)}$ corresponding to the first segment reads

$$\hat{\mathbf{A}}_{\text{int}}^{(1)} = \begin{bmatrix} \mathbf{1}_{n_x} - \Delta^{(1)} I_{11}^{(1)} \mathbf{A}_1^{(1)} & -\Delta^{(1)} I_{12}^{(1)} \mathbf{A}_1^{(1)} & \dots & -\Delta^{(1)} I_{1,N_1}^{(1)} \mathbf{A}_{N_1}^{(1)} \\ -\Delta^{(1)} I_{21}^{(1)} \mathbf{A}_1^{(1)} & \mathbf{1}_{n_x} - \Delta^{(1)} I_{22}^{(1)} \mathbf{A}_2^{(1)} & \dots & -\Delta^{(1)} I_{2,N_1}^{(1)} \mathbf{A}_{N_1}^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ -\Delta^{(1)} I_{N_1,1}^{(1)} \mathbf{A}_1^{(1)} & -\Delta^{(1)} I_{N_1,2}^{(1)} \mathbf{A}_2^{(1)} & \dots & \mathbf{1}_{n_x} - \Delta^{(1)} I_{N_1,N_1}^{(1)} \mathbf{A}_{N_1}^{(1)} \end{bmatrix} \tag{6.60}$$

For $1 < k \leq K$, the matrix is

$$\hat{\mathbf{A}}_{\text{int}}^{(k)} = \left[\begin{array}{c:cccc} -\mathbf{1}_{n_x} & \mathbf{1}_{n_x} - \Delta^{(k)} I_{11}^{(k)} \mathbf{A}_1^{(k)} & -\Delta^{(k)} I_{12}^{(k)} \mathbf{A}_1^{(k)} & \dots & -\Delta^{(k)} I_{1,N_k}^{(k)} \mathbf{A}_{N_k}^{(k)} \\ -\mathbf{1}_{n_x} & -\Delta^{(k)} I_{21}^{(k)} \mathbf{A}_1^{(k)} & \mathbf{1}_{\mathbf{n_x}} - \Delta^{(k)} I_{22}^{(k)} \mathbf{A}_2^{(k)} & \dots & -\Delta^{(k)} I_{2,N_k}^{(k)} \mathbf{A}_{N_k}^{(k)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -\mathbf{1}_{n_x} & -\Delta^{(k)} I_{N_k,1}^{(k)} \mathbf{A}_1^{(k)} & -\Delta^{(k)} I_{N_k,2}^{(k)} \mathbf{A}_2^{(k)} & \dots & \mathbf{1}_{n_x} - \Delta^{(k)} I_{N_k,N_k}^{(k)} \mathbf{A}_{N_k}^{(k)} \end{array}\right] \tag{6.61}$$

The columns left of the dashed line refer to the initial non-collocated point that is equal to the last point of the previous segment. The expression for $\hat{\mathbf{B}}_{\text{int}}^{(k)}$ is the same as in Eq. (6.36) when the standard LGR points are used. $\hat{\mathbf{q}}_{\text{int}}^{(k)}$ is given by

$$\hat{\mathbf{q}}_{\text{int}}^{(1)} = \begin{bmatrix} \Delta^{(1)} \sum_{j=1}^{N_1} I_{1j}^{(1)} \mathbf{q}_j^{(1)} + \mathbf{x}_0 \\ \Delta^{(1)} \sum_{j=1}^{N_1} I_{2j}^{(1)} \mathbf{q}_j^{(1)} + \mathbf{x}_0 \\ \vdots \\ \Delta^{(1)} \sum_{j=1}^{N_1} I_{N_1,j}^{(1)} \mathbf{q}_j^{(1)} + \mathbf{x}_0 \end{bmatrix}, \qquad \hat{\mathbf{q}}_{\text{int}}^{(k)} = \begin{bmatrix} \Delta^{(k)} \sum_{j=1}^{N_k} I_{1j}^{(k)} \mathbf{q}_j^{(k)} \\ \Delta^{(k)} \sum_{j=1}^{N_k} I_{2j}^{(k)} \mathbf{q}_j^{(k)} \\ \vdots \\ \Delta^{(k)} \sum_{j=1}^{N_k} I_{N_k,j}^{(k)} \mathbf{q}_j^{(k)} \end{bmatrix}, \quad k = 2, \dots, K \tag{6.62}$$

As the initial boundary condition is implicitly considered by adding $\mathbf{x}_0$ to $\hat{\mathbf{q}}^{(1)}$ and $\hat{\mathbf{q}}_{\text{int}}^{(1)}$, only the final boundary conditions in Eq. (6.40) are to be imposed as linear constraints.

Similar to the previous cases, the nonconvex constraint violations are obtained using Eqs. (5.31a) and (6.58).

6.1.3 First-Order-Hold Method

Although we focus on fuel-optimal transfers in this chapter, we also present the formulation when the final time is free for later use.

Fixed Final Time

In the first-order-hold discretization method considered in this dissertation, the time domain is divided into $N-1$ equally spaced segments

$$t_0 = t_1 < t_2 < \cdots < t_N = t_f \tag{6.63}$$

where N denotes the number of discretization points. Note that nodes and collocation points are the same in FOH. Given the controls $\mathbf{u}_k$ and $\mathbf{u}_{k+1}$ at nodes k and $k+1$, respectively, $k = 1, \ldots, N-1$, the continuous control profile $\mathbf{u}(t)$ is approximated by a piecewise affine function as follows:

$$\mathbf{u}(t) = \underbrace{\frac{t_{k+1}-t}{t_{k+1}-t_k}}_{=:\,\lambda_-(t)} \mathbf{u}_k + \underbrace{\frac{t-t_k}{t_{k+1}-t_k}}_{=:\,\lambda_+(t)} \mathbf{u}_{k+1} = \lambda_-(t)\,\mathbf{u}_k + \lambda_+(t)\,\mathbf{u}_{k+1}, \qquad t \in [t_k, t_{k+1}] \tag{6.64}$$

The linearized dynamics are given by [122]

$$\dot{\mathbf{x}}(t) = \mathbf{A}(t)\,\mathbf{x}(t) + \mathbf{B}\,\lambda_-(t)\,\mathbf{u}_k + \mathbf{B}\,\lambda_+(t)\,\mathbf{u}_{k+1} + \mathbf{q}(t) \tag{6.65}$$

which can be rewritten in discretized form as [182]

$$\mathbf{x}_{k+1} = \mathbf{A}_k\,\mathbf{x}_k + \mathbf{B}_k^-\,\mathbf{u}_k + \mathbf{B}_k^+\,\mathbf{u}_{k+1} + \mathbf{q}_k + \boldsymbol{\nu}_k \tag{6.66}$$

where $(\cdot)_k := (\cdot)(t_k)$. $\boldsymbol{\Phi}$ is the state transition matrix that satisfies

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\Phi}(t,t_0) = \mathbf{A}(t)\,\boldsymbol{\Phi}(t,t_0), \qquad \boldsymbol{\Phi}(t_0,t_0) = \mathbf{1} \tag{6.67}$$

and

$$\mathbf{A}_k = \boldsymbol{\Phi}(t_{k+1}, t_k) \tag{6.68a}$$

$$\mathbf{B}_k^- = \mathbf{A}_k \int_{t_k}^{t_{k+1}} \boldsymbol{\Phi}^{-1}(t,t_k)\,\mathbf{B}(t)\,\lambda_-(t)\,\mathrm{d}t \tag{6.68b}$$

$$\mathbf{B}_k^+ = \mathbf{A}_k \int_{t_k}^{t_{k+1}} \boldsymbol{\Phi}^{-1}(t,t_k)\,\mathbf{B}(t)\,\lambda_+(t)\,\mathrm{d}t \tag{6.68c}$$

$$\mathbf{q}_k = \mathbf{A}_k \int_{t_k}^{t_{k+1}} \boldsymbol{\Phi}^{-1}(t,t_k)\,\mathbf{q}(t)\,\mathrm{d}t \tag{6.68d}$$

At each node, the matrices $\mathbf{A}_k \in \mathbb{R}^{n_x \times n_x}$, $\mathbf{B}_k^- \in \mathbb{R}^{n_x \times n_u}$, $\mathbf{B}_k^+ \in \mathbb{R}^{n_x \times n_u}$, and $\mathbf{q}_k \in \mathbb{R}^{n_x}$ are calculated by integrating the state transition matrix in Eq. (6.67), the nonlinear dynamics in Eq. (5.11), and the integrands of Eqs. (6.68b)–(6.68d) simultaneously with an explicit fixed-step 8th-order Runge-Kutta method. n_x and n_u denote the number of states and controls, respectively. The reference state at $t \in [t_k, t_{k+1}]$ can be obtained using

$$\bar{\mathbf{x}}(t) = \bar{\mathbf{x}}_k + \int_{t_k}^{t} \mathbf{f}(\mathbf{x}(\xi), \mathbf{u}(\xi))\,\mathrm{d}\xi \tag{6.69}$$

A single equality constraint for the discretized dynamics is then created using

$$\left[\begin{array}{cc|cc|cc} \hat{\mathbf{A}}_1 & & \hat{\mathbf{B}}_1 & & \mathbf{1} & \\ & \mathbf{0} & & \mathbf{0} & & \mathbf{0} \\ \ddots & & \ddots & & \ddots & \\ \mathbf{0} & & \mathbf{0} & & \mathbf{0} & \\ & \hat{\mathbf{A}}_{N-1} & & \hat{\mathbf{B}}_{N-1} & & \mathbf{1} \end{array}\right] \begin{bmatrix} \mathbf{X} \\ \mathbf{U} \\ \boldsymbol{\nu} \end{bmatrix} = \begin{bmatrix} -\mathbf{q}_1 \\ \vdots \\ -\mathbf{q}_{N-1} \end{bmatrix} \tag{6.70}$$

with

$$\hat{\mathbf{A}}_k = \begin{bmatrix} \mathbf{A}_1 & -\mathbf{1} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_2 & -\mathbf{1} & \dots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & & \dots & \mathbf{0} & \mathbf{A}_{N-1} & -\mathbf{1} \end{bmatrix}, \quad \hat{\mathbf{B}}_k = \begin{bmatrix} \mathbf{B}_1^- & \mathbf{B}_1^+ & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_2^- & \mathbf{B}_2^+ & \dots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & & \dots & \mathbf{0} & \mathbf{B}_{N-1}^- & \mathbf{B}_{N-1}^+ \end{bmatrix} \tag{6.71}$$

where **1** and **0** denote the $n_x \times n_x$ identity and null matrix, respectively. $\mathbf{X}$, $\mathbf{U}$, and $\boldsymbol{\nu}$ are the concatenated states, controls, and virtual controls, respectively.

The nonconvex constraint violations in Eq. (5.31a) are computed by comparing the states at the end of segment k:

$$c_{\text{dyn},k}^{\text{noncvx}} = \mathbf{x}_{k+1} - \mathbf{x}_{k+1}^{\text{nonlin}}, \quad k = 1, \dots, N-1 \tag{6.72}$$

$\mathbf{x}_{k+1}$ denotes the state at t_{k+1} obtained from the optimization, and $\mathbf{x}_{k+1}^{\text{nonlin}}$ is computed using the nonlinear dynamics:

$$\mathbf{x}_{k+1}^{\text{nonlin}} = \mathbf{x}_k + \int_{t_k}^{t_{k+1}} \mathbf{f}_{\text{nonlin}}(\mathbf{x}(\zeta), \mathbf{u}(\zeta))\, \mathrm{d}\zeta, \quad k = 1, \dots, N-1 \tag{6.73}$$

where $\mathbf{x}_k = \mathbf{x}(t_k)$ is also the optimized state. Note that the integration is performed only segment-wise, i.e., the optimized state at $\mathbf{x}_k$ at t_k is used as the initial condition for the integration from t_k to t_{k+1}, and not the integrated state of the previous segment. This means that the nonlinear state trajectory $\mathbf{x}^{\text{nonlin}}$ is not continuous, but is "reset" at the beginning of a segment. This improves convergence as the integration error does not accumulate. The procedure is illustrated in Fig. 6.3.

Free Final Time

If the final time is free (for example in time-optimal problems where the final time is to be minimized), t_f becomes part of the optimization process. In order to keep the time horizon equally spaced, a time normalization is carried out such that an equivalent fixed-final-time problem is obtained [183]. Defining the normalized time $\xi \in [0, 1]$ such that

$$0 = \xi_1 < \xi_2 < \dots < \xi_N = 1 \tag{6.74}$$

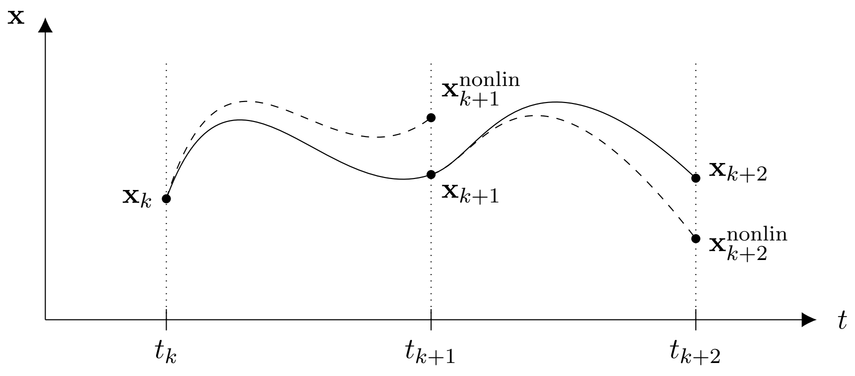


Figure 6.3: Optimized and integrated states for computing the nonconvex constraint violations in FOH.

and the dilation factor $\sigma := \mathrm{d}t/\mathrm{d}\xi$, the nonlinear dynamics become

$$\mathbf{x}'(t) := \frac{\mathrm{d}}{\mathrm{d}\xi}\mathbf{x}(t) = \frac{\mathrm{d}t}{\mathrm{d}\xi}\,\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{x}(t) = \sigma\,\dot{\mathbf{x}}(t) \tag{6.75}$$

As a consequence, t_f becomes an additional optimization variable, and the discretized optimal control problem is now formulated in the ξ domain with $\sigma = t_f$. Therefore, the linearized dynamics in Eq. (6.65) change to

$$\mathbf{x}'(\xi) = \sigma\,\mathbf{f}(\mathbf{x}(\xi), \mathbf{u}(\xi)) \tag{6.76}$$

and we obtain [183]

$$\mathbf{x}'(\xi) = \mathbf{A}(\xi)\,\mathbf{x}(\xi) + \mathbf{B}\,\lambda_-(\xi)\,\mathbf{u}_k + \mathbf{B}\,\lambda_+(\xi)\,\mathbf{u}_{k+1} + \mathbf{S}(\xi)\,\sigma + \mathbf{q}(\xi) \tag{6.77}$$

where the Jacobian matrices are now determined with respect to the modified dynamics $\mathbf{x}'(\xi)$. The discretized dynamics then read

$$\mathbf{x}_{k+1} = \mathbf{A}_k\,\mathbf{x}_k + \mathbf{B}_k^-\,\mathbf{u}_k + \mathbf{B}_k^+\,\mathbf{u}_{k+1} + \mathbf{S}_k\,\sigma + \mathbf{q}_k + \boldsymbol{\nu}_k \tag{6.78}$$

The matrices and vectors $\mathbf{A}_k$, $\mathbf{B}_k^-$, $\mathbf{B}_k^+$, and $\mathbf{q}_k$ are similar to Eq. (6.68), but in the ξ domain. $\mathbf{S}_k \in \mathbb{R}^{n_x}$ is given by

$$\mathbf{S}_k = \mathbf{A}_k \int_{\xi_k}^{\xi_{k+1}} \boldsymbol{\Phi}^{-1}(\xi, \xi_k)\,\mathbf{S}(\xi)\,\mathrm{d}\xi \tag{6.79}$$

The reference state at $\xi \in [\xi_k, \xi_{k+1}]$ and the nonlinear constraint violations are computed in the same way as in the formulation with a fixed final time.

6.1.4 Hermite–Legendre–Gauss–Lobatto Method

The arbitrary-order Legendre–Gauss–Lobatto discretization method relies on Hermite interpolation [116]. The idea is to use the information of the states and the dynamics at the nodal points and express the constraints at the collocation points by approximating the state variables with arbitrary-order polynomials in each segment [116, 117]. The method is briefly explained, and the interested reader is referred to [132] for details.

The total time of flight is divided into K segments. Each segment $[t_k, t_{k+1}]$ is mapped into the interval $[-1, 1]$ through the transformation

$$t \to \frac{h}{2}\,\xi + \frac{t_{k+1}+t_k}{2}, \qquad k = 1,\dots,K-1 \tag{6.80}$$

where $\xi \in [-1,1]$ and $h = t_{k+1} - t_k$ is the time step. In this work, nodes and collocation points are defined inside the interval $[-1,1]$ as the roots of the derivative of the $(n-1)$th order Legendre polynomial [117], where n is the order of the method. Given n, the state $\mathbf{x}^{(k)}(\xi) \in \mathbb{R}^{n_x}$ is approximated inside the kth segment as

$$\mathbf{x}^{(k)}(\xi) \approx \mathbf{a}_0^{(k)} + \mathbf{a}_1^{(k)}\,\xi + \cdots + \mathbf{a}_n^{(k)}\,\xi^n, \qquad k = 1,\dots,K \tag{6.81}$$

where the column vectors of coefficients $\mathbf{a}_m^{(k)} \in \mathbb{R}^{n_x}$, $m = 0,\dots,n$ are unknowns that are found by solving the following linear system:

$$\underbrace{\begin{bmatrix} \mathbf{1}_{n_x} & \theta_1\,\mathbf{1}_{n_x} & \theta_1^2\,\mathbf{1}_{n_x} & \dots & \theta_1^n\,\mathbf{1}_{n_x} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{1}_{n_x} & \theta_{n_p}\,\mathbf{1}_{n_x} & \theta_{n_p}^2\,\mathbf{1}_{n_x} & \dots & \theta_{n_p}^n\,\mathbf{1}_{n_x} \\ \mathbf{0}_{n_x} & \mathbf{1}_{n_x} & 2\,\theta_1\,\mathbf{1}_{n_x} & \dots & n\,\theta_1^{n-1}\,\mathbf{1}_{n_x} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{0}_{n_x} & \mathbf{1}_{n_x} & 2\,\theta_{n_p}\,\mathbf{1}_{n_x} & \dots & n\,\theta_{n_p}^{n-1}\,\mathbf{1}_{n_x} \end{bmatrix}}_{\boldsymbol{\theta}} \underbrace{\begin{bmatrix} \mathbf{a}_0^{(k)} \\ \vdots \\ \mathbf{a}_{n_p}^{(k)} \\ \vdots \\ \mathbf{a}_{n-1}^{(k)} \\ \mathbf{a}_n^{(k)} \end{bmatrix}}_{\mathbf{a}^{(k)}} = \underbrace{\begin{bmatrix} \mathbf{x}^{(k)}(\theta_1) \\ \vdots \\ \mathbf{x}^{(k)}(\theta_{n_p}) \\ \frac{h}{2}\,\mathbf{f}_l^{(k)}(\theta_1) \\ \vdots \\ \frac{h}{2}\,\mathbf{f}_l^{(k)}(\theta_{n_p}) \end{bmatrix}}_{\mathbf{b}^{(k)}} \tag{6.82}$$

In Eq. (6.82), θ_j are the positions of the nodal points, $n_p = (n+1)/2$ is the number of nodes in each segment, $\mathbf{1}_{n_x}$ is the $n_x \times n_x$ identity matrix, $\mathbf{0}_{n_x}$ the $n_x \times n_x$ null matrix, and $\mathbf{f}_l(\theta_j)$ the linearized dynamics as in Eq (2.33b). Once the coefficients $\mathbf{a}_m^{(k)}$ have been determined as $\mathbf{a}^{(k)} = \boldsymbol{\theta}^{-1}\,\mathbf{b}^{(k)}$, Eq. (6.81) can be used to define the state at the collocation points:

$$\mathbf{x}^{(k)}(\zeta) = \underbrace{\begin{bmatrix} \mathbf{1}_{n_x} & \zeta_1\,\mathbf{1}_{n_x} & \dots & \zeta_1^n\,\mathbf{1}_{n_x} \\ \mathbf{1}_{n_x} & \zeta_2\,\mathbf{1}_{n_x} & \dots & \zeta_2^n\,\mathbf{1}_{n_x} \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{1}_{n_x} & \zeta_{n_c}\,\mathbf{1}_{n_x} & \dots & \zeta_{n_c}^n\,\mathbf{1}_{n_x} \end{bmatrix}}_{\boldsymbol{\zeta}} \underbrace{\begin{bmatrix} \mathbf{a}_0^{(k)} \\ \mathbf{a}_1^{(k)} \\ \vdots \\ \mathbf{a}_n^{(k)} \end{bmatrix}}_{\mathbf{a}^{(k)}} = \boldsymbol{\zeta}\,\boldsymbol{\theta}^{-1}\,\mathbf{b}^{(k)} = \boldsymbol{\phi}\,\mathbf{b}^{(k)} \tag{6.83}$$

where ζ_j are the positions of the collocation points, and $n_c = (n-1)/2$ is the number of collocation points within each segment. The derivative of the state at the collocation points can be found in a similar fashion by deriving the matrix $\boldsymbol{\zeta}$. Similarly, the control $\mathbf{u}^{(k)}(\xi) \in \mathbb{R}^{n_u}$ is approximated in each segment as

$$\mathbf{u}^{(k)}(\xi) \approx \mathbf{a}_{u,0}^{(k)} + \mathbf{a}_{u,1}^{(k)}\xi + \cdots + \mathbf{a}_{u,n_p-1}^{(k)}\xi^{n_p-1}, \quad k = 1, \ldots, K \tag{6.84}$$

where the column vectors of coefficients $\mathbf{a}_{u,m}^{(k)} \in \mathbb{R}^{n_u}$, $m = 0, \ldots, n_p - 1$ are unknowns, obtained in a similar fashion as for the coefficients $\mathbf{a}_m^{(k)}$ inside Eq. (6.82). Note, however, that for the control no information about its dynamics is available and thus only the first n_p rows of the system can be considered. For this reason, the control is approximated by means of a polynomial of order $n_p - 1$. The quantities $\boldsymbol{\theta}_u$, $\mathbf{b}_u^{(k)}$, and $\boldsymbol{\phi}_u$ are defined accordingly. Once the matrices and vectors of all trajectory segments are computed, the dynamical constraints can be written as

$$\boldsymbol{\Delta} = \boldsymbol{\Phi}'\,\hat{\mathbf{b}} - \frac{h}{2}\left[\hat{\mathbf{f}}_f + \hat{\mathbf{A}}\left(\boldsymbol{\Phi}\,\hat{\mathbf{b}} - \boldsymbol{\Phi}\,\hat{\mathbf{b}}^*\right) + \hat{\mathbf{B}}\,\boldsymbol{\Phi}_u\,\hat{\mathbf{b}}_u\right] = \mathbf{0} \tag{6.85}$$

where the capital letters and $\hat{(\cdot)}$ indicate the concatenated quantities, and $\hat{\mathbf{f}}_f$ denotes the assembled unperturbed dynamics of the spacecraft.

6.2 Numerical Simulations

The performance of the discretization and trust-region methods is assessed in several thousand simulations. The number of converged simulations, iterations, final mass, computational time, propagation error, and the sparsity of the matrices of the discretized problem are compared. All simulations are carried out in MATLAB. The computational times are measured on an Intel Core i5-6300 2.30 GHz Laptop with four cores and 8 GB of RAM. The Embedded Conic Solver (ECOS) [86] is used to solve the second-order cone program in Eq. (5.23). The SCP algorithm converges if the maximum constraint violation and the relative change of the modified final mass $w(t_f)$ are lower than thresholds ε_c and ε_J, respectively (see also Eqs. (5.35) and (5.36) in Section 5.2). The algorithm also terminates without successful convergence to an optimal solution if there is no sufficient progress as per Eq. (5.37). Relevant SCP parameters are given in Table 6.1. Different combinations of ρ_i ($i = 0, 1, 2$), α, β, and δ for the hard, and λ_{TR}, $\lambda_{\text{TR},0}$, and ζ for the soft trust-region method were assessed in preliminary simulations. We found that the values in Table 6.1 perform well and are often a good compromise in terms of convergence, optimality, and number of iterations. The physical constants and scaling parameters are given in Table 6.2. Throughout this section, we refer to (F)RPM-D and (F)RPM-I for the differential and integral formulations of the (flipped) Radau pseudospectral method, HLGL for the Legendre–Gauss–Lobatto method based on Hermite interpolation, and FOH for the first-order-hold method.

Table 6.1: Parameters of the SCP algorithm.

Parameter	Value
Penalty weights λ_ν, λ_η	10.0, 10.0
Initial trust region R_0	100.0
ρ_0, ρ_1, ρ_2	0.01, 0.2, 0.85
α, β	1.5, 1.5
$\alpha_{\min}, \beta_{\min}$	1.01, 1.01
$\alpha_{\max}, \beta_{\max}$	4.0, 4.0
δ	1.0, 1.2
λ_{TR}, $\lambda_{\text{TR},0}$, ζ	10^{10}, 10^{7}, 5.0
ε_c, ε_J, ε_x	10^{-6}, 10^{-4}, 10^{-7}
Max. iterations	250

Table 6.2: Physical constants in all simulations.

Parameter	Value
Gravitational const. μ	$1.327\,124\,4 \times 10^{11}\,\text{km}^3\,\text{s}^{-2}$
Gravitational accel. g_0	$9.806\,65 \times 10^{-3}\,\text{km}\,\text{s}^{-2}$
Length unit LU = AU	$1.495\,978\,707 \times 10^{8}\,\text{km}$
Velocity unit VU	$\sqrt{\mu\,\text{AU}^{-1}}$
Time unit TU	$\text{AU}\,\text{VU}^{-1}$
Acceleration unit ACU	$\text{VU}\,\text{TU}^{-1}$
Mass unit MU	m_0

Table 6.3: Simulation values for SEL$_2$ to 2000 SG344, Earth-Venus and Earth-Dionysus transfers [39, 129, 131].

Param.	SEL$_2$ - 2000 SG344	Earth - Venus	Earth - Dionysus
$\mathbf{r}_0$, AU	$[-0.7018607, 0.7062324, -3.51115 \times 10^{-5}]$	$[0.9708322, 0.2375844, -1.67106 \times 10^{-6}]$	$[-0.0243177, 0.9833014, -1.51168 \times 10^{-5}]$
$\mathbf{v}_0$, VU	$[-0.7329695, -0.7159049, 4.40245 \times 10^{-5}]$	$[-0.2545390, 0.9686550, 1.50402 \times 10^{-5}]$	$[-1.0161293, -0.0284940, 1.69550 \times 10^{-6}]$
m_0, kg	22.6	1500	4000
$\mathbf{r}_f$, AU	$[0.4180680, 0.8289711, -0.0014338]$	$[-0.3277178, 0.6389172, 0.0276593]$	$[-2.0406178, 2.0517913, 0.5542890]$
$\mathbf{v}_f$, VU	$[-0.9699033, 0.4363022, -0.0012338]$	$[-1.0508770, -0.5435675, 0.0532095]$	$[-0.1423193, -0.4510880, 0.0189469]$
m_f, kg	free	free	free
$T_{\max}$, N	2.2519×10^{-3}	0.33	0.32
I_{sp}, s	3067	3800	3000
t_f, days	700	1000	3534

6.2.1 Overview of Simulations

Three different targets (near-Earth asteroid 2000 SG344, Venus, and asteroid Dionysus) where the complexity of the transfers increases, and two methods to generate (infeasible) initial guesses of different quality (perturbed shape-based cubic interpolation approach and propagation of dynamics) are taken into account. Details about the shape-based cubic interpolation can be found in the Appendix A. The simulation values of each target are given in Table 6.3, and Table 6.4 summarizes the number of initial guesses. The comparison of an optimal trajectory and the ones generated with cubic interpolation and propagation are illustrated in Fig. 6.4. Clearly, the trajectories of the initial guesses deviate significantly from the optimal one, and the final positions of the initial guesses are far from the target position. Note

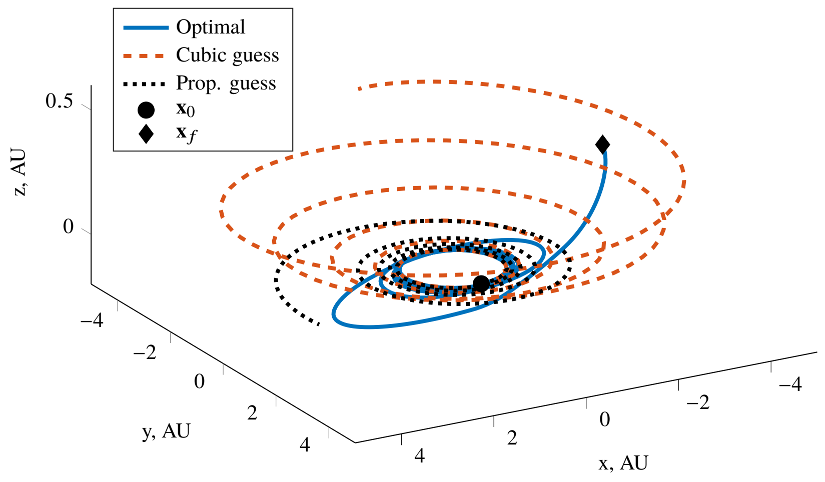


Figure 6.4: Optimal trajectory and initial guesses generated using cubic interpolation with 5.4 revolutions and propagation with $T = 0.5\,T_{\max}$ for a Dionysus transfer.

that the initial controls are set to zero in all simulations. Figures 6.5, 6.6, and 6.7 show typical optimized trajectories and the corresponding thrust profiles (linearly interpolated between the nodes) for the SEL$_2$-2000 SG344, Earth-Venus, and Earth-Dionysus transfers, respectively. SEL$_2$ refers to the Sun-Earth Lagrange point L$_2$.

Moreover, the trust-region and discretization methods described in Sections 5.2 and 6.1, different numbers of discretization points (ranging from 100 to 300) and orders of the interpolating polynomial (ranging from 3 to 23) are compared (see also Table 6.5). For each target, all combinations of different initial guesses, numbers of nodes, degrees of the interpolating polynomials, and trust-region methods are considered. An overview of the performed simulations is shown in Fig. 6.8.

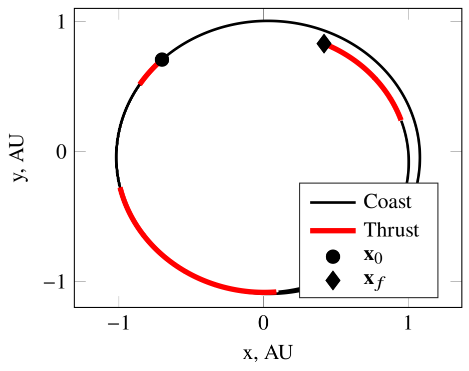
The total number of simulations n_{sim} for each discretization method is given in Table 6.6. It is determined using

$$n_{\text{sim}} = n_{\text{guess}} \cdot n_{\text{nodes}} \cdot n_{\text{TR}} \cdot n_{\text{orders}} \tag{6.86}$$

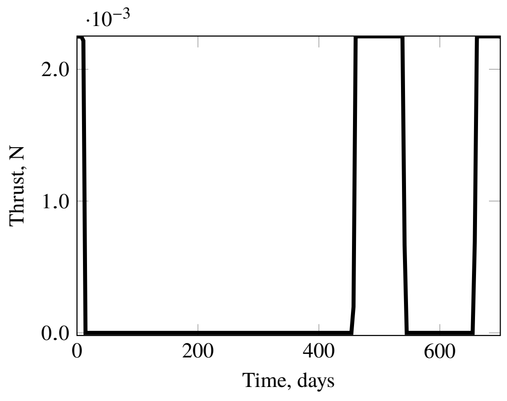
where n_{guess} is the total number of initial guesses obtained with the cubic-based and the propagation approaches, $n_{\text{nodes}} = 3$ the number of different nominal nodes, $n_{\text{TR}} = 3$ the number of trust-region methods, and $n_{\text{orders}} = 6$ the number of polynomial orders. Note that this term is only considered for HLGL and RPM/FRPM.

Table 6.4: Types and number of initial guesses for each transfer.

Parameter	SEL$_2$ - 2000 SG344	Earth - Venus	Earth - Dionysus
Cubic Interpolation	101	251	301
Propagation	101	101	101
Total	202	352	402

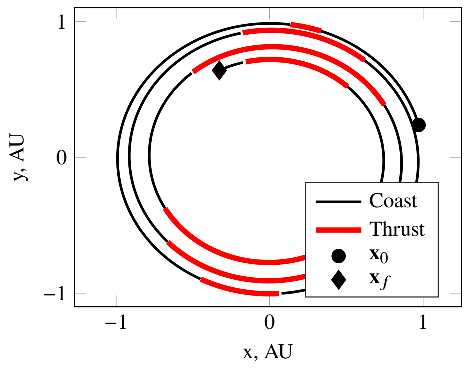


(a) Trajectory

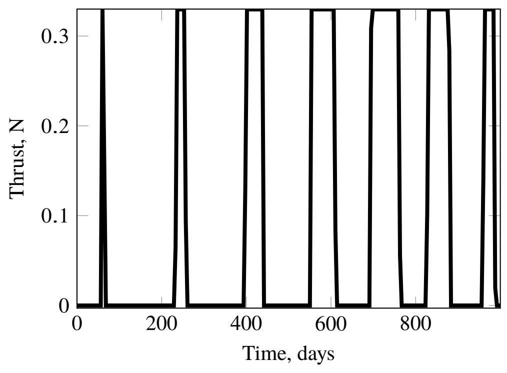


(b) Thrust profile

Figure 6.5: Typical SEL$_2$ to 2000 SG344 transfer trajectory and corresponding thrust profile.



(a) Trajectory

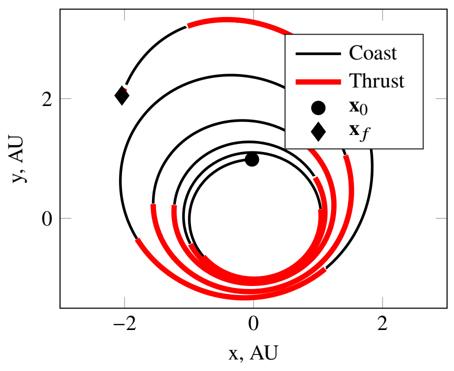


(b) Thrust profile

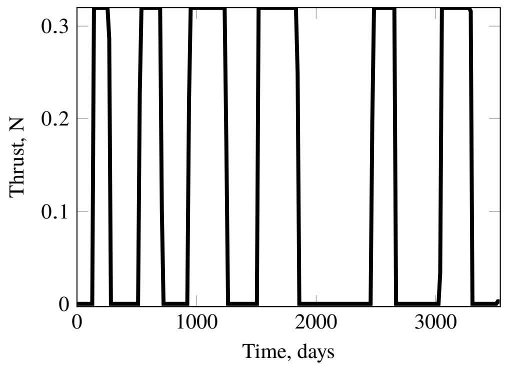
Figure 6.6: Typical Earth-Venus transfer trajectory and corresponding thrust profile.

6.2.2 Results

The performance of the algorithms is assessed by means of four comparisons for each of the transfers. For each of the aforementioned parameters (discretization and trust-region method, polynomial order, and number of nodes), we take the median of the obtained results by varying all the other parameters. For example, the comparison of the discretization methods is performed by taking the median of all

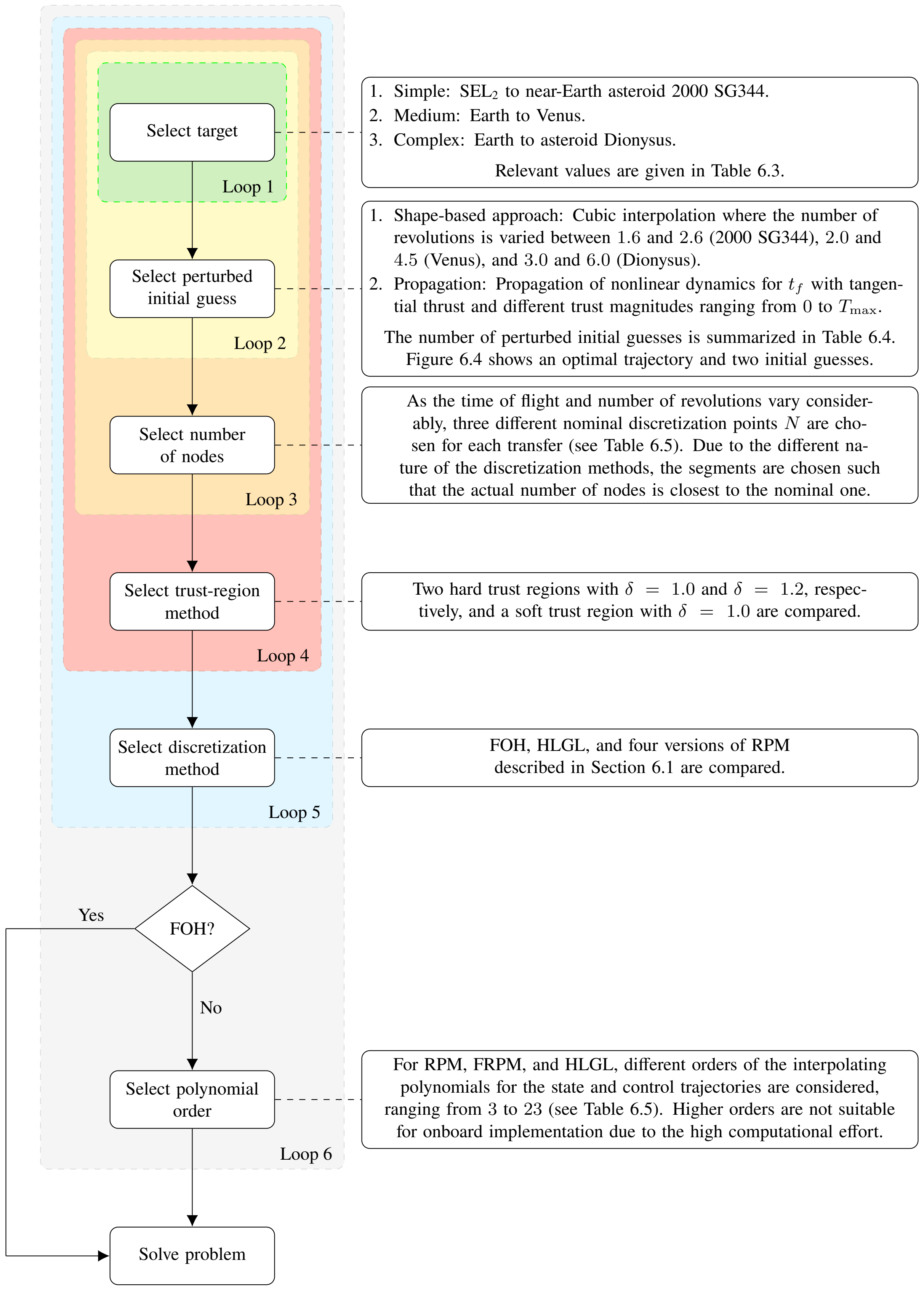


(a) Trajectory



(b) Thrust profile

Figure 6.7: Typical Earth-Dionysus transfer trajectory and corresponding thrust profile.



1. Simple: SEL_2 to near-Earth asteroid 2000 SG344.
2. Medium: Earth to Venus.
3. Complex: Earth to asteroid Dionysus.

Relevant values are given in Table 6.3.

1. Shape-based approach: Cubic interpolation where the number of revolutions is varied between 1.6 and 2.6 (2000 SG344), 2.0 and 4.5 (Venus), and 3.0 and 6.0 (Dionysus).
2. Propagation: Propagation of nonlinear dynamics for t_f with tangential thrust and different trust magnitudes ranging from 0 to $T_{\max}$.

The number of perturbed initial guesses is summarized in Table 6.4. Figure 6.4 shows an optimal trajectory and two initial guesses.

As the time of flight and number of revolutions vary considerably, three different nominal discretization points N are chosen for each transfer (see Table 6.5). Due to the different nature of the discretization methods, the segments are chosen such that the actual number of nodes is closest to the nominal one.

Two hard trust regions with $\delta = 1.0$ and $\delta = 1.2$, respectively, and a soft trust region with $\delta = 1.0$ are compared.

FOH, HLGL, and four versions of RPM described in Section 6.1 are compared.

For RPM, FRPM, and HLGL, different orders of the interpolating polynomials for the state and control trajectories are considered, ranging from 3 to 23 (see Table 6.5). Higher orders are not suitable for onboard implementation due to the high computational effort.

Figure 6.8: Overview of performed simulations.

Table 6.5: Number of nodes and orders of the interpolating polynomials for each transfer.

Parameter	SEL_2 - 2000 SG344	Earth - Venus	Earth - Dionysus
Nominal nodes	100, 150, 200	150, 200, 250	200, 250, 300
Polynomial orders	3, 7, 11, 15, 19, 23		

Table 6.6: Total number of simulations for each target and each discretization method.

Target	FOH	LGL, RPM-I, RPM-D, FRPM-I, FRPM-D
2000 SG344	1818	10908
Venus	3168	19008
Dionysus	3618	21708

results obtained with a given method for all polynomial orders, all considered nodes, and all trust-region strategies. Throughout our analysis, we comment on three key aspects: general performance (i.e., convergence, iterations, and final spacecraft mass), accuracy, and computational time and memory required by the discretization and trust-region methods. We also compare our results with the ones for the PDG problem in [124]. Finally, we assess the performance of our algorithms when compared with the state-of-the-art optimal control software GPOPS-II [175] in combination with the Sparse Nonlinear Optimizer (SNOPT) [184].

Convergence, Iterations, and Final Mass

One key objective is to understand the influence of the discretization method, the order of the interpolating polynomial, the number of nodes, and the trust-region method on the success rate, the iterations required to reach convergence, and the optimality of the solutions. The outcome of the analyses is reported in Figs. 6.9 – 6.11, where the results related to asteroid 2000 SG344, Venus, and asteroid Dionysus are presented, respectively. The error bars represent the 70th percentile of the considered quantity.

With regard to asteroid 2000 SG344, the largest number of converged cases is obtained with FOH (success rate of approximately $80\,\%$). HLGL yields only slightly fewer, and pseudospectral methods approximately $10\,\%$ fewer converged cases. The number of iterations is similar, even though FOH requires the fewest. The obtained final masses are almost the same for all methods. This is in accordance with the results of the PDG problem [124] where all methods achieve a similar fuel consumption. Notably, lower polynomial orders require fewer iterations than higher orders. Moreover, the success rate increases for higher polynomial degrees up to 15, and then remains almost constant. The convergence is slightly higher when N is increased, whereas iterations decrease for a larger number of nodes. Similar findings are reported in the PDG study. The hard trust-region with $\delta = 1.0$ and the soft trust-region method yield equivalent results. Remarkably, the hard trust-region approach with $\delta = 1.2$ requires only half as many iterations as with $\delta = 1.0$, at the cost of fewer converged simulations.

Regarding Venus, the tendency of the methods is opposite: the pseudospectral methods are able to find solutions in almost $60\,\%$ of the cases, therefore having a $10\,\%$ higher success rate compared to FOH and HLGL (see Fig. 6.10). Furthermore, the overall convergence is worse compared to 2000 SG344. The number of iterations and final masses are similar. Even though the convergence also improves for higher orders, the difference is less significant. The number of nodes and the trust-region method seem to have a small impact on the results (except for the fewer number of iterations when choosing $\delta = 1.2$).

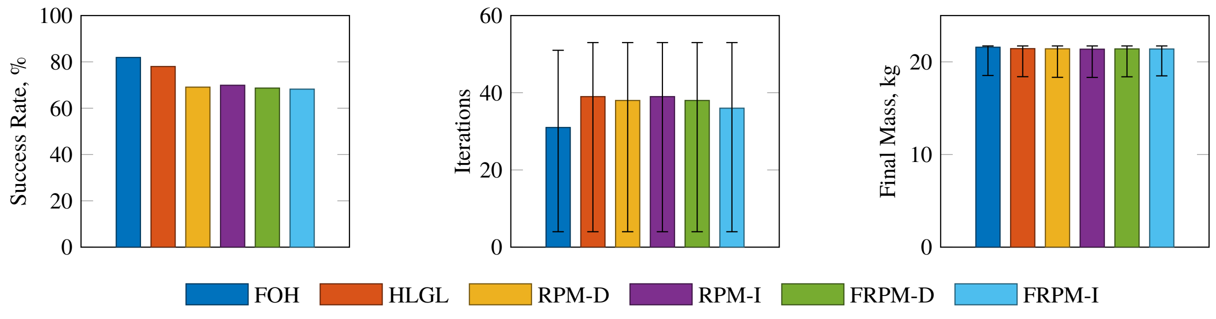
In the Dionysus case, FOH and HLGL yield $15\,\%$ more converged simulations than the pseudospectral methods (see Fig. 6.11). Remarkably, the behavior of the polynomial orders is opposite in this case: apart from the third order that yields the lowest success rate, the convergence is best for the 7th order and decreases as the order increases. The success rate slightly changes again depending on the number of nodes, whereas iterations and final mass are not particularly affected by N. The trust-region methods show the same behavior for the iterations and final mass as in the previous cases. Note, however, that the hard trust region with $\delta = 1.2$ achieves slightly higher success rates than the other methods in this example.

As the previous plots considered all orders, Fig. 6.12 shows the convergence for all targets for the most relevant polynomial degrees 7 and 11. This way the potentially poor performance of the third order does not bias the results. Apparently, the bars follow the same trend: FOH and HLGL seem to outperform the pseudospectral methods for the transfers to the asteroids 2000 SG344 and Dionysus. With regard to Venus, in contrast, RPM and FRPM achieve higher success rates.

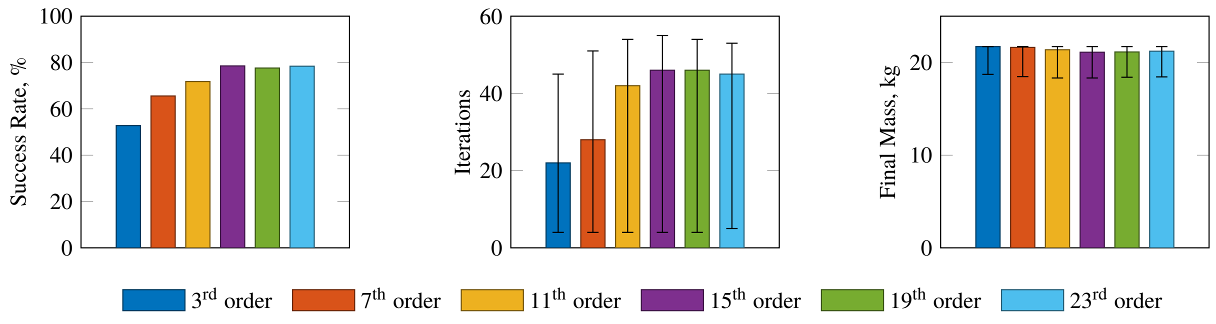
Figure 6.13 shows the convergence for the cubic interpolation and propagation guesses. Due to the similarity of the initial and final orbits, the success rate of approximately $70\,\%$ for the transfer to asteroid 2000 SG344 is high regardless of how the initial guess is generated. With regard to Venus and Dionysus, however, propagating with tangential thrust results in poor guesses that deviate considerably from the optimal trajectories. A success rate of almost $50\,\%$ is therefore remarkable. Still, using a cubic interpolation guess yields in general a larger number of converged cases.

Accuracy

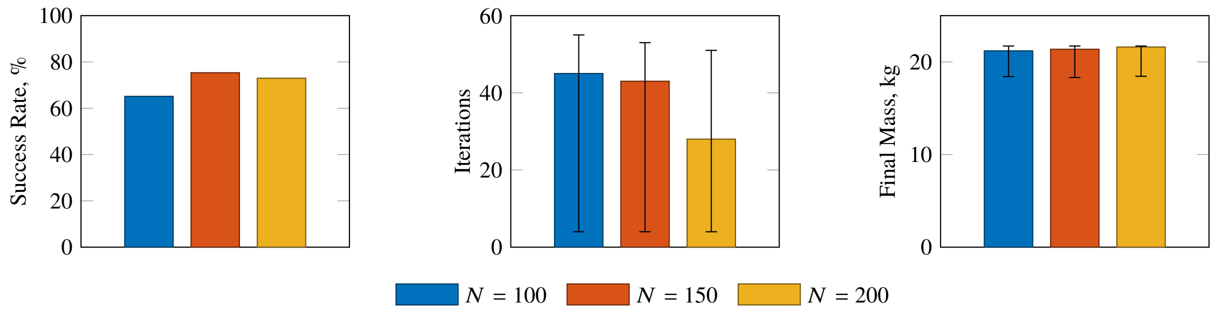
It is also crucial that a discretization method is able to achieve a certain accuracy. We define the propagation error for the position as $\|\mathbf{r}(t_f)_{\text{prop}} - \mathbf{r}(t_f)\|_2$ where $\mathbf{r}(t_f)_{\text{prop}}$ is the final position that is obtained by integrating the dynamics with the optimized controls (the error for the velocity is defined accordingly). As the thrust profile is only known at the discretization points, the controls need to be interpolated for the integration using Eqs. (6.3), (6.46), (6.64) and (6.84), respectively. Figure 6.14 shows



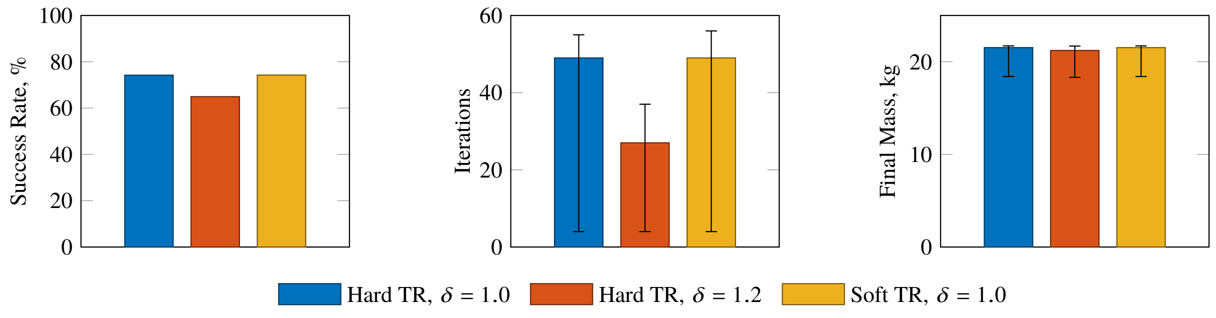
(a) Comparison of discretization methods.



(b) Comparison of the orders of the interpolating polynomial.

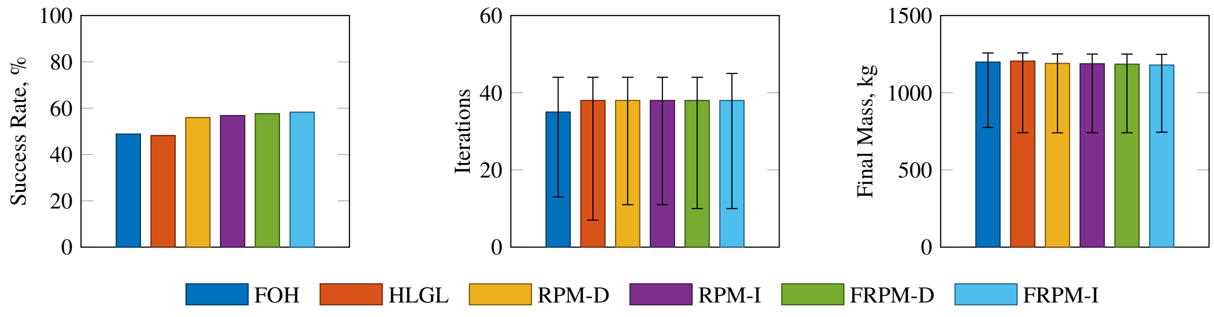


(c) Comparison of the number of nodes.

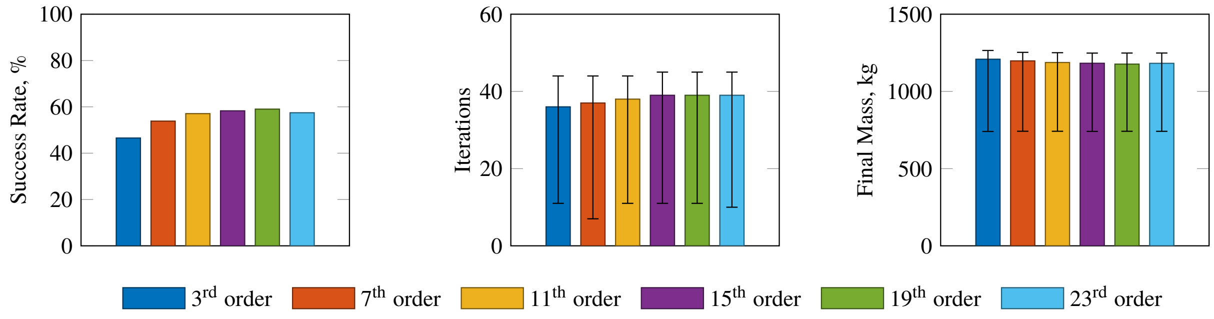


(d) Comparison of trust-region methods.

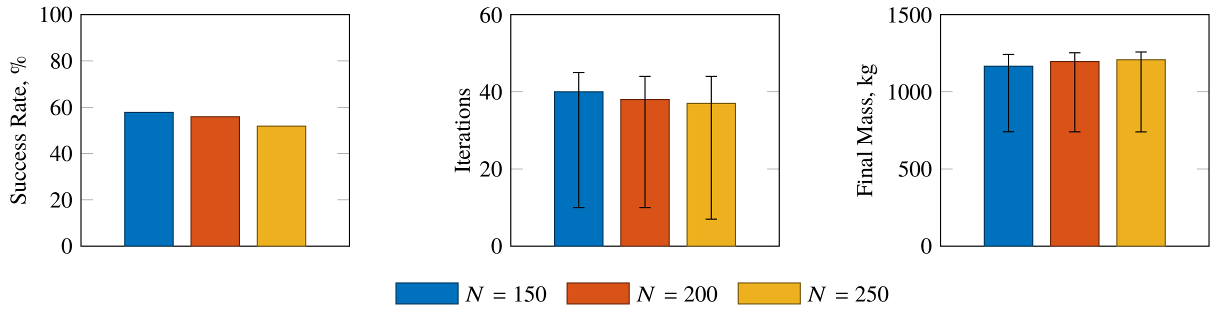
Figure 6.9: Influence of the discretization method, the order of the interpolating polynomial, the number of nodes, and the trust-region method on success rate, iterations, and final mass for the transfer to asteroid 2000 SG344.



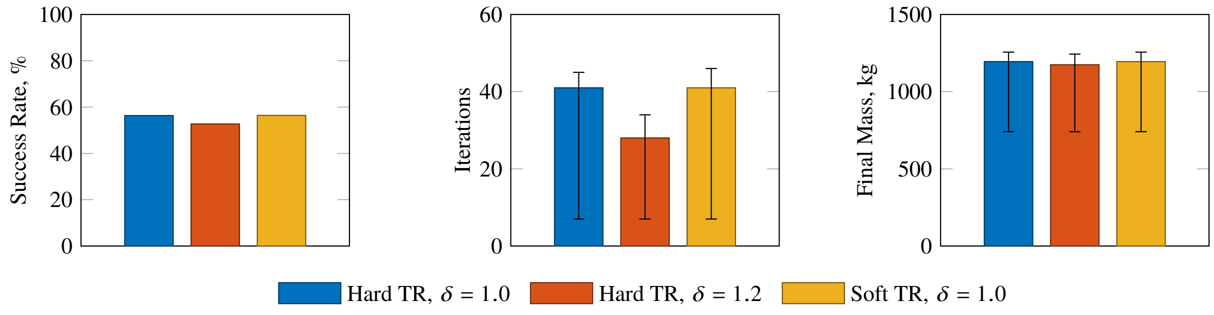
(a) Comparison of discretization methods.



(b) Comparison of the orders of the interpolating polynomial.

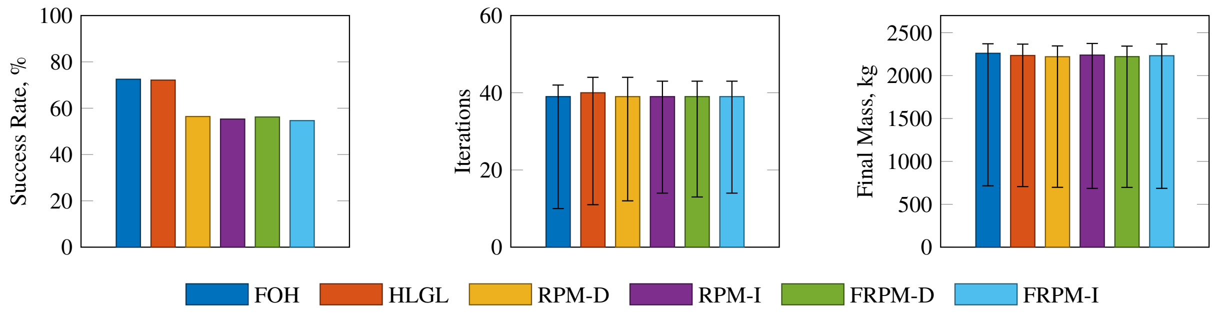


(c) Comparison of the number of nodes.

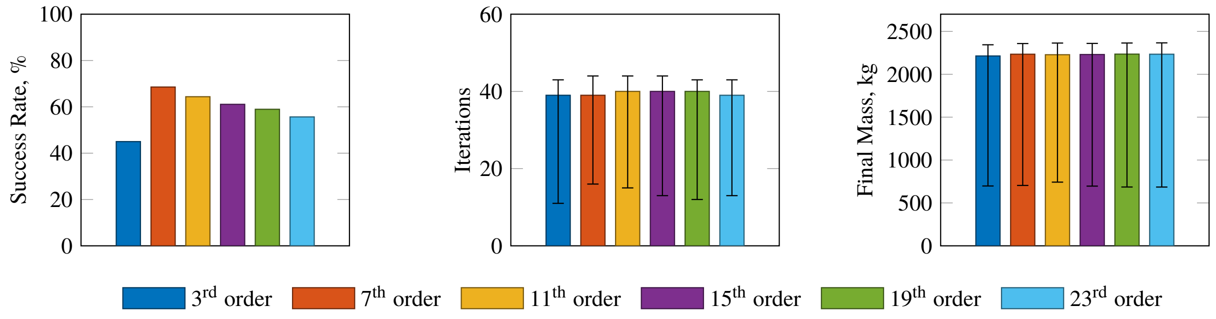


(d) Comparison of trust-region methods.

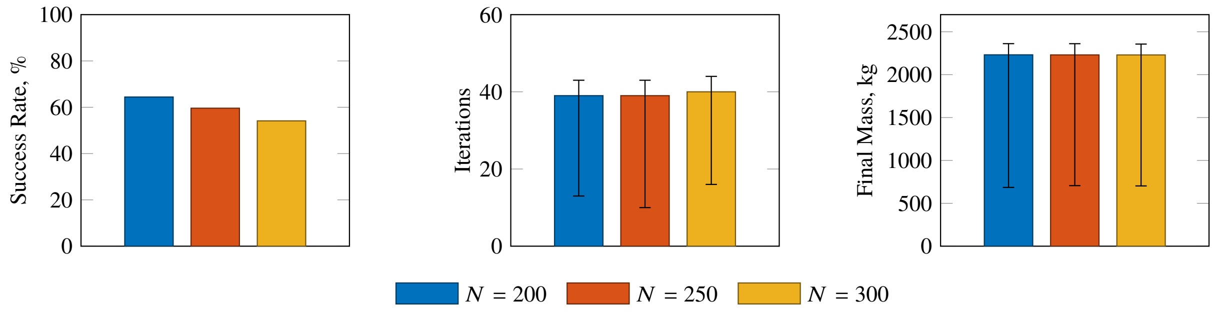
Figure 6.10: Influence of the discretization method, the order of the interpolating polynomial, the number of nodes, and the trust-region method on success rate, iterations, and final mass for the transfer to Venus.



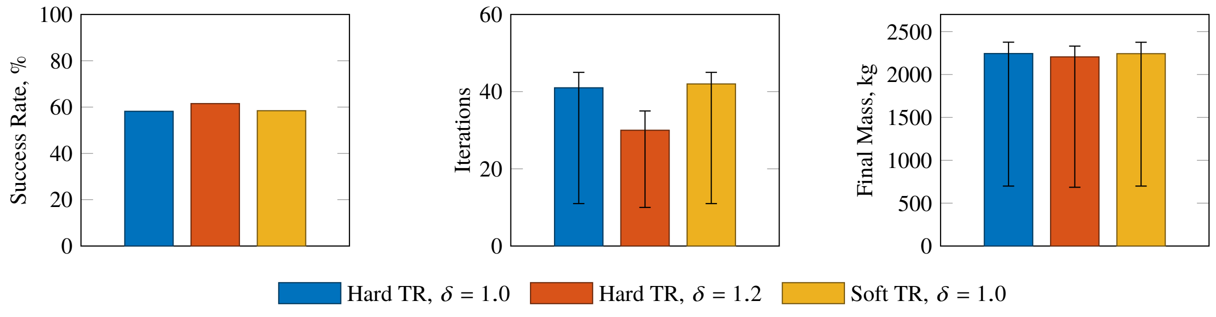
(a) Comparison of discretization methods.



(b) Comparison of the orders of the interpolating polynomial.



(c) Comparison of the number of nodes.



(d) Comparison of trust-region methods.

Figure 6.11: Influence of the discretization method, the order of the interpolating polynomial, the number of nodes, and the trust-region method on success rate, iterations, and final mass for the transfer to Dionysus.

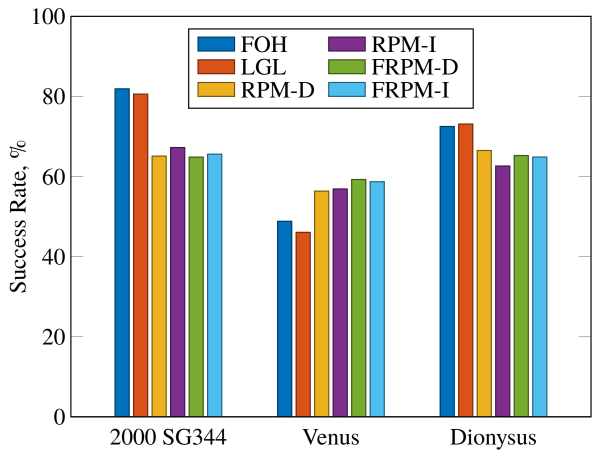


Figure 6.12: Comparison of success rate for polynomial orders 7 and 11 for all targets.

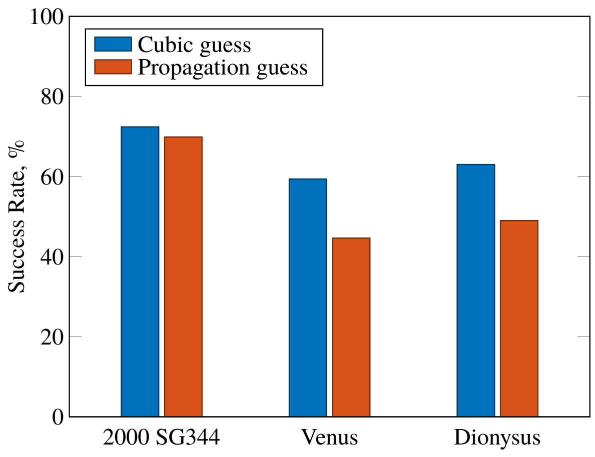


Figure 6.13: Comparison of success rate for different initial guesses for all targets.

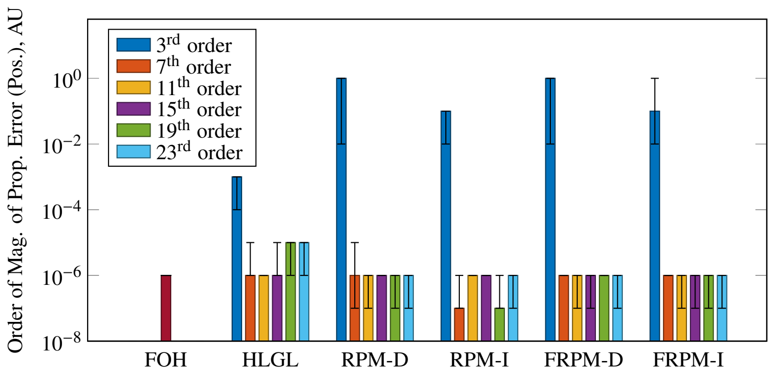


Figure 6.14: Comparison of the order of magnitude of the propagation error (position) for a Dionysus transfer.

the orders of magnitude of the propagation error for a Dionysus transfer. It is evident that all methods and polynomial orders achieved the desired accuracy of 10^{-6} AU except for the third order. More precisely, almost all methods found solutions with a median error of 10^{-7} AU or less; only some HLGL orders failed to do so in a few cases. The propagation error for the velocity shows a similar tendency, often being one order of magnitude smaller than the position error. The same statements are true for the other targets. Even though the time horizon is significantly smaller for the PDG problem, the errors on the final boundary conditions are very similar to our results (provided that a higher-order polynomial is used) [124]. This confirms the high accuracy of the proposed discretization methods. Although the propagation error is small for all methods, the interpolated controls violate the constraints on the thrust magnitude for HLGL and RPM/FRPM as shown in Fig. 6.15. The reason is that polynomial interpolation results in oscillations close to the edges of the segments for higher polynomial orders (Runge phenomenon). Only FOH is able to generate a bang-bang control profile that does not violate the bounds due to the linear interpolation. As expected, the same is true for the control profiles obtained when solving the PDG problem [124].

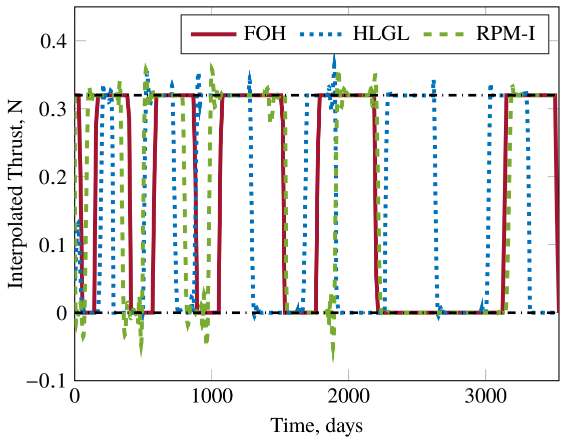


Figure 6.15: Interpolated control profiles obtained with FOH and 11th order polynomials (HLGL/RPM) for a Dionysus transfer.

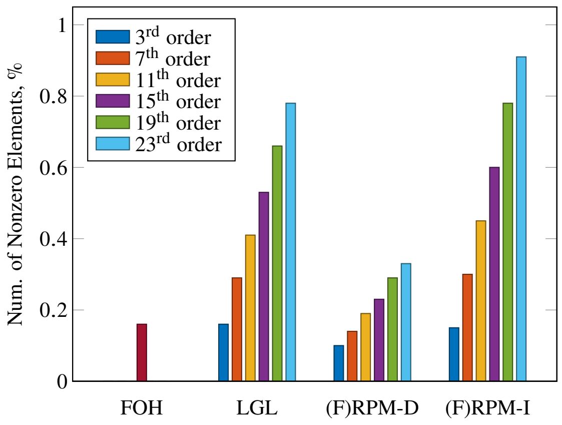
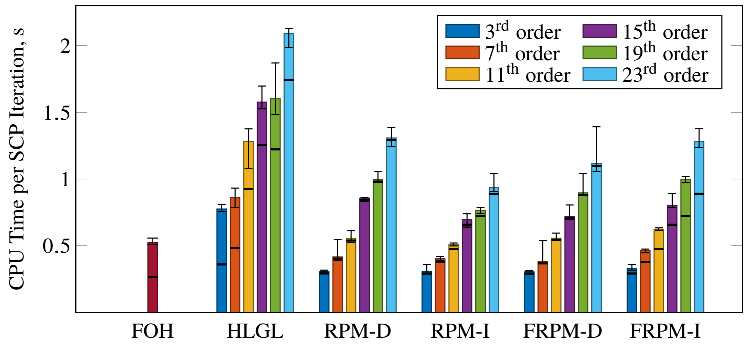


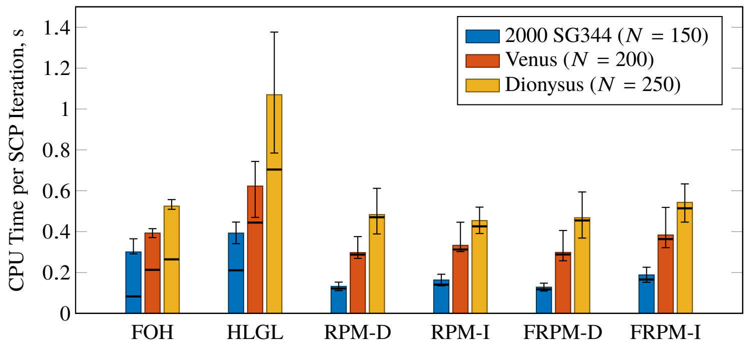
Figure 6.16: Comparison of the number of nonzero elements of the linear equality constraints matrix for a Dionysus transfer ($N = 250$).

Computational Time and Memory

Computational time and memory are two other important aspects to consider. As we are dealing with a large amount of optimization parameters, sparse linear algebra becomes crucial. Dense matrix operations would not only take longer to compute, but might also result in memory problems for large-scale optimization problems. The typical percentage of the nonzero elements in the linear equality constraints matrix is given in Fig. 6.16. Even though more than $99\,\%$ of the elements are zero for all methods, the integral formulations of RPM/FRPM and HLGL are several times denser than (F)RPM-D and FOH. This increases the time required to solve the second-order cone program, therefore resulting in a higher total CPU time when the number of iterations does not change. Figure 6.17a shows the computational times per SCP iteration for a typical Dionysus transfer with $N = 250$ discretization points. Median values with minimum and maximum values are shown. The heights of the bars show the total CPU times, the horizontal lines within each bar indicate the times required to solve one SOCP. As expected, the computational effort increases for higher orders as the matrices become denser. Remarkably, for the four pseudospectral methods the solver time accounts for the largest portion of the total time, whereas for FOH and HLGL the time outside the solver is larger. This is because of the integration (FOH) and the transformations due to the definition of the nodes and collocation points (HLGL). Moreover, HLGL requires the greatest computational effort among all methods regardless of the transfer (see Fig. 6.17b where the most relevant orders 7 and 11 are shown). The difference becomes more significant when the number of nodes is increased. In general, pseudospectral methods with orders 7 and 11 seem to be the fastest, directly followed by FOH which eventually outperforms all other methods when higher orders are considered. Given the typical number of iterations of 40, the maximum total CPU time is approximately 24 seconds for FOH and (F)RPM, and 52 seconds for HLGL. Although the integral formulations of



(a) CPU times for different polynomial orders for the Dionysus transfer.



(b) CPU times for polynomial orders 7 and 11 for all targets.

Figure 6.17: Comparison of CPU times per SCP iteration for different orders and targets.

(F)RPM are denser, their CPU times are sometimes lower than the ones obtained with (F)RPM-D due to the smaller number of solver iterations. Considering Figs. 6.9a, 6.10a, 6.11a, and 6.17b, it is interesting to note that the results regarding CPU time are partially in accordance with the findings for the PDG problem [124]: Even though the overall CPU time is not affected by the choice of either of the two methods, F(RPM) shows a higher sensitivity to the number of discretization points than FOH. Finally, albeit this chapter only considers fixed final boundary conditions, we will see in Chapter 9 that the computational effort required by the SCP algorithm does not significantly increase when a moving target is considered (see also [185]).

Comparison With GPOPS-II

The comparison of GPOPS-II and SCP for the transfers to 2000 SG344, Venus, and Dionysus is given in Fig. 6.18. In the bar charts, *SCP* refers to the median of the results obtained with the SCP algorithm when all discretization methods, orders, trust-region parameters and methods, and nodes are considered. *Best SCP* refers instead to the results when the best combination of all parameters (discretization and trust-region methods and polynomial order) is selected for each target. With regard to the 2000 SG344 transfer, the success rate of GPOPS-II is on average higher (approximately +20 %) compared to SCP. This discrepancy, however, reduces to only 5 % if the best SCP method (HLGL or FOH) is considered.

Yet, GPOPS-II often takes twice as long as SCP to find a solution. For this simple transfer, the final masses are nearly the same. The trend is similar with respect to the Venus transfer, albeit the success rate of SCP improves only slightly when choosing the best method (FRPM or RPM). In addition, the difference in the final mass becomes more evident now. Even though the success rate of GPOPS-II is slightly higher for the Dionysus transfer when considering the averaged SCP, *Best SCP* (11th order HLGL) outperforms GPOPS-II in terms of convergence, CPU time, and final mass. Especially the difference in computational effort is remarkable because even the slowest SCP method is several times faster than GPOPS-II. Furthermore, the final masses obtained with SCP are considerably larger. The propagation error is similar for all methods.

In general, GPOPS-II may on average have slightly higher success rates for the considered simulations. However, this is only true for extremely poor initial guesses where the initial constraint violations are large. In that case, SCP is often not able to find feasible solutions due to the linearized dynamics. If a more decent initial guess is provided, the difference becomes negligible. Even though modern nonlinear programming solvers like SNOPT can exploit sparsity, our simulations show that the required computational effort (and thus, CPU time) is still considerably higher compared to SCP. Note that if an upper bound on the CPU time is imposed, SCP would actually outperform GPOPS-II in terms of convergence in many cases, and this might be critical for real-time applications on hardware with limited resources.

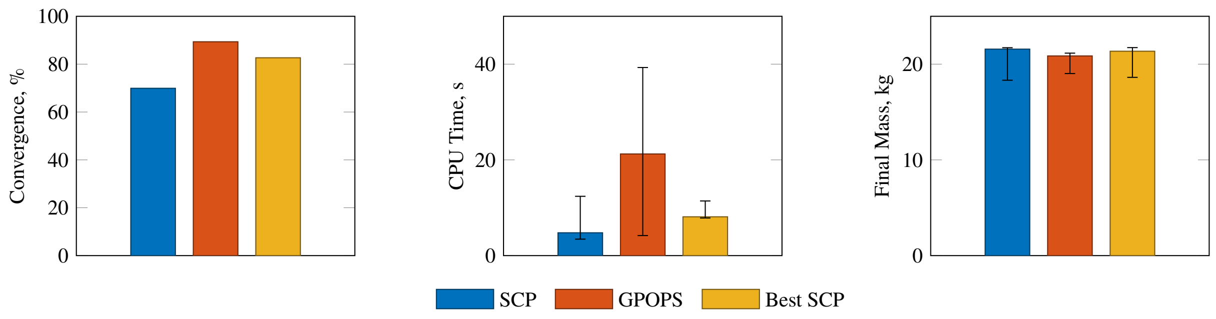
Typical thrust magnitude profiles are illustrated in Fig. 6.19. Note the jittery behavior of the control profiles obtained with GPOPS-II. Apparently, it has difficulties to find decent bang-bang trajectories if the controls are discontinuous [106, 186, 187]. Significant additional refinement would therefore be required to use such profiles on board. In contrast, SCP yields the desired bang-bang structure.

6.2.3 Performance Assessment

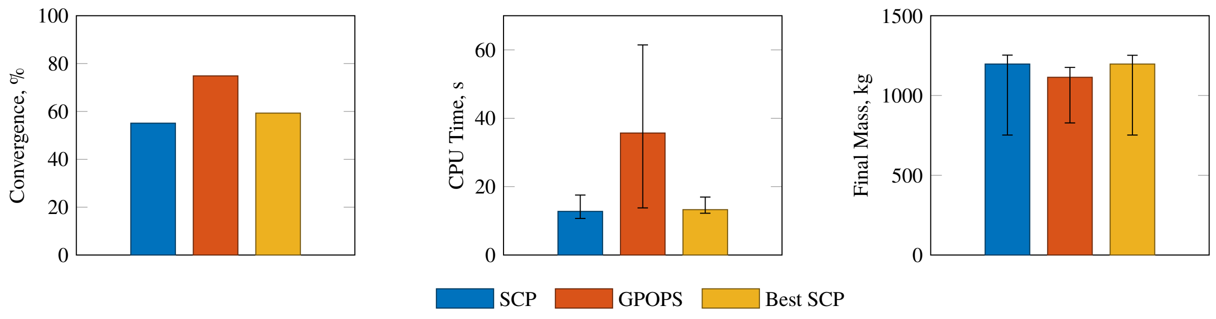
After the presentation of the results in the previous section, the performance of the methods is assessed from a general perspective, and with respect to their use for onboard guidance applications.

General Performance Assessment

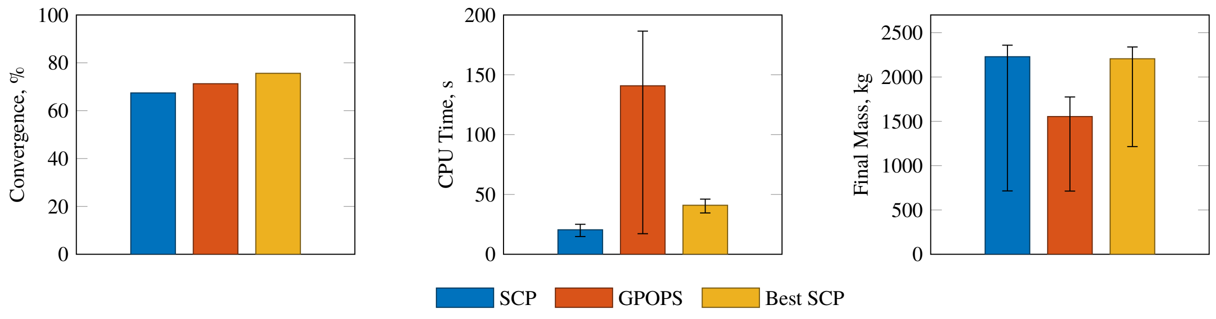
We summarize the general performance of the trust-region and discretization methods in Tables 6.7 and 6.8. The comparison criteria are: *convergence* (how often the method converges), *optimality* (performance index of the solution), *iterations* and *CPU time* to reach convergence, and *thrust profile*, i.e., to what extent a method is capable of accurately capturing the bang-bang structure of the optimal thrust profile. In addition, the discretization methods are also compared in terms of *sparsity* of their equality constraints



(a) Comparison for the transfer to asteroid 2000 SG344.

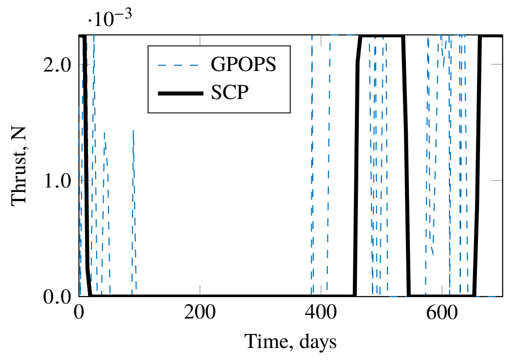


(b) Comparison for the transfer to Venus.

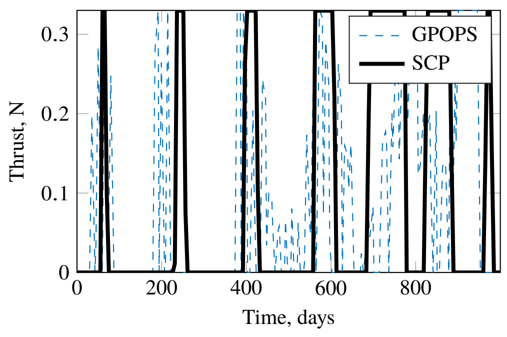


(c) Comparison for the transfer to Dionysus.

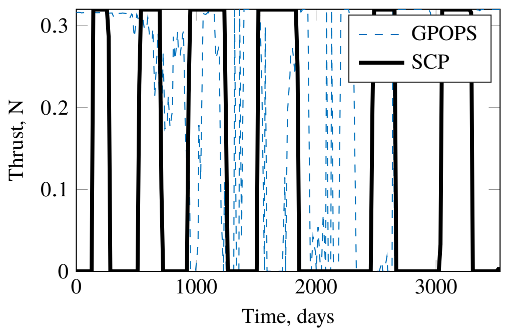
Figure 6.18: Comparison of GPOPS-II and SCP in terms of success rate, iterations, and final mass for the transfers to 2000 SG344, Venus, and Dionysus.



(a) Thrust profiles for the transfer to asteroid 2000 SG344.



(b) Thrust profiles for the transfer to Venus.



(c) Thrust profiles for the transfer to Dionysus.

Figure 6.19: Comparison of the thrust profiles obtained with GPOPS-II and SCP for the transfers to 2000 SG344, Venus, and Dionysus.

matrices, and *accuracy*, i.e., the error on the final boundary conditions when propagating the dynamics with the obtained controls.

In general, hard trust-region methods seem to be preferable due to the lower computational effort as no additional second-order cone constraints are needed when a quadratic (and hence differentiable) penalty function is used. Selecting $\delta > 1$ is often beneficial, and the value can be adjusted depending on the requirements on convergence and speed of the algorithm. Lower values tend to have higher success rates, whereas larger values result in fewer iterations, but also often less accurate control histories. Even though the results indicate that the performance of a discretization method can depend on the transfer, FOH seems to yield the best overall performance given the criteria in Table 6.8.

Table 6.7: Assessment of trust-region methods.

Criterion	Hard TR $\delta = 1.0$	Hard TR $\delta = 1.2$	Soft TR	Comments
Convergence	Good	Acceptable	Good	-
Optimality	Good	Good	Good	No influence of the trust region method
Iterations	Acceptable	Good	Acceptable	-
CPU time	Acceptable	Good	Bad	-
Thrust regularity	Good	Acceptable	Good	-

Table 6.8: Assessment of discretization methods.

Criterion	FOH	HLGL	RPM/FRPM	Comments
Convergence	Good	Good	Acceptable	Worst performance for third-order polyn.
Optimality	Good	Good	Good	No influence of discretization method
Iterations	Good	Acceptable	Acceptable	-
CPU time	Good	Bad	Acceptable	CPU time increases with order of the polyn.
Thrust profile	Good	Acceptable	Acceptable	Worst performance for high-order polyn.
Sparsity	Good	Acceptable	Acceptable	Few hundreds of kilobytes of memory needed
Accuracy	Good	Good	Good	Worst performance for third-order polyn.

Assessment of Onboard Guidance Requirements

There are several requirements for onboard guidance methods. We briefly comment on the results in terms of the onboard guidance requirements *reliability and robustness*, *onboard capability*, *accuracy*, and *optimality* in Table 6.9.

All of the fundamental requirements seem to be satisfied by almost all methods. With regard to the collocation methods, polynomial orders between 7 and 11 are preferable due to the acceptable computational effort and high success rates. As the repeatedly recomputed optimal trajectories in an onboard guidance scenario will not differ considerably, the previous solution can serve as a good initial guess for the next optimization. Therefore, the convergence is expected to increase and computational effort to decrease significantly (see also Chapter 9). Using a compiled language like C or C++ will also decrease the computational time. In addition, the flexibility and also reliability of the algorithm can be increased by incorporating planetary ephemeris for dynamic endpoint targets (see [185] and also Chapter 9). Yet, it is still to be investigated under what conditions convergence can be guaranteed, and how the methods perform on a real spacecraft onboard computer in a real mission scenario.

Table 6.9: Assessment of the methods in terms of high-level onboard guidance requirements.

Requirement	Comments
Reliability and robustness: The algorithm shall have a high success rate.	All discretization and trust-region methods achieve a high success rate. Previous optimal trajectories can be reused in an autonomous guidance scenario, so the convergence is expected to be close to $100\,\%$.
Optimality: The algorithm shall minimize the fuel consumption while respecting the other mission objectives/constraints.	All discretization and trust-region methods fulfil this requirement as they yield similar final masses that are close to the optimal ones found in literature.
Onboard capability: The algorithm shall be compatible with the limited hardware on board.	The obtained CPU times would result in a total computational time of few minutes for typical space-flight processors such as the LEON family (see, e.g., [169] and [188]), therefore being acceptable for deep-space cruise. High sparsity: only few hundreds of kilobytes of memory are required.
Accuracy: The propagation error shall be small; the optimized thrust profile shall have as few oscillations as possible.	All methods achieve a high accuracy. Interpolating the controls results in oscillations for HLGL and RPM/FRPM. The linear control interpolation in FOH seems therefore more suitable.

7 Assessment of State Vector Representations

In this chapter, various state vector representations are assessed. The goal is to investigate how the choice of the coordinates affects the performance of SCP. Therefore, a thorough assessment of the convergence and performance properties of four classical and two non-standard coordinate sets is carried out when poor initial guesses are provided. Moreover, two nonlinearity-like indices tailored to convex optimization are proposed to measure how various state vector representations affect the accuracy of the successive linearization approach within SCP. Parts of this chapter are taken from our work in [14].

7.1 State Vector Representations

There is a wide variety of different coordinate sets that are commonly used in astrodynamics, and in particular for low-thrust trajectory optimization. We consider Cartesian, spherical, and cylindrical coordinates as they are probably the most popular ones. Even though many modifications of the classical orbital elements have been developed over the past decades, previous work suggests that any set of (hybrid) orbital elements often performs similar or worse compared to modified equinoctial elements (MEE) [144]. We therefore restrict ourselves to the non-singular MEE as the only standard Keplerian element set in this dissertation. In addition, a recently developed set of modified orbital elements (MOE) [165] is considered. The main difference compared with other sets in the literature is that the dynamics are linear in the unperturbed case due to a time regularization. Moreover, we also take into account two variants of the non-minimal Kustaanheimo–Stiefel (KS) coordinates [189] that also result in linear or weakly nonlinear unperturbed dynamics. This property is expected to be beneficial for the successive linearization approach in SCP.

In the following subsections, the dynamics of each set are provided in the form

$$\mathbf{f}(\mathbf{x}, \mathbf{u}) = \mathbf{g}(\mathbf{x}) + \mathbf{B}(\mathbf{x})\,\mathbf{u} \tag{7.1}$$

where $\mathbf{g}(\mathbf{x})$ and $\mathbf{B}(\mathbf{x})$ are functions of the state variables $\mathbf{x}$, and $\mathbf{u}$ are the controls. If not stated otherwise, the control components are defined using the unit vectors of the corresponding coordinate frame. As Cartesian, spherical, and cylindrical coordinates along with modified equinoctial elements are well-known in astrodynamics, only a short overview is given (see also the Appendix B for relevant coordinate transformations). As the reader may not be familiar with MOE and KS coordinates, more elaborate derivations are presented.

7.1.1 Cartesian Coordinates

The equations of motion in Eq. (5.11) are considered:

$$\dot{\mathbf{r}} = \mathbf{v} \tag{7.2}$$

$$\dot{\mathbf{v}} = -\frac{\mu}{\|\mathbf{r}\|_2^3}\,\mathbf{r} + \boldsymbol{\tau} \tag{7.3}$$

$$\dot{w} = -\frac{\Gamma}{g_0\, I_{\text{sp}}} \tag{7.4}$$

The state $\mathbf{x} \in \mathbb{R}^7$ and control $\mathbf{u} \in \mathbb{R}^4$ vectors are

$$\mathbf{x}_{\text{cart}} = \left[\mathbf{r}^\top, \mathbf{v}^\top, w\right]^\top = [x, y, z, v_x, v_y, v_z, w]^\top \tag{7.5}$$

$$\mathbf{u}_{\text{cart}} = \left[\boldsymbol{\tau}^\top, \Gamma\right]^\top \tag{7.6}$$

The quantities $\mathbf{g}_{\text{cart}}(\mathbf{x})$ and $\mathbf{B}_{\text{cart}}(\mathbf{x})$ are

$$\mathbf{g}_{\text{cart}}(\mathbf{x}) = \begin{bmatrix} v_x \\ v_y \\ v_z \\ -\frac{\mu\, x}{(x^2+y^2+z^2)^{3/2}} \\ -\frac{\mu\, y}{(x^2+y^2+z^2)^{3/2}} \\ -\frac{\mu\, z}{(x^2+y^2+z^2)^{3/2}} \\ 0 \end{bmatrix}, \quad \mathbf{B}_{\text{cart}}(\mathbf{x}) = \begin{bmatrix} & \mathbf{0}_{3\times 4} & & \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -\frac{1}{g_0\, I_{\text{sp}}} \end{bmatrix} \tag{7.7}$$

7.1.2 Spherical Coordinates

The state $\mathbf{x} \in \mathbb{R}^7$ and control $\mathbf{u} \in \mathbb{R}^4$ vectors are

$$\mathbf{x}_{\text{sph}} = [r, \theta, \phi, v_r, v_\theta, v_\phi, w]^\top \tag{7.8}$$

$$\mathbf{u}_{\text{sph}} = \left[\boldsymbol{\tau}^\top, \Gamma\right]^\top \tag{7.9}$$

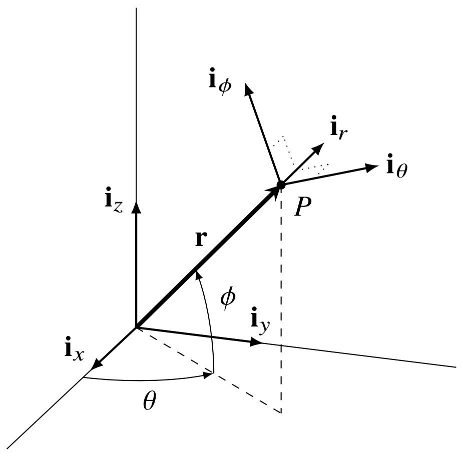


Figure 7.1: Spherical coordinates and rotating coordinate system.

where r, θ, and ϕ denote the radial distance, azimuthal, and polar angle, respectively (see Fig. 7.1). Following the derivations provided in the Appendix C.1, the quantities $\mathbf{g}_{\text{sph}}(\mathbf{x})$ and $\mathbf{B}_{\text{sph}}(\mathbf{x})$ are

$$\mathbf{g}_{\text{sph}}(\mathbf{x}) = \begin{bmatrix} v_r \\ \frac{v_\theta}{r} \\ \frac{v_\phi}{r\sin(\theta)} \\ \frac{v_\theta^2}{r} + \frac{v_\phi^2}{r} - \frac{\mu}{r^2} \\ -\frac{v_r\,v_\theta}{r} + \frac{v_\phi^2\,\cos(\theta)}{r\,\sin(\theta)} \\ -\frac{v_r\,v_\phi}{r} - \frac{v_\theta\,v_\phi\,\cos(\theta)}{r\,\sin(\theta)} \\ 0 \end{bmatrix}, \quad \mathbf{B}_{\text{sph}}(\mathbf{x}) = \begin{bmatrix} & \mathbf{0}_{3\times4} & & \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -\frac{1}{g_0\,I_{\text{sp}}} \end{bmatrix} \tag{7.10}$$

7.1.3 Cylindrical Coordinates

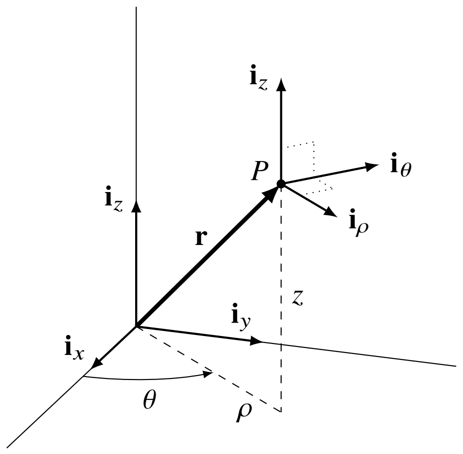


Figure 7.2: Cylindrical coordinates and rotating coordinate system.

The state $\mathbf{x} \in \mathbb{R}^7$ and control $\mathbf{u} \in \mathbb{R}^4$ vectors are

$$\mathbf{x}_{\text{cyl}} = [\rho, \theta, z, v_\rho, v_\theta, v_z, w]^\top \tag{7.11}$$

$$\mathbf{u}_{\text{cyl}} = \left[\boldsymbol{\tau}^\top, \Gamma\right]^\top \tag{7.12}$$

where ρ, θ, and z denote the radial distance, azimuthal angle, and the distance from the reference plane, respectively (see Fig. 7.2). The equations of motion are derived in the Appendix C.2, and the quantities $\mathbf{g}_{\text{cyl}}(\mathbf{x})$ and $\mathbf{B}_{\text{cyl}}(\mathbf{x})$ read

$$\mathbf{g}_{\text{cyl}}(\mathbf{x}) = \begin{bmatrix} v_\rho \\ \frac{v_\theta}{\rho} \\ v_z \\ \frac{v_\theta^2}{\rho} - \frac{\mu\,\rho}{(\rho^2+z^2)^{3/2}} \\ -\frac{v_\rho\, v_\theta}{\rho} \\ -\frac{\mu\, z}{(\rho^2+z^2)^{3/2}} \\ 0 \end{bmatrix}, \quad \mathbf{B}_{\text{cyl}}(\mathbf{x}) = \begin{bmatrix} & & \mathbf{0}_{3\times4} & \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -\frac{1}{g_0\, I_{\text{sp}}} \end{bmatrix} \tag{7.13}$$

7.1.4 Modified Equinoctial Elements

The state $\mathbf{x} \in \mathbb{R}^7$ and control $\mathbf{u} \in \mathbb{R}^4$ vectors are

$$\mathbf{x}_{\text{MEE}} = [p, e_x, e_y, h_x, h_y, l, w]^\top \tag{7.14}$$

$$\mathbf{u}_{\text{MEE}} = \left[\boldsymbol{\tau}^\top, \Gamma\right]^\top \tag{7.15}$$

The relationship between modified equinoctial and classical orbital elements is given by [190]

$$p = a\left(1 - e^2\right) \tag{7.16}$$

$$e_x = e\,\cos\left(\omega + \Omega\right) \tag{7.17}$$

$$e_y = e\,\sin\left(\omega + \Omega\right) \tag{7.18}$$

$$h_x = \tan\left(i/2\right)\,\cos\Omega \tag{7.19}$$

$$h_y = \tan\left(i/2\right)\,\sin\Omega \tag{7.20}$$

$$l = \omega + \Omega + \vartheta \tag{7.21}$$

where p denotes the semilatus rectum, a the semi-major axis, e the eccentricity, ω the argument of periapsis, Ω the longitude of the ascending node, i the inclination, l the true longitude, and ϑ the true anomaly. The quantities $\mathbf{g}_{\mathrm{MEE}}(\mathbf{x})$ and $\mathbf{B}_{\mathrm{MEE}}(\mathbf{x})$ are [148]

$$\mathbf{g}_{\mathrm{MEE}}(\mathbf{x}) = \left[0, 0, 0, 0, 0, \sqrt{\mu\, p}\left(\frac{\sigma}{p}\right)^2\right]^\top \tag{7.22}$$

$$\mathbf{B}_{\mathrm{MEE}}(\mathbf{x}) = \begin{bmatrix} 0 & \frac{2p}{\sigma}\sqrt{\frac{p}{\mu}} & 0 & 0 \\ \sqrt{\frac{p}{\mu}}\sin l & \sqrt{\frac{p}{\mu}}\frac{1}{\sigma}\left[(\sigma+1)\cos l + e_x\right] & -\sqrt{\frac{p}{\mu}}\frac{e_y}{\sigma}\left(h_x \sin l - h_y \cos l\right) & 0 \\ -\sqrt{\frac{p}{\mu}}\cos l & \sqrt{\frac{p}{\mu}}\frac{1}{\sigma}\left[(\sigma+1)\sin l + e_y\right] & \sqrt{\frac{p}{\mu}}\frac{e_x}{\sigma}\left(h_x \sin l - h_y \cos l\right) & 0 \\ 0 & 0 & \sqrt{\frac{p}{\mu}}\frac{b^2}{2\sigma}\cos l & 0 \\ 0 & 0 & \sqrt{\frac{p}{\mu}}\frac{b^2}{2\sigma}\sin l & 0 \\ 0 & 0 & \sqrt{\frac{p}{\mu}}\frac{1}{\sigma}\left(h_x \sin l - h_y \cos l\right) & 0 \\ 0 & 0 & 0 & -\frac{1}{g_0\, I_{\mathrm{sp}}} \end{bmatrix} \tag{7.23}$$

where

$$\sigma = 1 + e_x \cos l + e_y \sin l, \quad b^2 = 1 + h_y^2 + h_y^2 \tag{7.24}$$

7.1.5 Modified Orbital Elements

We start from the Hamiltonian $\mathcal{H}$ in spherical coordinates (the reader is referred to the Appendix B.1 for details)

$$\mathcal{H} = \frac{1}{2}\left(p_r^2 + \frac{p_\phi^2}{r^2} + \frac{p_\theta^2}{r^2 \cos^2\phi}\right) - \frac{\mu}{r} \tag{7.25}$$

where r, ϕ, and θ denote the radial distance, polar and azimuthal angle, respectively, and p_r, p_ϕ, and p_θ are the corresponding conjugate momenta:

$$p_r = \dot{r} \tag{7.26}$$

$$p_\theta = r^2\, \dot{\theta} \cos^2\phi \tag{7.27}$$

$$p_\phi = r^2\, \dot{\phi} \tag{7.28}$$

Applying a time regularization

$$\frac{\mathrm{d}\zeta}{\mathrm{d}t} = \frac{p_h}{r^2} \quad \Longleftrightarrow \quad \frac{\mathrm{d}}{\mathrm{d}t} = \frac{p_h}{r^2}\frac{\mathrm{d}}{\mathrm{d}\zeta} \tag{7.29}$$

with the angular momentum p_h

$$p_h = \sqrt{p_\phi^2 + \frac{p_\theta^2}{\cos^2\phi}} \tag{7.30}$$

and performing a change of variables yields the new set of coordinates $[\Lambda, \eta, s, \gamma, \kappa, \beta]$. Their relationship to spherical coordinates is given by [165]

$$\Lambda = \left(\frac{p_h}{r} - \frac{\mu}{p_h}\right)\sqrt{\frac{C}{\mu}} \tag{7.31}$$

$$\eta = p_r \sqrt{\frac{C}{\mu}} \tag{7.32}$$

$$s = \sin\phi \tag{7.33}$$

$$\gamma = \frac{p_\phi}{p_h}\cos\phi \tag{7.34}$$

$$\kappa = \frac{1}{p_h}\sqrt{C\,\mu} \tag{7.35}$$

$$\beta = \theta - \arcsin\left(\tan\phi\sqrt{\frac{p_\theta^2}{p_h^2 - p_\theta^2}}\right) \tag{7.36}$$

The standard gravitational parameter μ and some length unit C are included to make the quantities dimensionless. β is identical to the classical longitude of the ascending node Ω. The unperturbed equations of motion are then found to be [165]

$$\frac{\mathrm{d}\Lambda}{\mathrm{d}\zeta} = -\eta \tag{7.37a}$$

$$\frac{\mathrm{d}\eta}{\mathrm{d}\zeta} = \Lambda \tag{7.37b}$$

$$\frac{\mathrm{d}s}{\mathrm{d}\zeta} = \gamma \tag{7.37c}$$

$$\frac{\mathrm{d}\gamma}{\mathrm{d}\zeta} = -s \tag{7.37d}$$

$$\frac{\mathrm{d}\kappa}{\mathrm{d}\zeta} = 0 \tag{7.37e}$$

$$\frac{\mathrm{d}\beta}{\mathrm{d}\zeta} = 0 \tag{7.37f}$$

As the new independent variable is the true anomaly, the time can be obtained by integrating the additional equation $\mathrm{d}t/\mathrm{d}\zeta$. Using the relations

$$\cos\phi = \cos(\arcsin s) = \sqrt{1 - s^2} \tag{7.38}$$

$$\dot{\phi} = \frac{\gamma\, p_h}{r^2\cos\phi} \tag{7.39}$$

$$\dot{\theta} = \frac{p_\theta}{r^2\cos^2\phi} \tag{7.40}$$

and

$$r = p_h\left(\Lambda\sqrt{\frac{\mu}{C}} + \frac{\mu}{p_h}\right)^{-1} \tag{7.41}$$

$$p_h = \frac{1}{\kappa}\sqrt{C\,\mu} \tag{7.42}$$

the differential equation of the physical time t reads

$$\frac{\mathrm{d}t}{\mathrm{d}\zeta} = \frac{r^2}{p_h} = \frac{1}{\kappa\,(\kappa+\Lambda)^2}\sqrt{\frac{C^3}{\mu}} \tag{7.43}$$

The equations of motion with a perturbing acceleration $\mathbf{a}_p$ can be determined by computing the partial derivatives of each orbital element $\mathcal{O}$ with respect to the velocity $\mathbf{v}$ [190]:

$$\frac{\mathrm{d}\mathcal{O}}{\mathrm{d}t} = \left.\frac{\mathrm{d}\mathcal{O}}{\mathrm{d}t}\right|_{\text{unpert.}} + \frac{\partial \mathcal{O}}{\partial \mathbf{v}}\,\mathbf{a}_p, \qquad \mathcal{O} \in \{\Lambda, \eta, s, \gamma, \kappa, \beta\} \tag{7.44}$$

where the first term on the right-hand side refers to the unperturbed dynamics in Eqs. (7.37a)–(7.37f), and the second term to the perturbation. Applying Eq. (7.29) and changing the independent variable to ζ results in

$$\frac{\mathrm{d}\mathcal{O}}{\mathrm{d}\zeta} = \left.\frac{\mathrm{d}\mathcal{O}}{\mathrm{d}\zeta}\right|_{\text{unpert.}} + \frac{r^2}{p_h}\,\frac{\mathrm{d}\mathcal{O}}{\mathrm{d}\mathbf{v}}\,\mathbf{a}_p, \qquad \mathcal{O} \in \{\Lambda, \eta, s, \gamma, \kappa, \beta\} \tag{7.45}$$

Therefore, the partial derivatives of each element with respect to the Cartesian velocity $\mathbf{v}$ are needed to compute the equations of motion with a perturbing acceleration. Given the unit vectors $\mathbf{i}_r$, $\mathbf{i}_\theta$, $\mathbf{i}_\phi$ in spherical coordinates, the perturbing acceleration can be written as

$$\mathbf{a}_p = a_r\,\mathbf{i}_r + a_\theta\,\mathbf{i}_\theta + a_\phi\,\mathbf{i}_\phi \tag{7.46}$$

Using the partial derivatives in Eqs. (C.51)–(C.56) in the Appendix C.3, and expressing all quantities in terms of $[\Lambda, \eta, s, \gamma, \kappa, \beta]$, the equations of motion with respect to ζ are found to be

$$\frac{\mathrm{d}\Lambda}{\mathrm{d}\zeta} = -\eta + \frac{\sqrt{1-s^2-\gamma^2}\,(2\kappa+\Lambda)}{\sqrt{1-s^2}\,\kappa\,(\kappa+\Lambda)^3}\,\frac{C^2}{\mu}\,a_\theta + \frac{\gamma\,(2\kappa+\Lambda)}{\sqrt{1-s^2}\,\kappa\,(\kappa+\Lambda)^3}\,\frac{C^2}{\mu}\,a_\phi \tag{7.47a}$$

$$\frac{\mathrm{d}\eta}{\mathrm{d}\zeta} = \Lambda + \frac{1}{\kappa\,(\kappa+\Lambda)^2}\,\frac{C^2}{\mu}\,a_\rho \tag{7.47b}$$

$$\frac{\mathrm{d}\gamma}{\mathrm{d}\zeta} = -s - \frac{\gamma\,\sqrt{1-s^2-\gamma^2}}{\sqrt{1-s^2}\,\kappa\,(\kappa+\Lambda)^3}\,\frac{C^2}{\mu}\,a_\theta + \frac{1-s^2-\gamma^2}{\sqrt{1-s^2}\,\kappa\,(\kappa+\Lambda)^3}\,\frac{C^2}{\mu}\,a_\phi \tag{7.47c}$$

$$\frac{\mathrm{d}s}{\mathrm{d}\zeta} = \gamma \tag{7.47d}$$

$$\frac{\mathrm{d}\kappa}{\mathrm{d}\zeta} = -\frac{\sqrt{1-s^2-\gamma^2}}{\sqrt{1-s^2}\,(\kappa+\Lambda)^3}\,\frac{C^2}{\mu}\,a_\theta - \frac{\gamma}{\sqrt{1-s^2}\,(\kappa+\Lambda)^3}\,\frac{C^2}{\mu}\,a_\phi \tag{7.47e}$$

$$\frac{\mathrm{d}\beta}{\mathrm{d}\zeta} = -\frac{s\,\gamma}{(s^2+\gamma^2)\,\sqrt{1-s^2}\,\kappa\,(\kappa+\Lambda)^3}\,\frac{C^2}{\mu}\,a_\theta + \frac{s\,\sqrt{1-s^2-\gamma^2}}{(s^2+\gamma^2)\,\sqrt{1-s^2}\,\kappa\,(\kappa+\Lambda)^3}\,\frac{C^2}{\mu}\,a_\phi \tag{7.47f}$$

We observe that the EoMs are linear when no perturbation is present, but the perturbing terms seem rather complex. Therefore, we want to express the acceleration vector in the standard local-vertical local-horizontal rotating frame RTN:

$$\mathbf{a}_p = a_r\,\mathbf{i}_r + a_t\,\mathbf{i}_t + a_n\,\mathbf{i}_n \tag{7.48}$$

where the first axis is the radial unit vector $\mathbf{i}_r = \mathbf{r}/r$ that points along the position vector. The normal unit vector $\mathbf{i}_n$ points in the orbit normal direction with $\mathbf{r} \times \mathbf{v} = p_h\, \mathbf{i}_n$. The transversal unit vector $\mathbf{i}_t$ is found with the right-hand rule. The geometry of the inertial and orbital frame is illustrated in Fig. 7.3. The direction cosine matrix $\mathbf{R}_{\text{XYZ}\rightarrow\text{RTN}}$ that transforms the Cartesian inertial frame XYZ into the orbital frame RTN is given by the (3,1,3) Euler angle sequence corresponding to (Ω, i, δ), $\delta = \omega + \vartheta$ being the argument of latitude and ϑ the true anomaly.

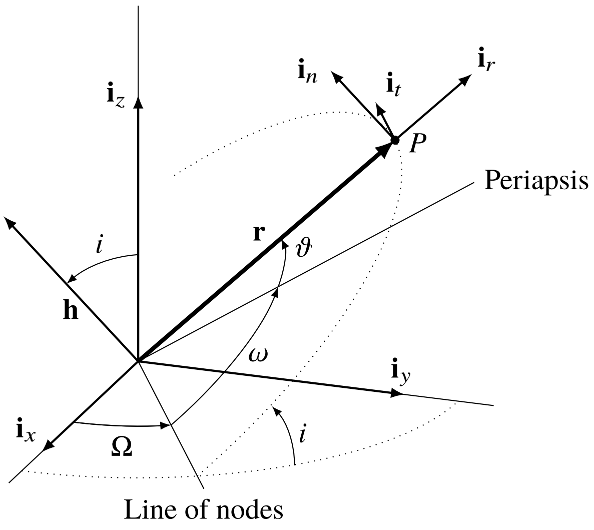


Figure 7.3: Orbital frame and geometry of classical orbital elements.

$$\begin{bmatrix} \mathbf{i}_r \\ \mathbf{i}_t \\ \mathbf{i}_n \end{bmatrix} = \underbrace{\begin{bmatrix} -\sin\Omega\,\cos i\,\sin\delta + \cos\Omega\,\cos\delta & \cos\Omega\,\cos i\,\sin\delta + \sin\Omega\,\cos\delta & \sin i\,\sin\delta \\ -\sin\Omega\,\cos i\,\cos\delta - \cos\Omega\,\sin\delta & \cos\Omega\,\cos i\,\cos\delta - \sin\Omega\,\sin\delta & \sin i\,\cos\delta \\ \sin\Omega\,\sin i & -\cos\Omega\,\sin i & \cos i \end{bmatrix}}_{=:\,\mathbf{R}_{\text{XYZ}\rightarrow\text{RTN}}} \begin{bmatrix} \mathbf{i}_x \\ \mathbf{i}_y \\ \mathbf{i}_z \end{bmatrix} \tag{7.49}$$

The transformation matrix $\mathbf{R}_{\text{XYZ}\rightarrow\text{SPH}}$ to convert Cartesian to spherical coordinates is given by a rotation about the third axis by θ, followed by a rotation about the second axis by $-\phi$:

$$\begin{bmatrix} \mathbf{i}_r \\ \mathbf{i}_\theta \\ \mathbf{i}_\phi \end{bmatrix} = \underbrace{\begin{bmatrix} \cos\theta\,\cos\phi & \sin\theta\,\cos\phi & \sin\phi \\ -\sin\theta & \cos\theta & 0 \\ -\cos\theta\,\sin\phi & -\sin\theta\,\sin\phi & \cos\phi \end{bmatrix}}_{=:\,\mathbf{R}_{\text{XYZ}\rightarrow\text{SPH}}} \begin{bmatrix} \mathbf{i}_x \\ \mathbf{i}_y \\ \mathbf{i}_z \end{bmatrix} \tag{7.50}$$

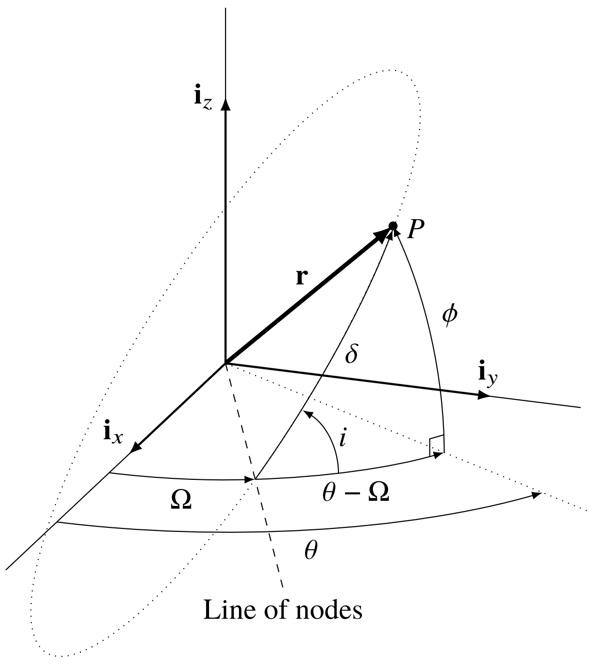


Figure 7.4: Spherical triangle for orbital motion.

The relationship between the spherical and RTN frame is then obtained using

$$\begin{bmatrix} \mathbf{i}_r \\ \mathbf{i}_t \\ \mathbf{i}_n \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos i/\cos\phi & \cos(\theta-\Omega)\sin i \\ 0 & -\cos(\theta-\Omega)\sin i & \cos i/\cos\phi \end{bmatrix}}_{=:\,\mathbf{R}_{\text{SPH}\to\text{RTN}}} \begin{bmatrix} \mathbf{i}_r \\ \mathbf{i}_\theta \\ \mathbf{i}_\phi \end{bmatrix} \tag{7.51}$$

with $\mathbf{R}_{\text{SPH}\to\text{RTN}} = \mathbf{R}_{\text{XYZ}\to\text{RTN}}\,\mathbf{R}^\top_{\text{XYZ}\to\text{SPH}}$, and where the following spherical trigonometric identities were used according to Fig. 7.4:

$$\cos\delta = \cos\phi\,\cos(\theta-\Omega) \tag{7.52a}$$

$$\sin\phi = \sin i\,\sin\delta \tag{7.52b}$$

$$\tan\phi = \tan i\,\sin(\theta-\Omega) \tag{7.52c}$$

Writing the perturbing acceleration as

$$\mathbf{a}_p = a_r\,\mathbf{i}_\mathbf{r} + a_\theta\,\mathbf{i}_\theta + a_\phi\,\mathbf{i}_\phi = a_r\,\mathbf{i}_r + a_t\,\mathbf{i}_t + a_n\,\mathbf{i}_n \tag{7.53}$$

and using the rotation matrix $\mathbf{R}^{\top}_{\text{SPH}\to\text{RTN}}$ in Eq. (7.51), we obtain a relationship between the perturbing acceleration in the spherical and RTN frame:

$$\begin{bmatrix} a_r \\ a_\theta \\ a_\phi \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos i/\cos\phi & -\cos(\theta-\Omega)\sin i \\ 0 & \cos(\theta-\Omega)\sin i & \cos i/\cos\phi \end{bmatrix} \begin{bmatrix} a_r \\ a_t \\ a_n \end{bmatrix} \tag{7.54}$$

Making use of the expressions

$$\sin i = \sqrt{s^2+\gamma^2} \iff i = \arcsin\sqrt{s^2+\gamma^2} \tag{7.55}$$

$$\cos i = \cos\left(\arcsin\sqrt{s^2+\gamma^2}\right) = \sqrt{1-s^2-\gamma^2} \tag{7.56}$$

$$\sin\delta = \frac{\sin\phi}{\sin i} \iff \delta = \arcsin\frac{s}{\sqrt{s^2+\gamma^2}} \tag{7.57}$$

$$\cos(\theta-\Omega) = \frac{\cos\delta}{\cos\phi} = \frac{\gamma}{\sqrt{s^2+\gamma^2}\,\sqrt{1-s^2}} \tag{7.58}$$

the perturbing accelerations can be written as

$$a_r = a_r \tag{7.59}$$

$$a_\theta = \frac{\sqrt{1-s^2-\gamma^2}}{\sqrt{1-s^2}}\,a_t - \frac{\gamma}{\sqrt{1-s^2}}\,a_n \tag{7.60}$$

$$a_\phi = \frac{\gamma}{\sqrt{1-s^2}}\,a_t + \frac{\sqrt{1-s^2-\gamma^2}}{\sqrt{1-s^2}}\,a_n \tag{7.61}$$

Substituting Eqs. (7.59)–(7.61) into Eqs. (7.47a)–(7.47f), the EoMs in the RTN orbital frame read:

$$\frac{\mathrm{d}\Lambda}{\mathrm{d}\zeta} = -\eta + \frac{2\,\kappa+\Lambda}{\kappa\,(\kappa+\Lambda)^3}\,\frac{C^2}{\mu}\,a_s \tag{7.62a}$$

$$\frac{\mathrm{d}\eta}{\mathrm{d}\zeta} = \Lambda + \frac{1}{\kappa\,(\kappa+\Lambda)^2}\,\frac{C^2}{\mu}\,a_r \tag{7.62b}$$

$$\frac{\mathrm{d}\gamma}{\mathrm{d}\zeta} = -s + \frac{\sqrt{1-s^2-\gamma^2}}{\kappa\,(\kappa+\Lambda)^3}\,\frac{C^2}{\mu}\,a_n \tag{7.62c}$$

$$\frac{\mathrm{d}s}{\mathrm{d}\zeta} = \gamma \tag{7.62d}$$

$$\frac{\mathrm{d}\kappa}{\mathrm{d}\zeta} = -\frac{1}{(\kappa+\Lambda)^3}\,\frac{C^2}{\mu}\,a_s \tag{7.62e}$$

$$\frac{\mathrm{d}\beta}{\mathrm{d}\zeta} = \frac{s}{(s^2+\gamma^2)\,\kappa\,(\kappa+\Lambda)^3}\,a_n \tag{7.62f}$$

Optimization Problem

With regard to the implementation, all quantities (including the perturbing accelerations, gravitational acceleration, and the specific impulse) are normalized before solving the optimization problem numerically. Therefore, the scaling factors C^2/μ and $\sqrt{C^3/\mu}$ in Eqs. (7.62a)–(7.62f) and (7.43), respectively, are dropped in the remainder of this dissertation as they are already implicitly included.

Using Eq. (7.43), the differential equation of the modified mass w

$$\frac{\mathrm{d}w}{\mathrm{d}t} = -\frac{\Gamma(t)}{g_0\, I_{\mathrm{sp}}} \tag{7.63}$$

is to be rewritten to account for the new independent variable ζ:

$$\frac{\mathrm{d}w}{\mathrm{d}\zeta} = -\frac{\Gamma(t)}{g_0\, I_{\mathrm{sp}}} \frac{1}{\kappa(\kappa+\Lambda)^2} \tag{7.64}$$

This makes the formerly linear formulation nonlinear. As we want to compute fuel-optimal trajectories with fixed final time, the time t is included as an additional state. The state $\mathbf{x} \in \mathbb{R}^8$ and control $\mathbf{u} \in \mathbb{R}^4$ vectors then read

$$\mathbf{x}_{\mathrm{MOE}} = [\Lambda, \eta, \gamma, s, \kappa, \beta, w, t]^\top \tag{7.65}$$

$$\mathbf{u}_{\mathrm{MOE}} = \left[\boldsymbol{\tau}^\top, \Gamma\right]^\top \tag{7.66}$$

The quantities $\mathbf{g}_{\mathrm{MOE}}(\mathbf{x})$ and $\mathbf{B}_{\mathrm{MOE}}(\mathbf{x})$ are

$$\mathbf{g}_{\mathrm{MOE}}(\mathbf{x}) = \begin{bmatrix} -\eta \\ \Lambda \\ -s \\ \gamma \\ 0 \\ 0 \\ 0 \\ \frac{1}{\kappa\,(\kappa+\Lambda)^2} \end{bmatrix}, \quad \mathbf{B}_{\mathrm{MOE}}(\mathbf{x}) = \begin{bmatrix} 0 & \frac{2\kappa+\Lambda}{\kappa\,(\kappa+\Lambda)^3} & 0 & 0 \\ \frac{1}{\kappa\,(\kappa+\Lambda)^2} & 0 & 0 & 0 \\ 0 & 0 & \frac{\sqrt{1-s^2-\gamma^2}}{\kappa\,(\kappa+\Lambda)^3} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & -\frac{1}{(\kappa+\Lambda)^3} & 0 & 0 \\ 0 & 0 & \frac{s}{(s^2+\gamma^2)\,\kappa\,(\kappa+\Lambda)^3} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{7.67}$$

As the independent variable is not time, the additional differential equation (7.43) is to be integrated to obtain the evolution of the elements with respect to time. For example, this is required for fuel-optimal problems with a fixed time of flight. In this case, the time can be added as an additional state in the optimization process, and the dynamics become nonlinear due to Eq. (7.43).

Remark 7.1. *Similar to classical orbital elements, the right ascension of the ascending node $\beta \equiv \Omega$ is not defined when the inclination is zero. Therefore, the modified orbital elements presented here cannot be used when $i \approx 0$. This can be overcome by defining a different time regularization:*

$$\frac{\mathrm{d}\tilde{\zeta}}{\mathrm{d}t} = \frac{p_h}{r^2 \cos^2\phi} \iff \frac{\mathrm{d}}{\mathrm{d}t} = \frac{p_h}{r^2} \frac{\mathrm{d}}{\mathrm{d}\tilde{\zeta}} \tag{7.68}$$

Even though the resulting unperturbed dynamics are nonlinear, the nonlinear term is small compared to the linear term, and the performance of SCP is expected to behave similarly. Yet, in the context of this chapter, only problems where $i \neq 0$ are considered.

7.1.6 Kustaanheimo–Stiefel Coordinates

Originally, the Kustaanheimo–Stiefel coordinates were developed to study the singularities of the system. Regularization is used to obtain a set of linear differential equations for the unperturbed motion [189]. In this section, the equations of motion for the KS elements are derived for the planar and spatial case.

We recall the general expression for the equations of motion

$$\ddot{\mathbf{r}} + \frac{\mu}{r^3}\,\mathbf{r} = -\frac{\partial V(\mathbf{r})}{\partial \mathbf{r}} + \mathbf{a}_p \tag{7.69}$$

where $r = \|\mathbf{r}\|_2$. $V(\mathbf{r})$ is a perturbing potential, and $\mathbf{a}_p$ a non-conservative perturbing acceleration. Throughout this section, we assume $V(\mathbf{r}) = 0$ and only consider the acceleration $\mathbf{a}_p$, for example due to thrust.

The specific orbital energy $\mathcal{E}$ is defined as

$$\mathcal{E} = \frac{1}{2}\,\|\dot{\mathbf{r}}\|^2 - \frac{\mu}{r} \tag{7.70}$$

Using the relation

$$\frac{\mathrm{d}}{\mathrm{d}t}\|\mathbf{y}\|^2 = \frac{\mathrm{d}}{\mathrm{d}t}\left(\mathbf{y}^\top \mathbf{y}\right) = 2\,\mathbf{y}^\top \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{y} \tag{7.71}$$

for any arbitrary vector $\mathbf{y}$, the time derivative of Eq. (7.70) is

$$\frac{\mathrm{d}\mathcal{E}}{\mathrm{d}t} = \dot{\mathbf{r}}^\top \ddot{\mathbf{r}} + \frac{\mu}{r^2}\,\dot{r} = \mathbf{v}^\top \dot{\mathbf{v}} + \frac{\mu}{r^2}\,\dot{r} \tag{7.72}$$

Substituting the equations of motion $\dot{\mathbf{r}} = \mathbf{v}$ and $\dot{\mathbf{v}} = -\mu/r^3 + \mathbf{a}_p$ into Eq. (7.72) yields

$$\frac{\mathrm{d}\mathcal{E}}{\mathrm{d}t} = \mathbf{v}^\top \left(-\frac{\mu}{r^3}\,r + \mathbf{a}_p\right) + \frac{\mu}{r^2}\,\dot{r} \tag{7.73}$$

It follows from

$$\frac{\mathrm{d}}{\mathrm{d}t}(\mathbf{r}^\top \mathbf{r}) = \frac{\mathrm{d}r^2}{\mathrm{d}t} = 2\,r\,\dot{r} \tag{7.74}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}(\mathbf{r}^\top \mathbf{r}) = \mathbf{r}^\top \dot{\mathbf{r}} + \dot{\mathbf{r}}^\top \mathbf{r} = 2\,\dot{\mathbf{r}}^\top \mathbf{r} \tag{7.75}$$

that

$$\mathbf{r}^\top \dot{\mathbf{r}} = r\,\dot{r} \tag{7.76}$$

This relation is used to simplify Eq. (7.73) to

$$\begin{aligned}\frac{\mathrm{d}\mathcal{E}}{\mathrm{d}t} &= -\frac{\mu}{r^3}\,\mathbf{v}^\top\mathbf{r} + \mathbf{v}^\top\mathbf{a}_p + \frac{\mu}{r^2}\,\dot{r} = -\frac{\mu}{r^3}\,r\,\dot{r} + \mathbf{v}^\top\mathbf{a}_p + \frac{\mu}{r^2}\,\dot{r} \\ &= \mathbf{v}^\top\mathbf{a}_p = \dot{\mathbf{r}}^\top\mathbf{a}_p\end{aligned} \tag{7.77}$$

For later use, we define h as the negative specific energy of the orbit:

$$h \coloneqq -\mathcal{E} = \frac{\mu}{r} - \frac{1}{2}\,\|\dot{\mathbf{r}}\|^2 \tag{7.78}$$

Planar Case

We introduce a time regularization with the new independent variable ξ [189]:

$$\mathrm{d}t = \frac{r}{c}\,\mathrm{d}\xi \tag{7.79}$$

This fictitious time ξ can be thought of an angle-like variable that represents an orbit anomaly if c is chosen appropriately. In this dissertation, $c = 1$ and $c = \sqrt{2\,h}$ are used that transform the equations of motion into a harmonic oscillator. Setting $c = 1$, Eq. (7.79) can be rewritten as

$$\frac{\mathrm{d}}{\mathrm{d}t} = \frac{1}{r}\frac{\mathrm{d}}{\mathrm{d}\xi} \tag{7.80}$$

Using the chain rule gives

$$\begin{aligned}\frac{\mathrm{d}^2}{\mathrm{d}t^2} &= \frac{\mathrm{d}}{\mathrm{d}t}\,\frac{\mathrm{d}}{\mathrm{d}t} = r^{-1}\frac{\mathrm{d}}{\mathrm{d}\xi}\left(r^{-1}\frac{\mathrm{d}}{\mathrm{d}\xi}\right) = r^{-1}\left(-r^{-2}\frac{\mathrm{d}r}{\mathrm{d}\xi}\frac{\mathrm{d}}{\mathrm{d}\xi} + r^{-1}\frac{\mathrm{d}^2}{\mathrm{d}\xi^2}\right) \\ &= -r^{-3}r'\frac{\mathrm{d}}{\mathrm{d}\xi} + r^{-2}\frac{\mathrm{d}^2}{\mathrm{d}\xi^2}\end{aligned} \tag{7.81}$$

where

$$(\cdot)' \coloneqq \frac{\mathrm{d}}{\mathrm{d}\xi} \tag{7.82}$$

denotes the differentiation with respect to ξ. Substituting Eqs. (7.80) and (7.81) into Eq. (7.69) and rearranging terms yields the equations of motion in the ξ domain:

$$r\,\mathbf{r}'' - r'\,\mathbf{r}' + \mu\,\mathbf{r} = r^3\,\mathbf{a}_p \tag{7.83}$$

For planar motion, we introduce the complex coordinates

$$\mathbf{r} = x + \mathrm{i}\,y \tag{7.84}$$

that satisfy

$$x = a\,(\cos E - e) \tag{7.85}$$

$$y = a\sqrt{1 - e^2}\,\sin E \tag{7.86}$$

where a is the semi-major axis, e the eccentricity, and E the eccentric anomaly. Taking the derivative with respect to time gives

$$\dot{x} = -a\,\dot{E}\,\sin E \tag{7.87}$$

$$\dot{y} = a\sqrt{1-e^2}\,\dot{E}\,\cos E \tag{7.88}$$

with

$$\dot{E} = \sqrt{\frac{\mu}{a}}\,\frac{1}{r} \tag{7.89}$$

$$r = \frac{b}{1+e\,\cos\vartheta} \tag{7.90}$$

$$b = a\left(1-e^2\right) \tag{7.91}$$

and the true anomaly ϑ. Introducing the complex coordinates

$$\mathbf{p} = p_1 + \mathrm{i}\,p_2 \tag{7.92}$$

in the parametric plane p_1, p_2, the following relationship describes the mapping onto the physical plane x, y [189]:

$$x + \mathrm{i}\,y = (p_1 + \mathrm{i}\,p_2)^2 \tag{7.93}$$

$$r := \|\mathbf{r}\|_2 = \mathbf{p}^\top\mathbf{p} = \|\mathbf{p}\|_2^2 \tag{7.94}$$

This regularization is called Levi-Civita transformation and given by

$$x = p_1^2 + p_2^2 \tag{7.95}$$

$$y = 2\,p_1\,p_2 \tag{7.96}$$

or equivalently

$$\mathbf{r} = \mathbf{L}(\mathbf{p})\,\mathbf{p} \tag{7.97}$$

with

$$\mathbf{L}(\mathbf{p}) = \begin{bmatrix} p_1 & -p_2 \\ p_2 & p_1 \end{bmatrix} \tag{7.98}$$

Using Eq. (7.97) and

$$\mathbf{r}' = 2\,\mathbf{L}(\mathbf{p})\,\mathbf{p}' \tag{7.99}$$

$$\mathbf{r}'' = 2\,\mathbf{L}(\mathbf{p}')\,\mathbf{p}' + 2\,\mathbf{L}(\mathbf{p})\,\mathbf{p}'' \tag{7.100}$$

$$r' = 2\,\mathbf{p}^\top\mathbf{p}' \tag{7.101}$$

the equation of motion in Eq. (7.83) can be rewritten in terms of $\mathbf{p}$:

$$2\,\|\mathbf{p}\|^2\,\mathbf{L}(\mathbf{p})\,\mathbf{p}'' + 2\,\|\mathbf{p}\|^2\,\mathbf{L}(\mathbf{p}')\,\mathbf{p}' - 4\,\mathbf{p}^\top\mathbf{p}'\,\mathbf{L}(\mathbf{p})\,\mathbf{p}' + \mu\,\mathbf{L}(\mathbf{p})\,\mathbf{p} = \|\mathbf{p}\|^6\,\mathbf{a}_p \tag{7.102}$$

Making use of

$$2\,\|\mathbf{p}\|^2\,\mathbf{L}(\mathbf{p}')\,\mathbf{p}' - 4\,\mathbf{p}^\top\mathbf{p}'\,\mathbf{L}(\mathbf{p})\,\mathbf{p}' = -2\,\mathbf{p}'^\top\,\mathbf{p}'\,\mathbf{L}(\mathbf{p})\,\mathbf{p} \tag{7.103}$$

and left-multiplying by $[\mathbf{L}(\mathbf{p})]^{-1} = 1/r\,[\mathbf{L}(\mathbf{p})]^\top = 1/\|\mathbf{p}\|^2\,[\mathbf{L}(\mathbf{p})]^\top$ gives

$$2\,\|\mathbf{p}\|^2\,\mathbf{p}'' + \left(\mu - 2\,\|\mathbf{p}'\|^2\right)\,\mathbf{p} = \|\mathbf{p}\|^4\,[\mathbf{L}(\mathbf{p})]^\top\,\mathbf{a}_p \tag{7.104}$$

The second term can be expressed in terms of h. The negative specific energy is

$$\begin{aligned} h &= \frac{\mu}{r} - \frac{1}{2}\,\|\frac{1}{r}\,\frac{\mathrm{d}}{\mathrm{d}\xi}\mathbf{r}\|^2 = \frac{\mu}{r} - \frac{1}{2}\,\frac{1}{r^2}\,\|\mathbf{x}'\|^2 \\ &= \frac{\mu - 2\,\|\mathbf{p}'\|^2}{\|\mathbf{p}\|^2} \end{aligned} \tag{7.105}$$

where we used

$$\|\mathbf{r}'\|^2 = [\mathbf{r}']^\top\mathbf{r}' = 4\,\|\mathbf{p}\|^2\,\|\mathbf{p}'\|^2 \tag{7.106}$$

Substituting Eq. (7.105) into Eq. (7.104) and simplifying yields the equation of motion with respect to $\mathbf{p}$ and ξ:

$$2\,\mathbf{p}'' + h\,\mathbf{p} = \|\mathbf{p}\|^2\,[\mathbf{L}(\mathbf{p})]^\top\,\mathbf{a}_p \tag{7.107}$$

This results in the following system of differential equations:

$$\frac{\mathrm{d}\mathbf{p}}{\mathrm{d}\xi} = \mathbf{p}' \tag{7.108a}$$

$$\frac{\mathrm{d}\mathbf{p}'}{\mathrm{d}\xi} = -\frac{h}{2}\,\mathbf{p} + \frac{1}{2}\,\|\mathbf{p}\|_2^2\,[\mathbf{L}(\mathbf{p})]^\top\,\mathbf{a}_p \tag{7.108b}$$

$$\frac{\mathrm{d}h}{\mathrm{d}\xi} = -2\,\mathbf{p}'^\top[\mathbf{L}(\mathbf{p})]^\top\,\mathbf{a}_p \tag{7.108c}$$

$$\frac{\mathrm{d}t}{\mathrm{d}\xi} = \|\mathbf{p}\|_2^2 \tag{7.108d}$$

where Eqs. (7.77), (7.80) and (7.99) were used to obtain the rate of change of h.

The unperturbed dynamics in Eqs. (7.108a)–(7.108c) are considered to be perturbed linear or weakly linear, and thus are expected to perform better under linearization. Even though the term $-h/2\,\mathbf{p}$ in Eq. (7.108b) is nonlinear, $\mathrm{d}\mathbf{p}'/(\mathrm{d}\xi)$ is expected to behave nearly linearly due to the only slowly varying variable h. As we are interested in spacecraft equipped with low-thrust propulsion systems, it is reasonable to assume that the thrust is a small perturbation compared to the relatively large acceleration caused by the gravitation of the central body. Similar to MOE, the differential equation (7.108d) is to be integrated to obtain the evolution of the time.

It is possible to obtain linear unperturbed dynamics by setting $c = \sqrt{2\,h}$, i.e.,

$$\frac{\mathrm{d}}{\mathrm{d}t} = \frac{\sqrt{2\,h}}{r}\,\frac{\mathrm{d}}{\mathrm{d}E} \tag{7.109}$$

Expressing this in terms of ξ gives

$$\frac{\mathrm{d}}{\mathrm{d}\xi} = \sqrt{2\,h}\,\frac{\mathrm{d}}{\mathrm{d}E} \tag{7.110}$$

Defining

$$\overset{\circ}{(\cdot)} \coloneqq \frac{\mathrm{d}}{\mathrm{d}E} \tag{7.111}$$

the second derivative is then

$$\begin{aligned}\frac{\mathrm{d}^2}{\mathrm{d}\xi^2} &= \frac{\mathrm{d}}{\mathrm{d}\xi}\,\frac{\mathrm{d}}{\mathrm{d}\xi} = \sqrt{2\,h}\,\frac{\mathrm{d}}{\mathrm{d}E}\left(\sqrt{2\,h}\,\frac{\mathrm{d}}{\mathrm{d}E}\right) = \sqrt{2\,h}\left(\frac{\mathrm{d}}{\mathrm{d}E}\sqrt{2\,h}\,\frac{\mathrm{d}}{\mathrm{d}E} + \sqrt{2\,h}\,\frac{\mathrm{d}^2}{\mathrm{d}E^2}\right)\\ &= \overset{\circ}{h}\,\frac{\mathrm{d}}{\mathrm{d}E} + 2\,h\,\frac{\mathrm{d}^2}{\mathrm{d}E^2}\end{aligned} \tag{7.112}$$

because

$$\frac{\mathrm{d}}{\mathrm{d}E}\sqrt{2\,h} = \frac{\sqrt{2}}{2\sqrt{h}}\,\overset{\circ}{h} \tag{7.113}$$

Substituting Eq. (7.112) into Eq. (7.107) allows us to rewrite the equation of motion in terms of the eccentric anomaly:

$$4\,h\,\overset{\circ\circ}{\mathbf{p}} + h\,\mathbf{p} + 2\,\overset{\circ}{h}\,\overset{\circ}{\mathbf{p}} = \|\mathbf{p}\|^2\,[\mathbf{L}(\mathbf{p})]^\top\,\mathbf{a}_p \tag{7.114}$$

Substituting

$$\overset{\circ}{h} = -\overset{\circ}{\mathbf{x}}^\top\mathbf{a}_p = -2\,\overset{\circ}{\mathbf{p}}^\top[\mathbf{L}(\mathbf{p})]^\top\mathbf{a}_p \tag{7.115}$$

into Eq. (7.114) and rearranging terms gives:

$$4\,\overset{\circ\circ}{\mathbf{p}} + \mathbf{p} = \frac{1}{h}\left[\|\mathbf{p}\|^2\,[\mathbf{L}(\mathbf{p})]^\top\,\mathbf{a}_p + 4\,\overset{\circ}{\mathbf{p}}^\top[\mathbf{L}(\mathbf{p})]^\top\mathbf{a}_p\,\overset{\circ}{\mathbf{p}}\right] \tag{7.116}$$

The equations of motion with respect to E are then

$$\frac{\mathrm{d}\mathbf{p}}{\mathrm{d}E} = \overset{\circ}{\mathbf{p}} \tag{7.117a}$$

$$\frac{\mathrm{d}\overset{\circ}{\mathbf{p}}}{\mathrm{d}E} = -\frac{1}{4}\,\mathbf{p} + \frac{1}{4\,h}\left[\|\mathbf{p}\|^2\,[\mathbf{L}(\mathbf{p})]^\top\,\mathbf{a}_p + 4\,\overset{\circ}{\mathbf{p}}^\top[\mathbf{L}(\mathbf{p})]^\top\mathbf{a}_p\,\overset{\circ}{\mathbf{p}}\right] \tag{7.117b}$$

$$\frac{\mathrm{d}h}{\mathrm{d}E} = -2\,\overset{\circ}{\mathbf{p}}^\top[\mathbf{L}(\mathbf{p})]^\top\mathbf{a}_p \tag{7.117c}$$

$$\frac{\mathrm{d}t}{\mathrm{d}E} = \frac{\|\mathbf{p}\|_2^2}{\sqrt{2\,h}} \tag{7.117d}$$

Three-Dimensional Case

It was found that a similar regularization in three dimensions is not possible. Therefore, a mapping from the physical $\mathbf{r} = [x, y, z]^\top \in \mathbb{R}^3$ to a four-dimensional parametric space $\mathbf{p} = [p_1, p_2, p_3, p_4]^\top \in \mathbb{R}^4$ was proposed to overcome the difficulties. The tranformation is given by [189]

$$x = p_1^2 - p_2^2 - p_3^2 + p_4^2 \tag{7.118}$$

$$y = 2\,(p_1\,p_2 - p_3\,p_4) \tag{7.119}$$

$$z = 2\,(p_1\,p_3 + p_2\,p_4) \tag{7.120}$$

The transformation matrix $\mathbf{L}(\mathbf{p})$ is

$$\mathbf{L}(\mathbf{p}) = \begin{bmatrix} p_1 & -p_2 & -p_3 & p_4 \\ p_2 & p_1 & -p_4 & -p_3 \\ p_3 & p_4 & p_1 & p_2 \\ p_4 & -p_3 & p_2 & -p_1 \end{bmatrix} \tag{7.121}$$

If ξ is the independent variable, the velocities $\dot{x}, \dot{y}, \dot{z}$ are obtained using

$$\dot{\mathbf{r}} = \frac{2}{||\mathbf{p}||^2}\,\mathbf{L}(\mathbf{p})\,\mathbf{p}' \tag{7.122}$$

which yields

$$\dot{x} = \frac{2}{||\mathbf{p}||^2}\,\left(p_1\,p_1' - p_2\,p_2' - p_3\,p_3' + p_4\,p_4'\right) \tag{7.123}$$

$$\dot{y} = \frac{2}{||\mathbf{p}||^2}\,\left(p_2\,p_1' + p_1\,p_2' - p_4\,p_3' - p_3\,p_4'\right) \tag{7.124}$$

$$\dot{z} = \frac{2}{||\mathbf{p}||^2}\,\left(p_3\,p_1' + p_4\,p_2' + p_1\,p_3' + p_2\,p_4'\right) \tag{7.125}$$

With regard to E, we obtain

$$\dot{\mathbf{r}} = \frac{2\sqrt{2\,h}}{||\mathbf{p}||^2}\,\mathbf{L}(\mathbf{p})\,\mathring{\mathbf{p}} \tag{7.126}$$

and therefore

$$\dot{x} = \frac{2\sqrt{2\,h}}{||\mathbf{p}||^2}\,(p_1\,\mathring{p}_1 - p_2\,\mathring{p}_2 - p_3\,\mathring{p}_3 + p_4\,\mathring{p}_4) \tag{7.127}$$

$$\dot{y} = \frac{2\sqrt{2\,h}}{||\mathbf{p}||^2}\,(p_2\,\mathring{p}_1 + p_1\,\mathring{p}_2 - p_4\,\mathring{p}_3 - p_3\,\mathring{p}_4) \tag{7.128}$$

$$\dot{z} = \frac{2\sqrt{2\,h}}{||\mathbf{p}||^2}\,(p_3\,\mathring{p}_1 + p_4\,\mathring{p}_2 + p_1\,\mathring{p}_3 + p_2\,\mathring{p}_4) \tag{7.129}$$

The inverse transformation leaves one degree of freedom undetermined. Therefore, if $x \geq 0$, p_1 and p_4 can be chosen such that [189]

$$p_1^2 + p_4^2 = \frac{1}{2}\,(x + r) \tag{7.130}$$

holds. It is then possible to solve for p_2 and p_3:

$$p_2 = \frac{y\,p_1 + z\,p_4}{x + r} \tag{7.131}$$

$$p_3 = \frac{z\,p_1 - y\,p_4}{x + r} \tag{7.132}$$

If $x < 0$, it is convenient to use the following inverse transformation [189]:

$$p_2^2 + p_3^2 = \frac{1}{2}\,(r - x) \tag{7.133}$$

$$p_1 = \frac{y\,p_2 + z\,p_3}{r - x} \tag{7.134}$$

$$p_2 = \frac{z\,p_2 - y\,p_3}{r - x} \tag{7.135}$$

The equations of motion are the same as in Eq. (7.117) with the only difference that the perturbing acceleration is to be augmented to be consistent with the four-dimensional space:

$$\mathbf{a}_p \longrightarrow \begin{bmatrix} \mathbf{a}_p \\ 0 \end{bmatrix} \tag{7.136}$$

Optimization Problem

The differential equation of w

$$\frac{\mathrm{d}w}{\mathrm{d}t} = -\frac{\Gamma(t)}{g_0\,I_{\text{sp}}} \tag{7.137}$$

is to be modified to account for the different independent variables:

$$\frac{\mathrm{d}w}{\mathrm{d}\xi} = -\frac{\Gamma(t)}{g_0\,I_{\text{sp}}}\,||\mathbf{p}||_2^2 \tag{7.138}$$

$$\frac{\mathrm{d}w}{\mathrm{d}E} = -\frac{\Gamma(t)}{g_0\,I_{\text{sp}}}\,\frac{||\mathbf{p}||_2^2}{\sqrt{2\,h}} \tag{7.139}$$

This makes the formerly linear formulation nonlinear. As we want to compute fuel-optimal trajectories with fixed final time, the time t is included as an additional state. The state $\mathbf{x} \in \mathbb{R}^{11}$ and control $\mathbf{u} \in \mathbb{R}^4$ vectors then read

$$\mathbf{x}_{\text{KS}}^{\xi} = \left[\mathbf{p}^\top, \mathbf{p}'^\top, h, w, t\right]^\top,\; \mathbf{x}_{\text{KS}}^{E} = \left[\mathbf{p}^\top, \mathring{\mathbf{p}}^\top, h, w, t\right]^\top \tag{7.140}$$

$$\mathbf{u}_{\text{KS}} = \mathbf{u}_{\text{KS}}^{\xi} = \mathbf{u}_{\text{KS}}^{E} = \left[\boldsymbol{\tau}^\top, \Gamma\right]^\top \tag{7.141}$$

where the superscripts ξ and E refer to the independent variables.

For the planar case, the inverse KS transformation is unique, and the final boundary condition is simply a linear equality constraint. In the spatial case, however, there is an additional degree of freedom. Therefore, one element of $\mathbf{p}$ can be chosen arbitrarily. This means that the previously fixed final state $\mathbf{p}(t_f)$ depends on the initial condition $\mathbf{p}(t_0)$ and the controls, and its value is to be obtained by integrating the dynamics. As a consequence, the final boundary condition is not a linear equality constraint anymore, but a nonlinear function of the initial condition. In discretized form, the final state $\left[[\mathbf{p}(t_f)]^\top, [\mathbf{p}'(t_f)]^\top\right]^\top$ and $\left[[\mathbf{p}(t_f)]^\top, [\mathring{\mathbf{p}}(t_f)]^\top\right]^\top$, respectively, can be mapped from KS to Cartesian coordinates, and the target state can be imposed in Cartesian coordinates:

$$\begin{bmatrix} \mathbf{r}_f \\ \mathbf{v}_f \end{bmatrix} = \mathcal{M}^{\xi}[\mathbf{p}(t_f), \mathbf{p}'(t_f)] \tag{7.142a}$$

$$\begin{bmatrix} \mathbf{r}_f \\ \mathbf{v}_f \end{bmatrix} = \mathcal{M}^{E}[\mathbf{p}(t_f), \mathring{\mathbf{p}}(t_f)] \tag{7.142b}$$

where $\mathcal{M}^j$, $j \in \{\xi, E\}$, denotes the nonlinear mapping from KS to Cartesian coordinates according to Eqs. (7.97), (7.122) and (7.126), respectively. The new final boundary conditions then read

$$\mathbf{r}_f = \mathbf{L}[\mathbf{p}(t_f)]\,\mathbf{p}(t_f), \quad \mathbf{v}_f = \frac{2}{\|\mathbf{p}(t_f)\|^2}\,\mathbf{L}[\mathbf{p}(t_f)]\,\mathbf{p}'(t_f) \tag{7.143a}$$

$$\mathbf{r}_f = \mathbf{L}[\mathbf{p}(t_f)]\,\mathbf{p}(t_f), \quad \mathbf{v}_f = \frac{2\sqrt{2h}}{\|\mathbf{p}(t_f)\|^2}\,\mathbf{L}[\mathbf{p}(t_f)]\,\mathring{\mathbf{p}}(t_f) \tag{7.143b}$$

These constraints are nonconvex and thus linearized about the reference $(\bar{\mathbf{p}}, \bar{\mathbf{p}}')$ or $(\bar{\mathbf{p}}, \mathring{\bar{\mathbf{p}}})$ according to Section 5.1. Defining

$$\psi^{\xi}[\mathbf{p}(t_f), \mathbf{p}'(t_f)] := \begin{bmatrix} \mathbf{L}[\mathbf{p}(t_f)]\,\mathbf{p}(t_f) \\ \dfrac{2}{\|\mathbf{p}(t_f)\|^2}\,\mathbf{L}[\mathbf{p}(t_f)]\,\mathbf{p}'(t_f) \end{bmatrix} \tag{7.144a}$$

$$\psi^{E}[\mathbf{p}(t_f), \mathring{\mathbf{p}}(t_f)] := \begin{bmatrix} \mathbf{L}[\mathbf{p}(t_f)]\,\mathbf{p}(t_f) \\ \dfrac{2\sqrt{2h}}{\|\mathbf{p}(t_f)\|^2}\,\mathbf{L}[\mathbf{p}(t_f)]\,\mathring{\mathbf{p}}(t_f) \end{bmatrix} \tag{7.144b}$$

the linearized constraints are then

$$\begin{bmatrix}\mathbf{r}_f\\ \mathbf{v}_f\end{bmatrix} = \boldsymbol{\psi}^\xi[\bar{\mathbf{p}}(t_f),\bar{\mathbf{p}}'(t_f)] + \nabla\boldsymbol{\psi}^\xi[\bar{\mathbf{p}}(t_f),\bar{\mathbf{p}}'(t_f)]\begin{bmatrix}\mathbf{p}(t_f)-\bar{\mathbf{p}}(t_f)\\ \mathbf{p}'(t_f)-\bar{\mathbf{p}}'(t_f)\end{bmatrix} + \boldsymbol{\nu}_{\text{KS}} \tag{7.145a}$$

$$\begin{bmatrix}\mathbf{r}_f\\ \mathbf{v}_f\end{bmatrix} = \boldsymbol{\psi}^E[\bar{\mathbf{p}}(t_f),\mathring{\bar{\mathbf{p}}}(t_f)] + \nabla\boldsymbol{\psi}^E[\bar{\mathbf{p}}(t_f),\mathring{\bar{\mathbf{p}}}(t_f)]\begin{bmatrix}\mathbf{p}(t_f)-\bar{\mathbf{p}}(t_f)\\ \mathring{\mathbf{p}}(t_f)-\mathring{\bar{\mathbf{p}}}(t_f)\end{bmatrix} + \boldsymbol{\nu}_{\text{KS}} \tag{7.145b}$$

A virtual control $\boldsymbol{\nu}_{\text{KS}} \in \mathbb{R}^6$ is added and penalized in the objective function as $\lambda_{\text{KS}}\,\|\boldsymbol{\nu}_{\text{KS}}\|_1$.

Remark 7.2. *In case of relaxed final boundary conditions*

$$\left|\boldsymbol{\psi}^\xi[\mathbf{p}(t_f),\mathbf{p}'(t_f)] - \begin{bmatrix}\mathbf{r}_f\\ \mathbf{v}_f\end{bmatrix}\right| \le \begin{bmatrix}\Delta\mathbf{r}\\ \Delta\mathbf{v}\end{bmatrix} \tag{7.146}$$

with arbitrary $\Delta\mathbf{r}$ and $\Delta\mathbf{v}$, the linearized constraints read

$$\left|\boldsymbol{\psi}^\xi[\bar{\mathbf{p}}(t_f),\bar{\mathbf{p}}'(t_f)] + \nabla\boldsymbol{\psi}^\xi[\bar{\mathbf{p}}(t_f),\bar{\mathbf{p}}'(t_f)]\begin{bmatrix}\mathbf{p}(t_f)-\bar{\mathbf{p}}(t_f)\\ \mathbf{p}'(t_f)-\bar{\mathbf{p}}'(t_f)\end{bmatrix} - \begin{bmatrix}\mathbf{r}_f\\ \mathbf{v}_f\end{bmatrix}\right| \le \begin{bmatrix}\Delta\mathbf{r}\\ \Delta\mathbf{v}\end{bmatrix} + \boldsymbol{\nu}_{\text{KS}} \tag{7.147}$$

The expressions for E are obtained accordingly.

If ξ is the independent variable, the quantities $\mathbf{g}^\xi_{\text{KS}}(\mathbf{x})$ and $\mathbf{B}^\xi_{\text{KS}}(\mathbf{x})$ are

$$\mathbf{g}^\xi_{\text{KS}}(\mathbf{x}) = \begin{bmatrix}\mathbf{p}'\\ -\frac{h}{2}\,\mathbf{p}\\ 0\\ 0\\ \|\mathbf{p}\|^2\end{bmatrix}, \quad \mathbf{B}^\xi_{\text{KS}}(\mathbf{x}) = \begin{bmatrix} & \mathbf{0}_{4\times4} & & \\ \frac{p_1}{2}\|\mathbf{p}\|^2 & \frac{p_2}{2}\|\mathbf{p}\|^2 & \frac{p_3}{2}\|\mathbf{p}\|^2 & 0\\ -\frac{p_2}{2}\|\mathbf{p}\|^2 & \frac{p_1}{2}\|\mathbf{p}\|^2 & \frac{p_4}{2}\|\mathbf{p}\|^2 & 0\\ -\frac{p_3}{2}\|\mathbf{p}\|^2 & -\frac{p_4}{2}\|\mathbf{p}\|^2 & \frac{p_1}{2}\|\mathbf{p}\|^2 & 0\\ \frac{p_4}{2}\|\mathbf{p}\|^2 & -\frac{p_3}{2}\|\mathbf{p}\|^2 & \frac{p_2}{2}\|\mathbf{p}\|^2 & 0\\ B^\xi_{\text{KS},91} & B^\xi_{\text{KS},92} & B^\xi_{\text{KS},93} & 0\\ 0 & 0 & 0 & -\frac{\|\mathbf{p}\|^2}{g_0\,I_{\text{sp}}}\\ 0 & 0 & 0 & 0\end{bmatrix} \tag{7.148}$$

where

$$B^\xi_{\text{KS},91} = 2\,(-p_1\,p_1' + p_2\,p_2' + p_3\,p_3' - p_4\,p_4') \tag{7.149a}$$

$$B^\xi_{\text{KS},92} = 2\,(-p_1\,p_2' - p_2\,p_1' + p_3\,p_4' + p_4\,p_3') \tag{7.149b}$$

$$B^\xi_{\text{KS},93} = 2\,(-p_1\,p_3' - p_2\,p_4' - p_3\,p_1' - p_4\,p_2') \tag{7.149c}$$

If E is the independent variable, the quantities $\mathbf{g}^E_{\mathrm{KS}}(\mathbf{x})$ and $\mathbf{B}^E_{\mathrm{KS}}(\mathbf{x})$ are given by

$$\mathbf{g}^E_{\mathrm{KS}}(\mathbf{x}) = \begin{bmatrix} \mathring{\mathbf{p}} \\ -\frac{1}{4}\,\mathbf{p} \\ 0 \\ 0 \\ \frac{\|\mathbf{p}\|^2}{\sqrt{2\,h}} \end{bmatrix}, \quad \mathbf{B}^E_{\mathrm{KS}}(\mathbf{x}) = \begin{bmatrix} & & \mathbf{0}_{4\times4} & \\ B^E_{\mathrm{KS},51} & B^E_{\mathrm{KS},52} & B^E_{\mathrm{KS},53} & 0 \\ B^E_{\mathrm{KS},61} & B^E_{\mathrm{KS},62} & B^E_{\mathrm{KS},63} & 0 \\ B^E_{\mathrm{KS},71} & B^E_{\mathrm{KS},72} & B^E_{\mathrm{KS},73} & 0 \\ B^E_{\mathrm{KS},81} & B^E_{\mathrm{KS},82} & B^E_{\mathrm{KS},83} & 0 \\ B^E_{\mathrm{KS},91} & B^E_{\mathrm{KS},92} & B^E_{\mathrm{KS},93} & 0 \\ 0 & 0 & 0 & -\frac{\|\mathbf{p}\|^2}{\sqrt{2h}\,g_0\,I_{\mathrm{sp}}} \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{7.150}$$

with

$$B^E_{\mathrm{KS},51} = \frac{1}{4\,h}\left(-4\,p_2\,\mathring{p}_1\,\mathring{p}_2 - 4\,p_3\,\mathring{p}_1\,\mathring{p}_3 + 4\,p_4\,\mathring{p}_1\,\mathring{p}_4 + 4\,p_1\,\mathring{p}_1^2 + p_1\,\|\mathbf{p}\|^2\right) \tag{7.151a}$$

$$B^E_{\mathrm{KS},52} = \frac{1}{4\,h}\left(4\,p_2\,\mathring{p}_1^2 + 4\,p_1\,\mathring{p}_1\,\mathring{p}_2 - 4\,p_4\,\mathring{p}_1\,\mathring{p}_3 - 4\,p_3\,\mathring{p}_1\,\mathring{p}_4 + p_2\,\|\mathbf{p}\|^2\right) \tag{7.151b}$$

$$B^E_{\mathrm{KS},53} = \frac{1}{4\,h}\left(4\,p_3\,\mathring{p}_1^2 + 4\,p_1\,\mathring{p}_1\,\mathring{p}_3 + 4\,p_4\,\mathring{p}_1\,\mathring{p}_2 + 4\,p_2\,\mathring{p}_1\,\mathring{p}_4 + p_3\,\|\mathbf{p}\|^2\right) \tag{7.151c}$$

$$B^E_{\mathrm{KS},61} = \frac{1}{4\,h}\left(4\,p_1\mathring{p}_1\,\mathring{p}_2 - 4\,p_3\,\mathring{p}_2\,\mathring{p}_3 + 4\,p_4\,\mathring{p}_2\,\mathring{p}_4 - 4\,p_2\,\mathring{p}_2^2 - p_2\,\|\mathbf{p}\|^2\right) \tag{7.151d}$$

$$B^E_{\mathrm{KS},62} = \frac{1}{4\,h}\left(4\,p_2\,\mathring{p}_1\,\mathring{p}_2 - 4\,p_4\,\mathring{p}_2\,\mathring{p}_3 - 4\,p_3\,\mathring{p}_2\,\mathring{p}_4 + 4\,p_1\,\mathring{p}_2^2 + p_1\,\|\mathbf{p}\|^2\right) \tag{7.151e}$$

$$B^E_{\mathrm{KS},63} = \frac{1}{4\,h}\left(4\,p_3\,\mathring{p}_1\,\mathring{p}_2 + 4\,p_1\,\mathring{p}_2\,\mathring{p}_3 + 4\,p_4\,\mathring{p}_2^2 + 4\,p_2\,\mathring{p}_2\,\mathring{p}_4 + p_4\,\|\mathbf{p}\|^2\right) \tag{7.151f}$$

$$B^E_{\mathrm{KS},71} = \frac{1}{4\,h}\left(4\,p_1\,\mathring{p}_1\,\mathring{p}_3 - 4\,p_2\,\mathring{p}_2\,\mathring{p}_3 - 4\,p_3\,\mathring{p}_3^2 + 4\,p_4\,\mathring{p}_3\,\mathring{p}_4 - p_3\,\|\mathbf{p}\|^2\right) \tag{7.151g}$$

$$B^E_{\mathrm{KS},72} = \frac{1}{4\,h}\left(4\,p_2\,\mathring{p}_1\,\mathring{p}_3 + 4\,p_1\mathring{p}_2\,\mathring{p}_3 - 4\,p_4\,{\circ}p_3^2 - 4\,p_3\,\mathring{p}_3\,\mathring{p}_4 - p_4\,\|\mathbf{p}\|^2\right) \tag{7.151h}$$

$$B^E_{\mathrm{KS},73} = \frac{1}{4\,h}\left(4\,p_3\,\mathring{p}_1\,\mathring{p}_3 + 4\,p_1\,{\circ}p_3^2 + 4\,p_4\,\mathring{p}_2\,\mathring{p}_3 + 4\,p_2\,\mathring{p}_3\,\mathring{p}_4 + p_1\,\|\mathbf{p}\|^2\right) \tag{7.151i}$$

$$B^E_{\mathrm{KS},81} = \frac{1}{4\,h}\left(4\,p_1\,\mathring{p}_1\,\mathring{p}_4 - 4\,p_2\,\mathring{p}_2\,\mathring{p}_4 - 4\,p_3\,\mathring{p}_3\,\mathring{p}_4 + 4\,p_4\,\mathring{p}_4^2 + p_4\,\|\mathbf{p}\|^2\right) \tag{7.151j}$$

$$B^E_{\mathrm{KS},82} = \frac{1}{4\,h}\left(4\,p_2\,\mathring{p}_1\,\mathring{p}_4 + 4\,p_1\,\mathring{p}_2\,\mathring{p}_4 - 4\,p_4\,\mathring{p}_3\,\mathring{p}_4 - 4\,p_3\,\mathring{p}_4^2 - p_3\,\|\mathbf{p}\|^2\right) \tag{7.151k}$$

$$B^E_{\mathrm{KS},83} = \frac{1}{4\,h}\left(4\,p_3\,\mathring{p}_1\,\mathring{p}_4 + 4\,p_1\,\mathring{p}_3\,\mathring{p}_4 + 4\,p_4\,\mathring{p}_2\,\mathring{p}_4 + 4\,p_2\,\mathring{p}_4^2 + p_2\,\|\mathbf{p}\|^2\right) \tag{7.151l}$$

$$B^E_{\mathrm{KS},91} = -2\,\left(p_1\,\mathring{p}_1 - p_2\,\mathring{p}_2 - p_3\,\mathring{p}_3 + p_4\,\mathring{p}_4\right) \tag{7.151m}$$

$$B^E_{\mathrm{KS},92} = -2\,\left(p_2\,\mathring{p}_1 + p_1\,\mathring{p}_2 - p_4\,\mathring{p}_3 - p_3\,\mathring{p}_4\right) \tag{7.151n}$$

$$B^E_{\mathrm{KS},93} = -2\,\left(p_3\,\mathring{p}_1 + p_1\,\mathring{p}_3 + p_4\,\mathring{p}_2 + p_2\,\mathring{p}_4\right) \tag{7.151o}$$

Table 7.1: Overview of the considered state vector representations.

Coordinates	Unpert. dynamics $\mathbf{g}(\mathbf{x})$	Pert. dynamics $\mathbf{B}(\mathbf{x})$	# states	Slow / fast variables
Cartesian	Nonlinear	Constant	6	0 / 6
Cylindrical	Nonlinear	Constant	6	0 / 6
Spherical	Nonlinear	Constant	6	0 / 6
MEE	Weakly nonlinear	Nonlinear	6	5 / 1
MOE	Linear*	Nonlinear	6†	2 / 4†
KS$_\xi$	Weakly nonlinear*	Nonlinear	9†	1 / 8†
KS$_E$	Linear*	Nonlinear	9†	1 / 8†

* The dynamics become nonlinear if time is added as a state variable.
† The number increases by one if time is added as a state variable.

7.1.7 Summary

We define the unperturbed $\mathbf{g}(\mathbf{x})$ and perturbed $\mathbf{B}(\mathbf{x})$ terms in Eq. (7.1) to be either linear, weakly nonlinear, or nonlinear. Table 7.1 characterizes the proposed coordinate sets in terms of the level of nonlinearity, the number of state variables (without considering the mass), and the number of slow and fast variables, slow meaning that the element is constant when no perturbations are present. The selected coordinates cover a wide variety of combinations. MEE is considered weakly nonlinear as all elements except one are constant in the unperturbed case. With regard to KS$_\xi$, multiplication with the slowly varying specific energy causes the unperturbed dynamics to become weakly nonlinear. The linear terms of each coordinate system are advantageous because no approximations are introduced when building the convex subproblems in Eq. (5.23). However, as highlighted in Table 7.1, no state vector representation has only linear terms. It is therefore not straightforward to indicate a certain set of coordinates as the most suitable. Moreover, if time is included in the state vector, the unperturbed dynamics of MOE and KS$_E$ become nonlinear.

7.2 Linearization Accuracy Index

Originally, a nonlinearity index based on the state transition matrix was introduced in [153] to measure the nonlinearity of a dynamical system. As this method requires the costates of the system, we pursue a different approach that is tailored to SCP. Due to the successive linearization of the nonlinear dynamics about a reference, we expect SCP to benefit from coordinates where the first-order Taylor approximation is as accurate as possible. We therefore propose the linearization accuracy index Ξ^f that evaluates the original nonlinear dynamics function $\mathbf{f}_k^{\text{nonlin}}$ and its first-order Taylor series $\mathbf{f}_k^{\text{lin}}$ at each node $k = 1, \dots, N$:

$$\Xi_k^f := \begin{cases} \left\| \mathbf{f}_k^{\text{nonlin}} - \mathbf{f}_k^{\text{lin}} \right\| & \text{for Cart., sph., cyl. elements, and MEE} \\ \left\| \hat{\mathbf{f}}_k^{\text{nonlin}} - \hat{\mathbf{f}}_k^{\text{lin}} \right\| + \dfrac{\left\| f_k^{\text{t,nonlin}} - f_k^{\text{t,lin}} \right\|}{f_{\max}^t} & \text{for MOE, KS}_\xi\text{, and KS}_E \text{ elements} \end{cases} \tag{7.152}$$

where

$$\mathbf{f}^{\text{lin}} = \mathbf{f}\left(\bar{\mathbf{x}}, \bar{\mathbf{u}}\right) + \nabla_x \mathbf{f}\left(\bar{\mathbf{x}}, \bar{\mathbf{u}}\right) \left(\mathbf{x} - \bar{\mathbf{x}}\right) + \nabla_u \mathbf{f}\left(\bar{\mathbf{x}}, \bar{\mathbf{u}}\right) \left(\mathbf{u} - \bar{\mathbf{u}}\right) \tag{7.153}$$

The physical time t is included as a separate term for MOE and KS, and $f_k^{\text{t,nonlin}}$ and $f_k^{\text{t,lin}}$ refer to the nonlinear and linearized differential equations of t, respectively. $f_{\max}^t$ is the maximum value over all k, and the hat notation $\hat{\mathbf{f}}$ denotes the dynamics without t. Equation (7.152) essentially compares the function values of the nonlinear dynamics with the values of the first-order Taylor series evaluated at a reference point $(\bar{\mathbf{x}}, \bar{\mathbf{u}})$. Our rationale is that the linearization error $E(x)$ becomes zero when $\Delta x \to 0$:

$$\lim_{\Delta x \to 0} E(x) = \lim_{\Delta x \to 0} f(\bar{x} + \Delta x) - \left[f(\bar{x}) + f'(\bar{x})\, \Delta x \right] = 0 \tag{7.154}$$

Therefore, the index indicates how well the linearization is able to represent the original nonlinear function. Smaller values suggest better approximations, and a value of zero means that the linearization is exact. Note that we deliberately do not normalize the difference. The reason is that $\mathbf{f}$ may take very small values or even zero, and therefore, dividing by some norm such as $\|\mathbf{f}\|$ would artificially increase the metric. Using the expression in Eq. (7.152) instead favors coordinate sets where the unperturbed dynamics are close to zero, and where the number of states is small. However, the time variable is normalized and added separately to lower its impact on the index. Our findings suggest that such a metric agrees best with the success rates of the performed simulations, and that a small variation in time is often not critical. As the index depends on the deviation from a reference point, we select a sufficiently large number n of perturbed trajectories that lie in the neighborhood of the reference. The nonlinearity-like index Ξ_k^f at each node is then given by the average over n samples:

$$\Xi_k^f := \frac{1}{n} \sum_{i=1}^{n} \Xi_{k,i}^f, \qquad k = 1, \dots, N, \quad i = 1, \dots, n \tag{7.155}$$

Remark 7.3. *It is also possible to use the maximum instead of the mean value for computing the index. Our simulations suggest that there is no significant difference if worst-case initial conditions are used to generate the perturbed trajectories (see also the following subsection).*

As the performance of SCP depends on the discretization method, we are interested in a practical metric to assess the performance of a coordinate set within FOH. In FOH, the state transition matrix is used to discretize the problem. The dynamics are integrated, and the state at t_{k+1} is obtained by evaluating Eq. (6.66). The violations of the nonlinear dynamical constraints c_k^{nonlin} are computed by comparing the states at the end of a segment k:

$$c_k^{\text{nonlin}} = \left\| \mathbf{x}_{k+1} - \mathbf{x}_{k+1}^{\text{nonlin}} \right\|, \qquad k = 1, \ldots, N-1 \tag{7.156}$$

where $\mathbf{x}_{k+1}$ denotes the state at t_{k+1} obtained from the optimization, and $\mathbf{x}_{k+1}^{\text{nonlin}}$ is computed using the nonlinear dynamics:

$$\mathbf{x}_{k+1}^{\text{nonlin}} = \mathbf{x}_k + \int_{t_k}^{t_{k+1}} \mathbf{f}_{\text{nonlin}}(\mathbf{x}(\zeta), \mathbf{u}(\zeta))\, \mathrm{d}\zeta, \qquad k = 1, \ldots, N-1 \tag{7.157}$$

where $\mathbf{x}_k \equiv \mathbf{x}(t_k)$ is also the optimized state. As $\mathbf{x}_{k+1}$ is obtained using Eqs. (6.66) and (6.68b)–(6.68d), the state transition matrix is required, and hence, the integration of the Jacobian matrices. Therefore, even if two coordinate sets yield a similar index Ξ^f, the constraint violation can be large as the error accumulates during the integration. We thus propose a second nonlinearity-like metric Ξ^x that is defined as follows:

$$\Xi_k^x := \begin{cases} \dfrac{\left\| \mathcal{M}_{\text{cart}}\left(\mathbf{x}_{k+1}^{\text{nonlin}}\right) - \mathcal{M}_{\text{cart}}\left(\mathbf{x}_{k+1}^{\text{lin}}\right) \right\|}{\left\| \mathbf{x}_{\text{cart}}^{\max} - \mathbf{x}_{\text{cart}}^{\min} \right\|} & \text{for Cart., sph., cyl. elements, MEE} \\[2ex] \dfrac{\left\| \mathcal{M}_{\text{cart}}\left(\mathbf{x}_{k+1}^{\text{nonlin}}\right) - \mathcal{M}_{\text{cart}}\left(\mathbf{x}_{k+1}^{\text{lin}}\right) \right\|}{\left\| \mathbf{x}_{\text{cart}}^{\max} - \mathbf{x}_{\text{cart}}^{\min} \right\|} + \dfrac{\left\| t_k^{\text{nonlin}} - t_k^{\text{lin}} \right\|}{t_{\max}} & \text{for MOE, KS}_\xi\text{, KS}_E \end{cases} \tag{7.158}$$

where the mapping $\mathcal{M}_{\text{cart}}(\mathbf{x})$ transforms the set $\mathbf{x}$ into Cartesian coordinates. $\mathbf{x}_{\text{cart}}^{\max}$ and $\mathbf{x}_{\text{cart}}^{\min}$ denote the maximum and minimum values of the state in Cartesian coordinates over all k, respectively. Again, t is added separately for MOE and KS, and it is normalized by its maximum value $t_{\max}$. The index evaluates the difference of the integrated states obtained using the nonlinear dynamics and the state transition matrix, respectively. We use Cartesian coordinates as the reference set to ensure a fair comparison given the different systems with different characteristics. Ξ_k^x therefore gives an indication on the constraint violation for a certain initial guess. As the magnitude of the violation affects convergence, such a metric is important to understand how well each state vector representation performs when the same initial guess is provided. This does not only depend on the differential equations, but also on the distribution of the nodes. Similar to the Ξ^f metric, several perturbed trajectories are considered, and the average over n samples is taken:

$$\Xi_k^x := \frac{1}{n} \sum_{i=1}^{n} \Xi_{k,i}^x, \qquad k = 1, \ldots, N-1, \quad i = 1, \ldots, n \tag{7.159}$$

Remark 7.4. *As time is a large, monotonically increasing variable, caution is advised when including it in the index for MOE and KS. One may simply take the norm of the variations from the reference without normalizing it, which would put more emphasis on the contribution of the time error. However, as this error is often not crucial and a linear evolution of the time is often a decent approximation, we add a normalization to not bias the index.*

Generation of Perturbed Trajectories

Given a reference trajectory $\bar{\mathbf{x}}(t)$ and its corresponding initial condition $\bar{\mathbf{x}}(t_0)$, we define a set of n worst-case initial condition variations $\delta\mathbf{x}_i(t_0)$, $i = 1, \ldots, n$ with $\|\delta\mathbf{x}_i(t_0)\| = \delta x_{\max}$, similar to what has been defined in previous work [153]. In particular, we use Cartesian coordinates to define

$$\delta x_{\max} = \left\| \begin{bmatrix} \delta\mathbf{r}_{\max} \\ \delta\mathbf{v}_{\max} \end{bmatrix} \right\| \tag{7.160}$$

where $\delta\mathbf{r}_{\max} \in \mathbb{R}^3$ and $\delta\mathbf{v}_{\max} \in \mathbb{R}^3$ are arbitrary vectors that define the position and velocity variations, respectively. The worst-case initial conditions therefore lie on a n-dimensional sphere of radius $\delta x_{\max}$. The perturbed initial condition $\mathbf{x}_i(t_0)$ is then

$$\mathbf{x}_i(t_0) = \bar{\mathbf{x}}(t_0) + \delta\mathbf{x}_i(t_0), \quad i = 1, \ldots, n \tag{7.161}$$

Similarly, worst-case control variations $\delta\mathbf{u}_i(t_k)$ with $\|\delta\mathbf{u}_i(t_k)\| = \|\delta\mathbf{u}_{\max}\| = \delta u_{\max}$ are defined at each time instant t_k, $k = 1, \ldots, N$, $i = 1, \ldots, n$, The perturbed control profiles $\mathbf{u}_i(t_k)$ read

$$\mathbf{u}_i(t_k) = \bar{\mathbf{u}}(t_k) + \delta\mathbf{u}_i(t_k), \qquad i = 1, \ldots, n, \quad k = 1, \ldots, N \tag{7.162}$$

In our case, $\delta\mathbf{u}_i(t_k) \equiv \delta\boldsymbol{\tau}_i(t_k)$, i.e., we only perturb the components $\boldsymbol{\tau} \in \mathbb{R}^3$ and then compute the magnitude using $\Gamma_i(t_k) = \|\boldsymbol{\tau}_i(t_k)\|$.

The nonlinear dynamics are then integrated using the perturbed control profiles $\mathbf{u}_i(t)$ obtained from Eq. (7.162), and the perturbed initial conditions $\mathbf{x}_i(t_0)$. The resulting n state trajectories $\mathbf{x}_i(t)$ deviate from the reference by $\delta\mathbf{x}_i(t)$:

$$\mathbf{x}_i(t) = \bar{\mathbf{x}}(t) + \delta\mathbf{x}_i(t), \qquad i = 1, \ldots, n \tag{7.163}$$

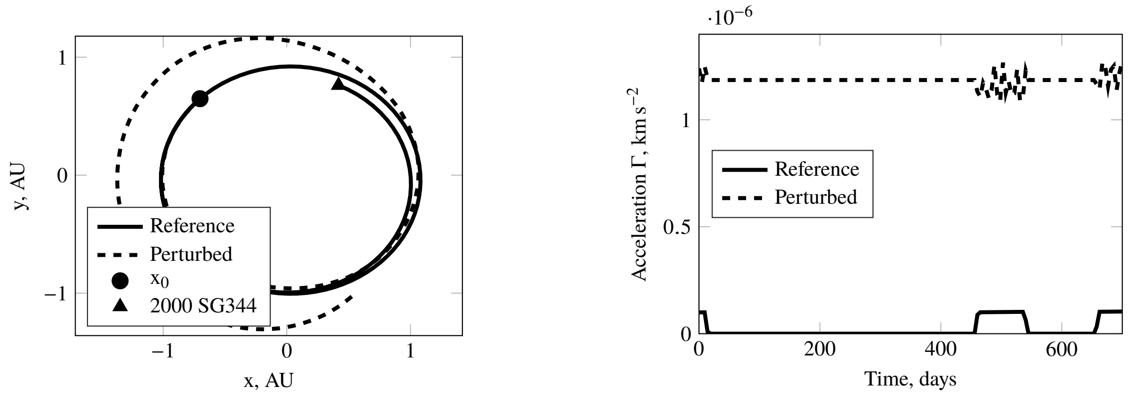
These trajectories are then transformed into the other coordinate sets to determine the linearization indices.

7.3 Numerical Simulations

The different coordinate sets are assessed in two analyses:

1) Linearization accuracy index: The two indices Ξ^f and Ξ^x are determined and compared for all coordinate sets.

2) Reliability: Using initial guesses of different quality, we compute 500 optimal trajectories for each set to compare the success rate, number of iterations, final mass, and CPU time.



(a) Transfer trajectory. **(b)** Control acceleration.

Figure 7.5: Typical reference and perturbed state and control trajectories for 2000 SG344.

We consider fuel-optimal transfers from the Sun-Earth Lagrange Point L$_2$ (SEL$_2$) to asteroid 2000 SG344, and from Earth to asteroid Dionysus. The number of revolutions is assumed to be known, and the J2000 reference frame is used where the xy-plane lies in the equatorial plane. The maximum thrust T_{max} and specific impulse I_{sp} are assumed constant. Relevant parameters of the transfers are given in Table 7.2, and Tables 7.3 and 7.4 define the SCP parameters and physical constants, respectively. All simulations are performed in MATLAB version 2018b on an Intel Core i5-6300 2.30 GHz Laptop with four cores and 8 GB of RAM.

Using the procedure explained in Section 7.2, $n = 500$ perturbed state and control trajectories are generated. The following variations are considered:

$$\begin{aligned}\text{2000 SG344:}\quad & \delta\mathbf{r}_{\text{max}} = \left[10^5, 10^5, 10^5\right]^\top \text{km},\ \delta\mathbf{v}_{\text{max}} = \left[10^{-2}, 10^{-2}, 10^{-2}\right]^\top \text{km}\,\text{s}^{-1}, \\ & \delta\boldsymbol{\tau}_{\text{max}} = [1.2, 1.2, 1.2]^\top \times 10^{-6}\,\text{km}\,\text{s}^{-2}\end{aligned} \tag{7.164}$$

$$\begin{aligned}\text{Dionysus:}\quad & \delta\mathbf{r}_{\text{max}} = \left[10^6, 10^6, 10^6\right]^\top \text{km},\ \delta\mathbf{v}_{\text{max}} = \left[10^{-1}, 10^{-1}, 10^{-1}\right]^\top \text{km}\,\text{s}^{-1}, \\ & \delta\boldsymbol{\tau}_{\text{max}} = [8.9, 8.9, 8.9]^\top \times 10^{-8}\,\text{km}\,\text{s}^{-2}\end{aligned} \tag{7.165}$$

Typical reference and perturbed trajectories are shown in Figs. 7.5 and 7.6. The obtained trajectories are used to determine the indices and serve as the initial guesses for the reliability analysis. Throughout this section, we use Cart, Sph, Cyl, MEE, MOE, KS$_\xi$, and KS$_E$ to refer to the coordinate sets described in Section 7.1.

Remark 7.5. *For the 2000 SG344 transfer, the control variations are several times larger than the maximum control magnitude of the reference trajectory. The reason is that smaller values would result in success rates of* $100\,\%$ *for all sets, which is not desirable for assessing the reliability.*

Table 7.2: Simulation values for SEL_2 to 2000 SG344, and Earth-Dionysus transfers [39, 129].

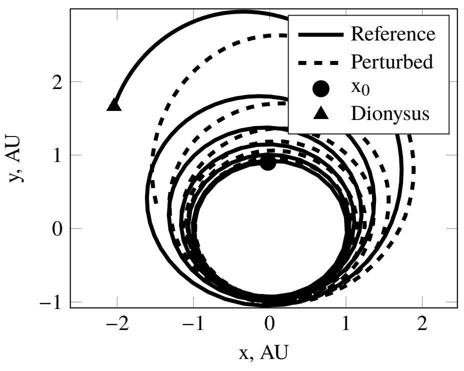
Parameter	SEL_2 - 2000 SG344	Earth - Dionysus
Initial epoch	04-Feb-2024 12:00:00 UTC	23-Dec-2012 00:00:00 UTC
$\mathbf{r}_0$, AU	$[-0.70186065, 0.64796956, 0.28089092]^\top$	$[-0.02372965, 0.90219612, 0.39111596]^\top$
$\mathbf{v}_0$, VU	$[-0.73296949, -0.65684737, -0.28473020]^\top$	$[-1.01593125, -0.02584808, -0.01116860]^\top$
m_0, kg	22.6	4000
$\mathbf{r}_f$, AU	$[0.41806795, 0.76113649, 0.32843028]^\top$	$[-2.04062009, 1.66199201, 1.32470365]^\top$
$\mathbf{v}_f$, VU	$[-0.96990332, 0.40079022, 0.17241904]^\top$	$[-0.14231814, -0.42140269, -0.16204985]^\top$
m_f, kg	free	free
$T_{\max}$, N	2.2519×10^{-3}	0.32
I_{sp}, s	3067	3000
t_f, days	700	3534

Table 7.3: Parameters of the SCP algorithm.

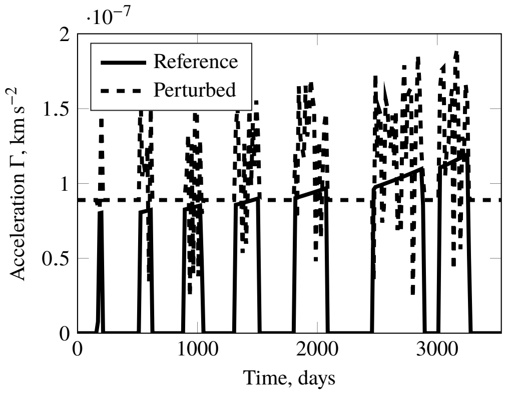
Parameter	Value
Feasibility tol. ε_c	10^{-6}
Optimality tol. ε_J	10^{-4}
Max. iterations	150
λ_ν, λ_η	10.0, 10.0
ρ_0, ρ_1, ρ_2	0.01, 0.25, 0.85
α, β, δ	1.5, 1.5, 1.0

Table 7.4: Physical constants in all simulations.

Parameter	Value
Gravitational const. μ	$1.327\,124\,4 \times 10^{11}\,\text{km}^3\,\text{s}^{-2}$
Gravitational accel. g_0	$9.806\,65 \times 10^{-3}\,\text{km}\,\text{s}^{-2}$
Length unit AU	$1.495\,978\,707 \times 10^{8}\,\text{km}$
Velocity unit VU	$\sqrt{\mu\,\text{AU}^{-1}}$
Time unit TU	$\text{AU}\,\text{VU}^{-1}$
Acceleration unit ACU	$\text{VU}\,\text{TU}^{-1}$
Mass unit MU	m_0



(a) Transfer trajectory.



(b) Control acceleration.

Figure 7.6: Typical reference and perturbed state and control trajectories for Dionysus.

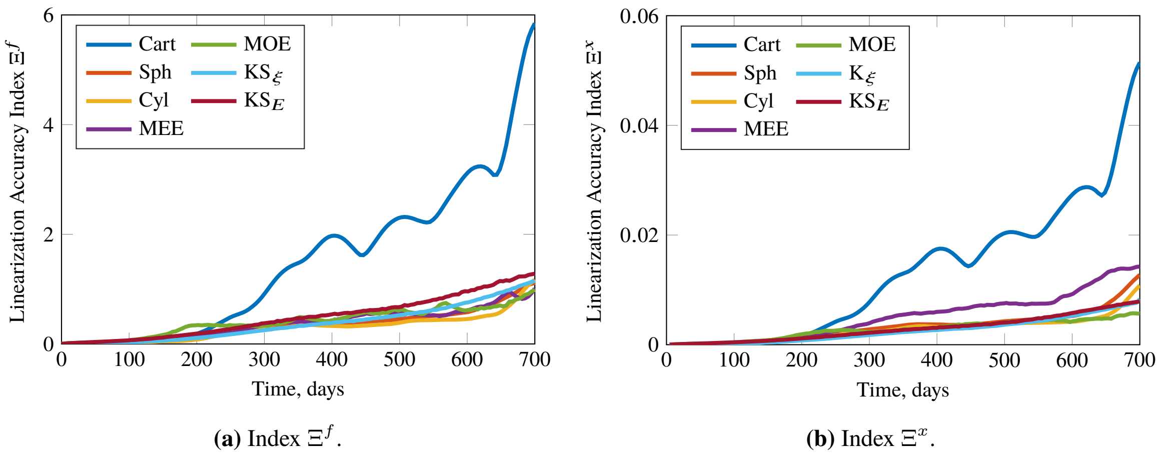


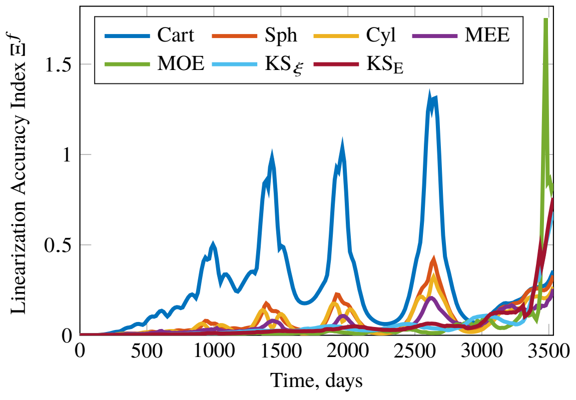
Figure 7.7: Indices Ξ^f and Ξ^x for the transfer to asteroid 2000 SG344.

7.3.1 Linearization Accuracy Index

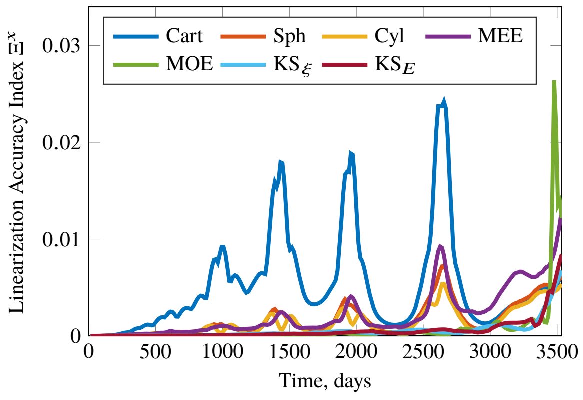
The metrics Ξ^f and Ξ^x are computed according to Eqs. (7.152) and (7.158). Their evolution for the transfers to asteroids 2000 SG344 and Dionysus is shown in Figs. 7.7 and 7.8, respectively. The values of Ξ^f and Ξ^x take different orders of magnitude over time, and Cartesian coordinates result in considerably larger values than the other coordinate sets. These, instead, yield relatively similar values. This is especially true for the 2000 SG344 transfer due to its simplicity and small number of thrust arcs. Despite the generally increasing nonlinearity around periapsis because of the rapidly changing state of the spacecraft, the evolution of the index is almost linear (except for Cartesian coordinates). One reason is the small eccentricity of the initial and final orbits which leads to a similar behavior for the whole transfer. Moreover, the duration and magnitude of the control actions are small. Therefore, the nonlinear control terms for MEE, MOE, and KS during thrust periods do not alter the index significantly.

The discrepancy becomes more significant for the Dionysus transfer. Again, Cartesian coordinates yield considerably larger values. Furthermore, the peaks become more evident for Cart, Cyl, Sph, and MEE due to the larger eccentricity of the orbits and the greater number of thrust arcs. Interestingly, MOE and KS do not follow the same trend, and result in smaller values because of the (close to) linear unperturbed dynamics (see also Figures 7.9a and 7.9b that use a logarithmic scale). Instead, their indices increase towards the end after approximately 3000 days. One reason is that the last burn occurs at apoapsis to insert the spacecraft into the final orbit. For MOE and KS, the highly nonlinear control terms dominate in that case, which results in a rise of the indices. Another reason is the error of the time variable that accumulates over time, thus contributing more towards the end of the transfer.

In general, it appears that MOE and KS result in the smallest indices, followed by cylindrical and spherical coordinates, and MEE. As the acceleration due to thrust is small with respect to the unperturbed two-body dynamics, it is reasonable that coordinate sets with highly nonlinear unperturbed dynamics (such as Cartesian) yield larger values.

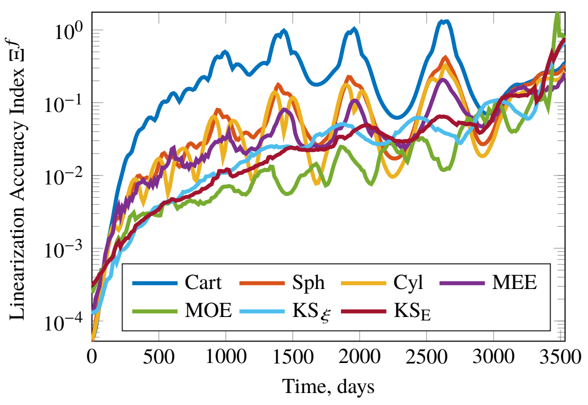


(a) Index Ξ^f.

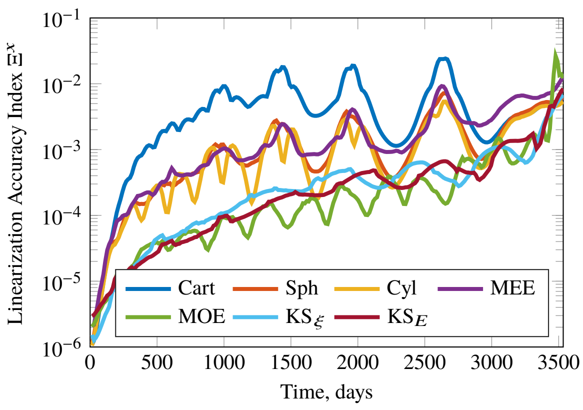


(b) Index Ξ^x.

Figure 7.8: Indices Ξ^f and Ξ^x for the transfer to Dionysus.



(a) Index Ξ^f.



(b) Index Ξ^x.

Figure 7.9: Indices Ξ^f and Ξ^x for the transfer to Dionysus (logarithmic scale).

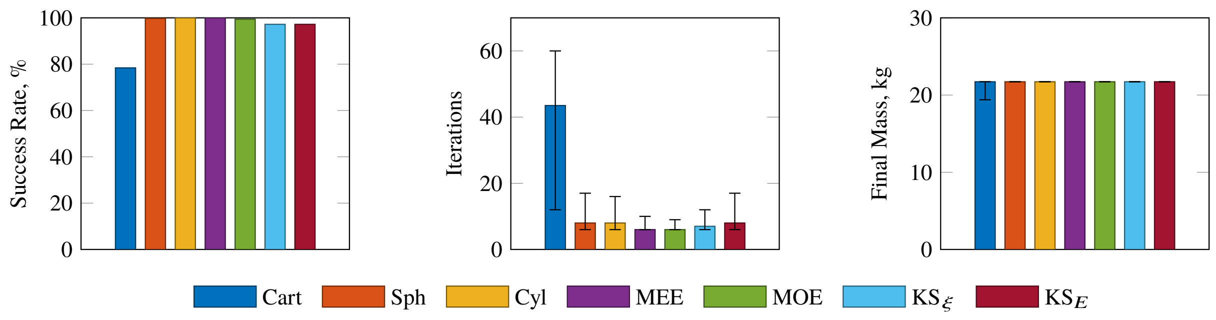


Figure 7.10: Comparison of success rate, iterations, and final mass for 2000 SG344 (fixed final value of the independent variable for MOE and KS).

7.3.2 Reliability Analysis

The convexified optimization problem is solved using the open-source Embedded Conic Solver (ECOS) [86]. Parameters of the SCP algorithm are given in Table 7.3, physical constants for the normalization in Table 7.4. The hard trust-region method with $\delta = 1.0$ is used (see also Section 5.2).

With regard to the transfer to asteroid 2000 SG344, the comparison of the success rate, number of iterations, and final mass for the 500 simulations is shown in Fig. 7.10. Median values are presented, and the error bars refer to the $90\,\%$ percentile of the respective quantity. Note that we assume the final value of the independent variable for MOE and KS to be fixed and known, e.g., from a previous optimization. All methods despite Cartesian coordinates (success rate of only $80\,\%$) converge in almost all cases, KS yielding slightly fewer optimal solutions compared to Sph, Cyl, MEE, and MOE. Cartesian elements require significantly more iterations than all other sets. The final mass is instead similar.

For Dionysus in Fig. 7.11, Sph, Cyl, MEE, and MOE outperform the other state representations in terms of success rate. Remarkably, all simulations converged successfully for MOE. Interestingly, Cart, Sph, Cyl, MEE, and KS$_\mathcal{E}$ require a similar amount of iterations (around 30), whereas MOE and KS$_E$ need only approximately five iterations. All methods yield a similar final mass.

The success rates seem in general to be in accordance with the nonlinearity index, because the highest success rates are obtained with MOE which also yield the smallest indices. Even though both KS sets result in smaller indices compared to Sph, Cyl, and MEE, the additional nonlinear final boundary constraint is probably the reason for the fewer converged cases. Moreover, MOE and KS$_E$ represent the only coordinate sets with linear unperturbed dynamics (except for the differential equation of the time) according to Table 7.1. Since the SCP process is based on linearization, it is reasonable that they are able to solve the considered problems in fewer iterations. This is particularly true for MOE, which has fewer state variables than KS$_E$ and linear final boundary conditions, thus resulting in the highest success rate.

As the independent variables of MOE and KS are anomaly-like quantities, their final values may not be known in advance. Instead of keeping the final value of the independent variable fixed, the problem can be transformed into a free final independent variable problem according to Section 6.1.3 (keeping in

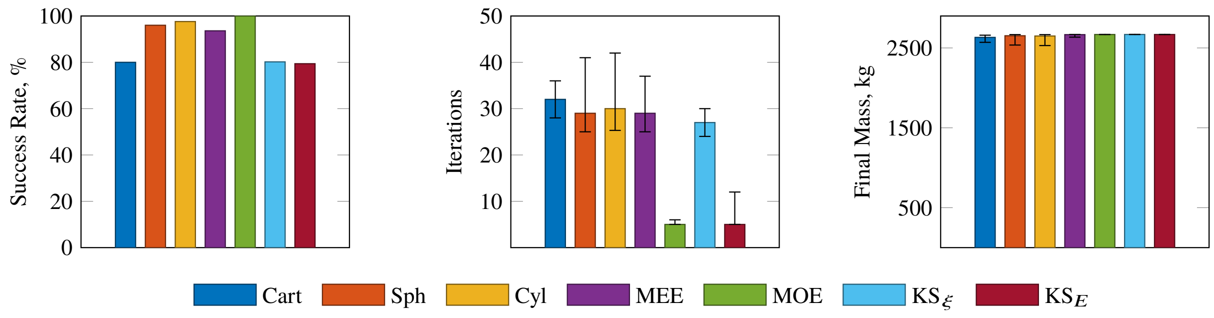


Figure 7.11: Comparison of success rate, iterations, and final mass for Dionysus (fixed final value of the independent variable for MOE and KS).

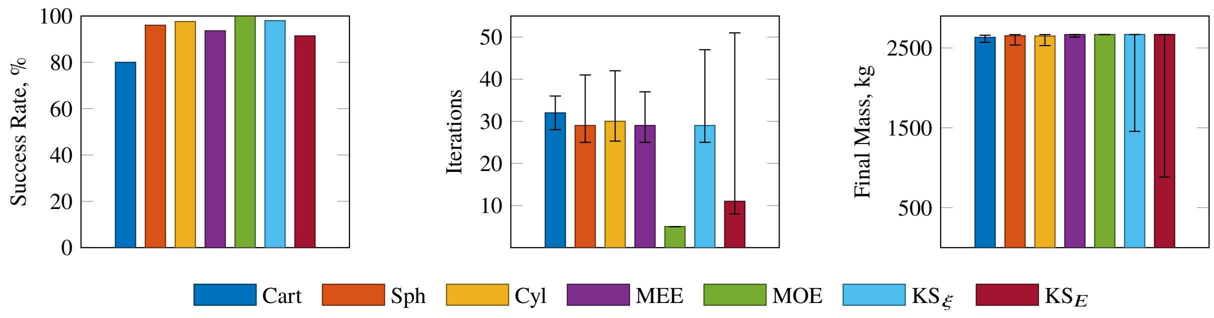


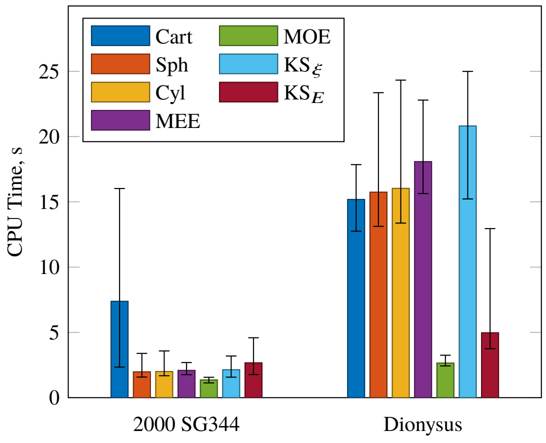
Figure 7.12: Comparison of success rate, iterations, and final mass for Dionysus (free final value of the independent variable for MOE and KS).

mind that the independent variable is not time, and that the actual time of flight does not change). This way, the final value of the independent variable is free, and therefore, an additional degree of freedom is added that might be beneficial for the solver. Figure 7.12 shows the results for the Dionysus transfer. It is evident that the success rates increase from 80 % to 98 % (KS$_\xi$) and from 79 % to 91 % (KS$_E$). As a consequence, the variations in the number of iterations and final mass rise considerably as different solutions are found that are not close to the reference. Still, keeping the final value of the independent variable free can be an effective means to increase convergence for KS. The results for the 2000 SG344 transfer do not change significantly compared to the fixed independent variable case.

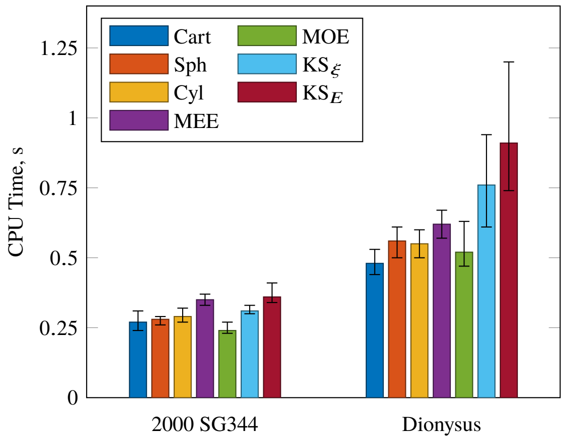
As expected, the median CPU time per simulation in Fig. 7.13a follows the same trend as the number of iterations. Yet, the CPU time per SCP iteration in Fig. 7.13b differs: KS$_\xi$ and KS$_E$ require more time per iteration due to the larger number of states, and hence, bigger matrices. KS$_E$ is the worst state representation in terms of CPU time per iteration due to the more complex control matrix $\mathbf{B}(\mathbf{x})$ compared to KS$_\xi$. Remarkably, MOE requires a similar amount of time compared to the standard sets despite the additional state.

7.3.3 Discussion

Table 7.5 gives an overview of the main advantages and disadvantages of each coordinate set.



(a) Total CPU time.



(b) CPU time per SCP iteration.

Figure 7.13: Comparison of total CPU time and CPU time per iteration.

As mentioned before, including time in the state vector causes the formerly linear dynamics to become nonlinear for MOE and KS$_E$. However, t is monotonically increasing, and its evolution is often close to linear (especially in case of quasi-circular orbits). Therefore, even if linearizing the differential equation of the time causes an error, the high success rate of MOE suggests that this does not significantly deteriorate the performance of the successive linearization approach. Similar statements can be made for KS$_E$ when the final value of the independent variable is free.

As low thrust can often be considered a small perturbation, the unperturbed dynamics dominate, and a highly nonlinear $\mathbf{B}(\mathbf{x})$ matrix as in MEE or KS$_E$ is not a major drawback. This is confirmed as both methods achieve high success rates despite the rather complex forms of the control terms.

An important aspect for state vector representations that use an anomaly-like independent variable is the distribution of the nodes. For example, MOE and KS place more nodes at periapsis. In many transfers, the controls are active mainly at periapsis due to the Oberth effect. Consequently, these coordinate sets can capture the rapidly changing state of the spacecraft (i.e., the highly nonlinear behavior) more accurately due to the larger number of nodes in this region compared to standard methods with equally-spaced temporal nodes. This also results in a smaller index for the Dionysus transfer in Figs. 7.8 and 7.9 because the thrust arcs occur near periapsis. The last burn after 3000 days, however, occurs at apoapsis to insert the spacecraft into the final orbit, and the indices for MOE and KS increase considerably towards the end of the transfer. Due to the smaller number of nodes at apoapsis compared to the conventional sets, this maneuver cannot be captured accurately, thus resulting in a larger error for MOE and KS.

Moreover, we found in our simulations that MOE and MEE are the only elements that converge if the initial and final states are linearly interpolated to generate the initial guess. This is appreciated as such a simple guess is often preferable due to its simplicity. Furthermore, MOE outperform the other coordinate sets in case the initial guess is generated by propagating the dynamics with tangential thrust. Regardless

Table 7.5: Main advantages and disadvantages of each state vector representation.

Set	Advantages	Disadvantages
Cart.	• Easy to include time-dependent constr. • Good for shape-based initial guesses	• Medium success rate • Low robustness against poor guesses
Cyl.	• High success rate • Easy to include time-dependent constr. • Good for shape-based guesses	• Medium robustness against poor guesses
Sph.	• High success rate • Easy to include time-dependent constr. • Good for shape-based guesses	• Medium robustness against poor guesses
MEE	• High success rate • Easy to include time-dependent constr. • Good for linear interpolation guesses • High robustness against poor guesses	• Highly nonlinear $\mathbf{B}(\mathbf{x})$ matrix
MOE	• High success rate • Good for linear interpolation guesses • High robustness against poor guesses	• Difficult to include time-dependent constr. • Additional time variable • Singularity
KS$_\xi$	• High success rate (free independent var.) • Medium robustness against poor guesses	• Low success rate (fixed independent var.) • Nonlinear final boundary cond. • Non-minimal set, additional time variable • Difficult to include time-dependent constr. • Highly nonlinear $\mathbf{B}(\mathbf{x})$ matrix
KS$_E$	• High success rate (free independent var.) • Medium robustness against poor guesses	• Low success rate (fixed independent var.) • Nonlinear final boundary cond. • Non-minimal set, additional time variable • Difficult to include time-dependent constr. • Highly nonlinear $\mathbf{B}(\mathbf{x})$ matrix

of the value of the thrust magnitude, MOE converge successfully in most cases, directly followed by MEE. Additionally, trajectories with more than ten revolutions for the Dionysus transfer could only be found with MOE.

Despite their disadvantages, MOE are an excellent choice for the preliminary design when feasible trajectories are to be generated reliably and efficiently using convex optimization. Yet, caution is advised because of the singularity for small inclinations.

8 Homotopic Approach for Trajectory Optimization in High-Fidelity Models

As adding high-fidelity models often increases the nonlinearity, a standard SCP algorithm may have difficulties to converge for complex problems. In indirect methods, homotopy-like procedures are often applied where the complexity of the problem is increased step by step. This chapter addresses the homotopic approach and how n-body dynamics, solar radiation pressure, and variable specific impulse and maximum thrust are incorporated in the SCP algorithm. Furthermore, no-thrust constraints are included in the first-order-hold method, and the effectiveness of the homotopic approach is compared with the state-of-the-art optimal control software GPOPS-II. Cartesian coordinates are used throughout this chapter. This chapter is based on our work in [9–11].

8.1 Embedded Homotopic Approach for High-Fidelity Models

In a homotopic approach, a parameter $\varepsilon \in [1, 0]$ is defined that is gradually reduced from $\varepsilon = 1$ (corresponds to a simpler problem) to $\varepsilon = 0$ (corresponds to the original, more complex problem that is eventually to be solved). Usually, each subproblem is solved to full optimality, and the step size n is fixed and defined by the user. This means that the algorithm requires at least n homotopic steps to achieve convergence. Often, a conservative (i.e., small) value of the step size is chosen to avoid non-convergence. This procedure is not ideal as it may result in a large number of homotopic steps and thus, iterations and CPU time increase. Even if a larger step size is chosen and then reduced by some heuristic when the solver fails, this would require additional (and unwanted) iterations. It would therefore be desirable to adjust the step size dynamically based on information that can be retrieved from the optimization problem. Thus, instead of keeping the step size fixed, it is adjusted dynamically in this dissertation. We propose a homotopy path based on the maximum constraint violation c_{max}. ε is updated at each SCP iteration depending on the constraint violation, which is a measure of the progress of the algorithm. Figure 8.1 shows three examples h_1, h_2, and h_3 that define different homotopy paths. A large c_{max} indicates that we are far from a feasible solution. Therefore, ε is also large, and we want to solve a simpler problem. Once the algorithm makes progress towards an optimal solution, it will select smaller ε values until the

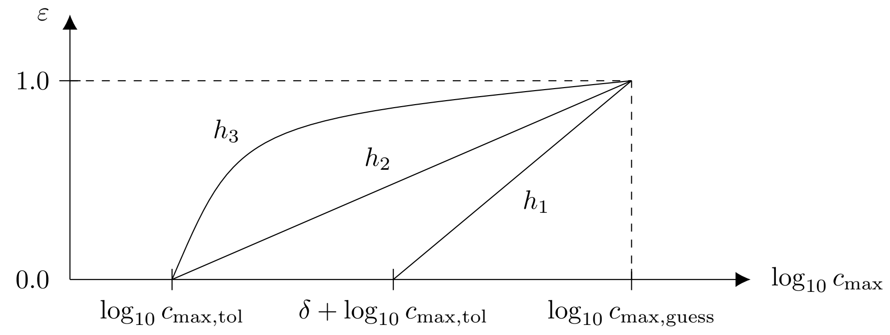


Figure 8.1: Three examples of smooth homotopy paths h_1, h_2, and h_3 that define the relationship between the homotopic parameter ε and the maximum constraint violation c_{max}.

desired optimal control problem is solved. For instance, given a relatively simple optimization problem where the initial guess is good and hence, the initial constraint violation is small, our method can exploit this knowledge and select larger step sizes. This results in faster convergence. On the other hand, if we intend to solve a difficult problem where only a poor initial guess is provided, the algorithm will select smaller step sizes as c_{max} will decrease only slowly during the iterations.

Selecting a homotopy path may depend on the problem. For example, h_1 and h_2 in Fig. 8.1 are more conservative towards the end, whereas h_3 decreases ε more slowly at the beginning of the optimization process. We found in our simulations that the choice is not critical, and a linear path like h_1 or h_2 is often a good compromise. The functions are defined at the beginning of the algorithm based on the maximum constraint violation of the initial guess $c_{\mathrm{max,guess}}$. Defining

$$m := \frac{\varepsilon_{\mathrm{max}} - \varepsilon_{\mathrm{min}}}{\log_{10}\left(c_{\mathrm{max,guess}}\right) - \log_{10}\left(c_{\mathrm{max,tol}}\right) - \delta} \tag{8.1}$$

the homotopy path h_1 that is used throughout this chapter can be written as follows:

$$h_1(\log_{10} c_{\mathrm{max}}) = m \log_{10}\left(c_{\mathrm{max}}\right) - m\left[\log_{10}\left(c_{\mathrm{max,tol}}\right) + \delta\right] \tag{8.2}$$

where $c_{\mathrm{max,guess}}$ and $c_{\mathrm{max,tol}}$ denote the maximum constraint violation of the initial guess and the desired feasibility tolerance, respectively. $\delta > 0$ is a constant to further adjust the path.

n-Body Dynamics

The fidelity of the model is increased by considering the perturbing accelerations $\mathbf{a}_{\text{nbody}}$ of other bodies in the solar system and the solar radiation pressure $\mathbf{a}_{\text{SRP}}$. Omitting the time dependency, the augmented dynamics then read

$$\mathbf{f}(\mathbf{x},\mathbf{u}) = \begin{bmatrix} \dot{\mathbf{r}} \\ \dot{\mathbf{v}} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} \mathbf{v} \\ -\mu\,\mathbf{r}/r^3 \\ 0 \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{a}_{\text{nbody}} \\ 0 \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{a}_{\text{SRP}} \\ 0 \end{bmatrix} + \mathbf{B}\,\mathbf{u} \tag{8.3}$$

With regard to the solar radiation pressure (SRP), a simple cannonball model is used where the projected area A_{SC} of the spacecraft is assumed constant [191], and we have

$$\mathbf{a}_{\text{SRP}} = \frac{S_{\text{Sun}}\,C_R\,A_{\text{SC}}}{m}\,\frac{\mathbf{r}}{r^3} \tag{8.4}$$

with

$$S_{\text{Sun}} = \frac{L_{\text{Sun}}}{4\,\pi\,c} \tag{8.5}$$

$$L_{\text{Sun}} = 4\,\pi\,C_{\text{Sun}}\,\text{AU}^2 \tag{8.6}$$

C_R is the reflectivity coefficient of the spacecraft, S_{Sun} the solar pressure constant, L_{Sun} the luminosity of the Sun, and C_{Sun} the solar constant. c and AU denote the speed of light and astronomical unit, respectively. Note that the mass m in the denominator of Eq. (8.4) is replaced by e^w in the implementation due to the change of variables (see Section 5.1).

Regarding the n-body dynamics, the $n = 10$ perturbations of the barycenters of Mercury, Venus, Earth, Moon, Mars, Jupiter, Saturn, Uranus, Neptune, and Pluto are considered. The acceleration is calculated as follows:

$$\mathbf{a}_{\text{nbody}} = \sum_i^n \mu_i \left(\frac{\mathbf{r}_{\text{sat},i}}{r^3_{\text{sat},i}} - \frac{\mathbf{r}_i}{r_i^3} \right) \tag{8.7}$$

μ_i is the gravitational constant of the ith body, $\mathbf{r}_i$ is the position of the ith body with respect to the Sun, and $\mathbf{r}_{\text{sat},i} = \mathbf{r}_i - \mathbf{r}$ denotes the position of the ith body with respect to the spacecraft. The time-dependent positions $\mathbf{r}_i$ of the perturbing bodies can be obtained using the software SPICE [192].

Due to the additional perturbing accelerations, the new dynamical system is highly nonlinear, and the dynamics are linearized as per Section 5.1. When including n-body dynamics, however, special treatment is required when the spacecraft flies in close proximity of another perturbing body (e.g., in case of flybys). This may become a problem as $r_{\text{sat},i}$ is close to zero and hence, the corresponding denominator in Eq. (8.7) is close to zero. When integrating the nonlinear dynamics and state transition matrix during the discretization, the position and velocity of the spacecraft can change rapidly between two integration steps. The reason is the large gravitational force in proximity of a planet. If a fixed-step integrator is used, the resulting perturbing accelerations can vary considerably at each step. As a consequence, the integration becomes inaccurate, and the SCP method eventually fails to converge. To overcome this issue, a constraint $r_{\text{sat},i} \geq r_{\text{min}}$ can be added to require a minimum distance r_{min} between spacecraft and planet; flybys are thus avoided. In this dissertation, however, we use a variable-step integrator for a segment when the condition $r_{\text{sat},i} \leq r_{\text{min}}$ is satisfied, i.e., when the spacecraft is close to a planet. This ensures high accuracy without rapid changes and does not unnecessarily constrain the solution space. As it is only used for very few segments, the computational load does not increase considerably.

Instead of solving the problem with n-body dynamics and SRP directly, we introduce the homotopic parameters $\varepsilon_{\text{nbody}}$ and ε_{SRP} and augment Eq. (8.3) as follows:

$$\mathbf{f}(\mathbf{x},\mathbf{u}) = \begin{bmatrix} \mathbf{v} \\ -\mu\,\mathbf{r}/r^3 \\ 0 \end{bmatrix} + (1-\varepsilon_{\text{nbody}}) \begin{bmatrix} \mathbf{0} \\ \mathbf{a}_{\text{nbody}} \\ 0 \end{bmatrix} + (1-\varepsilon_{\text{SRP}}) \begin{bmatrix} \mathbf{0} \\ \mathbf{a}_{\text{SRP}} \\ 0 \end{bmatrix} + \mathbf{B}\mathbf{u} \tag{8.8}$$

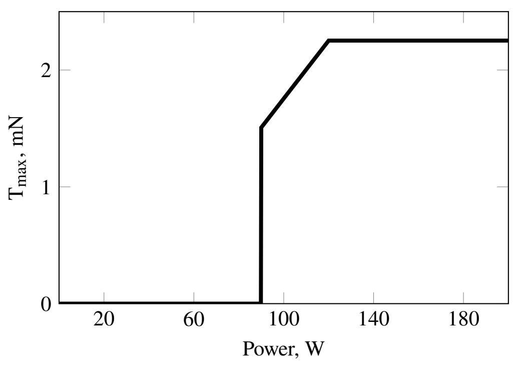
During the first iterations, the constraint violations are usually large, and $\varepsilon_{\text{nbody}}$ and ε_{SRP} are usually close to one. This means that the perturbing accelerations are small. When the maximum constraint violation decreases, ε is gradually decreased until $\varepsilon = 0$, i.e., the original problem is solved. Our rationale is that the solver is more likely to find a feasible solution when increasing the nonlinearity (and hence, the linearized term) only step by step compared to solving the full problem directly. Note that such an approach allows us to define different homotopy paths and ε for each perturbing force so that each term can be controlled separately. This may be useful if one perturbation is considered more critical and therefore requires a more conservative step size at the beginning (see also Eq. (8.2) and Fig. 8.1).

Remark 8.1. *If a good (e.g., close to feasible) initial guess is provided, it is straightforward to select an initial homotopic parameter ε_0 that is closer to zero, i.e., $0 < \varepsilon_0 < 1$, to shorten the homotopy path, and reduce the number of iterations.*

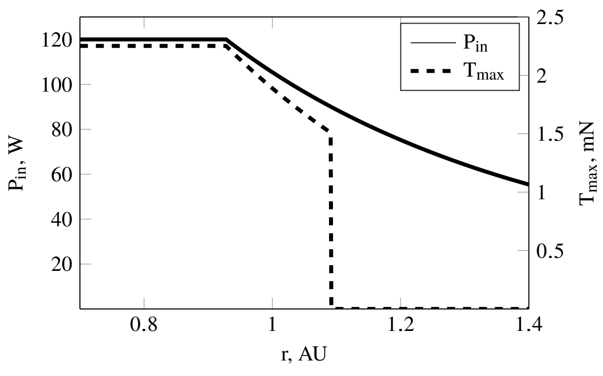
Variable Specific Impulse and Maximum Thrust

In a real thruster model, the maximum thrust $T_{\text{max}}(P_{\text{in}}(r))$ and specific impulse $I_{\text{sp}}(P_{\text{in}}(r))$ depend on the input power $P_{\text{in}}(r)$, which in turn is a function of the distance r to the Sun. Without loss of generality, the saturation logic of a real thruster is given by [129]

$$T_{\text{max}} = \begin{cases} T_{\text{max}}(P_{\text{in,max}}) & \text{if } P_{\text{in}}(r) > P_{\text{in,max}} \\ T_{\text{max}}(P_{\text{in}}(r)) & \text{if } P_{\text{in,min}} \leq P_{\text{in}}(r) \leq P_{\text{in,max}} \\ 0 & \text{if } P_{\text{in}}(r) < P_{\text{in,min}} \end{cases} \tag{8.9}$$



(a) $P_{\text{in}}(r)$ vs. $T_{\text{max}}(P_{\text{in}}(r))$ curve.



(b) $P_{\text{in}}(r)$ and $T_{\text{max}}(r)$ curves.

Figure 8.2: Typical input power and maximum thrust curves.

where $P_{\text{in,min}}$ and $P_{\text{in,max}}$ denote the minimum and maximum input power, respectively. Whenever the input power is lower than some threshold $P_{\text{in,min}}$, no thrust is available. Moreover, T_{max} cannot exceed the threshold $T_{\text{max}}(P_{\text{in,max}})$. These functions are often described by nth order polynomials. Typical $P_{\text{in}}(r)$ and $T_{\text{max}}(P_{\text{in}}(r))$ curves are shown in Fig. 8.2. Such a non-smooth representation increases complexity and results in additional nonlinear constraints. Recalling the dynamics in Eq. (5.11), it becomes evident that the previously constant Jacobian matrix $\mathbf{B}$ now depends on r, i.e., we have $\mathbf{B}(\mathbf{x})$ for the variable I_{sp} case. As a consequence, linearizing the dynamics would yield an additional term corresponding to the previous control history $\bar{\mathbf{u}}$. This dependence on $\bar{\mathbf{u}}$ is undesirable as it may deteriorate convergence [171]. Therefore, we propose the approximation $\mathbf{f}(\mathbf{x},\mathbf{u}) \approx \mathbf{p}(\mathbf{x}) + \mathbf{B}(\bar{\mathbf{x}})\mathbf{u}$ which results in a partial linearization of the dynamics at $\bar{\mathbf{x}}$:

$$\mathbf{f}(\mathbf{x},\mathbf{u}) \approx \mathbf{A}(\bar{\mathbf{x}})\mathbf{x} + \mathbf{B}(\bar{\mathbf{x}})\mathbf{u} + \mathbf{q}(\bar{\mathbf{x}}) \tag{8.10}$$

A similar approach is used for the upper bound of the thrust magnitude in Eq. (5.9) where $T_{\text{max}}(r)$ becomes a function of r. Instead of linearizing the constraint, $T_{\text{max}}(r)$ is approximated by

$$T_{\text{max}}(r) \approx T_{\text{max}}(\bar{r}) \tag{8.11}$$

where $\bar{r}$ denotes the radius of the previous iteration. As the algorithm progresses, $\|r - r^*\| \to 0$ and $\|r - \bar{r}\| \to 0$ due to the shrinking trust-region size, r^* being the radius at the optimal solution. Therefore, $r^* \approx \bar{r}$ when the algorithm determines an optimal solution. Our simulations suggest that this approach is often advantageous in terms of convergence.

Solving the optimal control problem with a real thruster model directly can be challenging, especially for large $P_{\text{in,min}}$ or when the spacecraft is far away from the Sun. An example is the transfer to asteroid Dionysus whose distance to the Sun can be larger than 3 AU, hence resulting in a low T_{max}. A standard way to improve convergence is to solve the problem with fixed I_{sp} and T_{max}, and then use the solution as the new initial guess to solve another optimization problem with the real thruster model. This is not ideal due to two reasons: First, the solver might not be able to find a solution even when the solution of the constant I_{sp} and T_{max} case is provided as the initial guess for the real model. This is especially true if the constant and real thruster models differ a lot (e.g., in the Dionysus transfer), or when the input power range is small. Secondly, such an approach would require solving another optimization problem with a potentially significantly different thruster curve, and therefore more computational effort and time. In this dissertation, two improvements are proposed:

1) A smooth representation of the input power P_{in} over T_{max} curve.

2) A homotopy from higher thrust levels to the real thruster model to enhance convergence.

Using the general hyperbolic tangent function [193], we propose the following smooth T_{max} function of the original relationship in Eq. (8.9):

$$T_{\text{max}} = \begin{cases} T_{\text{max}}(P_{\text{in,max}}) & \text{if } P_{\text{in}}(r) > P_{\text{in,max}} + \rho \wedge \rho > 0 \\ g(P_{\text{in}}(r)) & \text{if } P_{\text{in,max}} - \rho \leq P_{\text{in}}(r) \leq P_{\text{in,max}} + \rho \wedge \rho > 0 \\ \frac{1}{2} T_{\text{max}}(P_{\text{in}}(r)) \left[\tanh\left(\frac{P_{\text{in}}(r) - P_{\text{in,min}}}{\rho}\right) + 1\right] & \text{if } P_{\text{in}}(r) < P_{\text{in,max}} - \rho \wedge \rho > 0 \end{cases} \tag{8.12}$$

with some function $g(P_{\text{in}}(r))$ that depends on the order of the polynomial of the P_{in}-T_{max} curve, and a smoothing parameter ρ. If $\rho = 0$, this reduces to

$$T_{\text{max}} = T_{\text{max}}(P_{\text{in,max}}) \tag{8.13}$$

For example, given the following thruster model

$$T_{\text{max}}(P_{\text{in}}) = a_0 + a_1 P_{\text{in}} \tag{8.14a}$$

$$I_{\text{sp}}(P_{\text{in}}) = b_0 + b_1 P_{\text{in}} \tag{8.14b}$$

$$P_{\text{in}}(r) = \frac{1}{r^2} \tag{8.14c}$$

the piecewise defined functions in Eq. (8.9) are smoothly connected using a second-order polynomial:

$$g(P_{\text{in}}(r)) = c_0 + c_1\, P_{\text{in}} + c_2\, P_{\text{in}}^2 \tag{8.15}$$

where

$$c_0 = \frac{a_1\, P_{\text{in,max}}^2 + 2\, a_1\, P_{\text{in,max}}\, \rho - a_1\, \rho^2 + 4\, a_0 \rho}{4\, \rho} \tag{8.16a}$$

$$c_1 = \frac{a_1\, (P_{\text{in,max}} + \rho)}{2\, \rho} \tag{8.16b}$$

$$c_2 = -\frac{a_1}{4\, \rho} \tag{8.16c}$$

Higher orders can be included in a similar way by increasing the degree of the polynomial $g(P_{\text{in}}(r))$.

The homotopy from higher to lower thrust levels is achieved by increasing $P_{\text{in,max}}$ (see Fig. 8.3a) and by shifting the P_{in}-T_{max} curve along the x-axis (see Fig. 8.3b), hence increasing or decreasing T_{max} for a given input power which is a function of the distance to the Sun. The effect of ρ is also illustrated in Fig. 8.3b. Combining both yields the homotopic approach for the variable T_{max} case: For larger values of ε (i.e., larger constraint violations), the nominal curve (solid line corresponding to $\varepsilon = 0$ in Fig. 8.3b) is shifted to the left. Moreover, the maximum input power is increased, therefore increasing the available maximum thrust. At the same time, the curve is made smoother to have no abrupt changes in the thrust values, which is beneficial for convergence. As the algorithm progresses, the constraint violation reduces and consequently, the Power-T_{max} curve is shifted back towards the original one and the smoothness is reduced. This way, the complexity is gradually increased until the original problem is solved. We define mappings $\mathcal{M}_{\Delta P}$, $\mathcal{M}_{\rho}$, and $\mathcal{M}_{P_{\text{in,max}}}$ that map the shifting of the power curve ΔP, the smoothing ρ, and the change of $P_{\text{in,max}}$ to the homotopic parameter $\varepsilon \in [0,1]$ and vice versa:

$$\mathcal{M}_{\Delta P} : \varepsilon \longmapsto \Delta P \tag{8.17a}$$

$$\mathcal{M}_{\rho} : \varepsilon \longmapsto \rho \tag{8.17b}$$

$$\mathcal{M}_{P_{\text{in,max}}} : \varepsilon \longmapsto P_{\text{in,max}} \tag{8.17c}$$

We use the following linear relationship:

$$\mathcal{M}_{\Delta P} : \varepsilon \longmapsto \Delta P_{\text{min}} + \frac{\Delta P_{\text{max}} - \Delta P_{\text{min}}}{\varepsilon_{\text{max}} - \varepsilon_{\text{min}}} (\varepsilon - \varepsilon_{\text{min}}) \tag{8.18a}$$

$$\mathcal{M}_{\rho} : \varepsilon \longmapsto \rho_{\text{min}} + \frac{\rho_{\text{max}} - \rho_{\text{min}}}{\varepsilon_{\text{max}} - \varepsilon_{\text{min}}} (\varepsilon - \varepsilon_{\text{min}}) \tag{8.18b}$$

$$\mathcal{M}_{P_{\text{in,max}}} : \varepsilon \longmapsto P_{\text{in,max}} + \frac{P_{\text{max}} - P_{\text{in,max}}}{\varepsilon_{\text{max}} - \varepsilon_{\text{min}}} (\varepsilon - \varepsilon_{\text{min}}) \tag{8.18c}$$

with problem dependent constants ΔP_{min}, ΔP_{max}, ρ_{min}, ρ_{max}, and P_{max}. ε_{min} and ε_{max} are the lower and upper bound of the homotopic parameter, in general 0 and 1. Often, $\Delta P_{\text{min}} = 0$, and ΔP_{max} can be chosen such that $T_{\text{max}} = T_{\text{max}}(P_{\text{in,max}})$ over a sufficiently large input power range. ρ_{min} and ρ_{max} define the minimum and maximum smoothing parameter, respectively. Generally, values of the order 10^0 for ρ_{max} and values close to zero for ρ_{min} ensure a sufficient smoothing. P_{max} denotes the maximum allowable input power that is to be chosen such that $P_{\text{max}} \geq P_{\text{in,max}}$. The inverse mappings $\mathcal{M}_{\Delta P}^{-1}$, $\mathcal{M}_{\rho}^{-1}$, and $\mathcal{M}_{P_{\text{in,max}}}^{-1}$ are defined accordingly.

In case the same homotopy path h is used for n-body dynamics, SRP, and variable maximum thrust, it suffices to define only a single parameter ε that controls every homotopy in the optimization simultaneously. If different paths are used, one can define

$$\varepsilon := \max \left(\varepsilon_{\text{nbody}}, \varepsilon_{\text{SRP}}, \varepsilon_{T_{\text{max}}}\right) \tag{8.19}$$

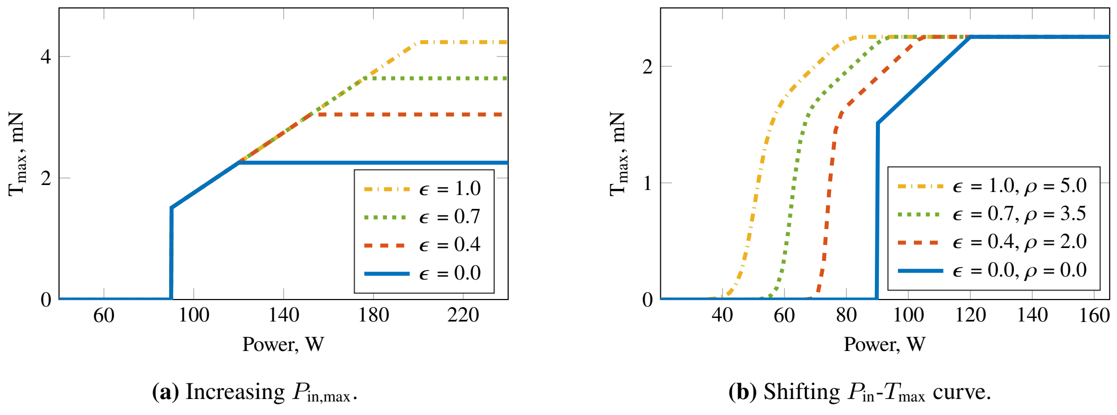


Figure 8.3: Overview of homotopy techniques for variable T_{max}.

to keep the number of parameters and complexity of the algorithm at a minimum.

Remark 8.2. *As in any direct method, the control constraints are only satisfied at the nodes. Due to the linear interpolation of the controls in FOH between the nodes, there will be a minor mismatch between the interpolated controls and the available maximum thrust if the $T_{max}(t)$ curve is nonlinear. This error, however, can be made arbitrarily small if more nodes are added in the corresponding segments.*

Adaptive Trust-Region Method

We recall the trust-region method of Section 5.2 that uses the ratio $\rho^{(k)}$ of the actual and predicted cost reductions at iteration k to measure the progress and decide whether to shrink or increase the trust-region size R (see, e.g., Eq. (5.38)). This procedure often works well in practice. However, when a homotopic approach is included, such methods can deteriorate convergence if not modified appropriately. If no homotopy is considered, the algorithm updates the trust region such that the trajectory is driven towards the optimal solution x^* as illustrated in Fig. 8.4. If a homotopy is included, the algorithm progresses in a similar way as long as ε does not change. We recall that the previous solution is used as the new initial guess. When ε_j changes to ε_{j+1}, the constraint violation increases because the constraints also (slightly) change. Therefore, the previously optimal trajectory will not satisfy all constraints anymore. In addition, the new optimal solution $x^*(\varepsilon_{j+1})$ will also be different. As a consequence, it might happen that the new optimal path associated with ε_{j+1} lies outside of the trust region defined by R. Even though the algorithm will still often be able to reduce the constraint violation in the following iterations by reducing the trust-region size, the followed path is different from the optimal one. This can eventually result in a non-feasible (and non-optimal) solution as shown in Fig. 8.5. This is especially true for difficult problems.

We propose an adaptive trust-region update to overcome this issue. Instead of using the same trust-region size when ε_j changes to ε_{j+1}, R is increased to expand the feasible region. Thus, the algorithm is able to reach the new optimal solution $x^*(\varepsilon_{j+1})$ as illustrated in Fig. 8.6. Even though expanding R to some large, fixed value might often be sufficient to reach the new optimal path, this approach would require a considerable amount of additional iterations to reduce the trust-region radius again. Moreover, finding an appropriate value for $R(\varepsilon_{j+1})$ is not intuitive.

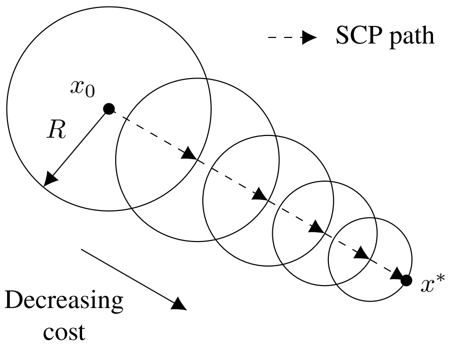


Figure 8.4: Change of the trust-region radius and typical SCP path without a homotopic approach.

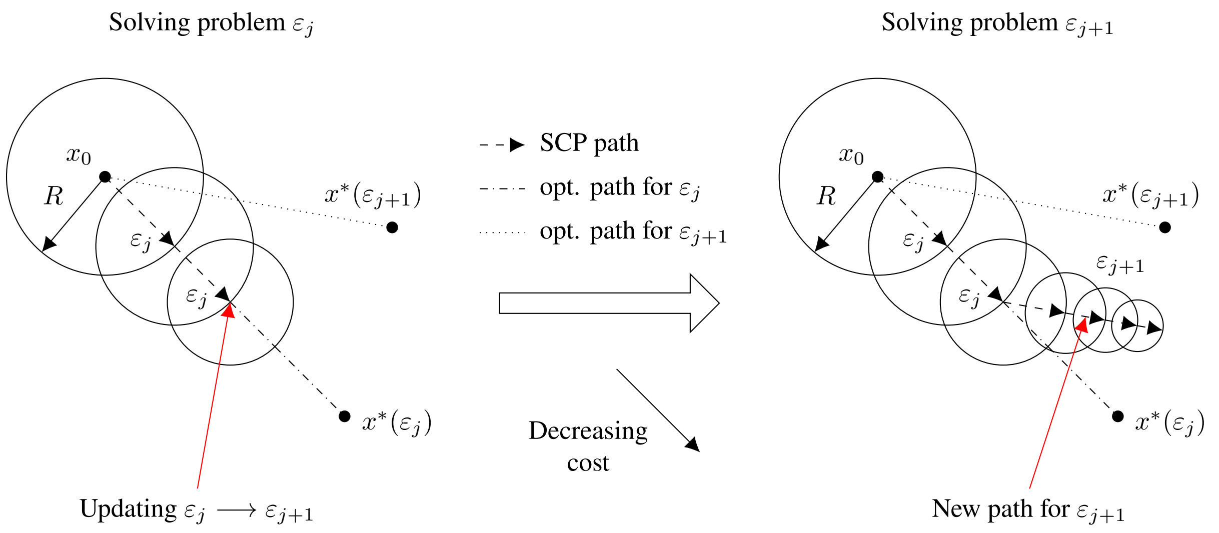

Solving problem ε_j
Solving problem ε_{j+1}
x_0
R
$x^*(\varepsilon_{j+1})$
ε_j
ε_j
$x^*(\varepsilon_j)$
Updating $\varepsilon_j \longrightarrow \varepsilon_{j+1}$
SCP path
opt. path for ε_j
opt. path for ε_{j+1}
Decreasing cost
x_0
R
$x^*(\varepsilon_{j+1})$
ε_{j+1}
ε_j
ε_j
$x^*(\varepsilon_j)$
New path for ε_{j+1}

Figure 8.5: Change of the trust-region radius and typical SCP path when the trust-region radius is not updated appropriately within a homotopic approach.

As the constraint violations c_{viol} change when the homotopic parameter is updated, it is reasonable to define an update mechanism based on the difference Δc_{viol} of the constraint violations. Figure 8.7 shows the ratio $\sigma = R(\varepsilon_{j+1})/R(\varepsilon_j) = R_{\text{new}}/R_{\text{old}}$ over $\Delta c_{\text{viol}} = \log_{10}(\|\mathbf{c}_{\text{viol}}(\varepsilon_{j+1})\|_1) - \log_{10}(\|\mathbf{c}_{\text{viol}}(\varepsilon_j)\|_1)$ for several hundreds of simulations with different targets and initial guesses, $\mathbf{c}_{\text{viol}}$ being a vector that contains the violations of all constraints. Note that we seek the maximum trust-region radius that is accepted after the homotopic parameter is updated to ensure the largest feasible region possible; this value is denoted by $R(\varepsilon_{j+1})$. We propose to fit the data points using the following equation:

$$\sigma = \frac{R(\varepsilon_{j+1})}{R(\varepsilon_j)} = C_0 + C_1 \left(\Delta c_{\text{viol}} - C_2\right)^n \tag{8.20}$$

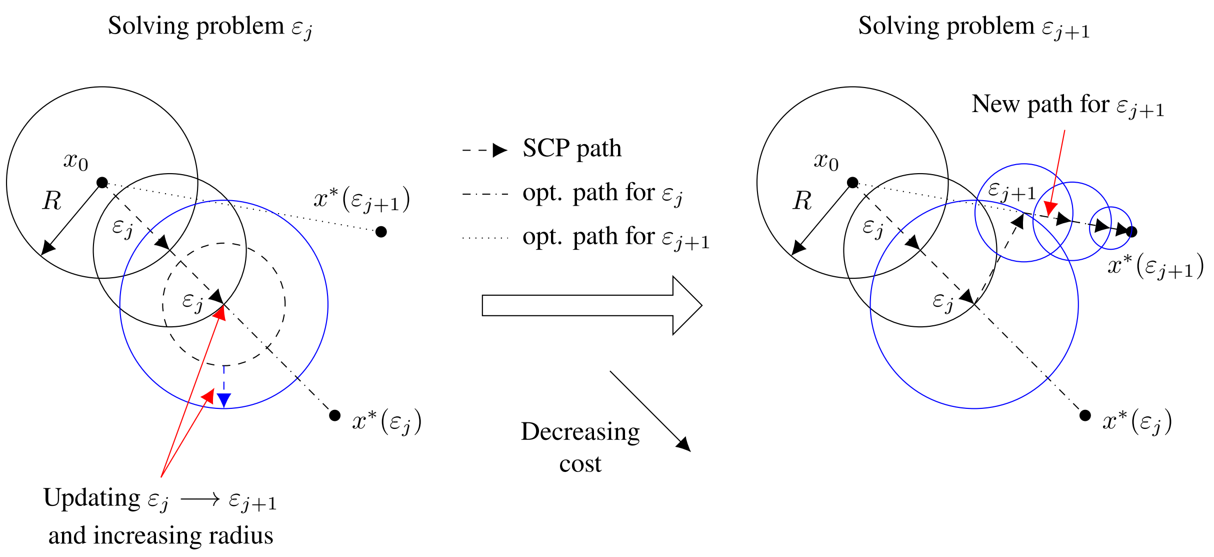

Solving problem ε_j
Solving problem ε_{j+1}
x_0
R
ε_j
ε_j
$x^*(\varepsilon_{j+1})$
$x^*(\varepsilon_j)$
SCP path
opt. path for ε_j
opt. path for ε_{j+1}
Decreasing cost
Updating $\varepsilon_j \longrightarrow \varepsilon_{j+1}$ and increasing radius
New path for ε_{j+1}
ε_{j+1}

Figure 8.6: Change of the trust-region radius and typical SCP path when the trust-region radius is updated appropriately within a homotopic approach.

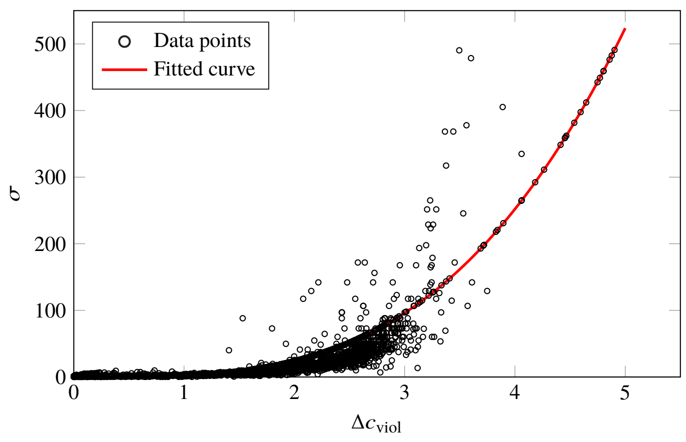


Figure 8.7: Relationship between the constraint violation and ratio of new and old trust-region radii.

where C_i $(i = 0, 1, 2)$ and n are the coefficients and exponent, respectively. In this dissertation, we use $C_0 = 1.8186$, $C_1 = 5.3518$, $C_1 = 0.3974$, and $n = 3$. The new trust-region radius $R(\varepsilon_{j+1})$ is then calculated by the relationship

$$R(\varepsilon_{j+1}) = \gamma\, \sigma\, R(\varepsilon_j) \tag{8.21}$$

with some factor $\gamma > 0$ to increase the flexibility of the update mechanism. Although one could simply set $R(\varepsilon_{j+1})$ equal to the initial trust-region size or some other large value, the proposed approach requires significantly fewer iterations as the factor σ is not fixed, but rather adjusted dynamically based on the current information at each iteration.

Flowchart of the Algorithm

The flowchart of the homotopic approach is illustrated in Fig. 8.8. After solving the convexified problem, the trust-region radius is updated as in the standard SCP method. The homotopic parameter is then adjusted based on the predefined homotopy path. Additional parameters $\Delta\varepsilon_{\text{min}}$ and $\Delta\varepsilon_{\text{max}}$ are introduced that denote the minimum and maximum allowable changes of ε, respectively. This ensures that the step size is neither too small (which would require more iterations) nor too large (which might result in non-convergence). Furthermore, the ε domain is divided into two regions with different values of $\Delta\varepsilon_{\text{min}}$ and $\Delta\varepsilon_{\text{max}}$. In particular, we have

1) $\Delta\varepsilon_{\text{min},1}$, $\Delta\varepsilon_{\text{max},1}$ if $\varepsilon \in [y_1, 1]$

2) $\Delta\varepsilon_{\text{min},2}$, $\Delta\varepsilon_{\text{max},2}$ if $\varepsilon \in [0, y_1)$

where $y_1 \in (0, 1)$ defines the switching between the different minimum and maximum step sizes. In this work, we use $y_1 = 0.7$ and $\Delta\varepsilon_{\text{min},1} = 0.0$, $\Delta\varepsilon_{\text{max},1} = 0.025$, and $\Delta\varepsilon_{\text{min},2} = 0.05$, $\Delta\varepsilon_{\text{max},2} = 0.10$. Based on our simulations, the most critical part of the algorithm is during the first iterations when $\varepsilon = 1$ and the linear constraint violations are large. This is often the case if the initial guess is poor. Therefore, it is convenient to use small values for $\Delta\varepsilon_{\text{min}}$ and ε_{max} at the beginning. This means that the optimal solution of the problem with the updated ε changes only slightly, and hence, the solver is more likely to find a solution. Once the algorithm makes progress, $\Delta\varepsilon_{\text{min}}$ and $\Delta\varepsilon_{\text{max}}$ are increased to avoid an excessive number of iterations. If the homotopic parameter changes, the constraint violations are updated and the trust-region size is expanded based on Eq. (8.21). The process continues until an optimal solution is found.

8.2 No-Thrust Constraints

Although operational constraints are important for real space missions, they are rarely included in the optimization process. The obtained thrust profiles in the literature often require continuous thrust for several weeks or even months. In reality, this is not feasible, for example due to hardware or mission constraints. Duty cycles are often imposed where thrusting and coasting periods alternate. In an autonomous guidance scenario where the spacecraft itself has to determine its reference trajectory, it is of paramount importance that mission-compliant trajectories are computed. In this dissertation, we focus on operational constraints where the thruster has to remain off for certain periods. We consider these the most challenging ones because other constraints, such as thruster pointing constraints, only restrict the direction of the thrust. We extend the approach in [136] where the ZOH discretization, linear dynamics, and a single and fixed thrusting period per segment are considered for small-body proximity operations.

Given a trajectory segment $[t_k, t_{k+1}]$, we define n required no-thrust periods in this segment as

$$[t_i, t_i + \Delta t_{\text{off},i}], \quad \forall i = 1, \ldots, n \tag{8.22}$$

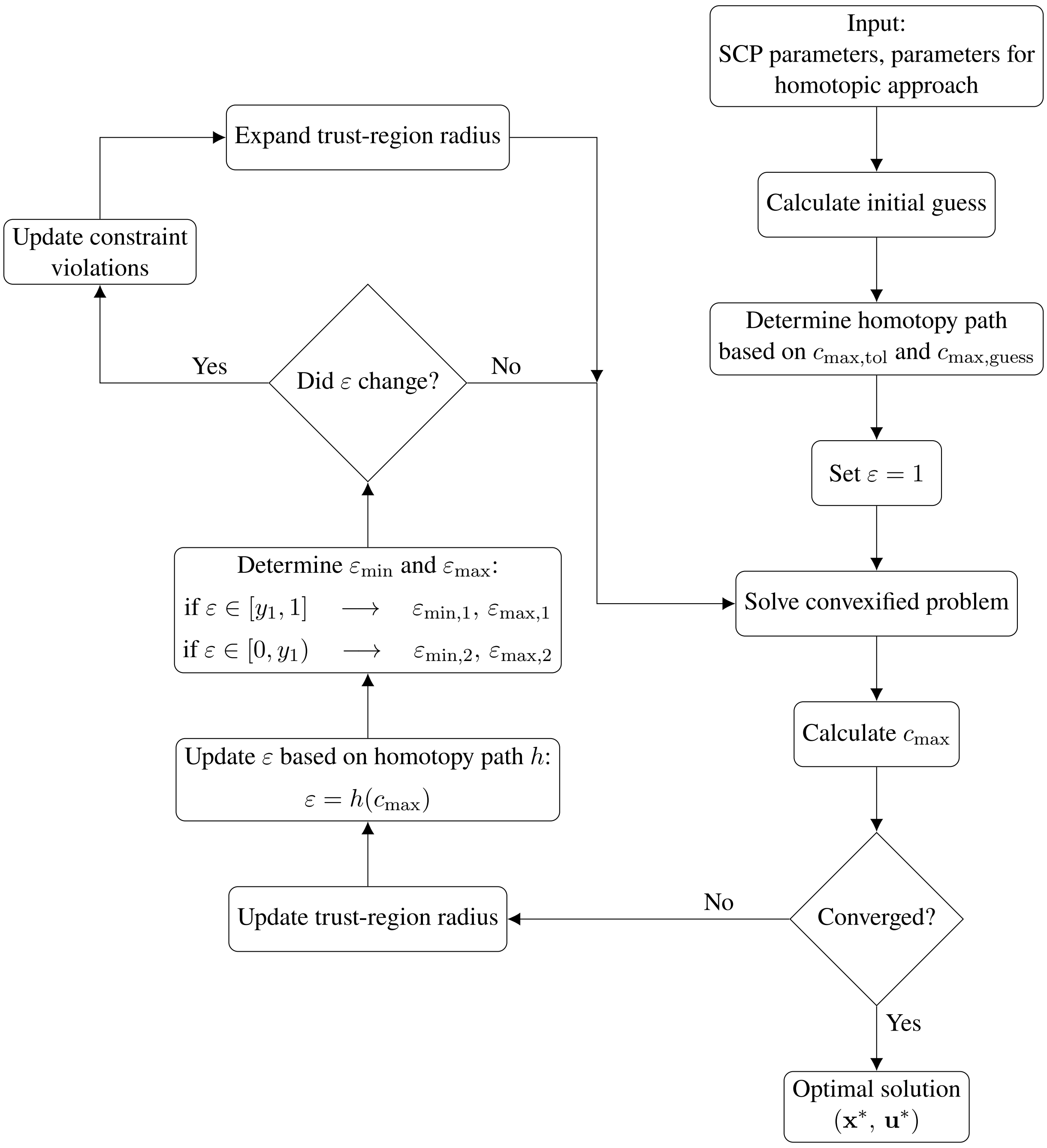

Input:
SCP parameters, parameters for homotopic approach
Calculate initial guess
Determine homotopy path based on $c_{\max,\mathrm{tol}}$ and $c_{\max,\mathrm{guess}}$
Set $\varepsilon = 1$
Solve convexified problem
Calculate $c_{\max}$
Converged?
Yes
Optimal solution $(\mathbf{x}^*,\, \mathbf{u}^*)$
No
Update trust-region radius
Update ε based on homotopy path h: $\varepsilon = h(c_{\max})$
Determine $\varepsilon_{\min}$ and $\varepsilon_{\max}$:
if $\varepsilon \in [y_1, 1] \longrightarrow \varepsilon_{\min,1},\ \varepsilon_{\max,1}$
if $\varepsilon \in [0, y_1) \longrightarrow \varepsilon_{\min,2},\ \varepsilon_{\max,2}$
Did ε change?
Yes
No
Update constraint violations
Expand trust-region radius

Figure 8.8: Flowchart of the homotopic approach.

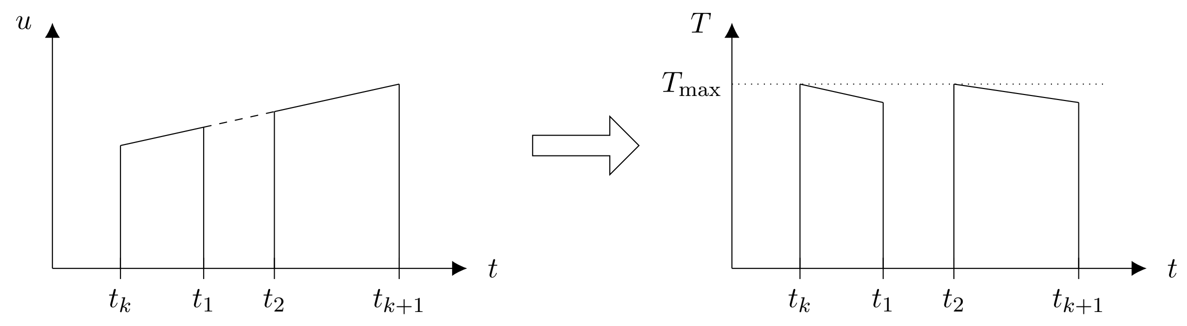


Figure 8.9: Standard linear interpolation for no-thrust periods and corresponding thrust magnitude curve.

where $t_i > t_k$ and $t_i + \Delta t_{\text{off},i} < t_{k+1}$ without loss of generality, i.e., the no-thrust periods $\Delta t_{\text{off},i}$ lie strictly within a trajectory segment. This assumption is reasonable because the location of the nodes (and thus, endpoints of a segment) can be chosen arbitrarily. Moreover, Δt_{off} is in general much smaller than one segment. If $i > 1$, it is assumed that the no-thrust periods are not overlapping, i.e., $t_{i+1} > t_i + \Delta t_{\text{off},i}$. Therefore,

$$\mathbf{u}(t) = \mathbf{0}, \qquad \forall t \in [t_i, t_i + \Delta t_{\text{off},i}] \tag{8.23}$$

and

$$\mathbf{u}(t) = \tilde{\lambda}_-(t)\,\mathbf{u}_k + \tilde{\lambda}_+(t)\,\mathbf{u}_{k+1}, \qquad \forall t \notin [t_i, t_i + \Delta t_{\text{off},i}] \tag{8.24}$$

where $\tilde{\lambda}_-(t)$ and $\tilde{\lambda}_+(t)$ are the factors that define the control interpolation. Note that these must be chosen differently compared to the standard formulation in Eq. (6.64). A fuel-optimal solution yields a bang-bang control structure where the magnitude $u = T_{\max}$ or $u = 0$. As we defined the accelerations as our control variables, this piecewise constant relationship does not hold anymore; instead, the magnitude of the acceleration Γ increases over time due to the decreasing mass when thrusting. A standard linear interpolation would yield a jump in the thrust magnitude curve as the acceleration is erroneously assumed increasing even during no-thrust periods. This scenario is illustrated in Fig. 8.9 with one no-thrust period $[t_1, t_2]$ in a segment $[t_k, t_{k+1}]$. Depending on whether the constant or variable maximum thrust and specific impulse case is considered, two different interpolation methods are presented in the following subsections to resolve this problem.

Regardless of the interpolation method and assuming one no-thrust period $[t_1, t_2]$ in a segment $[t_k, t_{k+1}]$, the state at t_1 can be computed as:

$$\begin{aligned}\mathbf{x}(t_1) = {}& \mathbf{\Phi}(t_1,t_k)\,\mathbf{x}(t_k) + \mathbf{\Phi}(t_1,t_k)\int_{t_k}^{t_1}\mathbf{\Phi}^{-1}(\xi,t_k)\,\tilde{\lambda}_-(\xi)\,\mathbf{B}(\xi)\,\mathrm{d}\xi\,\mathbf{u}(t_k)\\ &+ \mathbf{\Phi}(t_1,t_k)\int_{t_k}^{t_1}\mathbf{\Phi}^{-1}(\xi,t_k)\,\tilde{\lambda}_+(\xi)\,\mathbf{B}(\xi)\,\mathrm{d}\xi\,\mathbf{u}(t_1) + \mathbf{\Phi}(t_1,t_k)\int_{t_k}^{t_1}\mathbf{\Phi}^{-1}(\xi,t_k)\,\mathbf{q}(\xi)\,\mathrm{d}\xi\end{aligned} \tag{8.25}$$

Next, the final state $\mathbf{x}(t_2)$ of the coasting period is calculated:

$$\mathbf{x}(t_2) = \mathbf{\Phi}(t_2, t_1)\, \mathbf{x}(t_1) + \mathbf{\Phi}(t_2, t_1) \int_{t_1}^{t_2} \mathbf{\Phi}^{-1}(\xi, t_1)\, \mathbf{q}(\xi)\, \mathrm{d}\xi \tag{8.26}$$

Note that there is no control term. The final step is to calculate the state at the end of the segment:

$$\begin{aligned}\mathbf{x}(t_{k+1}) = {}& \mathbf{\Phi}(t_{k+1}, t_2)\, \mathbf{x}(t_2) + \mathbf{\Phi}(t_{k+1}, t_2) \int_{t_2}^{t_{k+1}} \mathbf{\Phi}^{-1}(\xi, t_2)\, \tilde{\lambda}_-(\xi)\, \mathbf{B}(\xi)\, \mathrm{d}\xi\, \mathbf{u}(t_2) \\ & + \mathbf{\Phi}(t_{k+1}, t_2) \int_{t_2}^{t_{k+1}} \mathbf{\Phi}^{-1}(\xi, t_2)\, \tilde{\lambda}_+(\xi)\, \mathbf{B}(\xi)\, \mathrm{d}\xi\, \mathbf{u}(t_{k+1}) \\ & + \mathbf{\Phi}(t_{k+1}, t_2) \int_{t_2}^{t_{k+1}} \mathbf{\Phi}^{-1}(\xi, t_k)\, \mathbf{q}(\xi)\, \mathrm{d}\xi \end{aligned} \tag{8.27}$$

Substituting Eq. (8.25) into Eq. (8.26), and Eq. (8.26) into Eq. (8.27) yields

$$\begin{aligned}\mathbf{x}(t_{k+1}) = {}& \mathbf{A}(t_2)\, \mathbf{A}(t_1)\, \mathbf{A}(t_k)\, \mathbf{x}(t_k) + \mathbf{A}(t_2)\, \mathbf{A}(t_1)\, \mathbf{B}^-(t_k)\, \mathbf{u}(t_k) + \mathbf{A}(t_2)\, \mathbf{A}(t_1)\, \mathbf{B}^+(t_k)\, \mathbf{u}(t_1) \\ & + \mathbf{A}(t_2)\, \mathbf{A}(t_1)\, \mathbf{q}(t_k) + \mathbf{A}(t_2)\, \mathbf{q}(t_1) + \mathbf{B}^-(t_2)\, \mathbf{u}(t_1) + \mathbf{B}^+(t_2)\, \mathbf{u}(t_2) + \mathbf{q}(t_2)\end{aligned} \tag{8.28}$$

where

$$\mathbf{A}(t_{k+1}, t_2) = \mathbf{\Phi}(t_{k+1}, t_2),\ \mathbf{A}(t_2, t_1) = \mathbf{\Phi}(t_2, t_1),\ \mathbf{A}(t_2, t_k) = \mathbf{\Phi}(t_2, t_k) \tag{8.29a}$$

$$\mathbf{B}^-(t_{k+1}, t_2) = \mathbf{\Phi}(t_{k+1}, t_2) \int_{t_2}^{t_{k+1}} \mathbf{\Phi}^{-1}(\xi, t_k)\, \mathbf{B}(\xi)\, \tilde{\lambda}_-(\xi)\, \mathrm{d}\xi \tag{8.29b}$$

$$\mathbf{B}^+(t_{k+1}, t_2) = \mathbf{\Phi}(t_{k+1}, t_2) \int_{t_2}^{t_{k+1}} \mathbf{\Phi}^{-1}(\xi, t_k)\, \mathbf{B}(\xi)\, \tilde{\lambda}_+(\xi)\, \mathrm{d}\xi \tag{8.29c}$$

$$\mathbf{B}^-(t_1, t_k) = \mathbf{\Phi}(t_1, t_k) \int_{t_k}^{t_1} \mathbf{\Phi}^{-1}(\xi, t_k)\, \mathbf{B}(\xi)\, \tilde{\lambda}_-(\xi)\, \mathrm{d}\xi \tag{8.29d}$$

$$\mathbf{B}^+(t_1, t_1) = \mathbf{\Phi}(t_1, t_k) \int_{t_k}^{t_1} \mathbf{\Phi}^{-1}(\xi, t_k)\, \mathbf{B}(\xi)\, \tilde{\lambda}_+(\xi)\, \mathrm{d}\xi \tag{8.29e}$$

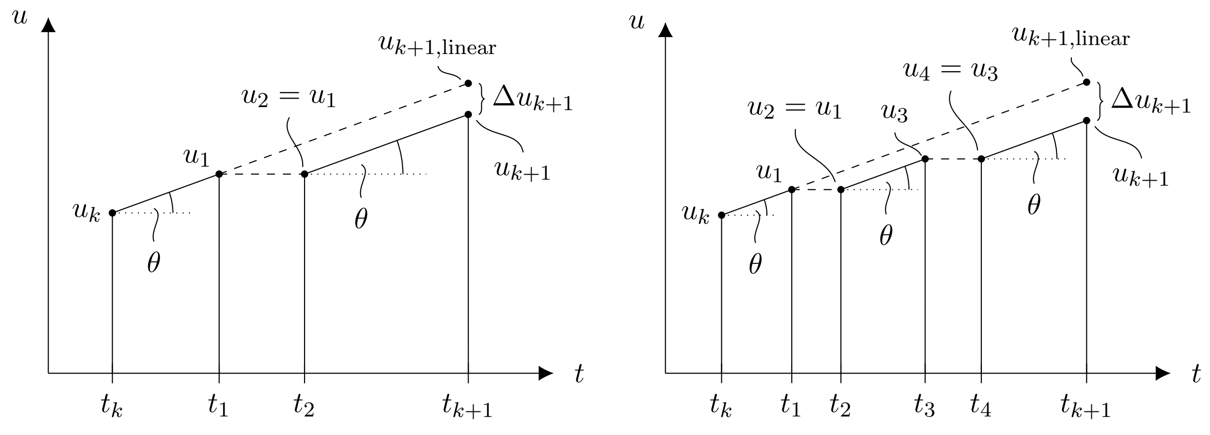
$$\mathbf{q}(t_{k+1}, t_2) = \mathbf{\Phi}(t_{k+1}, t_2) \int_{t_2}^{t_{k+1}} \mathbf{\Phi}^{-1}(\xi, t_k)\, \mathbf{q}(\xi)\, \mathrm{d}\xi \tag{8.29f}$$

$$\mathbf{q}(t_2, t_1) = \mathbf{\Phi}(t_2, t_1) \int_{t_1}^{t_2} \mathbf{\Phi}^{-1}(\xi, t_1)\, \mathbf{q}(\xi)\, \mathrm{d}\xi \tag{8.29g}$$

$$\mathbf{q}(t_1, t_k) = \mathbf{\Phi}(t_1, t_k) \int_{t_k}^{t_1} \mathbf{\Phi}^{-1}(\xi, t_k)\, \mathbf{q}(\xi)\, \mathrm{d}\xi \tag{8.29h}$$

$\mathbf{B}(\xi)$ and $\mathbf{q}(\xi)$ are the Jacobian matrix and constant part of the linearization, respectively, as defined in Eqs. (5.11) and (5.14) in Section 5.1. It is straightforward to extend this approach to more no-thrust periods.

The only unknowns are the interpolated controls $\mathbf{u}(t_1)$ and $\mathbf{u}(t_2)$. Two methods are proposed to show how they can be determined.



(a) One no-thrust period in one segment. **(b)** Two no-thrust periods in one segment.

Figure 8.10: Modified linear interpolation when no-thrust periods are considered with constant T_{max} and I_{sp}.

Constant Maximum Thrust and Specific Impulse

When T_{max} is constant, we have $T(t) = T_{\text{max}}$ for thrusting periods, $T(t) := \|\mathbf{T}(t)\|_2$. Therefore, $\dot{m}(t) = T_{\text{max}}/(g_0\, I_{\text{sp}})$ is also constant $\forall t \notin [t_i, t_i + \Delta t_{\text{off},i}]$ because $I_{\text{sp}} = \text{const.}$, and the mass decreases linearly. As $\Gamma(t) = T(t)/m(t)$, we thus propose a piecewise linear interpolation where the slope of $\Gamma(t)$ (and hence the angle θ) is constant. This way the resulting thrust magnitude is piecewise constant in that segment as expected. Figure 8.10a illustrates the interpolated thrust acceleration in one segment with one no-thrust period. Given the optimization variables $\mathbf{u}_k$ and $\mathbf{u}_{k+1}$ and times $\Delta t_1 = t_1 - t_k$ and $\Delta t_2 = t_{k+1} - t_2$, and using

$$\tan\theta = \frac{\Delta \mathbf{u}_1}{\Delta t_1} = \frac{\mathbf{u}_1 - \mathbf{u}_k}{\Delta t_1} \tag{8.30}$$

$$\tan\theta = \frac{\Delta \mathbf{u}_2}{\Delta t_2} = \frac{\mathbf{u}_{k+1} - \mathbf{u}_1}{\Delta t_2} \tag{8.31}$$

the interpolated controls $\mathbf{u}_1 = \mathbf{u}(t_1) = \mathbf{u}_2 = \mathbf{u}(t_2)$ can be computed as follows:

$$\mathbf{u}_1 = \mathbf{u}_2 = \underbrace{\frac{\Delta t_2}{\Delta t_1 + \Delta t_2}}_{\lambda_-(t)}\, \mathbf{u}_k + \underbrace{\frac{\Delta t_1}{\Delta t_1 + \Delta t_2}}_{\lambda_+(t)}\, \mathbf{u}_{k+1} \tag{8.32}$$

It is straightforward to determine the interpolated controls for $n > 1$ off periods. For example, for $n = 2$ (see Fig. 8.10b) we obtain

$$\mathbf{u}_1 = \mathbf{u}_2 = \frac{\Delta t_2 + \Delta t_3}{\Delta t_1 + \Delta t_2 + \Delta t_3}\, \mathbf{u}_k + \frac{\Delta t_1}{\Delta t_1 + \Delta t_2 + \Delta t_3}\, \mathbf{u}_{k+1} \tag{8.33}$$

$$\mathbf{u}_3 = \mathbf{u}_4 = \frac{\Delta t_3}{\Delta t_1 + \Delta t_2 + \Delta t_3}\, \mathbf{u}_k + \frac{\Delta t_1 + \Delta t_2}{\Delta t_1 + \Delta t_2 + \Delta t_3}\, \mathbf{u}_{k+1} \tag{8.34}$$

Variable Maximum Thrust and Specific Impulse

The variable maximum thrust and specific impulse case requires a different approach as $\dot{m}$ cannot be considered constant anymore. Instead of linearly interpolating the accelerations, we propose to approximate the thrust vector $\tilde{\mathbf{T}} := [\mathbf{T}^\top, T(t)]^\top$ with affine functions:

$$\tilde{\mathbf{T}}(t) = \underbrace{\frac{t_{k+1} - t}{t_{k+1} - t_k}}_{=: \lambda_-(t)} \tilde{\mathbf{T}}_k + \underbrace{\frac{t - t_k}{t_{k+1} - t_k}}_{=: \lambda_+(t)} \tilde{\mathbf{T}}_{k+1} = \lambda_-(t)\, \tilde{\mathbf{T}}_k + \lambda_+(t)\, \tilde{\mathbf{T}}_{k+1}, \qquad t \in [t_k, t_{k+1}] \tag{8.35}$$

Substituting $\tilde{\mathbf{T}}(t) = \mathbf{u}(t)\, m(t)$ into Eq. (8.35) and solving for $\mathbf{u}(t)$ yields

$$\begin{aligned} \mathbf{u}(t) &= \frac{\lambda_-(t)\, m(t_k)}{m(t)} \mathbf{u}_k + \frac{\lambda_+(t)\, m(t_{k+1})}{m(t)} \mathbf{u}_{k+1} \\ &\approx \frac{\lambda_-(t)\, \bar{m}(t_k)}{\bar{m}(t)} \mathbf{u}_k + \frac{\lambda_+(t)\, \bar{m}(t_{k+1})}{\bar{m}(t)} \mathbf{u}_{k+1} \end{aligned} \tag{8.36}$$

where the mass is approximated using the value of the reference trajectory. Evaluating Eq. (8.36) at t_i ($i = 1, 2$ for one no-thrust period) allows us to determine the interpolated controls with respect to the accelerations. The only unknown is $\bar{m}(t_1) = \bar{m}(t_2)$ which can be calculated by integrating Eq. (2.27):

$$\bar{m}(t_1) = \bar{m}(t_k) + \int_{t_k}^{t_1} \dot{\bar{m}}\, \mathrm{d}t = \bar{m}(t_k) - \frac{1}{g_0} \int_{t_k}^{t_1} \frac{\bar{T}(t)}{\bar{I}_{\text{sp}}(t)}\, \mathrm{d}t \tag{8.37}$$

As in Eq. (8.35), $\bar{T}(t)$ and $\bar{I}_{\text{sp}}(t)$ are assumed to be piecewise affine functions. Recalling that $m = \mathrm{e}^w$, $\bar{m}(t_1)$ can be computed analytically. It follows that

$$\mathbf{u}(t_1) = \underbrace{\frac{\lambda_-(t_1)\, \bar{m}(t_k)}{\bar{m}(t_1)}}_{=: \tilde{\lambda}_-(t_1)} \mathbf{u}_k + \underbrace{\frac{\lambda_+(t_1)\, \bar{m}(t_{k+1})}{\bar{m}(t_1)}}_{=: \tilde{\lambda}_+(t_1)} \mathbf{u}_{k+1} = \tilde{\lambda}_-(t_1)\, \mathbf{u}_k + \tilde{\lambda}_+(t_1)\, \mathbf{u}_{k+1} \tag{8.38}$$

$$\mathbf{u}(t_2) = \underbrace{\frac{\lambda_-(t_2)\, \bar{m}(t_k)}{\bar{m}(t_2)}}_{=: \tilde{\lambda}_-(t_2)} \mathbf{u}_k + \underbrace{\frac{\lambda_+(t_2)\, \bar{m}(t_{k+1})}{\bar{m}(t_2)}}_{=: \tilde{\lambda}_+(t_2)} \mathbf{u}_{k+1} = \tilde{\lambda}_-(t_2)\, \mathbf{u}_k + \tilde{\lambda}_+(t_2)\, \mathbf{u}_{k+1} \tag{8.39}$$

Our simulations suggest that this successive approximation approach yields a good compromise in terms of convergence, accuracy, and computational effort.

8.3 Bang-Bang Mesh Refinement

A common problem in direct methods is that the constraints are only satisfied at discrete points. Therefore, the solution is to be interpolated to obtain continuous state and control profiles. With regard to fuel-optimal problems with bang-bang control structures, approximating the control with higher-order polynomials results in oscillations at the edges of each segment due to the Runge phenomenon. Although low-order approximations may capture the discontinuous control profile more accurately, it is very unlikely that the initially chosen mesh can accurately represent the bang-bang behavior. Instead, controls are obtained that are neither $u_{\max}$ nor $u_{\min}$. Therefore, this section presents a mesh refinement method that yields accurate bang-bang control profiles.

The goal is to obtain a control profile where the control magnitude u_{mag} either takes its minimum $u_{\min}$ or maximum value $u_{\max}$, i.e., $u_{\mathrm{mag}} = u_{\min}$ or $u_{\mathrm{mag}} = u_{\max}$. Determining the switching times, that is, the times when the control changes from on to off or vice versa, requires the costates of the OCP. Once the switching times are known, the control structure is given, and the mesh refinement can be applied.

Computation of Costates and Switching Times for RPM/FRPM

After solving the discretized OCP, the resulting Lagrange multipliers $\mathbf{\Lambda}^{(k)} \in \mathbb{R}^{n_x \times N_k}$ are used to compute the values of the costates $\boldsymbol{\lambda}_i^{(k)} \in \mathbb{R}^{n_x}$ at the ith collocation point. The same notation as in Sections 6.1.1 and 6.1.2 is used.

As the first point of the first segment in FRPM is not collocated, the costates of the first segment are determined as follows using the covector mapping theorem in [119]:

$$\boldsymbol{\lambda}_0^{(1)} = -\mathbf{\Lambda}^{(1)}\,\mathbf{D}_{*0}^{(1)} \tag{8.40}$$

$$\boldsymbol{\lambda}_i^{(1)} = \frac{\mathbf{\Lambda}_{*i}^{(1)}}{w_i^{(1)}}, \qquad i = 1, \ldots, N_k \tag{8.41}$$

where the notation $(\cdot)_{*i}$ indicates the ith column. Recalling the linking condition in Eqs. (6.47)–(6.49), the costates of the remaining segments are:

$$\boldsymbol{\lambda}_i^{(k)} = \frac{\mathbf{\Lambda}_{*i}^{(k)}}{w_i^{(k)}}, \qquad i = 1, \ldots, N_k, \quad k = 2, \ldots, K \tag{8.42}$$

In RPM, the last node of a segment is not collocated. Therefore, the costates of the last segment are calculated using [119]

$$\boldsymbol{\lambda}_{N_K}^{(K)} = -\mathbf{\Lambda}^{(K)}\,\mathbf{D}_{*N_K}^{(K)} \tag{8.43}$$

$$\boldsymbol{\lambda}_i^{(K)} = \frac{\mathbf{\Lambda}_{*i}^{(K)}}{w_i^{(K)}}, \qquad i = 1, \ldots, N_K - 1 \tag{8.44}$$

For the remaining segments, we use the linking conditions in Eqs. (6.5)–(6.7) to obtain

$$\boldsymbol{\lambda}_i^{(k)} = \frac{\mathbf{\Lambda}_{*i}^{(k)}}{w_i^{(k)}}, \qquad i = 1, \ldots, N_k, \quad k = 1, \ldots, K - 1 \tag{8.45}$$

We rewrite the equations of motion as

$$\mathbf{f} = \begin{bmatrix} \dot{\mathbf{r}} \\ \dot{\mathbf{v}} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} \mathbf{v} \\ \mathbf{g}(\mathbf{r}) + \Gamma\,\boldsymbol{\alpha} \\ -\Gamma/(g_0\,I_{\mathrm{sp}}) \end{bmatrix} \tag{8.46}$$

with $\mathbf{g}(\mathbf{r}) = -\mu\,\mathbf{r}/r^3$, $r = \|\mathbf{r}\|_2$, and the thrust direction unit vector $\boldsymbol{\alpha}$. Considering the objective function $J = -w(t_f)$, the Hamiltonian of the problem can be stated as

$$\mathcal{H} = L + \boldsymbol{\lambda}^\top \mathbf{f} = \boldsymbol{\lambda}_r^\top \mathbf{v} + \boldsymbol{\lambda}_v^\top \left[\mathbf{g}(\mathbf{r}) + \Gamma\,\boldsymbol{\alpha}\right] - \lambda_w \frac{\Gamma}{g_0\,I_{\mathrm{sp}}} \tag{8.47}$$

because $L = 0$. $\boldsymbol{\lambda}_r$, $\boldsymbol{\lambda}_v$, and λ_w refer to the costates associated with the position, velocity, and modified mass, respectively. Substituting the optimal thrust direction $\boldsymbol{\alpha}^* = -\boldsymbol{\lambda}_v/\|\boldsymbol{\lambda}_v\|_2 = -\boldsymbol{\lambda}_v/\lambda_v$ with $\lambda_v = \|\boldsymbol{\lambda}_v\|_2$ into Eq. (8.47) and rearranging terms yields

$$\mathcal{H} = \boldsymbol{\lambda}_r^\top \mathbf{v} + \boldsymbol{\lambda}_v^\top \mathbf{g}(\mathbf{r}) + \Gamma \underbrace{\left(-\lambda_v - \frac{\lambda_w}{g_0\,I_{\mathrm{sp}}}\right)}_{=:\,S} \tag{8.48}$$

According to Pontryagin's minimum principle [17], an optimal trajectory minimizes the Hamiltonian and hence, the characteristic bang-bang control profile in fuel-optimal problems solely depends on the sign of the switching function S. The optimal Γ^* is therefore given by

$$\Gamma^* = \begin{cases} 0 & \text{if } S > 0 \\ \Gamma_{\max} & \text{if } S < 0 \end{cases} \tag{8.49}$$

As a consequence, the switching times (i.e., when S changes its sign) and control structure can be computed once the costates are known. However, because $\Gamma_{\max}$ depends on the mass and is therefore not constant, it is often beneficial to use the thrust components $\mathbf{T}$ and magnitude T as the control variables instead of accelerations, and the mass m instead of w. Using the corresponding objective function $J = -m(t_f)$, the Hamiltonian function $\tilde{\mathcal{H}}$ reads

$$\tilde{\mathcal{H}} = \boldsymbol{\lambda}_r^\top \mathbf{v} + \boldsymbol{\lambda}_v^\top \mathbf{g}(\mathbf{r}) + T \underbrace{\left(-\frac{\lambda_v}{m} - \frac{\lambda_m}{g_0\,I_{\mathrm{sp}}}\right)}_{=:\,\tilde{S}} \tag{8.50}$$

The optimal T^* is then

$$T^* = \begin{cases} 0 & \text{if } \tilde{S} > 0 \\ T_{\max} & \text{if } \tilde{S} < 0 \end{cases} \tag{8.51}$$

Therefore, assuming constant specific impulse and maximum thrust, $T_{\max}$ is also piecewise constant.

Remark 8.3. *Using the linearized dynamics in the formulation of the Hamiltonian*

$$\mathcal{H} = L + \boldsymbol{\lambda}^\top \left[\mathbf{A}(\bar{\mathbf{x}}, \bar{\mathbf{u}})\, \mathbf{x} + \mathbf{B}(\bar{\mathbf{x}}, \bar{\mathbf{u}})\, \mathbf{u} + \mathbf{q}(\bar{\mathbf{x}}, \bar{\mathbf{u}})\right] \tag{8.52}$$

to calculate the costates and switching times will yield similar results because a converged SCP solution satisfies the nonlinear dynamics. We compared both versions in our numerical simulations and did not notice significant differences.

Computation of Costates and Switching Times for FOH

As there is no equivalent covector mapping theorem for FOH, we use a different approach to determine the switching times. We make use of the necessary conditions in Eqs. (2.8), (2.9), (2.11) and (2.12)

$$\dot{\mathbf{x}} = \mathcal{H}_{\boldsymbol{\lambda}}^\top \tag{8.53}$$

$$\dot{\boldsymbol{\lambda}} = -\mathcal{H}_{\mathbf{x}}^\top \tag{8.54}$$

$$\left[\boldsymbol{\lambda}^\top - \phi_{\mathbf{x}} - \boldsymbol{\kappa}^\top \mathbf{h}_{\mathbf{x}}\right]_{t_f} = \mathbf{0}^\top \tag{8.55}$$

$$[\mathbf{h}]_{t_f} = \mathbf{0} \tag{8.56}$$

where

$$\phi(\mathbf{x}(t_f)) = -w(t_f) \tag{8.57}$$

$$\mathbf{h}(\mathbf{x}(t_f)) = \begin{bmatrix} \mathbf{r}(t_f) - \mathbf{r}_f \\ \mathbf{v}(t_f) - \mathbf{v}_f \end{bmatrix} \tag{8.58}$$

Assuming $\mathbf{x} = [\mathbf{r}^\top, \mathbf{v}^\top, w]^\top$ with $\mathbf{r} \in \mathbb{R}^{n_r}$, $\mathbf{v} \in \mathbb{R}^{n_v}$, and $w \in \mathbb{R}^{n_w}$, the first $q := n_r + n_v$ states are prescribed at the final time t_f. Because the final mass is free, Eq. (8.55) reduces to

$$\lambda_j(t_f) = \begin{cases} \kappa_j, & j = 1, \dots, q \\ \dfrac{\partial \phi}{\partial w} = -1, & j = q+1, \dots, n_x \end{cases} \tag{8.59}$$

In case of Cartesian coordinates, we have $n_x = 7$, $n_r = n_v = 3$, and $n_w = 1$. This means that the final costate of the modified final mass $w(t_f)$ is -1, and the final costates of the prescribed states $\mathbf{r}$ and $\mathbf{v}$ at t_f are simply the constant multipliers $\boldsymbol{\kappa} \in \mathbb{R}^q$. As these are obtained in the optimization process and returned by the solver as solution variables, it is possible to back-propagate the dynamics of the states and costates simultaneously using the optimized states and controls to determine the continuous profiles of the costates. According to Eqs. (8.53) and (8.54), taking the partial derivative of the Hamiltonian with respect to the costate $\boldsymbol{\lambda}$ and state $\mathbf{x}$ yields the full set of equations of motion:

$$\begin{bmatrix} \dot{\mathbf{r}} \\ \dot{\mathbf{v}} \\ \dot{w} \\ \dot{\boldsymbol{\lambda}}_r \\ \dot{\boldsymbol{\lambda}}_v \\ \dot{\lambda}_w \end{bmatrix} = \begin{bmatrix} \mathbf{v} \\ -\dfrac{\mu}{r^3}\,\mathbf{r} - \Gamma\,\dfrac{\boldsymbol{\lambda}_v}{\lambda_v} \\ -\dfrac{\Gamma}{g_0\,I_{\text{sp}}} \\ \dfrac{\mu}{r^3}\,\boldsymbol{\lambda}_v - \dfrac{3\,\mu}{r^5}\,\mathbf{r}^\top\,\boldsymbol{\lambda}_v\,\mathbf{r} \\ -\boldsymbol{\lambda}_r \\ 0 \end{bmatrix} \tag{8.60}$$

Remark 8.4. *Using the optimized control components $\boldsymbol{\tau}^* = \Gamma^*\,\boldsymbol{\alpha}^*$ directly in Eq.* (8.60) *for $\dot{\mathbf{v}}$ instead of the optimal thrust direction $\boldsymbol{\alpha}^* = -\boldsymbol{\lambda}_v/\lambda_v$ is equivalent due to the optimality of the solution.*

Mesh Refinement for RPM and FRPM

The solution to the optimal control problem is only an approximation as states and controls are known only at the discretization points. Therefore, any direct method can intrinsically not determine a discontinuous control structure accurately. The costates and switching times (that is, the zeros of S or $\tilde{S}$) are also only estimations. The values of the switching times can be refined by incorporating them into the optimization process to eventually obtain an accurate bang-bang control. In [143], a similar approach was applied to simple NLP problems. We adapt the bang-bang mesh refinement to solve complex low-thrust optimization problems in a convex programming environment for the first time. The complete procedure is depicted in Fig. 8.11. We limit ourselves to FRPM; it is straightforward to extend the approach to RPM.

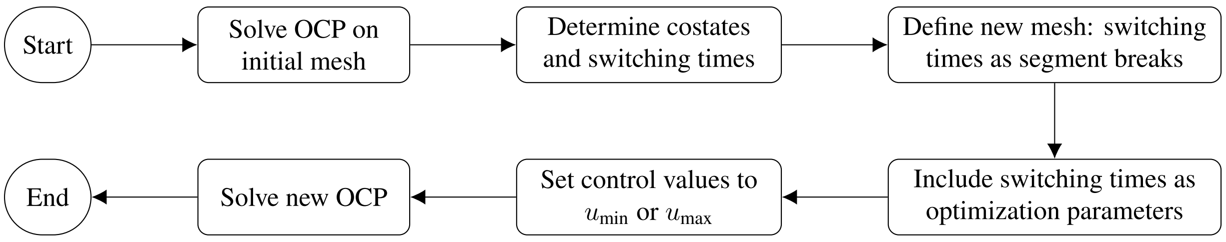

Start
Solve OCP on initial mesh
Determine costates and switching times
Define new mesh: switching times as segment breaks
Include switching times as optimization parameters
Set control values to u_{min} or u_{max}
Solve new OCP
End

Figure 8.11: Flowchart of bang-bang mesh refinement process for RPM/FRPM.

We define the vector of all switching times as $\mathbf{t}_s = [t_{s,1}, t_{s,2}, ..., t_{s,j}]^\top$ and divide the trajectory into $K = j + 1$ segments where j is the number of switching times. Thus, each $t_{s,j}$ lies on the corner of a segment as shown in Fig. 8.12.

The factors $\Delta^{(k)}$ for the time transformation then become

$$\Delta^{(1)} = \frac{t_{s,1} - t_0}{2}, \qquad \Delta^{(2)} = \frac{t_{s,2} - t_{s,1}}{2}, \qquad \ldots, \qquad \Delta^{(K)} = \frac{t_f - t_{s,j}}{2} \tag{8.61}$$

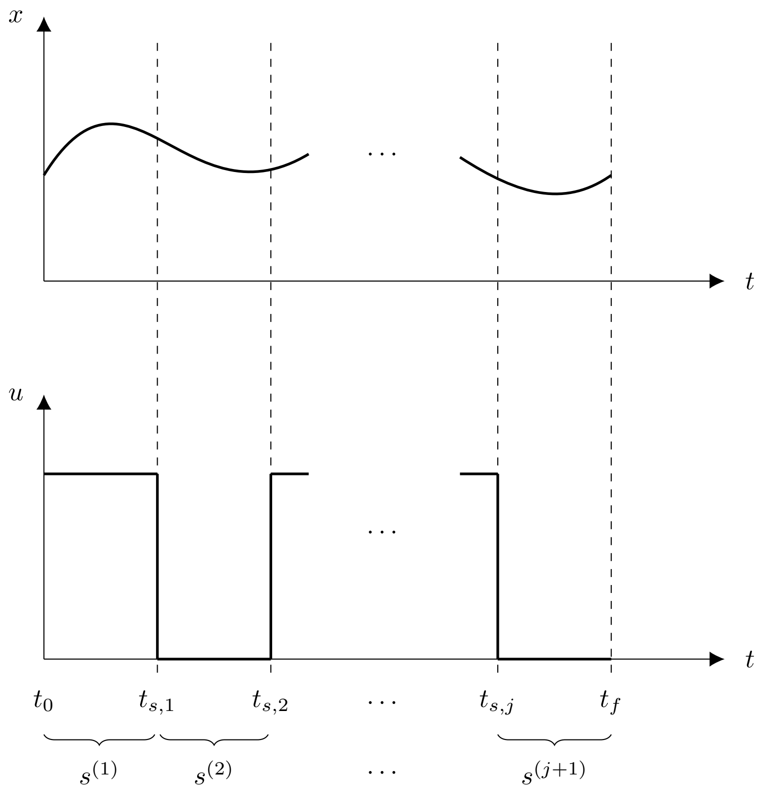


Figure 8.12: Two-dimensional illustration of switching times t_s and segments $s^{(k)}$ for some state x and control u curves.

and cause the formerly convex dynamical constraints in Eq. (6.54) to become nonconvex. Introducing and linearizing

$$\mathbf{F}(\mathbf{x}_i^{(k)}, \mathbf{u}_i^{(k)}, t_0^{(k)}, t_{N_k}^{(k)}) := \frac{t_{N_k}^{(k)} - t_0^{(k)}}{2}\,\mathbf{f}(\mathbf{x}_i^{(k)}, \mathbf{u}_i^{(k)}), \qquad i = 1, \dots, N_k, \quad k = 1, \dots, K \tag{8.62}$$

at $\left(\bar{\mathbf{x}}_i^{(k)}, \bar{\mathbf{u}}_i^{(k)}, \bar{t}_0^{(k)}, \bar{t}_{N_k}^{(k)}\right)$ yields the dynamical constraints:

$$\begin{gathered} D_{i0}^{(k)}\mathbf{x}_0^{(k)} + \sum_{j=1}^{N_k} D_{ij}^{(k)}\mathbf{x}_j^{(k)} = \mathbf{A}_i^{(k)}\,\mathbf{x}_i^{(k)} + \mathbf{B}_i^{(k)}\,\mathbf{u}_i^{(k)} + \mathbf{C}_i^{(k)}\,t_0^{(k)} + \mathbf{E}_i^{(k)}\,t_{N_k}^{(k)} + \mathbf{q}_i^{(k)} + \boldsymbol{\nu}_j^{(k)}, \\ i = 1, \dots, N_k, \quad k = 1, \dots, K \end{gathered} \tag{8.63}$$

with the same notation as in Section 6.1.2, and

$$\mathbf{C}_i^{(k)} := \nabla_{t_0}\mathbf{F}\left(\bar{\mathbf{x}}_\mathbf{i}^{(\mathbf{k})}, \bar{\mathbf{u}}_\mathbf{i}^{(\mathbf{k})}, \bar{\mathbf{t}}_\mathbf{0}^{(\mathbf{k})}, \bar{\mathbf{t}}_{\mathbf{N_k}}^{(\mathbf{k})}\right) \tag{8.64}$$

$$\mathbf{E}_i^{(k)} := \nabla_{t_{N_k}}\mathbf{F}\left(\bar{\mathbf{x}}_\mathbf{i}^{(\mathbf{k})}, \bar{\mathbf{u}}_\mathbf{i}^{(\mathbf{k})}, \bar{\mathbf{t}}_\mathbf{0}^{(\mathbf{k})}, \bar{\mathbf{t}}_{\mathbf{N_k}}^{(\mathbf{k})}\right) \tag{8.65}$$

$$\mathbf{q}_i^{(k)} := \mathbf{F}_i^{(k)} - \mathbf{A}_i^{(k)}\,\bar{\mathbf{x}}_\mathbf{i}^{(\mathbf{k})} - \mathbf{B}_i^{(k)}\,\bar{\mathbf{u}}_\mathbf{i}^{(\mathbf{k})} - \mathbf{C}_i^{(k)}\,\bar{t}_0^{(k)} - \mathbf{E}_i^{(k)}\,\bar{t}_{N_k}^{(k)} \tag{8.66}$$

$$\mathbf{F}_i^{(k)} := \mathbf{F}(\bar{\mathbf{x}}_i^{(k)}, \bar{\mathbf{u}}_i^{(k)}, \bar{t}_0^{(k)}, \bar{t}_{N_k}^{(k)}) \tag{8.67}$$

It follows that $\mathbf{C}_i^{(k)} \in \mathbb{R}^{n_x}$ and $\mathbf{E}_i^{(k)} \in \mathbb{R}^{n_x}$. The combined linear equality constraint in Eq. (6.21) changes to

$$\left[\begin{array}{ccc|ccc|ccc|c} \hat{\mathbf{A}}^{(1)} & & & \hat{\mathbf{B}}^{(1)} & & & \hat{\mathbf{1}}^{(1)} & & & \mathbf{T}_s^{(1)} \\ & & \mathbf{0} & & & \mathbf{0} & & & \mathbf{0} & \\ & \ddots & & & \ddots & & & \ddots & & \vdots \\ \mathbf{0} & & & \mathbf{0} & & & \mathbf{0} & & & \\ & & \hat{\mathbf{A}}^{(K)} & & & \hat{\mathbf{B}}^{(K)} & & & \hat{\mathbf{1}}^{(K)} & \mathbf{T}_s^{(K)} \end{array}\right] \begin{bmatrix} \mathbf{X} \\ \mathbf{U} \\ \boldsymbol{\nu} \\ \mathbf{t}_s \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{q}}^{(1)} \\ \vdots \\ \hat{\mathbf{q}}^{(K)} \end{bmatrix} \tag{8.68}$$

where

$$\mathbf{T}_s^{(1)} = \begin{bmatrix} -\mathbf{E}_1^{(1)} & \\ \vdots & \mathbf{0} \\ -\mathbf{E}_{N_1}^{(1)} & \end{bmatrix}, \mathbf{T}_s^{(2)} = \begin{bmatrix} -\mathbf{C}_1^{(2)} & -\mathbf{E}_1^{(2)} & \\ \vdots & \vdots & \mathbf{0} \\ -\mathbf{C}_{N_2}^{(2)} & -\mathbf{E}_{N_2}^{(2)} & \end{bmatrix}, \mathbf{T}_s^{(3)} = \begin{bmatrix} & -\mathbf{C}_1^{(3)} & -\mathbf{E}_1^{(3)} & \\ \mathbf{0}_{N_3 n_x \times 1} & \vdots & \vdots & \mathbf{0} \\ & -\mathbf{C}_{N_3}^{(3)} & -\mathbf{E}_{N_3}^{(3)} & \end{bmatrix},$$
$$\mathbf{T}_s^{(4)} = \begin{bmatrix} & -\mathbf{C}_1^{(4)} & -\mathbf{E}_1^{(4)} & \\ \mathbf{0}_{N_4 n_x \times 2} & \vdots & \vdots & \mathbf{0} \\ & -\mathbf{C}_{N_4}^{(4)} & -\mathbf{E}_{N_4}^{(4)} & \end{bmatrix}, \ldots, \mathbf{T}^{(K)} = \begin{bmatrix} & -\mathbf{C}_1^{(K)} \\ \mathbf{0} & \vdots \\ & -\mathbf{C}_{N_K}^{(K)} \end{bmatrix} \tag{8.69}$$

As the (estimated) switching times are known, the thrust magnitude $T(t_i^{(k)})$ at the ith node of segment k can be predefined based on the sign of the switching function $S(t_i^{(k)})$:

$$\begin{aligned} T(t_i^{(k)}) &= 0 && \text{if } S(t_i^{(k)}) > 0 \\ T(t_i^{(k)}) &= T_{\max} && \text{if } S(t_i^{(k)}) < 0 \end{aligned} \tag{8.70}$$

This is done by setting the upper and lower bounds of $T^{(k)}(t_i)$ accordingly. The linear constraints

$$t_0 \leq t_{s,1} \leq \ldots \leq t_{s,j} \leq t_f \tag{8.71}$$

are added to ensure the correct order of the switching times. The solution of the resulting optimization problem yields an accurate bang-bang control profile along with the optimized switching times.

Remark 8.5. *It is straightforward to include additional segment breaks to have more than $j+1$ segments, j being the number of switching times. Furthermore, the number of nodes per segment can be adjusted to achieve the desired accuracy.*

Mesh Refinement for FOH

Due to the explicit numerical integration of the dynamics and state transition matrix between two segments t_k and t_{k+1}, the accuracy of the solution is often higher compared to RPM/FRPM if a higher-

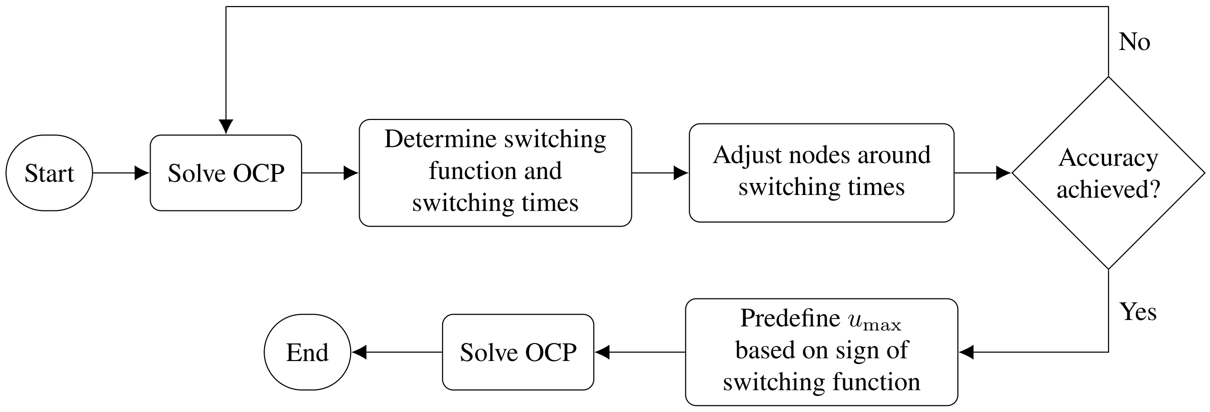

No
Start
Solve OCP
Determine switching function and switching times
Adjust nodes around switching times
Accuracy achieved?
Yes
Predefine $u_{\max}$ based on sign of switching function
Solve OCP
End

Figure 8.13: Flowchart of bang-bang mesh refinement process for FOH.

order integrator is chosen. Moreover, as the controls are interpolated linearly, the resulting control profile is often very accurate with respect to a bang-bang structure. It is therefore not necessary to include the switching times in the optimization. Instead, it is often sufficient to increase the number of nodes in the neighborhood of the switching times in order to achieve a sufficiently accurate control trajectory. The procedure is depicted in Fig. 8.13. First, the OCP is solved on the initial mesh. The switching times and switching function are determined using the costates. Once the locations of the switchings are known, the number of nodes can be increased in these regions to more accurately capture the on-off profile. These steps can be repeated until some desired accuracy is achieved, for example if the absolute or relative change of the computed switching times is smaller than some tolerance. Even though the control profile is often sufficiently accurate now, we can proceed similarly to RPM/FRPM and predefine the values of the control magnitude based on the sign of the switching function according to Eq. (8.70). This way, the control magnitude takes either its minimum or maximum value, thus representing the bang-bang structure accurately.

8.4 Numerical Simulations

The performance of the developed homotopic approach in terms of success rate, CPU time, and obtained final mass is compared with a standard SCP method without any homotopy, and the state-of-the-art optimal control software GPOPS-II [175] in combination with the Sparse Nonlinear Optimizer (SNOPT) [184]. In addition, it is assessed how SCP performs when no-thrust periods are included, and when mesh refinement is applied. All simulations are performed in MATLAB on an Intel Core i7-8565 1.80 GHz Laptop with four cores and 16 GB of RAM. The numerical integration of the state transition matrix in Eq. (6.67), the nonlinear dynamics in Eq. (5.11), and the integrands of Eqs. (6.68b)–(6.68d) within SCP is performed using a mex function. The second-order cone program in Eq. (5.23) is solved using the open-source Embedded Conic Solver (ECOS) [86]. In contrast, GPOPS-II solves the full nonlinear

Table 8.1: Simulation values for the transfers from SEL_2 to the asteroids 2000 SG344 and Dionysus [39, 129].

Parameter	SEL_2 - 2000 SG344	SEL_2 - Dionysus
Initial epoch	04-Feb-2024 12:00:00 UTC	23-Dec-2012 00:00:00 UTC
Initial position $\mathbf{r}_0$, AU	$[-0.70186065, 0.70623244, -3.51115 \times 10^{-5}]^\top$	$[-0.023941014, 0.99325372, -3.02763 \times 10^{-5}]^\top$
Initial velocity $\mathbf{v}_0$, VU	$[-0.73296949, -0.71590485, 4.40245 \times 10^{-5}]^\top$	$[-1.02637347, -0.02809721, 1.98538 \times 10^{-6}]^\top$
Initial mass m_0, kg	22.6	4000
Final position $\mathbf{r}_f$, AU	$[0.41806795, 0.82897114, -0.00143382]^\top$	$[-2.04061782, 2.05179130, 0.55428895]^\top$
Final velocity $\mathbf{v}_f$, VU	$[-0.96990332, 0.43630220, -0.00123381]^\top$	$[-0.14231932, -0.45108800, 0.01894690]^\top$
Final mass $m(t_f)$, kg	free	free
Min. input power $P_{\text{in,min}}$, W	90	62.5
Max. input power $P_{\text{in,max}}$, W	120	1000
Max. thrust T_{max}, N	2.2519×10^{-3}	0.5
Max. specific impulse $I_{\text{sp,max}}$, s	3067	3000
Spacecraft area A_{SC}, m^2	0.05	100
Reflectivity coefficient C_R	1.3	1.3
Time of flight t_f, days	700	3534

Table 8.2: Parameters of the algorithm.

Parameter	Value
Feasibility tol. ε_c	10^{-6}
Optimality tol. ε_J	10^{-4}, 10^{-5}
Max. iterations	1500, 3000
λ_ν, λ_η	10.0, 10.0
ρ_0, ρ_1, ρ_2	0.01, 0.25, 0.85
α, β	1.5, 1.5
r_{min}, AU	0.03

Table 8.3: Physical constants in all simulations.

Parameter	Value
Gravitational const. μ	$1.327\,124\,4 \times 10^{11}\,\mathrm{km^3\,s^{-2}}$
Gravitational accel. g_0	$9.806\,65 \times 10^{-3}\,\mathrm{km\,s^{-2}}$
Length unit LU = AU	$1.495\,978\,707 \times 10^{8}\,\mathrm{km}$
Velocity unit VU	$\sqrt{\mu\,\mathrm{AU}^{-1}}$
Time unit TU	$\mathrm{AU\,VU^{-1}}$
Acceleration unit ACU	$\mathrm{VU\,TU^{-1}}$
Mass unit MU	m_0

program directly. We compute fuel-optimal trajectories from the Sun-Earth Lagrange point L_2 (SEL_2) to the asteroids 2000 SG344 and Dionysus. The algorithms stop if the maximum constraint violation and change in the objective function are smaller than the feasibility ε_c and optimality ε_J tolerances, respectively. If not stated otherwise, the first-order-hold method is used to discretize the problem. Relevant parameters for the transfers and algorithms are given in Tables 8.1 and 8.2. Physical constants are given in Table 8.3, and parameters related to the homotopic approach in Table 8.4. n-body dynamics,

Table 8.4: Homotopy parameters for the transfers from SEL$_2$ to the asteroids 2000 SG344 and Dionysus.

Parameter	SEL$_2$ - 2000 SG344	SEL$_2$ - Dionysus
δ	2.0	
n, C_0, C_1, C_2	3, 1.8186, 5.3518, 0.3974	
γ	1.0	
y_1	0.75	
$\Delta\varepsilon_{\text{min},1}, \Delta\varepsilon_{\text{max},1}$	0.0, 0.025	
$\Delta\varepsilon_{\text{min},2}, \Delta\varepsilon_{\text{max},2}$	0.05, 0.10	
ΔP_{min}, W	0	0
ΔP_{max}, W	180	1500
ρ_{min}	10^{-4}	10^{-4}
ρ_{max}	5.0	5.0
P_{max}, W	250	2000

solar radiation pressure, and a real thruster model with variable specific impulse and maximum thrust are considered. In particular, the following thruster model is used for the transfer to 2000 SG344 [129]:

$$T_{\text{max}}(P_{\text{in}}) = a_0 + a_1 P_{\text{in}} + a_2 P_{\text{in}}^2 + a_3 P_{\text{in}}^3 + a_4 P_{\text{in}}^4 \tag{8.72a}$$

$$I_{\text{sp}}(P_{\text{in}}) = b_0 + b_1 P_{\text{in}} + b_2 P_{\text{in}}^2 + b_3 P_{\text{in}}^3 + b_4 P_{\text{in}}^4 \tag{8.72b}$$

$$P_{\text{in}}(r) = c_0 + c_1 r + c_2 r^2 + c_3 r^3 + c_4 r^4 \tag{8.72c}$$

with $a_0 = -0.7253\,\text{mN}$, $a_1 = 0.024\,81\,\text{mN}\,\text{W}^{-1}$, $a_2 = a_3 = a_4 = 0$, $b_0 = 2652$ s, $b_1 = -18.123\,\text{s}\,\text{W}^{-1}$, $b_2 = 0.3887\,\text{s}\,\text{W}^{-2}$, $b_3 = -0.001\,74\,\text{s}\,\text{W}^{-3}$, $b_4 = 0$, and $c_0 = 840.11\,\text{W}$, $c_1 = -1754.3\,\text{W}\,\text{AU}^{-1}$, $c_2 = 1625.01\,\text{W}\,\text{AU}^{-2}$, $c_3 = -739.87\,\text{W}\,\text{AU}^{-3}$, $c_4 = 134.45\,\text{W}\,\text{AU}^{-4}$, and r in AU. For Dionysus, we use a modified version of the model presented in [126]:

$$T_{\text{max}}(P_{\text{in}}) = \tilde{a}_0 + \tilde{a}_1 P_{\text{in}} \tag{8.73a}$$

$$I_{\text{sp}}(P_{\text{in}}) = I_{\text{sp,max}} \tag{8.73b}$$

$$P_{\text{in}}(r) = \frac{1}{r^2} \tag{8.73c}$$

where $\tilde{a}_1 = 0.1069\,\text{N}$, $\tilde{a}_1 = 0.1069\,\text{N}$, $\tilde{a}_2 = 3.9307 \times 10^{-4}\,\text{N}\,\text{W}^{-1}$, and r in AU.

8.4.1 Embedded Homotopic Approach for High-Fidelity Models

Different numbers of nodes and optimality tolerances are considered for GPOPS-II to account for potential discrepancies in the performance if an inappropriate value is selected. The problem is discretized using 5 and 10 nodes per segment for each transfer (see also Table 8.6). 30 and 15 segments are used for 2000 SG344, respectively, and 50 and 25 segments for Dionysus, respectively. Hence, the total number of

Table 8.5: Overview of simulations for each target and initial guess.

Initial guess	2000 SG344	Dionysus
Cubic interpolation	101	301
Propagation	101	51

Table 8.6: Number of nodes and segments for GPOPS-II.

Target	Nodes per segment	Segments
2000 SG344	5, 10	30, 15
Dionysus	5, 10	50, 25

nodes is 150 (2000 SG344) and 250 (Dionysus), respectively. The same number of points is chosen for the first-order-hold discretization method. Two different methods are used to generate the initial guesses:

1) A simple perturbed cubic interpolation that results in infeasible trajectories that neither satisfy the dynamical nor the endpoint constraints (see also Appendix A). The number of revolutions of the initial guesses ranges from 1.6 to 2.6 for 2000 SG344, and 4 to 7 for Dionysus. The total number of simulations is therefore 101 (2000 SG344) and 301 (Dionysus), respectively. The controls are set to zero.

2) Propagation of the two-body dynamics without SRP and with constant tangential thrust. The thrust magnitude for each initial guess is varied from 0 to T_{max} (2000 SG344) and 0 to $0.5\,T_{\text{max}}$ (Dionysus), respectively. This results in dynamically feasible trajectories, but the endpoints are far from the target positions.

An overview of the number of simulations and initial guesses is given in Table 8.5. Note that no-thrust periods are not included to ensure a fair comparison with GPOPS-II. The results are shown in Figs. 8.14 and 8.15. SCP is the standard SCP method without homotopy, SCP$_{\text{H}}$ the homotopic approach. The notation GPOPS$_i^k$ refers to the number of nodes $i \in \{5, 10\}$ per segment, and $k \in \{-4, -5\}$ to the optimality tolerance for GPOPS-II, i.e., $k = -4 \triangleq 10^{-4}$ and $k = -5 \triangleq 10^{-5}$. Even though we use 10^{-4} for SCP, selecting a smaller value would not change the results as feasiblity is the main factor for SCP.

With regard to the 2000 SG344 transfer, the benefit of a homotopic approach becomes clear in Fig. 8.14a. When using a cubic interpolation to generate the initial guess, the success rate is only 39 $\%$ if the full problem is solved directly with a SCP method. GPOPS-II achieves a slightly lower rate of approximately 31 – 36 $\%$ regardless of the number of nodes and optimality tolerance. In contrast, SCP$_{\text{H}}$ converges in more than 81 $\%$ of the cases, thus doubling the success rate. As the perturbing accelerations due to the n bodies and SRP are small, the thrust continuation is often the most critical part for convergence. However, there are cases where a continuation of the dynamics is also required to achieve convergence. This is especially true for poor initial guesses where the initial constraint violation is large. Judging from Fig. 8.14a, GPOPS-II benefits from an initial guess that satisfies the dynamics. Remarkably, considerably more simulations converge successfully for SCP ($67\,\%$) and GPOPS-II (50 – $56\,\%$) if the propagation guess is used. In this case, the convergence of SCP$_{\text{H}}$ increases only slightly to $70\,\%$. The reason is that

Table 8.7: Number of iterations (median) for SCP and SCP$_{\mathrm{H}}$.

Target / Method	SEL$_2$ - 2000 SG344 cubic	SEL$_2$ - 2000 SG344 propagation	SEL$_2$ - Dionysus cubic	SEL$_2$ - Dionysus propagation
SCP	19	24	47	47
SCP$_{\mathrm{H}}$	33	28	105	102

the initial guesses become very poor for larger thrust magnitudes, and even a homotopic approach is not able to find feasible solutions anymore.

According to Fig. 8.14b, the CPU times are lowest for the propagation guess for all methods. The standard SCP algorithm requires the least CPU time (median of approximately $9\,\mathrm{s}$), followed by SCP$_{\mathrm{H}}$ that requires a few more iterations (see Table 8.7) with 15 – $26\,\mathrm{s}$, and GPOPS-II (18 – $54\,\mathrm{s}$). As expected, increasing the number of nodes and decreasing the optimality tolerance result in higher CPU times for GPOPS-II. Remarkably, regardless of the parameters, the error bars and thus variation of the results are substantially larger for GPOPS-II. For example, there are cases where the NLP solver requires more than $120\,\mathrm{s}$ to find an optimal solution. Although the final masses are similar for all methods, the solutions obtained with both SCP algorithms are more consistent and closer to the optimal value compared to GPOPS-II. Furthermore, all methods yield an excellent accuracy as the propagation error (i.e., the difference between the optimized final state and the state obtained when integrating the dynamics with the obtained controls) is of the order $10^0\,\mathrm{km}$ (position) and $10^{-7}\,\mathrm{km\,s^{-1}}$ (velocity) only.

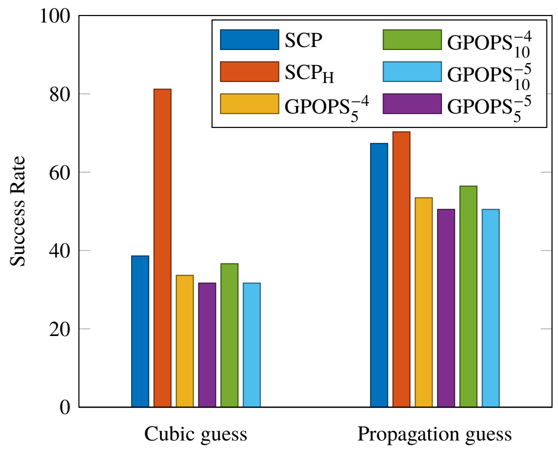
With regard to the Dionysus transfer, SCP converges in only $33\,\%$ (propagation guess) and $46\,\%$ (cubic interpolation guess) of the simulations, therefore performing worst among all methods. Depending on the optimality threshold, GPOPS-II achieves a success rate of 53 – $65\,\%$ ($\varepsilon_J = 10^{-4}$) and 35 – $55\,\%$ ($\varepsilon_J = 10^{-5}$). As opposed to the 2000 SG344 transfer, the cubic guess yields more converged simulations than the propagation guess for GPOPS-II for the Dionysus transfer. Again, the success rate can approximately be doubled if the embedded homotopic approach is used, resulting in more than $95\,\%$ converged simulations. Remarkably, it is evident from Fig. 8.15b that both SCP methods require up to one order of magnitude fewer seconds to converge than GPOPS-II. Even though the CPU time per iteration is similar for SCP and SCP$_{\mathrm{H}}$, the latter requires twice as many iterations as shown in Table 8.7, and therefore, also twice as much CPU time. The CPU time of GPOPS-II can in general be reduced significantly (sometimes up tp $50\,\%$) if the propagation guess is used. The discrepancy in the final mass becomes more significant now as shown in Fig. 8.15c. Especially if a larger optimality threshold is chosen, the final masses obtained with GPOPS-II are considerably lower compared to both SCP methods. Interestingly, SCP$_{\mathrm{H}}$ does not only improve convergence, but also finds more optimal solutions than the standard SCP. The propagation error is of the order 10^1 to $10^2\,\mathrm{km}$ (position) and 10^{-6} to $10^{-5}\,\mathrm{km\,s^{-1}}$ (velocity), therefore again being very small.

Typical transfer trajectories and control profiles are shown in Figs. 8.16 and 8.17. These plots confirm that SCP and GPOPS-II find different solutions. Moreover, the controls obtained with GPOPS-II are often jittery (see also, e.g., [187]). A much smaller optimality threshold would be required to yield more accurate bang-bang structures. With regard to the Dionysus transfer, it is evident from Fig. 8.17b that the available maximum thrust decreases considerably over time as the spacecraft gets farther away from the Sun.

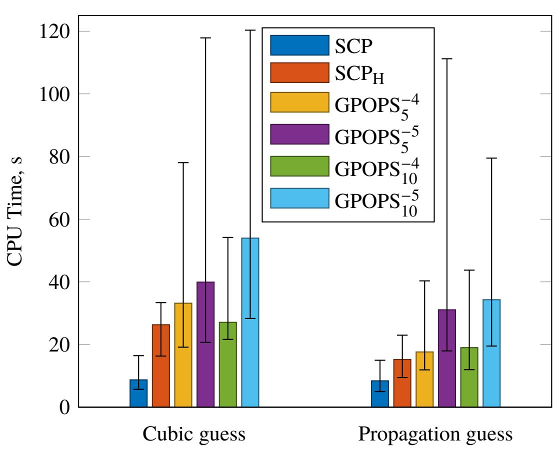
Even though a homotopic approach requires in general more iterations, the simulations confirm that convergence can be improved significantly by gradually increasing the complexity of the dynamical model. Often, only a continuation from higher thrust levels to the real thruster model is sufficient to achieve convergence. The reason is that the additional accelerations due to the perturbing bodies and SRP are small, and therefore, a continuation of the dynamics may not be needed. Yet, for large initial constraint violations (e.g., if the final target state of the initial guess deviates considerably from the desired target state), we observed that a continuation of the dynamics is also required so that the algorithm converges successfully. Due to the rapid speed of SCP, the additional iterations may not be as relevant when compared to methods that usually require more computational effort, such as nonlinear programming solvers. Instead of using SCP only to generate a decent initial guess for an indirect method that makes use of a continuation technique [106], our approach embeds the homotopy directly into SCP. Remarkably, the number of iterations can be reduced considerably with an embedded approach compared to the method in [10] where each optimization problem is solved to full optimality. Our simulations show that methods based on convex optimization can yield a high accuracy even if high-fidelity models are considered.

Convergence Analysis

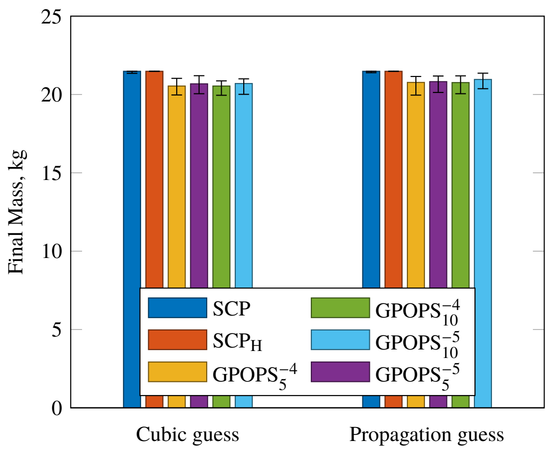
Clearly, a homotopic approach is beneficial when solving more complex problems with convex optimization. Ideally, it would be desirable to have some criterion that defines whether a homotopy is needed, or if a standard SCP method suffices. This way, the additional iterations required by $\mathrm{SCP_H}$ could be avoided. The complexity of the problem depends on the problem itself (e.g., number of revolutions, dynamical system), but also on the quality of the initial guess. For example, a highly nonlinear problem can still be solved if a decent initial guess is provided. Our simulations suggest that non-perturbed initial guesses where the final boundary conditions are satisfied and the constraint violations of the dynamics are not too large, SCP is in general able to converge successfully even if high-fidelity models are considered. There-



(a) Comparison of success rate.

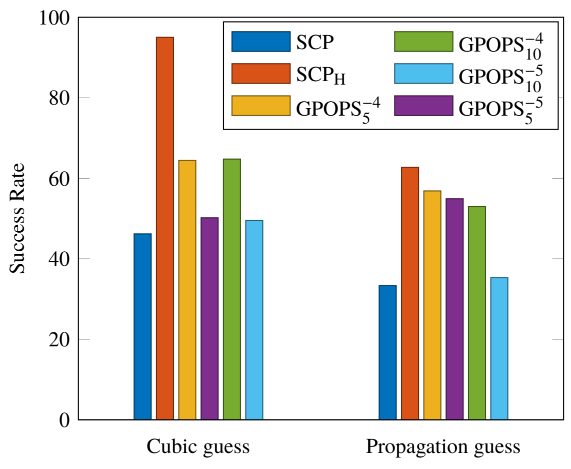


(b) Comparison of CPU time.

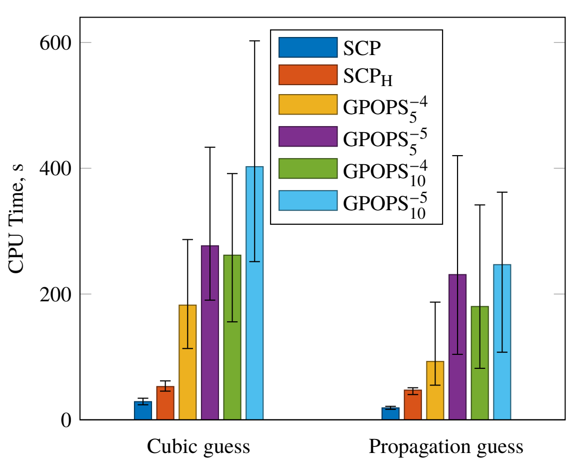


(c) Comparison of final mass.

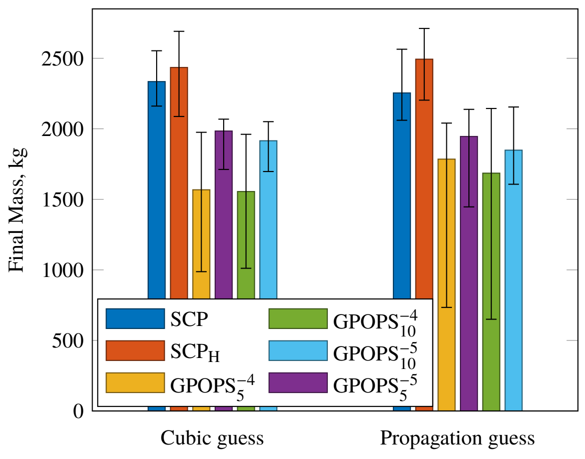
Figure 8.14: Comparison of SCP, SCP$_H$ and GPOPS-II in terms of success rate, CPU time, and final mass for the 2000 SG344 transfer. Median values are shown, and the error bars refer to the 80th and 20th percentile of the corresponding quantity.



(a) Comparison of success rate.

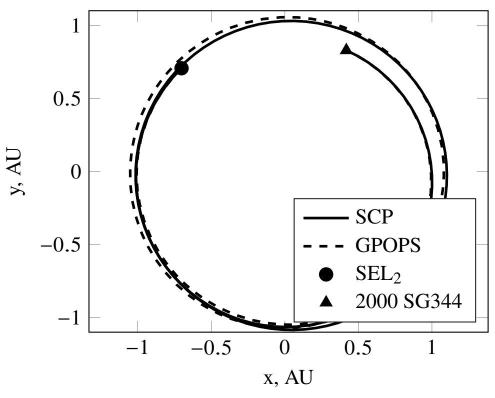


(b) Comparison of CPU time.

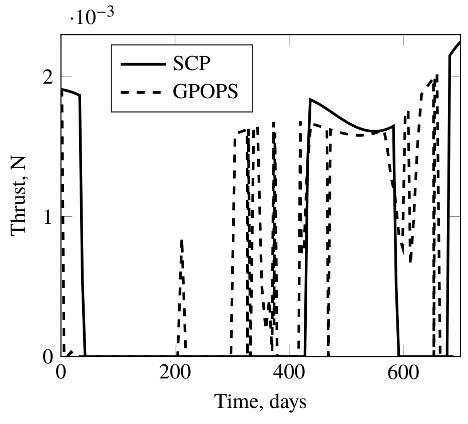


(c) Comparison of final mass.

Figure 8.15: Comparison of SCP, SCP$_H$ and GPOPS-II in terms of success rate, CPU time, and final mass for the Dionysus transfer. Median values are shown, and the error bars refer to the 80th and 20th percentile of the corresponding quantity.

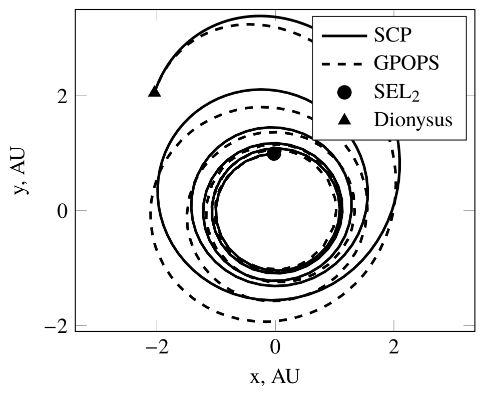


(a) Transfer trajectory.

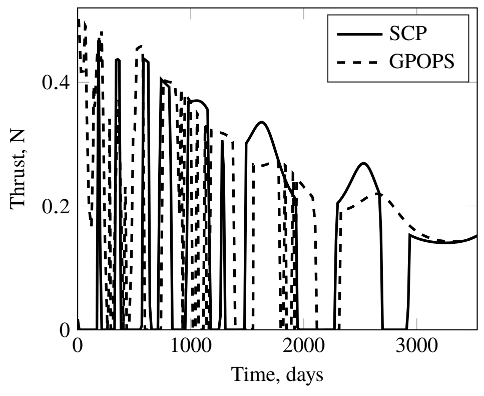


(b) Thrust magnitude.

Figure 8.16: Typical transfer and control trajectories for 2000 SG344 obtained with SCP and GPOPS-II.



(a) Transfer trajectory.



(b) Thrust magnitude.

Figure 8.17: Typical transfer and control trajectories for Dionysus obtained with SCP and GPOPS-II.

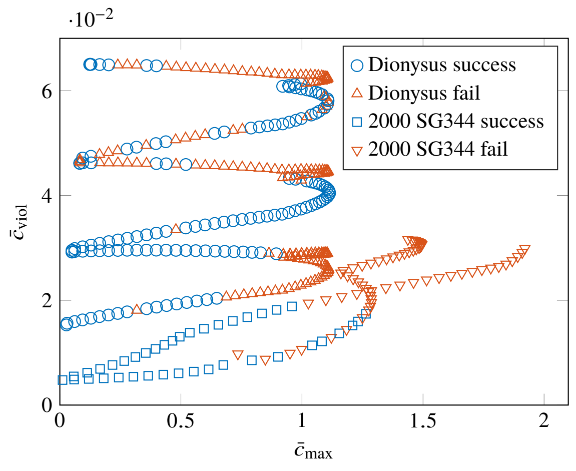
fore, we define the normalized maximum constraint violation $\bar{c}_{\text{max}}$ and normalized constraint violation $\bar{c}_{\text{viol}}$ as follows:

$$\bar{c}_{\text{max}} = \frac{\|\mathbf{c}_{\text{viol}}\|_{\infty}}{\|\mathbf{x}\|_{\infty}} \tag{8.74}$$

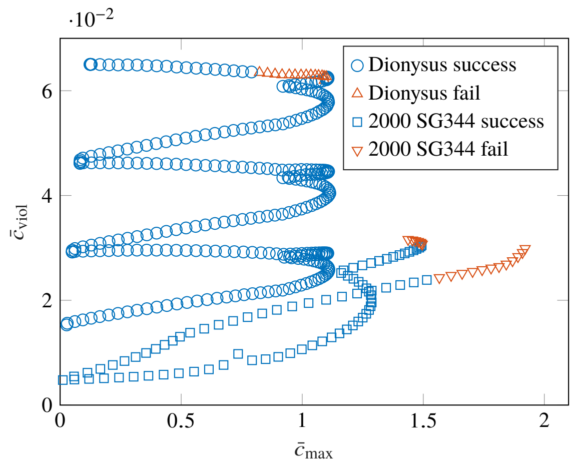
$$\bar{c}_{\text{viol}} = \frac{\|\mathbf{c}_{\text{viol}}\|_{1}}{\|\mathbf{x}\|_{1}} \tag{8.75}$$

where $\mathbf{c}_{\text{viol}}$ is a vector that contains all constraint violations, and $\mathbf{x}$ is the solution vector. The normalization accounts for the strong dependence on the initial and final boundary conditions (e.g., larger values of the coordinates if the target is farther away from the primary body). Figure 8.18 illustrates $\bar{c}_{\text{max}}$ and $\bar{c}_{\text{viol}}$ for each cubic interpolation guess for SCP and SCP$_{\text{H}}$. It is apparent that the closer both values are to zero, the more likely it is that SCP converges successfully (see Fig. 8.18a). Yet, there is no unique pair of values that clearly defines success or failure of the standard algorithm. For example, SCP fails if $(\bar{c}_{\text{max}}, \bar{c}_{\text{viol}}) = (0.9, 0.02)$, but converges again if $\bar{c}_{\text{viol}}$ becomes larger (e.g., 0.045 and 0.055). Still, $\bar{c}_{\text{max}} \geq 1.1$ seems to be the threshold where the standard SCP cannot find optimal solutions anymore regardless of the normalized constraint violation. As shown in Fig. 8.18b, this value increases to approximately 1.4 if a homotopic approach is used. Moreover, a much higher constraint violation is required so that SCP$_{\text{H}}$ fails. Interestingly, the failure and success cases follow a more regular pattern than in the non-homotopic case. In general, a homotopic approach seems to be useful for a perturbed cubic guess whenever $\bar{c}_{\text{max}} \geq 0.7 \wedge \bar{c}_{\text{viol}} \geq 0.02$. SCP$_{\text{H}}$ seems to fail if $\bar{c}_{\text{max}} \geq 1.4$ or $\bar{c}_{\text{max}} \geq 0.8 \wedge \bar{c}_{\text{viol}} \geq 0.06$.

With regard to the propagation guess in Fig. 8.19, only the transfer to 2000 SG344 shows a pattern. If $\bar{c}_{\text{max}} \geq 1.5 \wedge \bar{c}_{\text{viol}} \geq 0.015$, both algorithms seem to fail (even though there very few other cases where SCP does not converge, see Fig. 8.19a). For Dionysus, however, there does not seem to be a visible correlation. It therefore may be beneficial to make use of a homotopic approach whenever $\bar{c}_{\text{max}} \geq 0.5$, even though SCP$_{\text{H}}$ will fail at some point due to the large initial error on the final boundary condition.

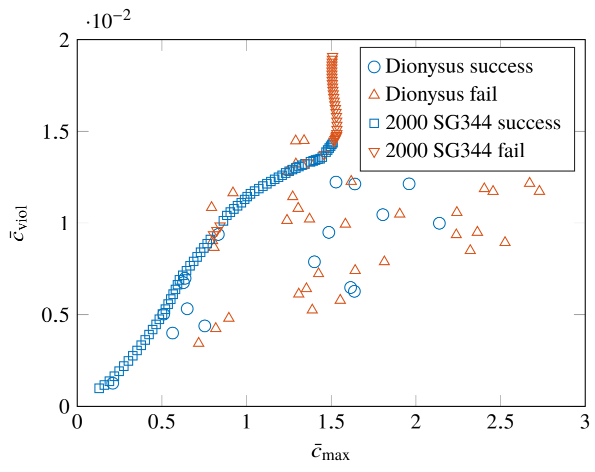


(a) Standard SCP.

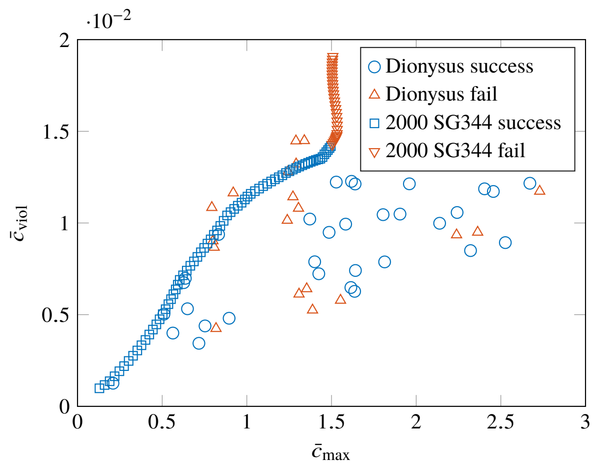


(b) SCP with homotopic approach.

Figure 8.18: Overview of the constraint violation of each initial guess generated by cubic interpolation for SCP and SCP$_{\mathrm{H}}$.



(a) Standard SCP.



(b) SCP with homotopic approach.

Figure 8.19: Overview of the constraint violation of each initial guess generated by propagation for SCP and SCP$_{\mathrm{H}}$.

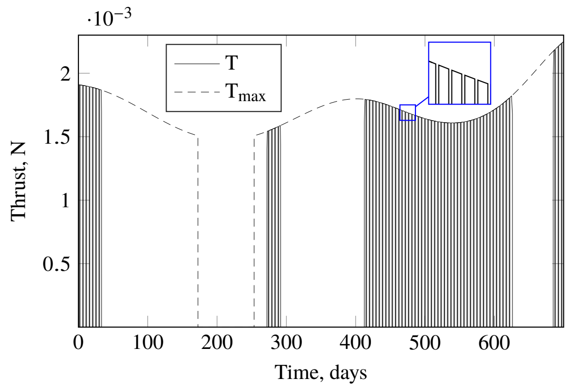
8.4.2 No-Thrust Constraints

We briefly assess the performance and solutions when no-thrust periods are included. In this context, periods of one (2000 SG344) and two (Dionysus) days per segment are considered where no thrust is available. Typical profiles of the thrust magnitude are shown in Figs. 8.20a and 8.20b. The dashed black line refers to the maximum available thrust, and the vertical lines correspond to the on and off switches. Clearly, the thrust magnitude follows the available maximum thrust, being either zero or taking the maximum value that depends on the distance to the Sun. Note that the available thrust is zero between 175 and 250 days in Fig. 8.20a because the input power drops below the required minimum value of 90 W. Due to the long transfer time to Dionysus, there are many no-thrust periods, making this problem difficult to solve. The corresponding trajectories are depicted in Figs. 8.21a and 8.21b. The thrust arcs in red are discontinuous due to the no-thrust periods.

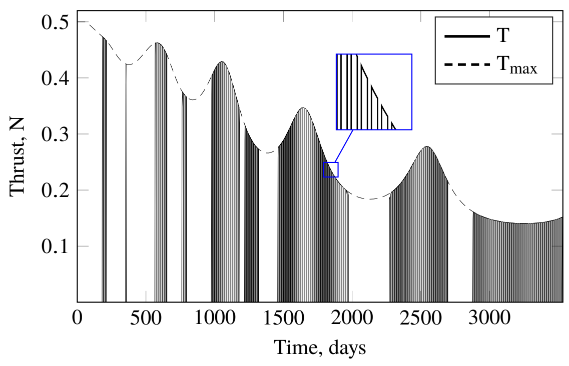
The comparison of the number of iterations and CPU time with the standard SCP method (i.e., continuous thrust) is shown in Fig. 8.22. Median values are given, and the error bars refer to the 80th and 20th percentile of the corresponding quantity. $\mathrm{SCP_{NT}}$ denotes the case with no-thrust periods. With regard to the transfer to 2000 SG344, the number of iterations increases from 19 to 31 when no-thrust periods are enforced. The reason is that the trust-region size grows rapidly first, but then has to shrink again which requires additional iterations. Interestingly, $\mathrm{SCP_{NT}}$ requires slightly fewer iterations to find a solution for the Dionysus transfer. The CPU time behaves accordingly (see Fig. 8.22b), even though the discrepancy is larger for 2000 SG344 due the greater difference in the number of iterations. Most of the CPU time is required outside of the convex solver. The CPU time of the convex solver is approximately the same regardless of whether no-thrust periods are considered or not. The reason is that the convex program itself does not significantly change. However, the integration of the integrands in Eq. (8.29) requires more time compared to Eq. (6.68) due to the additional terms. As each segment is divided into subsegments in the no-thrust case, the integration intervals become smaller. Therefore, if the same integrator is chosen, the accuracy increases. Thus, a lower-order integrator may be used to reduce the computational effort without sacrificing accuracy.

8.4.3 Bang-Bang Mesh Refinement

We demonstrate the effectiveness of the bang-bang mesh refinement procedure described in Section 8.3 for the transfers to 2000 SG344 (using FOH) and to Dionysus (using FRPM). For both cases, the corresponding OCPs on the initial mesh are solved using FOH and FRPM, respectively. Figures 8.23 and 8.24 show the evolution of the throttle factor (interpolated linearly between the discretization points for better readability) and switching function. It is clear that the determined switching functions cross zero precisely at the times when the throttle factor switches from 0 to 1 or vice versa. However, the

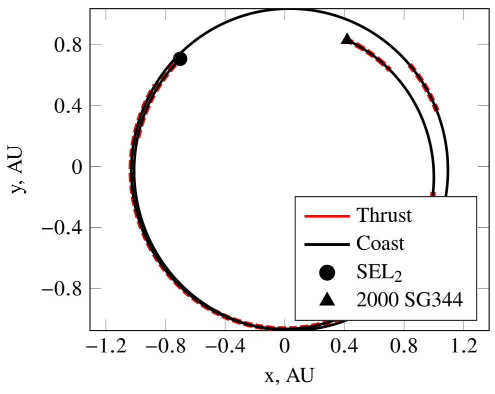


(a) Thrust profile for transfer to 2000 SG344.

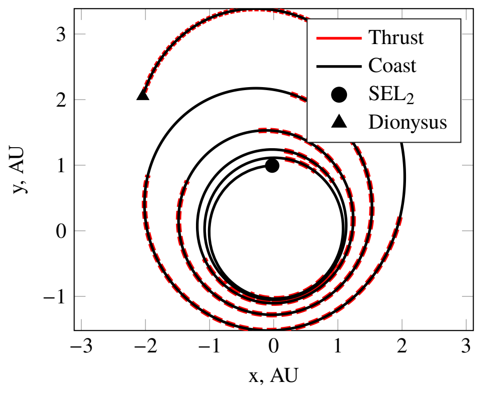


(b) Thrust profile for transfer to Dionysus.

Figure 8.20: Typical thrust profiles for the transfers to 2000 SG344 and Dionysus when no-thrust periods are considered.

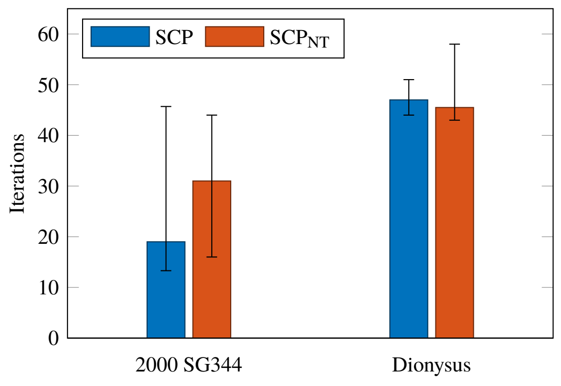


(a) Trajectory for transfer to 2000 SG344.

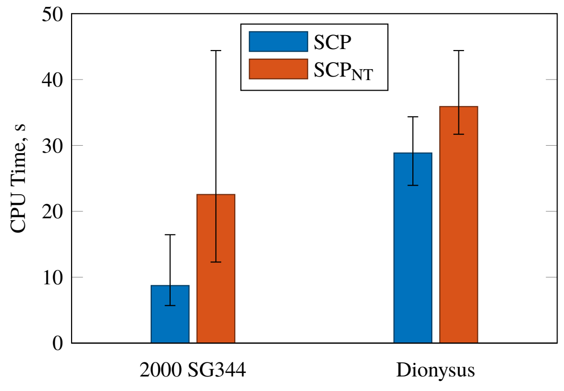


(b) Trajectory for transfer to Dionysus.

Figure 8.21: Typical trajectories for the transfers to 2000 SG344 and Dionysus when no-thrust periods are considered.



(a) Number of iterations.



(b) CPU time.

Figure 8.22: Comparison of the number of iterations and CPU time for standard SCP and SCP_{NT} where no-thrust periods are considered.

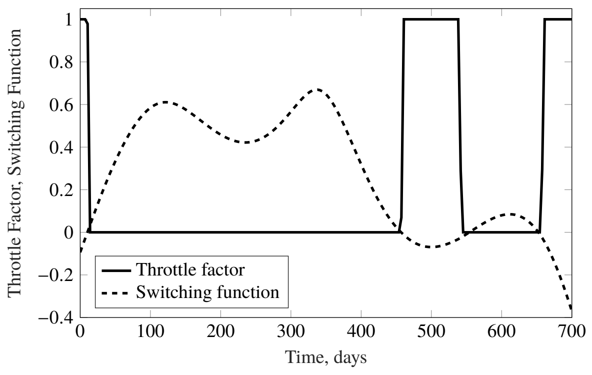


Figure 8.23: Throttle factor and switching function for 2000 SG344 transfer and initial mesh (FOH).

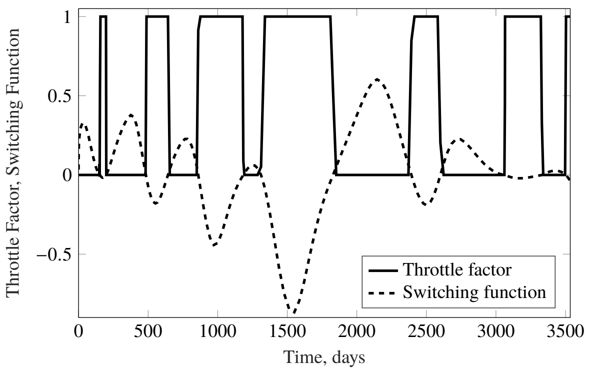
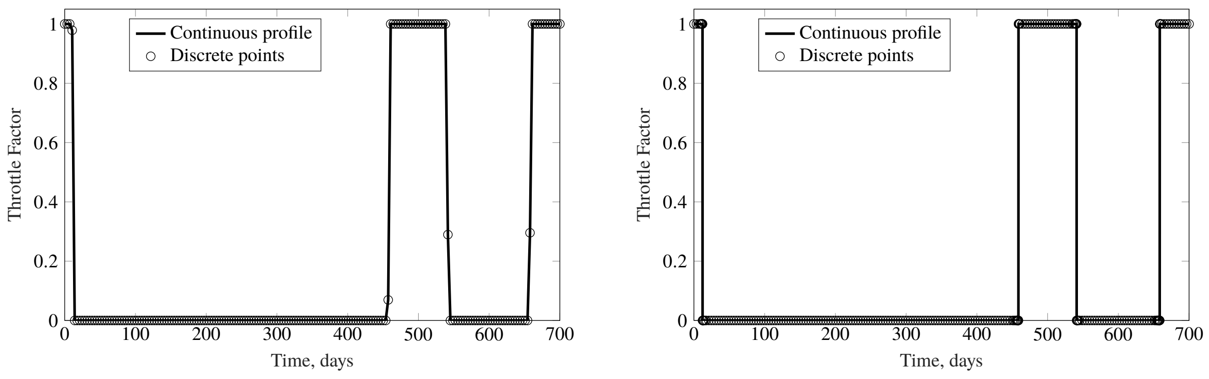


Figure 8.24: Throttle factor and switching function for Dionysus transfer and initial mesh (FRPM).



(a) Throttle factor before refinement.

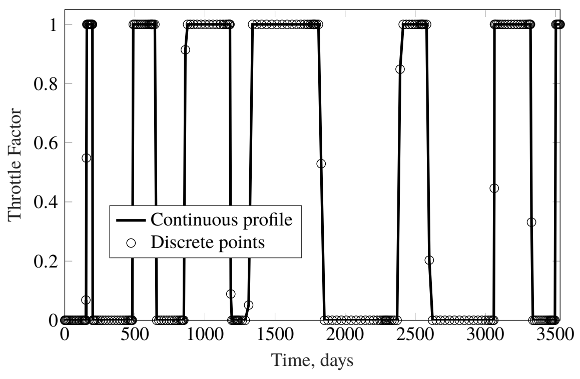
(b) Throttle factor after refinement.

Figure 8.25: Continuous throttle factor profile and discrete points for 2000 SG344 transfer before and after bang-bang mesh refinement (FOH).

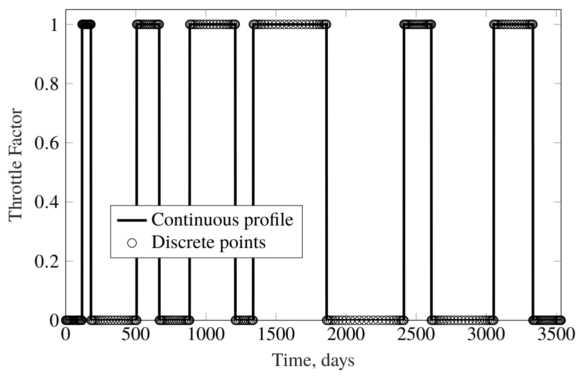
thrust profile is not completely regular. The reason is that some values of the throttle factor are neither 0 nor 1. This becomes clear in Figs. 8.25a and 8.26a where the discrete values are shown along with the continuous profiles.

Following the procedure in Fig. 8.13 for FOH, the number of nodes is increased in the neighborhood of the switching locations by solving a series of optimization problems. Note that often only one iteration is required if the previous solution is used as the initial guess. We stop when the absolute change of the switching times is smaller than 10^{-5}, and then predefine the values of the thrust magnitude based on the sign of the switching function. The refined thrust profile shown in Fig. 8.25b is now regular because it does not contain any values between zero and one anymore.

The more complex thrust profile for the transfer to Dionysus using FRPM is refined in a similar way according to Fig. 8.11. The solver is able to optimize the switching times, and the resulting thrust magnitudes have a bang-bang structure as illustrated in Fig. 8.26.



(a) Throttle factor before refinement.



(b) Throttle factor after refinement.

Figure 8.26: Continuous throttle factor profile and discrete points for Dionysus transfer before and after bang-bang mesh refinement (FRPM).

9 Closed-Loop Guidance in Deep Space

In the autonomous guidance scenario, the spacecraft shall repeatedly determine its reference trajectory autonomously on board and in real time. This is desirable for long-duration transfers in an unknown environment due to several reasons, for example to allow for immediate correction maneuvers without the dependence on the communication with ground stations. Moreover, the additional uncertainties introduced by autonomous navigation require a different method than the traditional tracking of a reference trajectory because of the larger deviations. In this dissertation, we propose a deep-space closed-loop guidance approach where trajectory optimization and guidance are combined. Instead of tracking a given reference trajectory, the control actions are repeatedly reoptimized on board using data acquired from the environment. This is expected to be more robust against uncertainties, unmodeled perturbations, and possible failures or deviations from nominal conditions.

The first part of this chapter addresses the processor-in-the-loop (PIL) experiment to assess the performance of the SCP algorithm as a guidance approach on hardware similar to spaceflight processors. The closed-loop guidance is explained, and numerical simulations are performed. Moreover, the moving target scenario is introduced where the target state is not fixed anymore, but rather varies with time. Finally, details about the real-time implementation of the SCP algorithm are addressed. Parts of this chapter are based on our work in [15].

9.1 Processor-in-the-Loop Simulation

The setup of the processor-in-the-loop experiment is illustrated in Fig. 9.1. It consists of two main parts: the guidance system and the orbit propagator. The former is represented by a single-board computer that is comparable to real spacecraft hardware. In this dissertation, a Raspberry Pi 3 Model B+ [194] is used because its computational power is similar to CubeSat onboard computers such as the LEON family that is often used in European space missions [188]. The Raspberry Pi is shown in Fig. 9.2, and the most relevant technical specifications are given in Table 9.1. The initialization step initializes the SCP algorithm and loads all data on the processor. As the first reference trajectory can be computed on ground, the initial guesses of the state and control trajectories are also transferred to the guidance computer as input data. The orbit propagator represents the environment and the navigation system of the spacecraft.

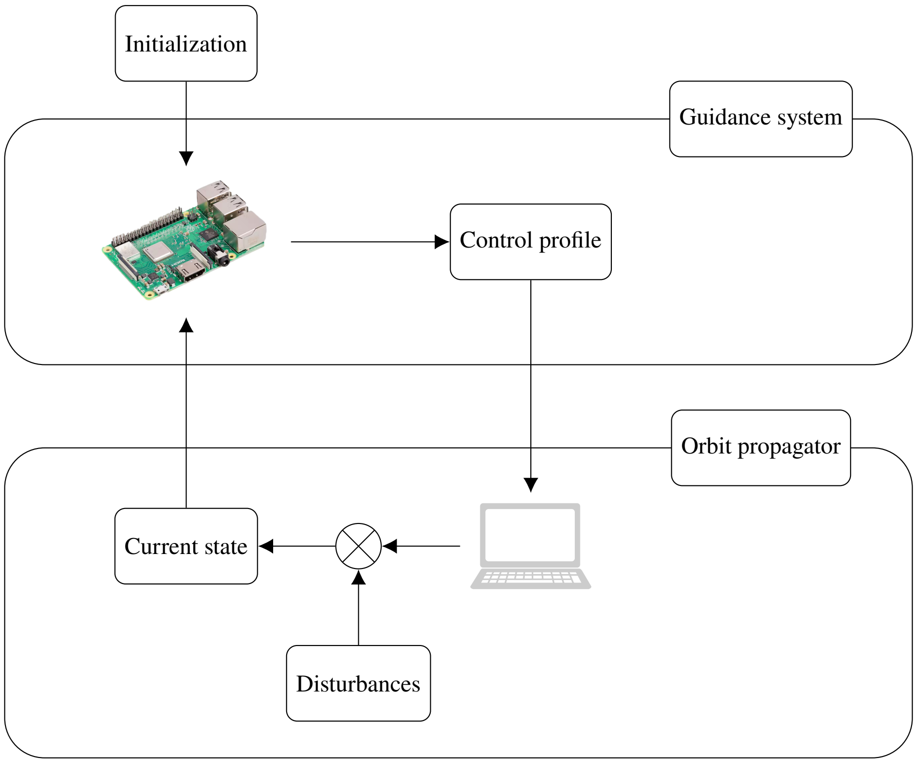


Figure 9.1: Overview of the processor-in-the-loop simulation.

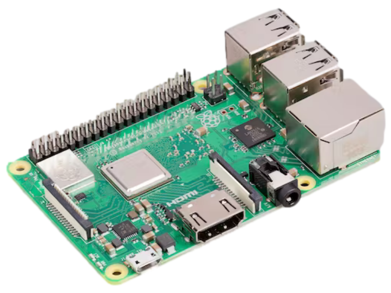


Figure 9.2: Raspberry Pi 3 Model B+ [194].

Table 9.1: Technical specifications of Raspberry Pi 3 Model B+ [194].

Processor	RAM	Operating system
Cortex-A53 (ARMv8), 64 bit, 1.4 GHz	1 GB	Raspberry Pi OS

It propagates the dynamics using a high-fidelity dynamical model and the obtained controls from the guidance system. Disturbances are added to account for uncertainties and errors in the models. The propagated final state is fed back to the guidance system and serves as the new current state. Note that, in reality, the state of the spacecraft is determined through an orbit determination process that utilizes sensor data measurements, instead of relying on the propagation of dynamics to retrieve the state. The guidance system then reoptimizes the trajectory, and the obtained controls are again used for the integration. This process is followed recursively until the spacecraft reaches the target. A more detailed flowchart of the closed-loop guidance simulation is shown in Fig. 9.3. We predefine the vector $\mathbf{t}_{\text{opt}} = [t_1, \dots, t_L]^\top$ with $t_L < t_f$ during the initialization that contains the times when a (new) trajectory is to be determined. The frequency of recomputing the states and controls strongly depends on navigational and other operational constraints, and can range from few days to several weeks. After computing a new trajectory, the equations

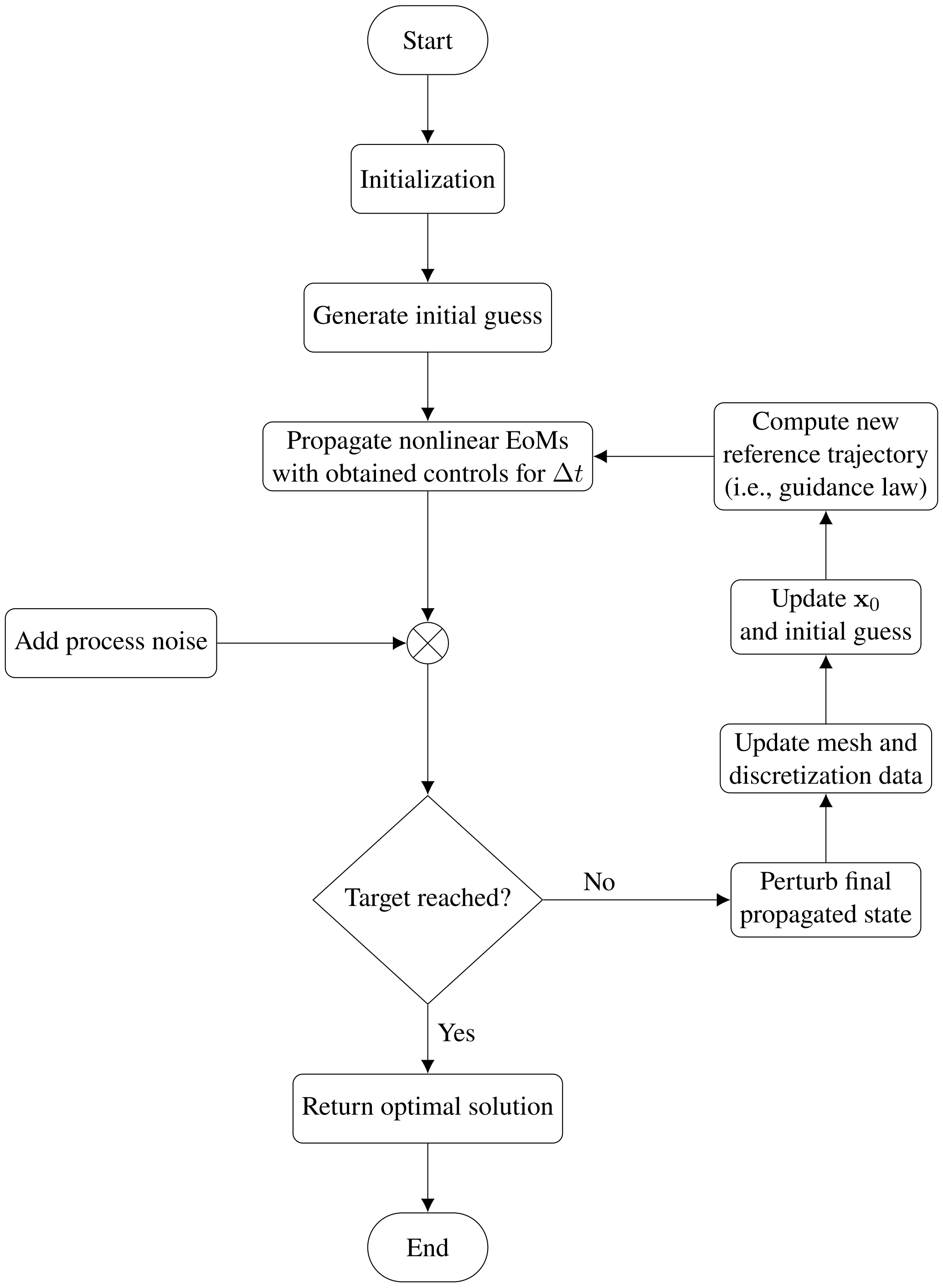

Start
Initialization
Generate initial guess
Propagate nonlinear EoMs with obtained controls for Δt
Compute new reference trajectory (i.e., guidance law)
Add process noise
Update $\mathbf{x}_0$ and initial guess
Update mesh and discretization data
Target reached?
No
Perturb final propagated state
Yes
Return optimal solution
End

Figure 9.3: Flowchart of the closed-loop guidance simulation.

of motion are propagated for Δt to simulate the space flight. The high-fidelity model of Chapter 8 is used where n-body dynamics, solar radiation pressure, variable specific impulse and maximum thrust, and no-thrust constraints are considered. Yet, several sources of errors (for example due to the dynamical system, an unknown environment, and the propulsion or navigation system) result in a deviation of the actual spacecraft state from the computed one. One major error is caused by the misalignment of the thrusters, leading to perturbed thrust values compared to the nominal ones determined by the guidance system. We consider such correlated process noises by modeling the thrust components and magnitude as Gauss–Markov processes. The perturbed thrust magnitude $\hat{T}(t)$ for integrating the dynamics from t_k to t_{k+1} is then given by [195, 196]

$$\hat{T}(t) = \kappa\, \gamma(t) + T \tag{9.1}$$

where

$$\gamma(t) = \gamma_j\, \mathrm{e}^{-\varphi\,(t-t_k)} + \omega_j\, \sqrt{1 - \mathrm{e}^{-2\,\varphi\,(t-t_k)}}, \qquad t \in [t_k, t_{k+1}] \tag{9.2}$$

κ denotes the variance of the process, T the nominal (i.e., unperturbed) thrust magnitude, and φ the inverse of the correlation time. γ_j and ω_j are the jth elements of vectors $\boldsymbol{\gamma}$ and $\boldsymbol{\omega}$ that contain random numbers from a Gaussian distribution with mean 0 and standard deviation κ. The expression for the thrust components $\mathbf{T}(t)$ is obtained accordingly. Other sources of noise that can be represented by Gauss–Markov processes, such as the accelerations caused by solar radiation pressure or other bodies, are very small and therefore neglected as they have no significant impact on the solution.

If the target is not yet reached, the spacecraft must acquire its new state. Optical navigation is a promising technique for CubeSats to determine the position and velocity autonomously on board [79]. Still, the relatively large uncertainty results in larger errors compared to ground-based navigation methods. To account for navigational errors, other uncertainties and unmodeled errors, each component of the propagated final state is perturbed by random values between $\mathbf{r}_{\min}(t)$ and $\mathbf{r}_{\max}(t)$ (position), and $\mathbf{v}_{\min}(t)$ and $\mathbf{v}_{\max}(t)$ (velocity). In this context, we use time-varying functions of the minimum and maximum values. In real missions, optical navigation is often the main error source. According to [80], the orbit determination error for targets close to Earth usually decreases over time. The reason is that a higher accuracy is required when the spacecraft approaches the target, for example by using a higher frequency of observations for estimating the state. Therefore, we model the minimum and maximum values as linear functions that define a truncated cone as illustrated in Fig. 9.4. The simulations in [80] suggest errors of 100 – $2000\,\mathrm{km}$ and 0.1 – $1\,\mathrm{m\,s^{-1}}$ for each component of the position and velocity, respectively, for missions to Venus and Mars. As we consider transfers to near-Earth asteroids, we expect similar errors and choose over-conservative values of $r_{\max} = 20\,000\,\mathrm{km}$ and $v_{\max} = 10\,\mathrm{m\,s^{-1}}$. The minimum values towards the end of the transfer should be close to zero to ensure that the target can actually be reached, for example $r_{\min} = 100\,\mathrm{km}$ and $v_{\min} = 0.1\,\mathrm{m\,s^{-1}}$. The reason for these extremely large

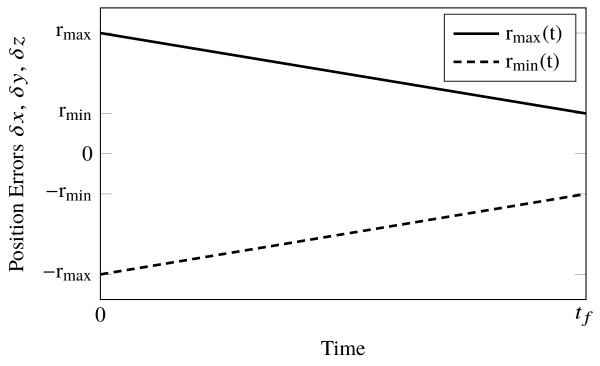


Figure 9.4: Definition of minimum $r_{\mathrm{min}}(t)$ and maximum $r_{\mathrm{max}}(t)$ error functions.

values during the cruise phase is to account for additional, not modeled errors, and thus to demonstrate that feasible solutions can be obtained even in such cases. Additional simulations with $r_{\mathrm{max}} = 2000\,\mathrm{km}$ and $v_{\mathrm{max}} = 3\,\mathrm{m\,s^{-1}}$ are performed to investigate how different errors affect the results.

After perturbing the propagated final state, the current mesh and discretization data are to be updated. As the reference trajectories change only slightly between the iterations of the closed-loop simulation, it is reasonable to use the previous solution as the new initial guess. However, as the time of flight decreases by the propagation time Δt, the mesh data is to be adjusted accordingly. We intend to maintain the accuracy of the initial mesh, that is, the number of nodes per time unit shall be similar. Therefore, the total number of nodes is reduced, and the states and controls are interpolated on this new mesh. The new initial state $\mathbf{x}_0$ is updated along with the constraints and time of flight, and the next reference trajectory is reoptimized. This process continues until the spacecraft reaches the target. If the calculated trajectory is close to the propagated (i.e., real) trajectory, the algorithm converges within few iterations only. Our rationale is that the new solution is expected to lie in the neighborhood of the previous solution. Although a proof of the convergence is beyond the scope of this dissertation, the numerical simulations strongly support this statement.

9.2 Moving Target

In the nominal case, the errors are assumed to remain within the expected range, and no unforeseen events occur. However, some components may fail for certain periods during a real mission. Even though SCP can handle large deviations from the nominal condition, this becomes critical when the spacecraft gets closer to the target. The reason is that if the remaining time of flight is short and the spacecraft significantly deviates from the nominal trajectory, a feasible solution may not exist anymore. An example is a failure of the propulsion system so that thrust cannot be provided anymore for some days. This is equivalent to perturbations of up to $50\,000\,\mathrm{km}$ and $30\,\mathrm{m\,s^{-1}}$. In this section, we simulate such an event where the spacecraft is near the target and the perturbations considerably exceed the expected values.

Even though the target is often considered to be a fixed point in space, the asteroid (or planet) is in fact orbiting some primary body. This means that the spacecraft may still be able to reach the target at a future point in time. The only constraint that results in infeasibility is the predefined time of flight because it cannot be satisfied anymore given the original target position. Therefore, the basic idea is to relax the problem such that the target is considered moving, and hence, the new arrival time is free. The goal is to regain feasibility at the cost of a longer time of flight. Even though increasing t_f may at first glance not seem practical from an operational point of view, we believe that this kind of flexibility is necessary for self-driving spacecraft to avoid mission failure. To keep the arrival time (and hence, fuel consumption) as close as possible to the nominal one, we keep the final time fixed during the cruise phase and switch to the moving target problem only if needed (i.e., in case of some failure condition that leads to infeasibility). As we will see in the simulations section, the new arrival date differs only a few days from the nominal one, which we consider acceptable for real missions.

The position $\mathbf{r}_t(t)$ and velocity $\mathbf{v}_t(t)$ of the target are functions of time, and the relaxed final boundary conditions are simply

$$|\mathbf{r}(t_f) - \mathbf{r}_t(t_f)| \leq \Delta\mathbf{r} \tag{9.3a}$$

$$|\mathbf{v}(t_f) - \mathbf{v}_t(t_f)| \leq \Delta\mathbf{v} \tag{9.3b}$$

where $\Delta\mathbf{r}$ and $\Delta\mathbf{v}$ are given, and define the desired accuracy. Even though the ephemerides of the target body are assumed to be known at all times, determining $\mathbf{r}_t(t)$ and $\mathbf{v}_t(t)$ (e.g., retrieving the data from a lookup table or integrating the dynamics of the target body) is in general a nonlinear and nonconvex operation. As a consequence, the boundary conditions in Eqs. (9.3) become nonconvex. Therefore, $\mathbf{r}_t(t)$ and $\mathbf{v}_t(t)$ are approximated using a first-order Taylor series about a reference final time $\bar{t}_f$ [185]:

$$\mathbf{r}_t(t_f) \approx \mathbf{r}_t(\bar{t}_f) + \left.\frac{\mathrm{d}\mathbf{r}_t(t)}{\mathrm{d}t}\right|_{\bar{t}_f} (t_f - \bar{t}_f) \tag{9.4a}$$

$$\mathbf{v}_t(t_f) \approx \mathbf{v}_t(\bar{t}_f) + \left.\frac{\mathrm{d}\mathbf{v}_t(t)}{\mathrm{d}t}\right|_{\bar{t}_f} (t_f - \bar{t}_f) \tag{9.4b}$$

As $\mathrm{d}\mathbf{r}_t(t)/\mathrm{d}t = \mathbf{v}_t(t)$ and $\mathrm{d}\mathbf{v}_t(t)/\mathrm{d}t = \mathbf{a}_t(t)$, Eqs. (9.4a) and (9.4b) can be rewritten as

$$\mathbf{r}_t(t_f) \approx \mathbf{r}_t(\bar{t}_f) + \mathbf{v}_t(\bar{t}_f)\ (t_f - \bar{t}_f) \tag{9.5a}$$

$$\mathbf{v}_t(t_f) \approx \mathbf{v}_t(\bar{t}_f) + \mathbf{a}_t(\bar{t}_f)\ (t_f - \bar{t}_f) \tag{9.5b}$$

Considering a n-body model, the acceleration $\mathbf{a}_t(\bar{t}_f)$ of the target body can be calculated using Eq. (8.7)

$$\mathbf{a}_t(\bar{t}_f) = -\frac{\mu\,\mathbf{r}_t(\bar{t}_f)}{r_t(\bar{t}_f)^3} + \sum_{i=1}^{n} \mu_i \left(\frac{\mathbf{r}_{\mathrm{t},i}(\bar{t}_f)}{r^3_{\mathrm{t},i}(\bar{t}_f)} - \frac{\mathbf{r}_i(\bar{t}_f)}{r^3_i(\bar{t}_f)} \right) \tag{9.6}$$

where $\mathbf{r}_{\mathrm{t},i}(\bar{t}_f) = \mathbf{r}_i(\bar{t}_f) - \mathbf{r}_{\mathrm{t}}(\bar{t}_f)$. The relaxed final boundary conditions are thus

$$\left|\mathbf{r}(t_f) - \mathbf{r}_t(\bar{t}_f) - \mathbf{v}_t(\bar{t}_f)\ (t_f - \bar{t}_f)\right| \le \Delta\mathbf{r} + \boldsymbol{\zeta}_r \tag{9.7a}$$

$$\left|\mathbf{v}(t_f) - \mathbf{v}_t(\bar{t}_f) - \mathbf{a}_t(\bar{t}_f)\ (t_f - \bar{t}_f)\right| \le \Delta\mathbf{v} + \boldsymbol{\zeta}_v \tag{9.7b}$$

Note that slack variables $\boldsymbol{\zeta}_r \ge \mathbf{0}$ and $\boldsymbol{\zeta}_v \ge \mathbf{0}$ are included to avoid artificial infeasibility due to the linearized constraints. These are again penalized by adding

$$\lambda_\zeta \sum_{i=1}^{m} \max(0, \zeta_i) \tag{9.8}$$

to the cost function, where ζ_i denotes the ith component of $\boldsymbol{\zeta} := [\boldsymbol{\zeta}_r^\top, \boldsymbol{\zeta}_v^\top]^\top \in \mathbb{R}^m$.

In case of a moving target, the final time t_f and final boundary conditions in Eqs. (5.23g) or (5.24) cannot be considered fixed anymore. Therefore, the problem becomes a free-final-time problem with t_f being an optimization parameter. The dilation factor $\sigma \equiv t_f$ is introduced, and a time normalization is carried out according to Section 6.1.3. Given N discretization points, the resulting convex optimization problem is then given by Problem 2.

Problem 2. Find the vectors $\mathbf{x}$, $\mathbf{u}$, $\boldsymbol{\nu}$, $\boldsymbol{\eta}$, $\boldsymbol{\zeta}$, and t_f that solve the following second-order cone program:

$$\min_{t_f,\mathbf{x},\mathbf{u},\boldsymbol{\nu},\boldsymbol{\eta},\boldsymbol{\zeta}} \quad -w_N + \lambda_\nu \sum_{i=1}^{N-1} \|\boldsymbol{\nu}_i\|_1 + \lambda_\eta \sum_{i=1}^{N} \max(0, \eta_i) + \lambda_\zeta \sum_{i=1}^{m} \max(0, \zeta_i) \tag{9.9a}$$

subject to:

$$\mathbf{x}_{k+1} = \mathbf{A}_k\,\mathbf{x}_k + \mathbf{B}_k^-\,\mathbf{u}_k + \mathbf{B}_k^+\,\mathbf{u}_{k+1} + \mathbf{S}_k\,t_f + \mathbf{q}_k + \boldsymbol{\nu}_k, \quad k = 1, \ldots, N-1 \tag{9.9b}$$

$$\Gamma_k \le T_{\max}(\bar{r}_k)\,\mathrm{e}^{-\bar{w}_k}\,(1 - w_k + \bar{w}_k) + \eta_k, \quad k = 1, \ldots, N \tag{9.9c}$$

$$\|\boldsymbol{\tau}_k\|_2 \le \Gamma_k, \quad k = 1, \ldots, N \tag{9.9d}$$

$$\|\mathbf{y} - \bar{\mathbf{y}}\|_1 \le R \tag{9.9e}$$

$$t_{f,\mathrm{lb}} \le t_f \le t_{f,\mathrm{ub}} \tag{9.9f}$$

$$\mathbf{r}_1 = \mathbf{r}_0,\ \mathbf{v}_1 = \mathbf{v}_0,\ w_1 = w_0 \tag{9.9g}$$

$$\left|\begin{bmatrix}\mathbf{r}_N\\ \mathbf{v}_N\end{bmatrix} - \begin{bmatrix}\mathbf{r}_t(\bar{t}_f) + \mathbf{v}_t(\bar{t}_f)\ (t_f - \bar{t}_f)\\ \mathbf{v}_t(\bar{t}_f) + \mathbf{a}_t(\bar{t}_f)\ (t_f - \bar{t}_f)\end{bmatrix}\right| \le \begin{bmatrix}\Delta\mathbf{r}\\ \Delta\mathbf{v}\end{bmatrix} + \boldsymbol{\zeta} \tag{9.9h}$$

where $\mathbf{y} := [\mathbf{x}^\top, t_f]^\top$ in Eq. (9.9e), i.e., we include states and the final time in the trust-region constraint. Our simulations suggest that neglecting the controls does not affect the results, but reduces the problem size. The constraint in Eq. (9.9f) refers to the lower and upper bound of t_f, respectively.

As the final time is free, minor modifications are required if no-thrust constraints are to be taken into account as per Section 8.2. All no-thrust periods are normalized with respect to the reference time of flight $\bar{t}_f$ during the discretization step, and evaluated with respect to the reference solution. As t_f may change between consecutive iterations, the no-thrust periods may not lie strictly within a trajectory segment anymore if the number of nodes is kept constant. Therefore, N is adjusted such that the number of nodes per time unit is approximately the same in all iterations. This ensures a consistent mesh, and also that the required off periods are strictly within $[t_{k+1}, t_k]$, $k = 1, \dots, N-1$.

Remark 9.1. *Although we consider a moving target only for the failure case, it is possible to target a moving body for the whole cruise phase. For example, if the exact final state of an asteroid is not known with enough confidence at launch and must be determined on the fly, it may be necessary to switch to the moving target problem formulation during the transfer.*

9.3 Real-Time Implementation

In this section, details about the implementation of the SCP algorithm are addressed. We refer to real-time implementation because we seek source code in a compiled language like C/C++ that shall comply with certain standards (see for example [197] and [198]). Once the optimal control problem is discretized, the objective function and constraints are to be transformed into the following standard SOCP form that can be handled by computers:

$$\begin{aligned}
&\text{minimize} && \mathbf{c}^\top \mathbf{y} && (9.10\text{a})\\
&\text{subject to:} && \mathbf{A}\,\mathbf{y} = \mathbf{b} && (9.10\text{b})\\
& && \mathbf{G}\,\mathbf{y} \le \mathbf{h} && (9.10\text{c})
\end{aligned}$$

where $\mathbf{y}$ is the decision vector, and $\mathbf{c}$, $\mathbf{A}$, $\mathbf{b}$, $\mathbf{G}$, and $\mathbf{h}$ are given vectors and matrices that define the constraints. Equation (9.10c) represents linear inequality and second-order cone constraints. We refer to this transformation as *parsing*. Modeling languages like CVX [99] allow the user to formulate the objective function and constraints using a simple syntax that requires little effort. All constraints are transformed automatically into the standard form of Eq. (9.10). As such tools must be kept as general as possible to deal with a variety of input parameters, the parsing process requires a significant amount of time that is often several times larger than solving the actual cone program. Even though this is convenient, the computational effort is unacceptable for real-time applications where we require as little computational resources as possible. Therefore, we follow a different approach in this dissertation, and parse the problem manually into standard form. Instead of aiming at a general tool, the parsing is tailored to the low-thrust trajectory optimization problem, and the complete process is divided into 1) tasks that are performed *offline* only once, and 2) tasks that are repeatedly executed *online* at runtime. An overview of the SCP steps is illustrated in Fig. 9.5. Given an optimal control problem, the analytical constraints are convexified, and the SCP parameters are selected as in Chapter 5 (for example, tolerances and trust-region parameters). The problem is discretized (for instance, using the first-order-hold method in Section 6.1.3),

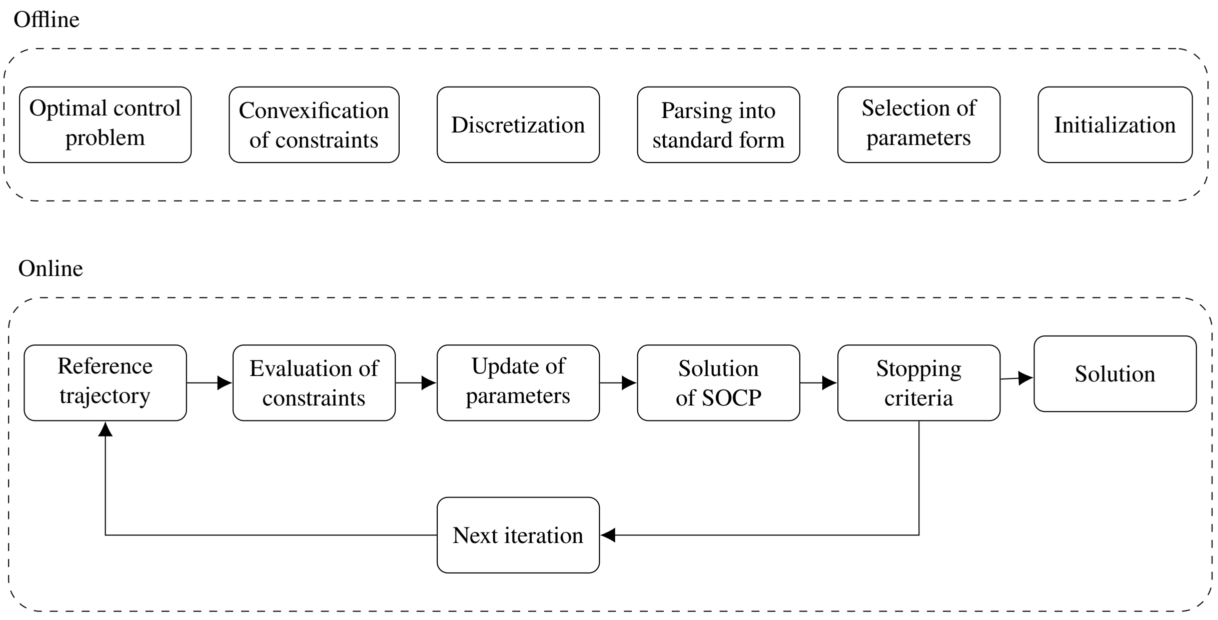


Figure 9.5: Overview of the SCP steps performed offline and online.

parsed into standard form, and remaining parameters are initialized. All of these steps can be performed offline because a significant amount of data (including parts of the constraint matrices and vectors) does not change once the number of discretization points is selected.

Naturally, the actual *solve* step is to be carried out at runtime because the reference trajectory and the corresponding elements of the matrices and vectors in Eq. (9.10) change as the algorithm proceeds. In particular, the approximations of the constraints of Problem 1 in Eq. (5.23) are to be re-evaluated, and the corresponding entries of $\mathbf{c}$, $\mathbf{A}$, $\mathbf{b}$, $\mathbf{G}$, and $\mathbf{h}$ are to be updated. The SOCP is then solved, and if the stopping criteria are not met, the next iteration begins.

In the following subsections, the parsing as a key element of the SCP algorithm is explained in more detail. Moreover, additional aspects of the implementation are addressed, with a particular focus on the steps performed offline and online. If not stated otherwise, we refer to the fuel-optimal problem with fixed final time throughout this section. Moreover, we consider the first-order-hold method for discretizing the OCP, and the high-fidelity model with n-body dynamics, SRP, and variable specific impulse and maximum thrust.

9.3.1 Sequential Convex Programming: Offline

We first address the steps that can be done offline: parsing of the low-thrust trajectory optimization problem, initialization, and parsing into standard form.

Parsing of Low-Thrust Trajectory Optimization Problem

In the context of this dissertation, we require three main variations of the low-thrust trajectory optimization problem:

1) Fixed final time and linear, fixed final boundary conditions
2) Fixed final time and nonconvex final boundary conditions
3) Free final time and nonconvex final boundary conditions (moving target)

The first category considers fixed final time problems with fixed final states, similar to the simulations in Chapters 6, 7, and 8. The only exception concerns KS coordinates in Chapter 7 that belong to category 2) due to the nonconvex final boundary conditions. In the closed-loop guidance scenario, we consider a moving target, i.e., the final time is free and the final boundary conditions are nonconvex.

For a real-time implementation, we require all constraints to be formulated in a way computers can deal with. For example, the discontinuous behavior of the absolute or maximum value operators must be transformed into a series of linear constraints. Moreover, quadratic constraints can be rewritten as second-order cone constraints. For the sake of conciseness, the parsing of relevant operators and constraints is given in the Appendix D.1 and used throughout this section to parse the fuel-optimal problem into an equivalent problem. The discretized fuel-optimal problem with fixed, linear final boundary conditions is given by Problem 3.

Problem 3. Find the vectors $\mathbf{x}$, $\mathbf{u}$, $\boldsymbol{\nu}$, and $\boldsymbol{\eta}$ that solve the following second-order cone program:

$$\underset{\mathbf{x},\mathbf{u},\boldsymbol{\nu},\boldsymbol{\eta}}{\text{minimize}} \quad -w_N + \lambda_\nu \sum_{i=1}^{N-1} \|\boldsymbol{\nu}_i\|_1 + \lambda_\eta \sum_{i=1}^{N} \max(0, \eta_i) \tag{9.11a}$$

$$\text{subject to:} \quad \mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{u}_{k+1}, \boldsymbol{\nu}_k), \quad k = 1, \dots, N-1 \tag{9.11b}$$

$$\Gamma_k \le T_{\max}\, \mathrm{e}^{-\bar{w}_k} (1 - w_k + \bar{w}_k) + \eta_k, \quad k = 1, \dots, N \tag{9.11c}$$

$$\|\boldsymbol{\tau}_k\|_2 \le \Gamma_k, \quad k = 1, \dots, N \tag{9.11d}$$

$$\|\mathbf{x} - \bar{\mathbf{x}}\|_1 \le R \tag{9.11e}$$

$$\mathbf{r}_1 = \mathbf{r}_0,\ \mathbf{v}_1 = \mathbf{v}_0,\ w_1 = w_0 \tag{9.11f}$$

$$|\mathbf{r}_N - \mathbf{r}_f| \le \Delta\mathbf{r}_f,\ |\mathbf{v}_N - \mathbf{v}_f| \le \Delta\mathbf{v}_f \tag{9.11g}$$

with $k = 1, \dots, N$, and the concatenated states $\mathbf{x} = [\mathbf{x}_1^\top, \dots, \mathbf{x}_N^\top]^\top$, nominal states $\bar{\mathbf{x}} = [\bar{\mathbf{x}}_1^\top, \dots, \bar{\mathbf{x}}_N^\top]^\top$, controls $\mathbf{u} = [\mathbf{u}_1^\top, \dots, \mathbf{u}_N^\top]^\top$, virtual controls $\boldsymbol{\nu} = [\boldsymbol{\nu}_1^\top, \dots, \boldsymbol{\nu}_{N-1}^\top]^\top$, and $\boldsymbol{\eta} = [\eta_1, \dots, \eta_N]$. The virtual control $\boldsymbol{\nu}_k \in \mathbb{R}^{n_x}$ for the dynamics is unconstrained, whereas $\eta_k \in \mathbb{R}_{\ge 0}$. Note that we impose the trust region only on the states to reduce the number of variables. Our simulations suggest that there is no significant difference if the controls are included in the trust-region constraint.

Using the parsing of the absolute value, $\max$ operator and 1-norm in the Appendix D.1, the reformulated problem is given by Problem 4.

Problem 4. Find the vectors $\mathbf{x}$, $\mathbf{u}$, $\boldsymbol{\nu}$, $\boldsymbol{\eta}$, $\mathbf{s}_\nu$, $\mathbf{s}_\eta$, and $\mathbf{s}_{\text{TR}}$ that solve the following second-order cone program:

$$\underset{\mathbf{x},\mathbf{u},\boldsymbol{\nu},\boldsymbol{\eta},\mathbf{s}_\nu,\mathbf{s}_\eta,\mathbf{s}_{\text{TR}}}{\text{minimize}} \quad -w_N + \lambda_\nu \mathbf{1}^\top \mathbf{s}_\nu + \lambda_\eta \mathbf{1}^\top \mathbf{s}_\eta \tag{9.12a}$$

$$\text{subject to:} \quad \mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{u}_{k+1}, \boldsymbol{\nu}_k), \quad k = 1, \dots, N-1 \tag{9.12b}$$

$$\boldsymbol{\nu}_k \leq \mathbf{s}_{\nu,k}, \; -\boldsymbol{\nu}_k \leq \mathbf{s}_{\nu,k}, \quad k = 1, \dots, N-1 \tag{9.12c}$$

$$-s_{\eta,k} \leq 0, \; \eta_k \leq s_{\eta,k}, \; -\eta_k \leq 0, \quad k = 1, \dots, N \tag{9.12d}$$

$$\Gamma_k \leq T_{\max} \mathrm{e}^{-\bar{w}_k} (1 - w_k + \bar{w}_k) + \eta_k, \quad k = 1, \dots, N \tag{9.12e}$$

$$\|\boldsymbol{\tau}_k\|_2 \leq \Gamma_k, \quad k = 1, \dots, N \tag{9.12f}$$

$$\mathbf{x} \leq s_{\text{TR}} + \bar{\mathbf{x}}, \; -\mathbf{x} \leq s_{\text{TR}} - \bar{\mathbf{x}}, \; \mathbf{1}^\top \mathbf{s}_{\text{TR}} = R \tag{9.12g}$$

$$\mathbf{r}_1 = \mathbf{r}_0, \; \mathbf{v}_1 = \mathbf{v}_0, \; w_1 = w_0 \tag{9.12h}$$

$$\mathbf{r}_N \leq \Delta \mathbf{r}_f + \mathbf{r}_f, \; -\mathbf{r}_N \leq \Delta \mathbf{r}_f - \mathbf{r}_f \tag{9.12i}$$

$$\mathbf{v}_N \leq \Delta \mathbf{v}_f + \mathbf{v}_f, \; -\mathbf{v}_N \leq \Delta \mathbf{v}_f - \mathbf{v}_f \tag{9.12j}$$

with slack variables $\mathbf{s}_{\nu,k} \in \mathbb{R}^{n_x}$, $s_{\eta,k} \in \mathbb{R}$, and $\mathbf{s}_{\text{TR}}$, $\mathbf{s}_{\text{TR}} \in \mathbb{R}^{N\,n_x}$. $\mathbf{s}_\nu = [\mathbf{s}_{\nu,1}^\top, \dots, \mathbf{s}_{\nu,N-1}^\top]^\top$, $\mathbf{s}_\eta = [\mathbf{s}_{\eta,1}, \dots, \mathbf{s}_{\eta,N}]^\top$ refer to the concatenated quantities. Equations (9.12c) and (9.12d) refer to the penalized virtual controls of the dynamics and upper bound on the thrust magnitude, respectively. The parsed trust-region constraint is given by Eq. (9.12g), and Eqs. (9.12i) and (9.12j) are the relaxed final boundary conditions due to the absolute value. The initial $\mathbf{r}_0$, $\mathbf{v}_0$, w_0 and final boundary conditions $\mathbf{r}_f$, $\mathbf{v}_f$ are fixed. Note that no additional bounds on the states and controls are imposed to keep the number of constraints at a minimum.

In case of a moving target, the final time is free because the final boundary conditions $\mathbf{r}_t(t_f)$ and $\mathbf{v}_t(t_f)$ are nonconvex functions of time. The convexified optimization problem is given by Problem 2 in Section 9.2. Rewriting the norms and absolute values yields Problem 5.

Problem 5. Find the vectors $\mathbf{x}$, $\mathbf{u}$, $\boldsymbol{\nu}$, $\boldsymbol{\eta}$, $\boldsymbol{\zeta}$, $\mathbf{s}_\nu$, $\mathbf{s}_\eta$, $\mathbf{s}_\zeta$, $\mathbf{s}_{\text{TR}}$, and t_f that solve the following second-order cone program:

$$\min_{\substack{t_f,\mathbf{x},\mathbf{u},\boldsymbol{\nu},\boldsymbol{\eta},\boldsymbol{\zeta},\\ \mathbf{s}_\nu,\mathbf{s}_\eta,\mathbf{s}_\zeta,\mathbf{s}_{\text{TR}}}} \quad -w_N + \lambda_\nu \mathbf{1}^\top \mathbf{s}_\nu + \lambda_\eta \mathbf{1}^\top \mathbf{s}_\eta + \lambda_\zeta \mathbf{1}^\top \mathbf{s}_\zeta \tag{9.13a}$$

$$\text{subject to:} \quad \text{Eqs. (9.12b)–(9.12h)} \tag{9.13b}$$

$$-t_f \leq -t_{f,\text{lb}}, \; t_f \leq t_{f,\text{ub}} \tag{9.13c}$$

$$-\mathbf{s}_\zeta \leq \mathbf{0}, \; \boldsymbol{\zeta} \leq \mathbf{s}_\zeta, \; -\boldsymbol{\zeta} \leq \mathbf{0} \tag{9.13d}$$

Table 9.2: Number of elements of each component of the solution vector.

$\mathbf{x}$	$\mathbf{u}$	$\boldsymbol{\nu}$	$\mathbf{s}_{\nu}$	$\boldsymbol{\eta}$	$\mathbf{s}_{\eta}$	$\mathbf{s}_{\text{TR}}$
$n_x\,N$	$n_u\,N$	$n_x\,(N-1)$	$n_x\,(N-1)$	N	N	$n_x\,N$

$$\begin{bmatrix}\mathbf{r}_N\\ \mathbf{v}_N\end{bmatrix} - \begin{bmatrix}\mathbf{r}_t(\bar{t}_f)+\mathbf{v}_t(\bar{t}_f)\;(t_f-\bar{t}_f)\\ \mathbf{v}_t(\bar{t}_f)+\mathbf{a}_t(\bar{t}_f)\;(t_f-\bar{t}_f)\end{bmatrix} \leq \begin{bmatrix}\Delta\mathbf{r}_f\\ \Delta\mathbf{v}_f\end{bmatrix} + \boldsymbol{\zeta} \tag{9.13e}$$

$$-\begin{bmatrix}\mathbf{r}_N\\ \mathbf{v}_N\end{bmatrix} + \begin{bmatrix}\mathbf{r}_t(\bar{t}_f)+\mathbf{v}_t(\bar{t}_f)\;(t_f-\bar{t}_f)\\ \mathbf{v}_t(\bar{t}_f)+\mathbf{a}_t(\bar{t}_f)\;(t_f-\bar{t}_f)\end{bmatrix} \leq \begin{bmatrix}\Delta\mathbf{r}_f\\ \Delta\mathbf{v}_f\end{bmatrix} + \boldsymbol{\zeta} \tag{9.13f}$$

Note that t_f is included in the trust-region constraint in Eq. (9.12g). The case with fixed final time and nonconvex final boundary conditions is given in the Appendix D.2, along with the reformulation of the quadratic objective function for the energy-optimal problem.

Initialization

Even though we are only interested in the optimal state and control profiles, we have seen that additional (slack) variables are needed to formulate the problem in standard form. We therefore define the solution vector $\mathbf{y} \in \mathbb{R}^{n_y}$ to be

$$\mathbf{y} := \left[\mathbf{x}^\top, \mathbf{u}^\top, \boldsymbol{\nu}^\top, \mathbf{s}_{\nu}^\top, \boldsymbol{\eta}^\top, \mathbf{s}_{\eta}^\top, \mathbf{s}_{\text{TR}}^\top\right]^\top \tag{9.14}$$

The number of elements of each component is given in Table 9.2. The total number of elements n_y of $\mathbf{y}$ is

$$n_y = n_x\,N + n_u\,N + n_x\,(N-1) + n_x\,(N-1) + N + N + n_x\,N = 2\,N - 2\,n_x + 4\,n_x\,N + n_u\,N \tag{9.15}$$

In case of a moving target, the variables $t_f \in \mathbb{R}$, $\boldsymbol{\zeta} \in \mathbb{R}^{n_x-1}$, and $\mathbf{s}_{\zeta} \in \mathbb{R}^{n_x-1}$ are simply appended to $\mathbf{y}$, and the length of $\mathbf{s}_{\text{TR}}$ increases by one. The following main steps are performed within the initialization step:

1) Setup of the dynamical model: Based on the desired model (for example, two-body or n-body dynamics), relevant data is defined and loaded. This includes the gravitational constants of the perturbing bodies for the n-body model and the ephemeris. The latter is retrieved offline, stored on board, and then interpolated to obtain the required states. In addition, parameters for the solar radiation pressure are calculated and stored.
2) Normalization: All quantities are normalized using some time unit, length unit, velocity unit, acceleration unit, force unit, and power unit.

3) Jacobian matrices: The Jacobian matrices are computed analytically based on the dynamical model, and stored as functions that can be called during the solution process.
4) Discretization: Depending on the discretization method, the number of collocation points, segments, mesh data, etc., are initialized. For FOH, this includes the matrices and vectors $\mathbf{A}_k$, $\mathbf{B}_k^-$, $\mathbf{B}_k^+$, and $\mathbf{q}_k$, $k = 1, \ldots, N-1$, in Eq. (6.66). Each matrix $(\cdot)_k$ is stored as a 1D array, and the matrices are concatenated column-wise to obtain a single 2D array.
5) Initial guess: The initial guess is generated, for example using the shape-based approach in the Appendix A.
6) Sizes and indices: The sizes of the matrices and vectors are defined, and the corresponding indices of the entries of the solution vector are stored.
7) Initialization of parameters: All parameters are initialized, for example tolerances, trust-region parameters, etc.

Parsing into Standard Form

Besides the parsing of the discretized low-thrust trajectory optimization problem, the matrices and vectors in Eq. (9.10) are to be populated. This step is very important for a real-time implementation because several entries remain constant and can thus be defined offline. Note that the matrices are not created explicitly due to their sparse nature, but rather a so-called sparse triplet is stored for each matrix. It consists of three 1D arrays that contain the actual data, the row index, and the column index in column compressed storage format. This way, a large, sparse matrix can be stored efficiently. Details on the structure of the matrices and vectors for the constraints and objective function can be found in the Appendix D.3.

The number of rows, nonzero elements, and changing entries of the matrices and vectors are illustrated in Tables 9.3 and 9.4, respectively. We assume $n_x = 7$ states and $n_u = 4$ controls. Note that the elements are updated only if a solution is accepted. If it is rejected, only the trust-region radius in $\mathbf{b}_{\mathrm{TR}}$ changes for the next iteration.

The sparsity and specific values for $N = 100$ and $N = 300$ discretization points are given in Table 9.5. These values refer to the fixed final time problem because the moving target case has only slightly more elements. Apparently, the matrices $\mathbf{A}$ and $\mathbf{G}$ are extremely sparse as more than $99.5\,\%$ of the elements are zero. Most of the data that requires an update is due to the dynamics, whereas a large portion of the remaining constraints is constant and can be initialized offline.

9.3.2 Sequential Convex Programming: Online

Algorithm 1 summarizes the steps that are carried out at runtime after the parsing and initialization. Instead of using the software SPICE [192] to retrieve the positions of the perturbing bodies, the ephemerides are

Table 9.3: Overview of number of rows, nonzero elements, and changing entries for each matrix. *MT* refers to the moving target case.

Matrix	Number of rows	Number of nonzero elements	Number of changing elements
$\mathbf{A}_{\text{dyn}}$	$7\,(N-1)$	$107\,(N-1)$ [MT: $114\,(N-1)$]	$93\,(N-1)$ [MT: $100\,(N-1)$]
$\mathbf{A}_{x0}$	7	7	0
$\mathbf{A}_{\text{TR}}$	1	$7\,N$ [MT: $7\,N+1$]	0
$\mathbf{G}_{\text{thrust}}$	N	$3\,N$	N
$\mathbf{G}_{\text{TR}}$	$14\,N$	$28\,N$ [MT: $28\,N+4$]	0
$\mathbf{G}_{x_f}$	12	12 [MT: 36]	0 [MT: 6]
$\mathbf{G}_{\nu}$	$14\,(N-1)$	$28\,(N-1)$	0
$\mathbf{G}_{\eta}$	$3\,N$	$4\,N$	0
$\mathbf{G}_{\zeta}{}^{*}$	18	24	0
$\mathbf{G}_{\text{bounds}}{}^{*}$	2	2	0
$\mathbf{G}_{\text{socc}}$	$4\,N^{\dagger}$	$4\,N$	0
Total	$43\,N-1$	$153\,N-60$	$94\,N-93$
Total MT	$43\,N+19$	$153\,N-34$	$101\,N-94$

* For the moving target problem only.
† Corresponds to N second-order cone constraints.

Table 9.4: Overview of elements and changing entries for each vector. *MT* refers to the moving target case.

Vector	Number of elements	Number of changing elements
$\mathbf{b}_{\text{dyn}}$	$7\,(N-1)$	$7\,(N-1)$
$\mathbf{b}_{x0}$	7	0
$\mathbf{b}_{\text{TR}}$	1	1
$\mathbf{h}_{\text{thrust}}$	N	N
$\mathbf{h}_{\text{TR}}$	$14\,N$ [MT: $14\,N+2$]	$14\,N$ [MT: $14\,N+2$]
$\mathbf{h}_{x_f}$	12	0 [MT: 12]
$\mathbf{h}_{\nu}$	$14\,(N-1)$	0
$\mathbf{h}_{\eta}$	$3\,N$	0
$\mathbf{h}_{\zeta}{}^{*}$	18	0
$\mathbf{h}_{\text{bounds}}{}^{*}$	2	0
$\mathbf{h}_{\text{socc}}$	$4\,N^{\dagger}$	0
$\mathbf{c}$	$8\,N-6$ [MT: $8\,N$]	0
Total	$51\,N-7$	$22\,N-6$
Total MT	$51\,N+13$	$22\,N+8$

* For the moving target problem only.
† Corresponds to N second-order cone constraints.

Table 9.5: Nonzero elements for $N = 100$ and $N = 300$.

N	Quantity	Sparsity, %	Nonzero elements	Changing elements	Pct. of changing elm., %
	$\mathbf{c}$	–	794	0	0
	$\mathbf{A}$	99.52	11 300	9207	81.45
100	$\mathbf{b}$	–	701	694	99.00
	$\mathbf{G}$	99.95	6684	100	1.50
	$\mathbf{h}$	–	3598	1500	41.69
	$\mathbf{c}$	–	2394	0	0
	$\mathbf{A}$	99.84	34 100	27 807	81.55
300	$\mathbf{b}$	–	2101	2094	99.67
	$\mathbf{G}$	99.98	20 084	300	1.49
	$\mathbf{h}$	–	10 798	4500	41.67

stored on board and interpolated. This way, no additional software is required, and the computationally expensive calls of external functions is avoided. In the closed-loop guidance or the moving target scenario with no-thrust constraints, the number of discretization points is adjusted if the time of flight changes. Thus, the size of the matrices and vectors may change, too. In that case, slightly more effort is required to update the respective quantities. Yet, we will see in the following section that most of the CPU time is required to solve the second-order cone program.

9.4 Numerical Simulations

We consider transfers to the asteroids 2000 SG344 and 2010 UE51 as they are potential targets of ESA's Miniaturised Asteroid Remote Geophysical Observer (M-ARGO) mission [129]. For each transfer, a Monte-Carlo analysis with 1000 closed-loop guidance simulations is performed to assess the reliability of the proposed approach. In that context, the perturbations of the spacecraft state are generated randomly according to Fig. 9.4. The standard SCP algorithm of Section 5.2 is used with Cartesian coordinates and the high-fidelity model that is described in Chapter 8. No-thrust cycles are included where the thruster has to remain off for one day, followed by 6.3 and 7.4 days of thrust, respectively. These values are similar to the duty cycles of the M-ARGO mission [129]. The same thruster model as in Section 8.4 is used, and the reoptimization cycle is 14.5 and 16.8 days, respectively. We found in our simulations that varying the values between 7 and 21 days does not significantly change the results. The reason is that the errors (e.g., thruster misalignment or modeling errors) do not accumulate considerably in such a short amount of time if a high-fidelity model is used to compute the reference trajectory. Orbit determination is performed every ten days within optical navigation simulations in [80]. As the position and velocity

Algorithm 1 SCP steps performed online

```
Input: Reference trajectory $(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \bar{t}_f)$, SCP parameters, discretization data, flag movingTarget
Output: Solution $(\mathbf{x}^*, \mathbf{u}^*, t_f^*)$
1: Set converged = false
2: while converged = false do
3:     for each segment $k = 1, \ldots, N-1$ do
4:         Interpolate ephemeris to obtain positions of perturbing bodies
5:         Compute $\mathbf{A}_k$, $\mathbf{B}_k^-$, $\mathbf{B}_k^+$, $\mathbf{q}_k$
6:         if movingTarget = true then
7:             Compute $\mathbf{S}_k$
8:         end if
9:     end for
10:    Update changing elements of $\mathbf{A}_{\text{dyn}}$, $\mathbf{b}_{\text{dyn}}$, $\mathbf{G}_{\text{thrust}}$, $\mathbf{h}_{\text{thrust}}$, $\mathbf{h}_{\text{TR}}$
11:    if movingTarget = true then
12:        Update changing elements of $\mathbf{G}_{x_f}$, $\mathbf{h}_{x_f}$
13:        if $N$ changed then
14:            Update $\mathbf{c}$, $\mathbf{A}$, $\mathbf{b}$, $\mathbf{G}$, $\mathbf{h}$ accordingly
15:        end if
16:    end if
17:    Set accept = false
18:    while accept = false do
19:        Solve SOCP to obtain current solution $(\mathbf{x}^*, \mathbf{u}^*, t_f^*)$
20:        if movingTarget = true then
21:            Update mesh to ensure similar number of nodes per time unit
22:            Interpolate obtained states and controls onto new mesh
23:        end if
24:        Calculate actual $\varphi$ and predicted cost $\hat{\varphi}$
25:        if stopping criteria satisfied then
26:            Set converged = true
27:            return solution $(\mathbf{x}^*, \mathbf{u}^*, t_f^*)$
28:        else
29:            Determine ratio $\rho$ of the actual and predicted cost decrease
30:            Compute new trust-region radius
31:            if solution is accepted then
32:                Set accept = true
33:                Set $(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \bar{t}_f) = (\mathbf{x}^*, \mathbf{u}^*, t_f^*)$
34:            end if
35:        end if
36:        Update entry of trust-region radius in $\mathbf{b}$
37:    end while
38: end while
```

errors increase if the state is determined less frequently, we select slightly larger values of 14.5 and 16.8 days to assess the performance when larger errors are present.

The ephemeris of the perturbing bodies (and asteroid in case of a moving target) are stored on the single-board computer. We observed that the time resolution of the data points is not critical, and a sampling time in the order of one day is often sufficiently accurate. Cubic splines are created offline and also stored on board for the interpolation. These can then be evaluated efficiently at runtime to retrieve the desired states. The SCP algorithm is implemented in C++ where we use the library *Eigen* [199] for linear algebra operations. The second-order cone program is solved using ECOS [86] because it was shown that its performance is comparable to an onboard, flight-tested interior-point method [101]. The optimization of the reference trajectories is performed on the Raspberry Pi 3 Model B+ whose technical specifications are given in Table 9.1. Moreover, we also investigate how the CPU time changes when an underclocked version of the Raspberry Pi with only $600\,\mathrm{MHz}$ is used.

Additional relevant parameters for the transfers and SCP algorithm are given in Tables 9.6 and 9.7, respectively. Note that we select values of $\varepsilon_c = 10^{-6}$ and $\varepsilon_J = 10^{-4}$ at the beginning to keep the number of iterations at a minimum, and then switch to smaller feasibility and optimality tolerances 150 days before arrival to ensure a high final accuracy. Physical constants are given in Table 9.8.

We first address the nominal case where the perturbations stay within the expected values, and then consider a failure condition with considerably larger disturbances towards the end of the transfer. Finally, we assess the performance of the SCP algorithm on the Raspberry Pi.

9.4.1 Monte-Carlo Analysis: Nominal Case

In order to assess the reliability of the proposed approach, we perform 1000 closed-loop guidance simulations for each transfer. One simulation consists of the complete process depicted in Fig. 9.3, i.e., the algorithm is initialized, data is loaded onto the Raspberry Pi, and then propagating and reoptimizing the trajectory alternate until the spacecraft reaches the target. The initial number of nodes is $N_0 = 180$, and the minimum number is $N_{\min} = 10$.

Transfer to Asteroid 2000 SG344

Before a new trajectory is determined, the state of the spacecraft is randomly perturbed. Typical magnitudes with $r_{\max} = 20\,000\,\mathrm{km}$ and $v_{\max} = 10\,\mathrm{m\,s^{-1}}$ are illustrated in Fig. 9.6. It is evident that the imposed deviations reduce over time, and exceed values of $30\,000\,\mathrm{km}$ (position) and $15\,\mathrm{m\,s^{-1}}$ (velocity) at the beginning of the transfer. We also consider smaller values of $r_{\max} = 2000\,\mathrm{km}$ and $v_{\max} = 3\,\mathrm{m\,s^{-1}}$ to investigate whether the error magnitude affects the results. Examples of the corresponding perturbations are shown in Fig. 9.7.

Table 9.6: Simulation values for the transfers from SEL$_2$ to the asteroids 2000 SG344 and 2010 UE51 [129].

Parameter	SEL$_2$ - 2000 SG344	SEL$_2$ - 2010 UE51
Initial epoch	04-Feb-2024 12:00:00 UTC	25-Jan-2024 12:00:00 UTC
Initial position $\mathbf{r}_0$, AU	$[-0.70186065, 0.70623244, -3.51115 \times 10^{-5}]^\top$	$[-0.565594862, 0.817852054, -4.21043 \times 10^{-5}]^\top$
Initial velocity $\mathbf{v}_0$, VU	$[-0.73296949, -0.71590485, 4.40245 \times 10^{-5}]^\top$	$[-0.847308876, -0.578399661, 3.70665 \times 10^{-5}]^\top$
Initial mass m_0, kg	22.6	4000
Final position $\mathbf{r}_f$, AU	$[0.967546555, 0.166086135, 6.86064 \times 10^{-5}]^\top$	$[0.460719330, 0.884256883, 6.59008 \times 10^{-3}]^\top$
Final velocity $\mathbf{v}_f$, VU	$[-0.236236309, 0.978953388, -1.98707 \times 10^{-3}]^\top$	$[-0.928506079, 0.458224786, 7.99479 \times 10^{-3}]^\top$
Final mass $m(t_f)$, kg	free	free
Min. input power $P_{\text{in,min}}$, W	20	
Max. input power $P_{\text{in,max}}$, W	120	
Max. thrust T_{max}, N	2.2519×10^{-3}	
Max. specific impulse $I_{\text{sp,max}}$, s	3067	
Spacecraft area A_{SC}, m^2	0.05	
Reflectivity coefficient C_R	1.3	
Time of flight t_f, days	650	750
Accepted pos. error $\Delta\mathbf{r}$, km	1000	
Accepted vel. error $\Delta\mathbf{v}$, $\text{m}\,\text{s}^{-1}$	1	
Reoptimization cycle, days	14.5	16.8
No-thrust cycle, days	6.3 on, 1 off	7.4 on, 1 off
Gauss–Markov param. κ	0.05	
Gauss–Markov param. φ, days	0.0417	

Table 9.7: Parameters of the algorithm.

Parameter	Value
Feasibility tol. ε_c	10^{-6}, 10^{-7}
Optimality tol. ε_J	10^{-4}, 10^{-5}
Max. iterations	50
λ_ν, λ_η, λ_ζ	10.0, 10.0, 10.0
ρ_0, ρ_1, ρ_2	0.01, 0.25, 0.85
α, β	1.5, 1.5
N_0	180
N_{min}	10

Table 9.8: Physical constants in all simulations.

Parameter	Value
Gravitational const. μ	$1.327\,124\,4 \times 10^{11}\,\text{km}^3\,\text{s}^{-2}$
Gravitational accel. g_0	$9.806\,65 \times 10^{-3}\,\text{km}\,\text{s}^{-2}$
Length unit LU = AU	$1.495\,978\,707 \times 10^{8}\,\text{km}$
Velocity unit VU	$\sqrt{\mu\,\text{AU}^{-1}}$
Time unit TU	AU VU^{-1}
Acceleration unit ACU	VU TU^{-1}
Mass unit MU	m_0

Table 9.9: Results for 1000 closed-loop guidance simulations to asteroid 2000 SG344 with fixed final time (median values, and maximum values in brackets).

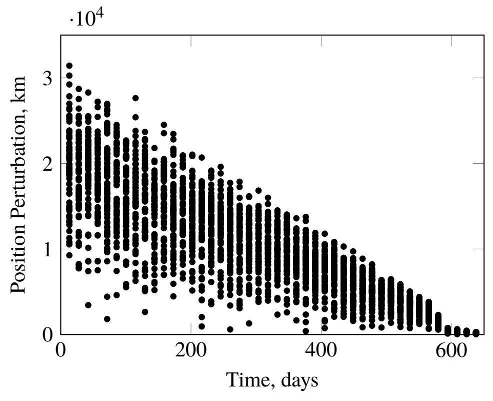
Perturb.	Success rate, %	Iterations	CPU time, s	Propellant, kg	Δr_{prop}, km	Δv_{prop}, $\frac{\text{m}}{\text{s}}$
Small	100	2 (22)	2.9 (58.8)	1.21 (1.22)	6 (403)	0.009 (0.04)
Large	100	2 (23)	3.2 (63.3)	1.21 (1.25)	6 (455)	0.009 (0.05)

The results of 1000 simulations to asteroid 2000 SG344 are summarized in Table 9.9. The two rows refer to the smaller and larger perturbations, respectively. All simulations converged successfully, i.e., the final position and velocity errors are smaller than $1000\,\text{km}$ and $1\,\text{m}\,\text{s}^{-1}$ as defined in Section 3.2. As expected, the median number of iterations of 2 is very small.

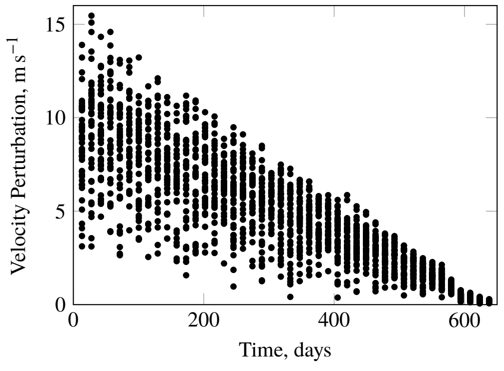
According to Fig. 9.8a, all reoptimizations of the trajectory required only very few iterations (median values are shown, and the error bars refer to the minimum and maximum values). Only some cases at the beginning and end of the transfer required more iterations due to the reduction of the trust-region size. This confirms that the obtained solutions lie in the neighborhood of the nominal trajectory. The median CPU time per optimization is around $3\,\text{s}$ on the Raspberry Pi, with maximum values of approximately $60\,\text{s}$ when more than 20 iterations are needed. The CPU time is consistent with the evolution of the iterations, and decreases over time because the number of nodes also diminishes (see Figs. 9.8b and 9.8c). If the simulations are performed on an underclocked Raspberry Pi with $600\,\text{MHz}$, the solving times approximately double.

The median fuel consumption is $1.21\,\text{kg}$, thus being the same as the nominal one. Depending on the perturbations, the algorithm finds also solutions where the required propellant is slightly larger. Apparently, more fuel is needed if the perturbations increase. The final position Δr_{prop} and velocity Δv_{prop} errors are shown in Fig. 9.9. The median is $6\,\text{km}$ (position) and $0.009\,\text{m}\,\text{s}^{-1}$ (velocity), and all values are smaller than $500\,\text{km}$ and $0.05\,\text{m}\,\text{s}^{-1}$, respectively. Moreover, more than $99\,\%$ of the errors are below $80\,\text{km}$, therefore resulting in sufficiently accurate solutions. The errors increase only if the disturbances near the target become large. Furthermore, the maximum errors are slightly bigger if larger disturbances are considered. Besides that, the magnitude of the perturbation does not have a significant impact on the results.

Figure 9.10a shows the nominal thrust magnitude and a typical profile obtained in the closed-loop guidance simulation. Even though they are very similar, they do not overlap exactly due to the imposed perturbations. The corresponding profile with no-thrust constraints is illustrated in Fig. 9.10b, and the transfer trajectory is shown in Fig. 9.10c.

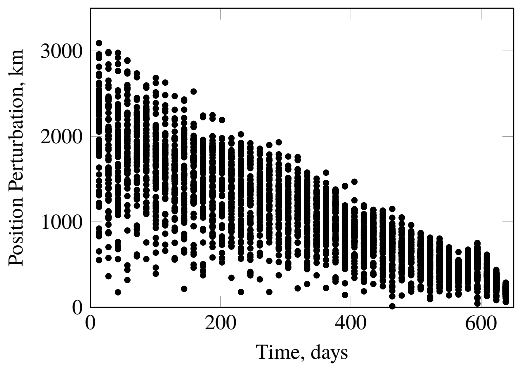


(a) Position perturbations.

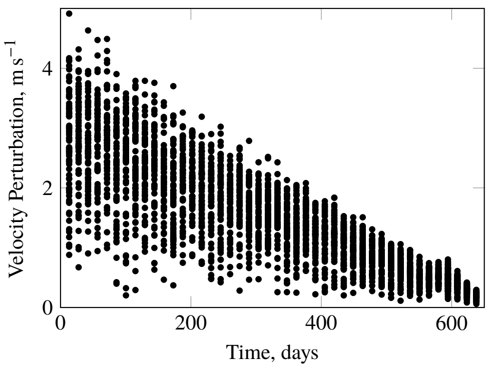


(b) Velocity perturbations.

Figure 9.6: Typical perturbations for position and velocity over time for the transfer to asteroid 2000 SG344.

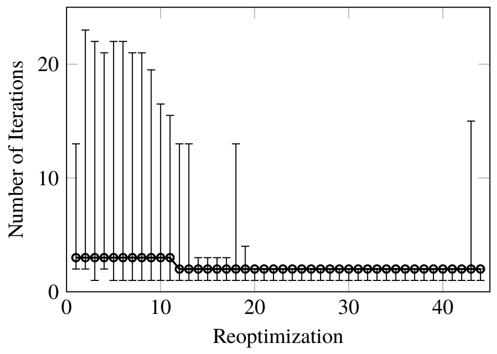


(a) Position perturbations.

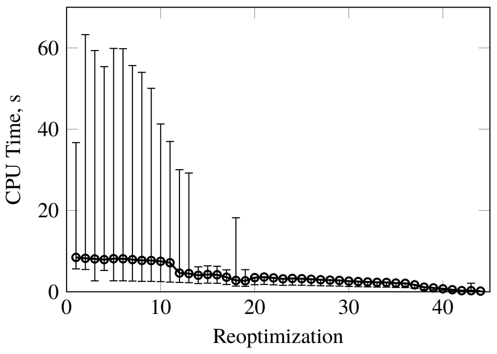


(b) Velocity perturbations.

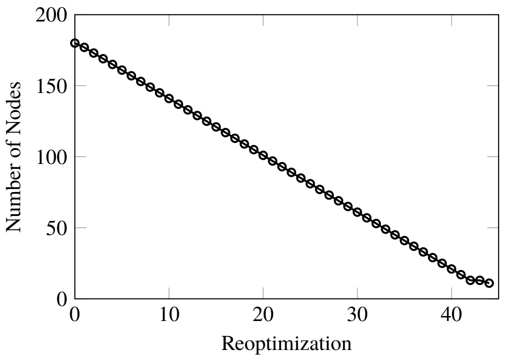
Figure 9.7: Smaller perturbations for position and velocity over time for the transfer to asteroid 2000 SG344.



(a) Number of iterations.

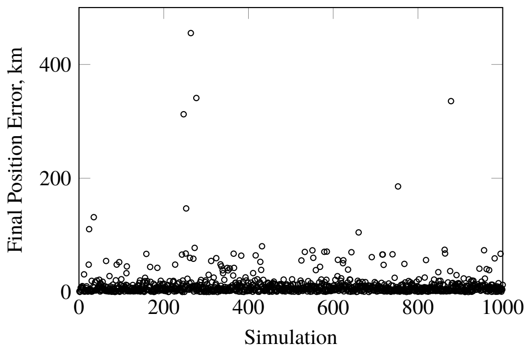


(b) CPU time.

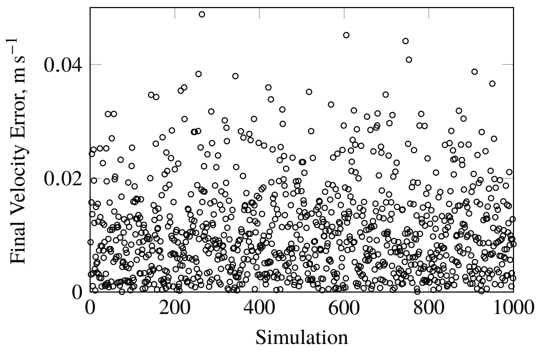


(c) Number of nodes.

Figure 9.8: Evolution of number of iterations, CPU time, and number of nodes for the closed-loop guidance simulation to asteroid 2000 SG344.

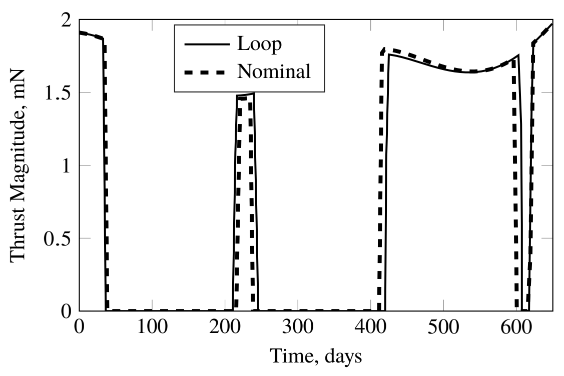


(a) Final position errors.

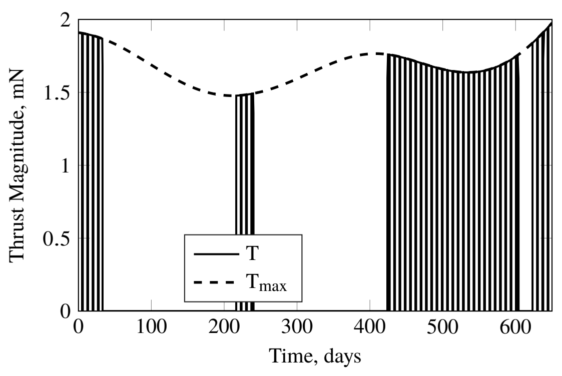


(b) Final velocity errors.

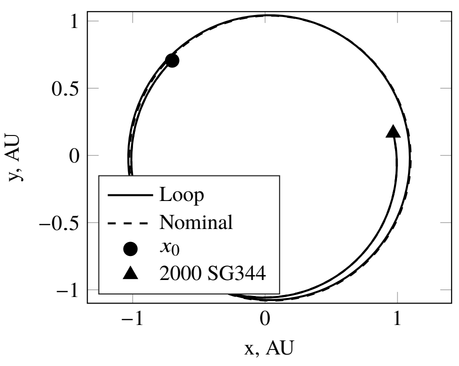
Figure 9.9: Propagation errors for position and velocity for 1000 simulations for the transfer to asteroid 2000 SG344.



(a) Nominal and closed-loop thrust magnitude.



(b) Thrust magnitude with no-thrust periods.



(c) Transfer trajectory.

Figure 9.10: Thrust magnitude and trajectory obtained from the closed-loop simulation to asteroid 2000 SG344.

Table 9.10: Results for 1000 closed-loop guidance simulations to asteroid 2010 UE51 with fixed final time (median values, and maximum values in brackets).

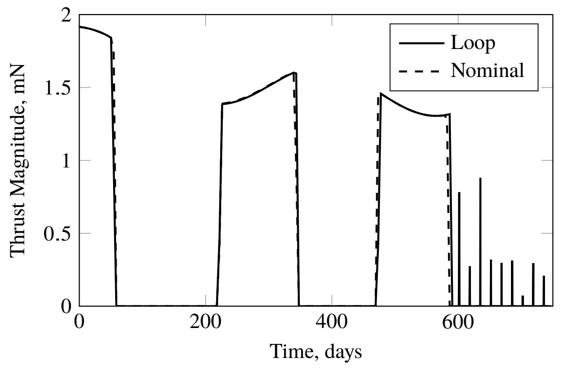
Success rate, %	Iterations	CPU time, s	Propellant, kg	Δr_{prop}, km	Δv_{prop}, $\frac{\text{m}}{\text{s}}$
100	2 (20)	3.1 (54.0)	1.19 (1.24)	2 (91)	3.7×10^{-4} (0.011)

Transfer to Asteroid 2010 UE51

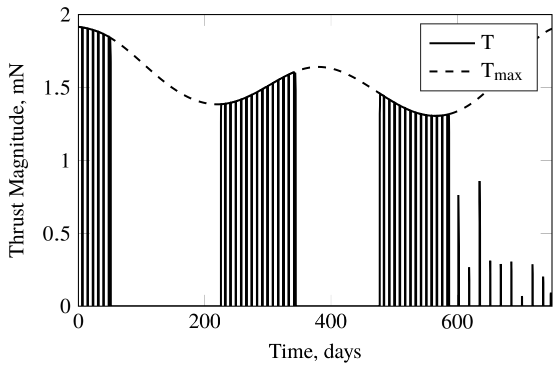
The transfer time to asteroid 2010 UE51 is 750 days, therefore being 100 days longer. Only the results corresponding to $r_{\max} = 20\,000\,\text{km}$ and $v_{\max} = 10\,\text{m}\,\text{s}^{-1}$ are reported in Table 9.10. Again, all 1000 simulations converged successfully, and the number of iterations and CPU time are similar to the previous transfer. The median fuel consumption of $1.19\,\text{kg}$ is slightly larger than the nominal one of $1.17\,\text{kg}$. The final position and velocity errors are smaller than the ones for the transfer to 2000 SG344. One reason is the shape of the state and control profiles that are shown in Fig. 9.11. We observe that the nominal trajectory ends with a long coast arc. However, due to the perturbations in the closed-loop simulation, the spacecraft needs to perform some small correction maneuvers at the end of the transfer in order to reach the target. This is in contrast to the transfer to asteroid 2000 SG344 where the thruster operates over a significantly larger time span towards the end. Therefore, no additional control actions are visible because the spacecraft simply needs to alter the thrust directions instead of adding new thrust arcs. The small control spikes in the 2010 UE51 transfer result in a smaller final error due to the higher flexibility, but also require slightly more fuel.

9.4.2 Monte-Carlo Analysis: Failure Case

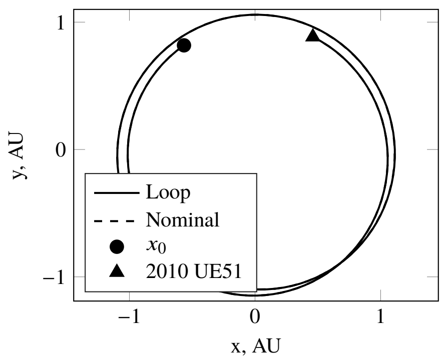
We perform an additional Monte-Carlo analysis with 1000 simulations for the 2000 SG344 transfer to assess the performance when the failure conditions described in Section 9.2 occur. In this context, each component of the position and velocity is randomly perturbed by values of up to $50\,000\,\text{km}$ and $30\,\text{m}\,\text{s}^{-1}$, respectively (see Fig. 9.12). This is similar to a failure of the propulsion system when there is no thrust for several days. The time when this failure occurs is selected randomly between 550 and 610 days. Depending on the time and magnitude of the disturbance, a feasible trajectory may not exist anymore. In that case, the algorithm switches to the moving target formulation and tries to regain feasibility by increasing the time of flight. When a feasible solution is found, the algorithm switches back to the fixed final time problem. The results are given in Table 9.11. Despite these large deviations, the algorithm is always able to determine a feasible trajectory and successfully reach the target. The column Δt_f indicates the additional time of flight because of the moving target. Apparently, only one day is often sufficient to reach the target at a future point. However, there are cases where the time of flight increases by more than 16 days as shown in Fig. 9.13. The number of iterations and CPU time per optimization solely refer to



(a) Nominal and closed-loop thrust magnitude.

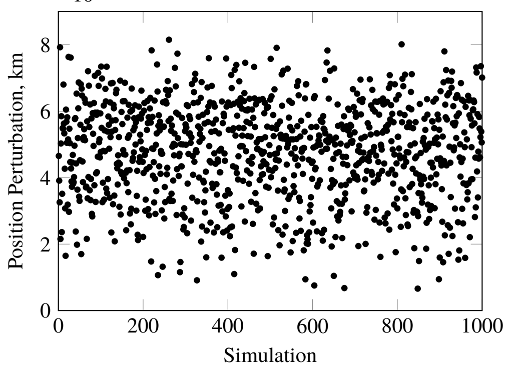


(b) Thrust magnitude with no-thrust periods.

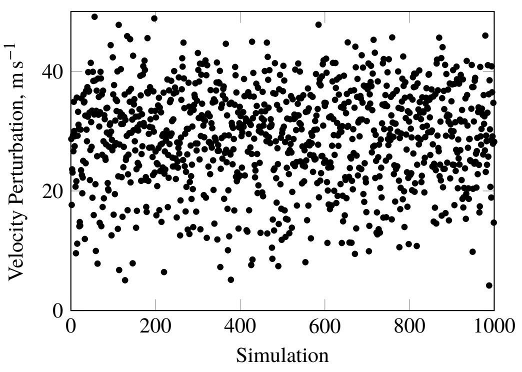


(c) Transfer trajectory.

Figure 9.11: Thrust magnitude and trajectory obtained from the closed-loop simulation to asteroid 2010 UE51.



(a) Position perturbations.



(b) Velocity perturbations.

Figure 9.12: Perturbations for position and velocity over time for the transfer to asteroid 2000 SG344 with failure conditions.

Table 9.11: Results for the 1000 closed-loop guidance simulations to asteroid 2000 SG344 with failure conditions (median values, and maximum values in brackets).

Success rate, %	Δt_f, days	Iterations	CPU time, s	Propellant, kg	Δr_{prop}, km	Δv_{prop}, $\frac{\mathrm{m}}{\mathrm{s}}$
100	0.9 (16.4)	26 (38)	11.0 (17.6)	1.22 (1.27)	7 (83)	0.009 (0.047)

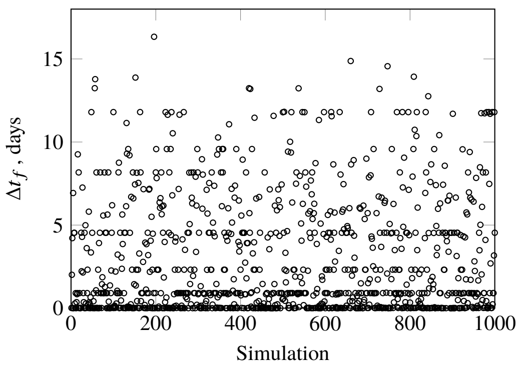


Figure 9.13: Additional flight time when solving the moving target problem in the failure scenario.

the simulations when the moving target problem is solved. It becomes evident that the median number of iterations is considerably larger compared to the nominal case. The reason is the more complex problem in combination with the substantial perturbations. As expected, the required propellant is higher because of the additional control actions. Interestingly, the final accuracy is also higher. One explanation is the higher flexibility by allowing the asteroid state to vary. Keeping the arrival time free can therefore also contribute to reducing the error on the final boundary conditions.

9.4.3 Discussion

Our simulations suggest that it is in general possible to reoptimize low-thrust trajectories in real time within the deep-space cruise. The proposed closed-loop guidance approach is robust against extremely large disturbances and very reliable as it was able to determine feasible solutions at any time. We found that perturbations during the cruise phase are often not critical, and can be handled by SCP as long as a feasible solution exists. This, however, may not be the case anymore if greater errors occur when the spacecraft is close to the target, for example in the context of some failure condition. The moving target approach has shown to be an effective means to regain feasibility by allowing the target state to vary.

Due to the similar computational capability of the Raspberry Pi and real spacecraft hardware, the reported CPU times are a useful indicator of the real-time performance. The CPU times range from few seconds to one minute per optimization, which we regard excellent for the deep-space cruise that can last years. Although the solving times approximately double when a less powerful onboard computer is considered, they seem still acceptable. Yet, the Raspberry Pi used in this dissertation is not flight-proven, and additional steps are required before the approach becomes ready for real applications. Especially more detailed analyses regarding the uncertainty and system noise are recommended.

Interestingly, SCP finds solutions that are close to the nominal ones despite the large perturbations. Therefore, reoptimizing the trajectory frequently may replace the traditional tracking of a given reference without sacrificing optimality and accuracy.

Even though there is still a tremendous effort required to verify and validate the proposed approach for real space missions, it is a promising first step towards the real-time guidance of low-thrust spacecraft.

9.4.4 Performance on a Single-Board Computer

We briefly comment on the performance of the SCP algorithm on the Raspberry Pi 3 Model B+. In particular, we consider the problem of optimizing a trajectory using a different number of discretization points and different models. Therefore, Figs. 9.14 and 9.15 compare typical CPU times per SCP iteration for the following cases:

1) Fixed final time, high-fidelity model, no-thrust constraints (referred to as SCP)
2) Fixed final time, high-fidelity model, without no-thrust constraints (SCP_{NT})
3) Moving target, high-fidelity model, no-thrust constraints (SCP_{MT})
4) Fixed final time, high-fidelity model, no-thrust constraints, single-board computer underclocked to $600\,\text{MHz}$ (SCP_{600})

The last item refers to the case where we reduced the clock frequency of the Raspberry Pi from $1.4\,\text{GHz}$ to $600\,\text{MHz}$. Even though it is expected that flight processors of CubeSats become increasingly powerful, we want to investigate how the SCP algorithm performs on a single-board computer with lower performance. Without loss of generality, we select the transfer to asteroid 2000 SG344 (see Table 9.6) with $N = 100$ and $N = 300$ discretization points to measure the CPU time.

According to Fig. 9.14, all variants require similar CPU times per iteration. This is also true for the time needed to solve the second-order cone program with ECOS (indicated by the horizontal line within each bar). Yet, neglecting no-thrust constraints results in a slightly faster solving time, and considering the moving target problem takes often marginally longer. Interestingly, the time within ECOS accounts for approximately $50\,\%$ of the total time for $N = 100$, and becomes more dominant when the number of nodes increases. The CPU time ranges from 2 to $7\,\text{s}$ per SCP iteration. Considering a maximum number of iterations of approximately 25 that we observed in the closed-loop guidance simulations, this would result in a total time of 50 to $175\,\text{s}$ per optimization. Therefore, even if a larger number of nodes is needed, we consider a computational time of three minutes still acceptable for interplanetary transfers.

Figure 9.15 compares the CPU times of SCP for the standard and underclocked Raspberry Pi. A SCP iteration takes approximately twice as long in case of the $600\,\text{MHz}$ version, hence resulting in a maximum

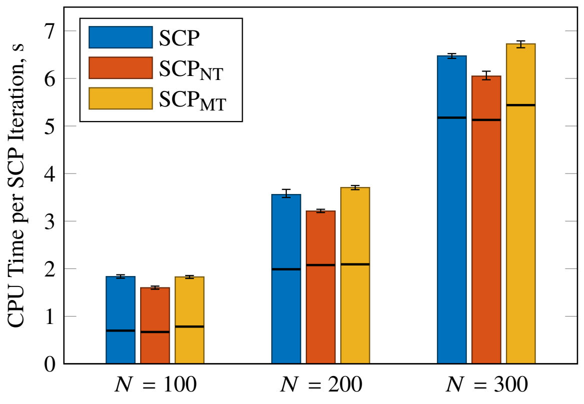


Figure 9.14: Comparison of CPU time per SCP iteration for different models.

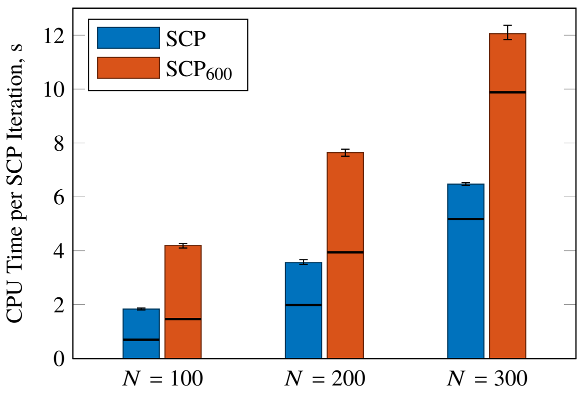


Figure 9.15: Comparison of CPU time per SCP iteration for standard and under-clocked Raspberry Pi.

time of 12 s for $N = 300$. This yields CPU times of approximately 100 to 300 s per optimization for the previous example. Even this seems to be an acceptable amount of time for our application.

Although it may seem surprising that using a high-fidelity model with n-body dynamics does not result in extremely high CPU times, we have shown that the computational burden can still be handled by a single-board computer. The SOCP solving time does not increase considerably because the matrices and vectors are similar compared to the ones for lower-fidelity models. The additional floating point operations during the propagation step require only little time if a compiled language like C or C++ is used. Furthermore, ephemeris can be stored on board, and interpolating the data can be performed efficiently at runtime if the splines are created offline.

Yet, it is still to be investigated how other processes influence the performance, and whether the required memory or memory access become critical in real missions.

10 Koopman Operator Theory

In this chapter, we introduce the Koopman operator theory, and provide the theoretical background to make the reader familiar with the formalism. Moreover, we include non-conservative perturbations (e.g., external controls due to thrust) in the formulation to transform highly nonlinear dynamics into a bilinear system of higher dimension. Moreover, a data-driven approach to obtain a completely linear system is presented. The accuracy of the Koopman-transformed systems is assessed when propagating the dynamics. In addition, the linearization accuracy index is compared for the original and Koopman system. The Duffing oscillator is used to demonstrate the procedure, and it is described how the KOT can be applied to the unperturbed and perturbed Keplerian motion. Parts of this chapter are taken from our work in [16].

10.1 Theoretical Background

The Koopman operator is an infinite-dimensional, linear operator that describes the evolution of observable functions. Given a general nonlinear dynamical system

$$\frac{\mathrm{d}\mathbf{x}(t)}{\mathrm{d}t} = \mathbf{f}(\mathbf{x}), \quad \mathbf{x}(t_0) = \mathbf{x}_0 \tag{10.1}$$

where $\mathbf{x} \in \mathbb{R}^{n_x}$ and $\mathbf{f} \in \mathbb{R}^{n_x}$ are defined in the n_x-dimensional state space. Let $g(\mathbf{x})$ be an observable function. KOT describes how these functions evolve in an extended, infinite-dimensional Hilbert space [159]:

$$\frac{\mathrm{d}g(\mathbf{x})}{\mathrm{d}t} = \mathcal{K}\left(g(\mathbf{x})\right) = \left(\nabla_{\mathbf{x}} g(\mathbf{x})\right) \frac{\mathrm{d}\mathbf{x}(t)}{\mathrm{d}t} = \left(\nabla_{\mathbf{x}} g(\mathbf{x})\right) \mathbf{f}(\mathbf{x}) \tag{10.2}$$

where $\mathcal{K}$ is the Koopman operator. The following relationship holds as the operator is linear:

$$\mathcal{K}\left(c_1\, g_1(\mathbf{x}) + c_2\, g_2(\mathbf{x})\right) = c_1\, \mathcal{K}\left(g_1(\mathbf{x})\right) + c_2\, \mathcal{K}\left(g_2(\mathbf{x})\right) \tag{10.3}$$

with constants c_1 and c_2. The major goal is to find a new set of coordinates that results in a linear representation of the dynamics. Due to the linear property of the Koopman operator, we can perform an eigendecomposition:

$$\mathcal{K}\left(\phi(\mathbf{x})\right) = \frac{\mathrm{d}\phi(\mathbf{x})}{\mathrm{d}t} = \left(\nabla_{\mathbf{x}} \phi(\mathbf{x})\right) \frac{\mathrm{d}\mathbf{x}(t)}{\mathrm{d}t} = \lambda\, \phi(\mathbf{x}) \tag{10.4}$$

where ϕ is a right eigenfunction, and λ the corresponding eigenvalue. The evolution of the system is therefore given by a linear combination of the eigenfunctions. If each component of the vector valued function $\mathbf{g}(\mathbf{x}) \in \mathbb{R}^{n_g}$ lies in the span of the eigenfunctions, we have

$$\mathbf{g}(\mathbf{x}) = \sum_{i=1}^{\infty} \phi_i(\mathbf{x})\, \mathbf{v}_i \tag{10.5}$$

Once the eigenfunctions are known, the evolution of the observables is given by Eq. (10.4) where the observables are projected onto the span of the eigenfunctions. This operation is also referred to as the Koopman mode decomposition, and $\mathbf{v}_i$ are the Koopman modes [159]:

$$\mathbf{v}_i = \begin{bmatrix} \langle \phi_i, g_1 \rangle \\ \langle \phi_i, g_2 \rangle \\ \vdots \\ \langle \phi_i, g_{n_g} \rangle \end{bmatrix} \tag{10.6}$$

The partial differential equation (10.4) is approximated using the Galerkin method to determine the eigenfunctions, and eventually a linear representation of the system [165]. The idea is to project the Koopman operator onto a subspace that is defined by orthonormal basis functions. These projections $\langle \cdot, \cdot \rangle$ are computed using the inner product of two functions $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$:

$$\langle f_1(\mathbf{x}), f_2(\mathbf{x}) \rangle = \int_{\Omega} f_1(\mathbf{x})\, f_2(\mathbf{x})\, w(\mathbf{x})\, \mathrm{d}\mathbf{x} \tag{10.7}$$

where $w(\mathbf{x})$ is a weighting function, and Ω the domain of the integration. Defining L_i as the ith basis function and recalling Eq. (10.2), we can write

$$\frac{\mathrm{d}L_i(\mathbf{x})}{\mathrm{d}t} = \frac{\mathrm{d}L_i(\mathbf{x})}{\mathrm{d}t} = \left(\nabla_{\mathbf{x}} L_i(\mathbf{x})\right)\, \mathbf{f}(\mathbf{x}) \tag{10.8}$$

Using the inner product, the Koopman operator can be projected onto the basis functions to obtain a finite-dimensional matrix representation of $\mathcal{K}$ [165]:

$$K_{ij} = \langle \left(\nabla_{\mathbf{x}} L_i(\mathbf{x})\right)\, \mathbf{f}(\mathbf{x}), L_j(\mathbf{x}) \rangle = \int_{\Omega} \left(\nabla_{\mathbf{x}} L_i(\mathbf{x})\right)\, \mathbf{f}(\mathbf{x})\, L_j(\mathbf{x})\, w(\mathbf{x})\, \mathrm{d}\mathbf{x} \tag{10.9}$$

where K_{ij} are the elements of the Koopman matrix $\mathbf{K}$. Considering all basis functions, Eq. (10.2) can be rewritten as

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{L}(\mathbf{x}) = \mathbf{K}\, \mathbf{L}(\mathbf{x}) \tag{10.10}$$

where $\mathbf{L}(\mathbf{x}) = [L_1(\mathbf{x}), L_2(\mathbf{x}), \ldots, L_m(\mathbf{x})]$ is a vector of m basis functions. Provided that $\mathbf{K}$ is diagonalizable, its eigendecomposition reads

$$\mathbf{K}\, \mathbf{V}_r = \mathbf{V}_r\, \mathbf{D} \tag{10.11}$$

where $\mathbf{V}_r$ is the matrix that contains the right eigenvectors in its columns, and $\mathbf{D}$ is a diagonal matrix of eigenvalues. Defining $\mathbf{\Phi}(\mathbf{x}) := \mathbf{V}_r^{-1}\,\mathbf{L}(\mathbf{x})$ as the vector of eigenfunctions, substituting $\mathbf{L}(\mathbf{x}) = \mathbf{V}_r\,\mathbf{\Phi}(\mathbf{x})$ into Eq. (10.10) yields

$$\mathbf{V}_r\,\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{\Phi}(\mathbf{x}) = \mathbf{K}\,\mathbf{V}_r\,\mathbf{\Phi}(\mathbf{x}) \tag{10.12}$$

Making use of Eq. (10.11) and simplifying gives

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{\Phi}(\mathbf{x}) = \mathbf{D}\,\mathbf{\Phi}(\mathbf{x}) \tag{10.13}$$

which is the vector representation of Eq. (10.4). Its solution is given by

$$\mathbf{\Phi}(\mathbf{x}) = \mathrm{e}^{\mathbf{D}\,t}\,\mathbf{\Phi}(\mathbf{x}_0) \tag{10.14}$$

This allows us to compute the evolution of the basis functions:

$$\mathbf{L}(\mathbf{x}) = \mathbf{V}_r\,\mathbf{\Phi}(\mathbf{x}) = \mathbf{V}_r\,\mathrm{e}^{\mathbf{D}\,t}\,\mathbf{\Phi}(\mathbf{x}_0) \tag{10.15}$$

As the observables can also be projected onto the basis functions, we have the relationship

$$\mathbf{g}(\mathbf{x}) = \mathbf{P}\,\mathbf{L}(\mathbf{x}) \tag{10.16}$$

with the projection matrix $\mathbf{P}$ and its elements $P_{ij} = \langle g_i, L_j\rangle$. Therefore, the evolution of the observable functions is obtained by substituting Eq. (10.15) into Eq. (10.16):

$$\mathbf{g}(\mathbf{x}) = \mathbf{P}\,\mathbf{V}_r\,\mathrm{e}^{\mathbf{D}\,t}\,\mathbf{\Phi}(\mathbf{x}_0) = \mathbf{P}\,\mathbf{V}_r\,\mathrm{e}^{\mathbf{D}\,t}\,\mathbf{V}_r^{-1}\,\mathbf{L}(\mathbf{x}_0) \tag{10.17}$$

Often, we are interested in the identity observable, i.e., $\mathbf{g}(\mathbf{x}) \equiv \mathbf{x}$. Thus, the state $\mathbf{x}(t)$ at time t solely depends on the initial condition $\mathbf{x}_0 = \mathbf{x}(t_0)$, and can be computed analytically using Eq. (10.17) once the eigenvalues and eigenvectors are known. This, however, is only true for certain dynamical systems [165]. Moreover, if non-conservative forces (e.g., external control actions) are considered, a different approach is required as we will see in the following section.

Remark 10.1. *If $\mathbf{K}$ is not diagonalizable, we cannot perform an eigendecomposition and use the eigenvalues to solve the linear system in Eq.* (10.13). *In this case, other approaches must be used, for example the Schur decomposition [200].*

10.2 Bilinearization and Full Linearization of Control-Affine Systems

This section presents the procedure to bilinearize a control-affine system. Furthermore, a method is proposed to fully linearize the dynamics.

10.2.1 Bilinearization

We now add an external control $\mathbf{u} \in \mathbb{R}^{n_u}$ and consider control-affine systems of the following form:

$$\dot{\mathbf{x}} = \mathbf{p}(\mathbf{x}) + \sum_{i=1}^{n_u} \mathbf{b}_i(\mathbf{x})\, u_i \tag{10.18}$$

In this case, it is possible to apply the KOT and transform Eq. (10.18) into bilinear form. Let $\mathbf{T}(\mathbf{x}) = [T_1(\mathbf{x}), \dots, T_n(\mathbf{x})]^\top$ be a vector of n eigenfunctions related to the dynamics $\mathbf{p}(\mathbf{x})$, i.e., the unperturbed system. This corresponds to the procedure in the previous section where the eigendecomposition of the Koopman matrix can be performed to obtain a finite number of eigenfunctions ϕ_i, $i = 1, \dots, n$. The ith entry of $\mathbf{T}(\mathbf{x})$ is defined as follows [201]:

$$T_i(\mathbf{x}) = \phi_i(\mathbf{x}) \quad \text{if } \phi_i(\mathbf{x}) \in \mathbb{R} \tag{10.19a}$$

$$[T_i(\mathbf{x}), T_{i+1}(\mathbf{x})]^\top = [2\,\mathrm{Re}(\phi_i(\mathbf{x})), -2\,\mathrm{Im}(\phi_i(\mathbf{x}))]^\top \quad \text{if } \phi_i(\mathbf{x}) \in \mathbb{C} \tag{10.19b}$$

where we assume that $[\phi_i,\, \phi_{i+1}]$ is a complex conjugate pair if $\phi_i(\mathbf{x}) \in \mathbb{C}$. Using again the Galerkin method with basis functions $\mathbf{L}(\mathbf{x})$, the Koopman matrices $\mathbf{K}^{b_i}$ with respect to the control vector fields $\mathbf{b}_i$ can be obtained by calculating $\mathcal{K}^{b_i}(\mathbf{T}(\mathbf{x}))$:

$$K_{jk}^{b_i} = \left\langle \frac{\mathrm{d}T_j^{b_i}}{\mathrm{d}t}, L_k \right\rangle \tag{10.20}$$

The total derivatives are given by

$$\frac{\mathrm{d}T_j^{b_i}}{\mathrm{d}t} = \frac{\partial T_j^{b_i}}{\partial x_1}\, b_{i,1} + \frac{\partial T_j^{b_i}}{\partial x_2}\, b_{i,2} + \cdots = \sum_{k=1}^{n_x} \frac{\partial T_j^{b_i}}{\partial x_k}\, b_{i,k} \tag{10.21}$$

where $b_{i,k}$ denotes the kth entry of $\mathbf{b}_{i,k}$. However, we require that the Koopman operators with respect to the control vector fields lie in the span of the eigenfunctions corresponding to the unactuated system [161], i.e.,

$$\nabla_x \mathbf{T}(\mathbf{x})\, \mathbf{b}_i(\mathbf{x}) = \sum_{j=1}^{n} T_j(\mathbf{x})\, \mathbf{v}_j^{b_i} \tag{10.22}$$

where $\mathbf{v}_j^{b_i}$ refer to the Koopman modes for $\mathbf{b}_i$. If Eq. (10.22) holds true for all $i = 1, \dots, n$, Eq. (10.18) can be transformed into a bilinear system [201]:

$$\dot{\mathbf{z}} = \mathbf{D}\,\mathbf{z} + \sum_{i=1}^{n_u} \mathbf{B}_i\,\mathbf{z}\, u_i \tag{10.23}$$

$$\mathbf{x} = \mathbf{C}\,\mathbf{z} \tag{10.24}$$

with the transformed states $\mathbf{z}$, and constant matrices $\mathbf{D}$, $\mathbf{B}_i$, and $\mathbf{C}$. $\mathbf{D}$ is a block diagonal matrix with the entry $D_{i,i} = \lambda_i$ if ϕ_i is real-valued, λ_i being the eigenvalue associated with the eigenfunction ϕ_i. Otherwise,

$$\begin{bmatrix} D_{i,i} & D_{i,i+1} \\ D_{i+1,i} & D_{i+1,i+1} \end{bmatrix} = |\lambda_i| \begin{bmatrix} \cos\left(\mathrm{Arg}\left(\lambda_i\right)\right) & \sin\left(\mathrm{Arg}\left(\lambda_i\right)\right) \\ -\sin\left(\mathrm{Arg}\left(\lambda_i\right)\right) & \cos\left(\mathrm{Arg}\left(\lambda_i\right)\right) \end{bmatrix} \tag{10.25}$$

where $\mathrm{Arg}\left(\cdot\right)$ denotes the argument of a complex number. $\mathbf{B}_i$ are comprised of the Koopman modes of $\mathcal{K}^{b_i}(\mathbf{T}(\mathbf{x}))$, and $\mathbf{C}$ contains the Koopman modes of the observable:

$$\mathbf{B}_i = \mathbf{K}^{b_i}\,\mathbf{V}_r, \qquad i = 1, \ldots, n_u \tag{10.26}$$

$$\mathbf{C} = \mathbf{P}\,\mathbf{V}_r \tag{10.27}$$

Let $\mathbf{v}_i$ be the ith column of $\mathbf{C}$, i.e., $\mathbf{C} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n]$. Then [201],

$$\mathbf{v}_i = \mathbf{v}_i \quad \text{if } \phi_i(\mathbf{x}) \in \mathbb{R} \tag{10.28a}$$

$$[\mathbf{v}_i, \mathbf{v}_{i+1}] = [\mathrm{Re}(\mathbf{v}_i), \mathrm{Im}(\mathbf{v}_i)] \quad \text{if } \phi_i(\mathbf{x}) \in \mathbb{C} \tag{10.28b}$$

The matrices $\mathbf{B}_i$ are adjusted in a similar way using Eqs. (10.28a) and (10.28b). The matrix $\mathbf{P}$ denotes the projection of the observables onto the basis functions and is defined in Eq. (10.16). This system describes the evolution of the eigenfunctions $\mathbf{z} \equiv \mathbf{T}(\mathbf{x})$ instead of the original states $\mathbf{x}$. These can be retrieved using the linear relationship in Eq. (10.24). Note, however, that Eq. (10.18) is often only an approximation because of the truncated decomposition. The new number of states n_z (and hence, all combinations of the polynomials that define the basis) of the transformed system depends on the number of states, n_x, and order of the basis functions, q:

$$n_z = \frac{(q + n_x)!}{q!\,n_x!} \tag{10.29}$$

For example, if there are $n_x = 3$ states and 4th order basis functions are considered, the number of dimensions increases to $n_z = 35$.

Figure 10.1 presents the procedure to transform a given system into bilinear form using the KOT. Even though the first step seems trivial, we will see in the following subsections that selecting an appropriate state vector representation for the Koopman formalism is in fact crucial. Although the projection of the Koopman operator onto a finite subspace allows us to calculate the eigenfunctions (and thus, the evolution of the states), it is important to understand that this is only an approximation. We can only use a limited number of basis functions and eigenfunctions, and therefore, the approximation is more accurate the more linear the original system is. As we found that the choice of the basis functions is not critical, we use orthonormal Legendre polynomials throughout this chapter. The reason is that the weighting function in Eq. (10.9) is $w(\mathbf{x}) = 1$, i.e., constant. This eases the computation of the integrals. Once the finite-dimensional Koopman matrix is obtained, its eigenfunctions, eigenvalues, and eigenvectors can be determined. The Koopman modes are found using Eq. (10.6), and the Koopman canonical transform in Eq. (10.19) is used to transform the original coordinates $\mathbf{x}$ into a higher-dimensional space spanned by the eigenfunctions. The final step is a Koopman mode decomposition with respect to the control term to obtain a bilinear system.

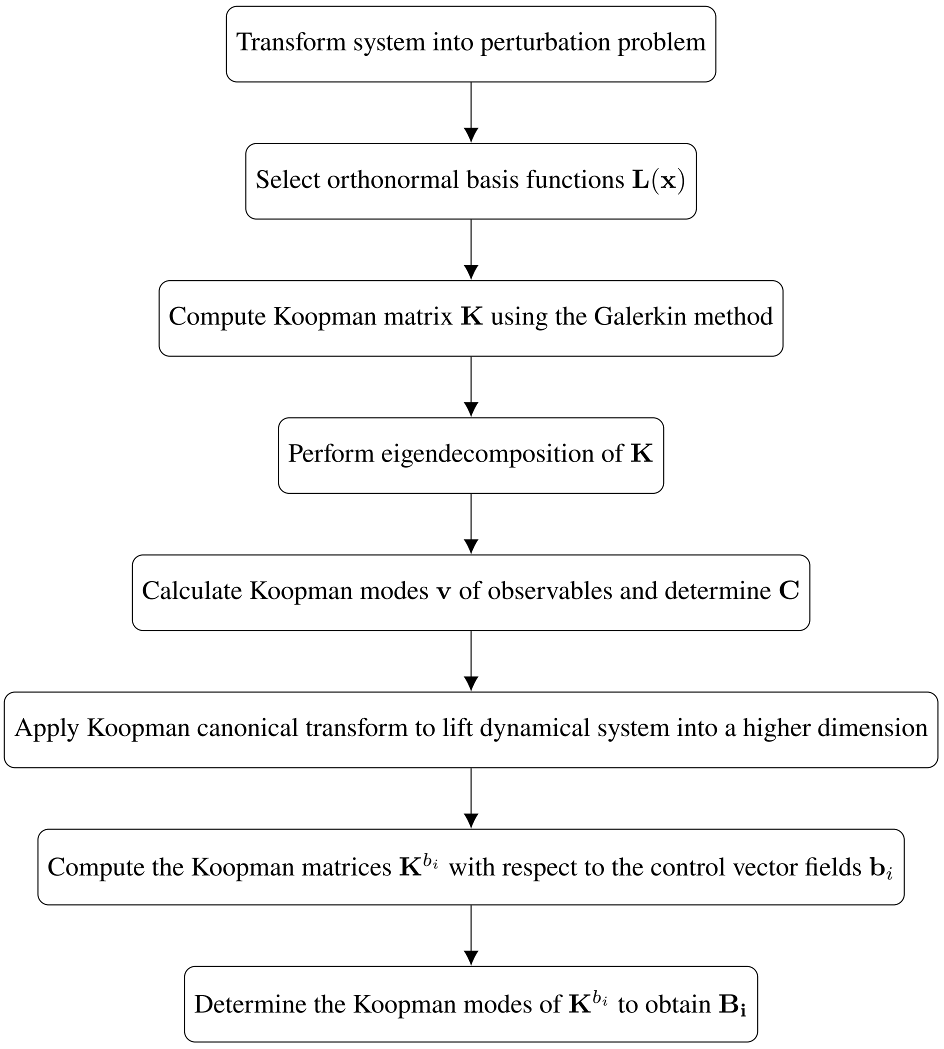

Transform system into perturbation problem
Select orthonormal basis functions $\mathbf{L}(\mathbf{x})$
Compute Koopman matrix $\mathbf{K}$ using the Galerkin method
Perform eigendecomposition of $\mathbf{K}$
Calculate Koopman modes $\mathbf{v}$ of observables and determine $\mathbf{C}$
Apply Koopman canonical transform to lift dynamical system into a higher dimension
Compute the Koopman matrices $\mathbf{K}^{b_i}$ with respect to the control vector fields $\mathbf{b}_i$
Determine the Koopman modes of $\mathbf{K}^{b_i}$ to obtain $\mathbf{B_i}$

Figure 10.1: Procedure to transform a dynamical system into bilinear form.

10.2.2 Full Linearization

Although we expect that a bilinear form can be handled more easily than a highly nonlinear system, a completely linear representation of the dynamics is, naturally, preferable. If the left-hand side of Eq. (10.22) is constant, i.e.,

$$\nabla_x \mathbf{T}(\mathbf{x})\, \mathbf{b}_i(\mathbf{x}) = \text{const.} =: \mathbf{B} \qquad \forall i = 1, \ldots, n \tag{10.30}$$

the dynamics in Eq. (10.23) reduce to linear form [161]:

$$\dot{\mathbf{z}} = \mathbf{D}\,\mathbf{z} + \mathbf{B}\,\mathbf{u} \tag{10.31}$$

Yet, this condition is in general not satisfied if a limited number of eigenfunctions is used. In this dissertation, we apply a procedure called extended dynamic mode decomposition (EDMD) to obtain an approximate linear model. In literature, this approach is often used to determine a finite-dimensional representation of the Koopman operator [159, 202]. Let $\mathbf{F}$ denote the map that solves the discrete-time equivalent of the system $\dot{\mathbf{x}}$ in Eq. (10.18) such that

$$\mathbf{x}^{+} = \mathbf{F}(\mathbf{x}, \mathbf{u}) \tag{10.32}$$

where $\mathbf{x}^{+}$ denotes the resulting state at the next time step. We are then interested in the corresponding linear form expressed in the transformed coordinates $\mathbf{z}$:

$$\mathbf{z}^{+} = \mathbf{D}\,\mathbf{z} + \mathbf{B}\,\mathbf{u} \tag{10.33}$$

Given a set of M data points $\left[\mathbf{x}_j^{+}, \mathbf{x}_j, \mathbf{u}_j\right]$, $j = 1, \ldots, M$, that satisfy Eq. (10.32), the basic idea of EDMD is to solve the following optimization problem [202]:

$$\min_{\mathbf{D},\mathbf{B}} \; \sum_{j=1}^{M} \left\| \mathbf{T}(\mathbf{x}_j^{+}) - \mathbf{D}\,\mathbf{T}(\mathbf{x}_j) - \mathbf{B}\,\mathbf{u} \right\|_2^2 \tag{10.34}$$

Collecting all data in matrices, Eq. (10.34) can be rewritten to obtain [202]

$$\min_{\mathbf{D},\mathbf{B}} \; \left\| \mathbf{X}_{\text{lift}}^{+} - \mathbf{D}\,\mathbf{X}_{\text{lift}} - \mathbf{B}\,\mathbf{U} \right\|_F \tag{10.35}$$

where $\|\cdot\|_F$ refers to the Frobenius norm of a matrix, and

$$\mathbf{X}^{+} = \left[\mathbf{x}_1^{+}, \ldots, \mathbf{x}_M^{+}\right], \; \mathbf{X} = \left[\mathbf{x}_1, \ldots, \mathbf{x}_M\right], \; \mathbf{U} = \left[\mathbf{u}_1, \ldots, \mathbf{u}_M\right] \tag{10.36}$$

$$\mathbf{X}_{\text{lift}}^{+} = \left[\mathbf{T}(\mathbf{x}_1^{+}), \ldots, \mathbf{T}(\mathbf{x}_M^{+})\right], \; \mathbf{X}_{\text{lift}} = \left[\mathbf{T}(\mathbf{x}_1), \ldots, \mathbf{T}(\mathbf{x}_M)\right] \tag{10.37}$$

Therefore, the obtained solution can be considered the best linear fit over one step. Equation (10.35) can be solved analytically using the Moore–Penrose pseudoinverse $(\cdot)^{\dagger}$ to obtain $\mathbf{D}$ and $\mathbf{B}$ [202]:

$$[\mathbf{D}, \mathbf{B}] = \mathbf{X}_{\text{lift}}^{+} \, [\mathbf{X}_{\text{lift}}, \mathbf{U}]^{\dagger} \tag{10.38}$$

As we already computed the matrix $\mathbf{D}$ with the Galerkin method, we pursue a slightly different approach. In particular, we only solve for $\mathbf{B}$, and use the matrix $\mathbf{D}$ that we obtained with the eigendecomposition. Thus, Eq. (10.38) changes to

$$\mathbf{B} = \left(\mathbf{X}_{\text{lift}}^{+} - \mathbf{D}\,\mathbf{X}_{\text{lift}}\right) \, \mathbf{U}^{\dagger} \tag{10.39}$$

As EDMD minimizes the prediction error over one time step only, the error will accumulate over time. Even though more accurate solutions may be obtained if a larger training set is used, caution is advised if the linear form is intended for long time prediction.

10.3 Linearization Accuracy Index

Even though we mentioned that it is expected that a bilinear form may be easier to handle, the dynamics are still nonconvex. As a first-order Taylor series is the standard method to approximate nonlinear dynamics, we now want to assess whether linearizing the bilinear form is beneficial compared to linearizing the original system directly. For this reason, we determine the linearization accuracy index defined in Eq. (7.158) for a given set of perturbed reference trajectories. Our findings from Section 7.3 suggest that this metric can be a good indicator of how accurate a linearized system is. We refer the reader to Chapter 7 for details, and present the results for the Duffing oscillator and the space-flight mechanics problem in Section 10.5.

10.4 Applications

The transformations into bilinear and linear systems is applied to two examples: the Duffing oscillator and the perturbed Kepler problem. We provide a step-by-step example for the Duffing oscillator, and describe the characteristics and modifications that are needed to apply the Koopman formalism to problems in orbital mechanics.

10.4.1 Duffing Oscillator

We consider the following system of differential equations for the Duffing oscillator:

$$\dot{x}_1 = x_2 \tag{10.40a}$$

$$\dot{x}_2 = -x_1 - \varepsilon\, x_1^3 + u \tag{10.40b}$$

where x_1, x_2 are the states, ε is a small parameter, and u an external control. This system is fully polynomial, and consists of a leading linear term $-x_1$ and a small perturbation $-\varepsilon\, x_1^3$ in Eq. (10.40b). No further modifications are necessary to compute the Koopman matrix because the system is already completely polynomial.

In the following, we provide a step-by-step example of how all relevant quantities for the bilinearization are computed to make the reader familiar with the Koopman formalism.

Step 1: Selection of Basis Functions

According to Fig. 10.1, orthonormal basis functions $\mathbf{L}(\mathbf{x})$ are to be selected where $\mathbf{x} := [x_1, x_2]^\top$. For demonstration purposes, we limit ourselves to 2nd order basis functions. The first three Legendre polynomials are given by

$$P_1(x) = 1 \tag{10.41a}$$

$$P_2(x) = x \tag{10.41b}$$

$$P_3(x) = \frac{1}{2}\,\left(3\,x^2 - 1\right) \tag{10.41c}$$

As these are orthogonal but not orthonormal, they are normalized such that

$$\langle P_i(x), P_j(x)\rangle = \int_{-1}^{1} P_i(x)\,P_j(x)\,w(x)\,\mathrm{d}x = \delta_{ij}, \qquad i = 1,2,3,\ j = 1,2,3 \tag{10.42}$$

where δ_{ij} denotes the Kronecker delta, and $w(x) = 1$. The normalization factor α_i for the ith Legendre plynomial is therefore given by

$$\alpha_i = \sqrt{\frac{2\,(i-1)+1}{2}}, \qquad i = 1,2,3 \tag{10.43}$$

and the normalized Legendre polynomials $\tilde{P}_i(x) = \alpha_i\,P_i(x)$ read

$$\tilde{P}_1(x) = \frac{1}{\sqrt{2}} \tag{10.44a}$$

$$\tilde{P}_2(x) = \sqrt{\frac{3}{2}}\,x \tag{10.44b}$$

$$\tilde{P}_3(x) = \sqrt{\frac{5}{2}}\,\frac{1}{2}\,\left(3\,x^2 - 1\right) \tag{10.44c}$$

Making use of Eq. (10.29), there are $n_z = 6$ combinations of the orthonormal Legendre polynomials that yield the basis functions $L_i(\mathbf{x})$, $i = 1, \ldots, 6$, of maximum degree two:

$$L_1(\mathbf{x}) = \tilde{P}_1(x_1)\,\tilde{P}_1(x_2) = \frac{1}{2} \tag{10.45a}$$

$$L_2(\mathbf{x}) = \tilde{P}_2(x_1)\,\tilde{P}_1(x_2) = \frac{\sqrt{3}}{2}\,x_1 \tag{10.45b}$$

$$L_3(\mathbf{x}) = \tilde{P}_1(x_1)\,\tilde{P}_2(x_2) = \frac{\sqrt{3}}{2}\,x_2 \tag{10.45c}$$

$$L_4(\mathbf{x}) = \tilde{P}_3(x_1)\,\tilde{P}_1(x_2) = \frac{\sqrt{5}}{4}\,\left(3\,x_1^2 - 1\right) \tag{10.45d}$$

$$L_5(\mathbf{x}) = \tilde{P}_2(x_1)\,\tilde{P}_2(x_2) = \frac{3}{2}\,x_1\,x_2 \tag{10.45e}$$

$$L_6(\mathbf{x}) = \tilde{P}_1(x_1)\,\tilde{P}_3(x_2) = \frac{\sqrt{5}}{4}\,\left(3\,x_2^2 - 1\right) \tag{10.45f}$$

Step 2: Computation of Koopman Matrix and Eigendecomposition

Recalling that the entries K_{ij} of the Koopman matrix associated with the uncontrolled problem (i.e., $u = 0$) is given by Eq. (10.9), we need to compute the derivatives of $L_i(\mathbf{x})$ with respect to time:

$$\frac{\mathrm{d}L_i(\mathbf{x})}{\mathrm{d}t} = \nabla_{\mathbf{x}} L_i(\mathbf{x})\,\mathbf{f}(\mathbf{x}), \qquad i = 1,\dots,6 \tag{10.46}$$

Using $\mathbf{f}(\mathbf{x}) = [\dot{x}_1, \dot{x}_2]^\top$ and Eq. (10.40), this results in

$$\frac{\mathrm{d}L_1(\mathbf{x})}{\mathrm{d}t} = 0 \tag{10.47a}$$
$$\frac{\mathrm{d}L_2(\mathbf{x})}{\mathrm{d}t} = \frac{\sqrt{3}}{2}\,x_2 \tag{10.47b}$$
$$\frac{\mathrm{d}L_3(\mathbf{x})}{\mathrm{d}t} = -\frac{\sqrt{3}}{2}\,\left(x_1 + \varepsilon\,x_1^3\right) \tag{10.47c}$$
$$\frac{\mathrm{d}L_4(\mathbf{x})}{\mathrm{d}t} = \frac{3\sqrt{5}}{2}\,x_1\,x_2 \tag{10.47d}$$
$$\frac{\mathrm{d}L_5(\mathbf{x})}{\mathrm{d}t} = -\frac{3}{2}\,\left(x_1^2 - x_2^2 + \varepsilon\,x_1^4\right) \tag{10.47e}$$
$$\frac{\mathrm{d}L_6(\mathbf{x})}{\mathrm{d}t} = -\frac{3\sqrt{5}}{2}\,x_2\,\left(x_1 + \varepsilon\,x_1^3\right) \tag{10.47f}$$

As there are two states, each element K_{ij} is then found by calculating the following double integral:

$$K_{ij} = \int_{-1}^{1}\int_{-1}^{1} \frac{\mathrm{d}L_i(\mathbf{x})}{\mathrm{d}t}\,L_j(\mathbf{x})\,w(\mathbf{x})\,\mathrm{d}x_1\,\mathrm{d}x_2, \qquad i = 1,\dots,6,\ j = 1,\dots,6 \tag{10.48}$$

where $w(\mathbf{x}) = 1$, and the integration domain is $[-1, 1]$ for Legendre polynomials. The only nonzero elements of $\mathbf{K}$ are $K_{23} = 1$, $K_{32} = -1 - 3\,\varepsilon/5$, $K_{45} = \sqrt{5}$, $K_{51} = -3\,\varepsilon/5$, $K_{54} = -2\,(7+6\,\varepsilon)/(7\sqrt{5})$, $K_{56} = 2\sqrt{5}/5$, and $K_{65} = -(5 + 3\,\varepsilon)/\sqrt{5}$.

Performing an eigendecomposition of $\mathbf{K}$ yields the vector of eigenvalues $\boldsymbol{\lambda}$:

$$\boldsymbol{\lambda} = \left[0,\ 0,\ -\sqrt{-1 - \frac{3}{5}\,\varepsilon},\ \sqrt{-1 - \frac{3}{5}\,\varepsilon},\ -\sqrt{-4 - \frac{102}{35}\,\varepsilon},\ \sqrt{-4 - \frac{102}{35}\,\varepsilon}\right]^\top \tag{10.49}$$

The matrix of right eigenvectors $\mathbf{V}_r$ reads

$$\mathbf{V}_r = \begin{bmatrix} -\frac{2\sqrt{5}\,(7+6\,\varepsilon)}{21\,\varepsilon} & \frac{2\sqrt{5}}{3\,\varepsilon} & 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{\sqrt{5}}{\sqrt{-3\,\varepsilon-5}} & \frac{\sqrt{5}}{\sqrt{-3\,\varepsilon-5}} & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & -\frac{5}{3\,\varepsilon+5} & -\frac{5}{3\,\varepsilon+5} \\ 0 & 0 & 0 & 0 & \sqrt{\frac{2}{7}}\,\frac{\sqrt{-51\,\varepsilon-70}}{3\,\varepsilon+5} & -\sqrt{\frac{2}{7}}\,\frac{\sqrt{-51\,\varepsilon-70}}{3\,\varepsilon+5} \\ 0 & 1 & 0 & 0 & 1 & 1 \end{bmatrix} \tag{10.50}$$

and the vector of eigenfunctions $\mathbf{\Phi}(\mathbf{x})$ is then

$$\mathbf{\Phi}(\mathbf{x}) = \mathbf{V}_r^{-1}\,\mathbf{L}(\mathbf{x}) = \begin{bmatrix} \frac{\sqrt{5}\,(63\,\varepsilon\,x_1^2-42\,\varepsilon+105\,x_1^2+105\,x_2^2-70)}{204\,\varepsilon+280} \\ \frac{21\,\sqrt{5}\,\varepsilon\,(3\,\varepsilon+5)}{1020\,\varepsilon+1400} + \frac{5\,\sqrt{5}(6\,\varepsilon+7)\,(3\,x_2^2-1)}{204\,\varepsilon+280} + \frac{\sqrt{5}\,(3\,\varepsilon+5)\,(6\,\varepsilon+7)\,(3\,x_1^2-1)}{204\,\varepsilon+280} \\ \frac{\sqrt{3}\,x_2}{4} - \frac{\sqrt{15}\,x_1\sqrt{-3\,\varepsilon-5}}{20} \\ \frac{\sqrt{3}\,x_2}{4} + \frac{\sqrt{15}\,x_1\sqrt{-3\,\varepsilon-5}}{20} \\ \frac{7\,\sqrt{5}\,(3\,\varepsilon+5)\,(3\,x_2^2-1)}{408\,\varepsilon+560} - \frac{21\,\sqrt{5}\,\varepsilon\,(3\,\varepsilon+5)}{2040\,\varepsilon+2800} - \frac{\sqrt{5}\,(3\,\varepsilon+5)\,(6\,\varepsilon+7)\,(3\,x_1^2-1)}{408\,\varepsilon+560} + \frac{3\,\sqrt{14}\,x_1\,x_2\,(3\,\varepsilon+5)}{8\,\sqrt{-51\,\varepsilon-70}} \\ \frac{7\,\sqrt{5}\,(3\,\varepsilon+5)\,(3\,x_2^2-1)}{408\,\varepsilon+560} - \frac{21\,\sqrt{5}\,\varepsilon\,(3\,\varepsilon+5)}{2040\,\varepsilon+2800} - \frac{\sqrt{5}\,(3\,\varepsilon+5)\,(6\,\varepsilon+7)\,(3\,x_1^2-1)}{408\,\varepsilon+560} - \frac{3\,\sqrt{14}\,x_1\,x_2\,(3\,\varepsilon+5)}{8\,\sqrt{-51\,\varepsilon-70}} \end{bmatrix} \tag{10.51}$$

Step 3: Computation of Koopman Modes and Koopman Canonical Transform

We require the elements P_{ij} of the projection matrix $\mathbf{P}$. It is determined using Eq. (10.16) with $\mathbf{g}(\mathbf{x}) \equiv \mathbf{x}$:

$$P_{ij} = \int_{-1}^{1}\int_{-1}^{1} x_i\,L_j(\mathbf{x})\,w(\mathbf{x})\,\mathrm{d}x_1\,\mathrm{d}x_2, \qquad i = 1, 2,\ j = 1, \ldots, 6 \tag{10.52}$$

The only nonzero elements are $P_{12} = P_{23} = 2\sqrt{3}/3$. The matrix $\mathbf{C}$ that contains the Koopman modes is then computed using Eq. (10.27):

$$\mathbf{C} = \mathbf{P}\,\mathbf{V}_r = \begin{bmatrix} 0 & 0 & -\frac{2\,\sqrt{15}}{3\,\sqrt{-3\,\varepsilon-5}} & \frac{2\,\sqrt{15}}{3\,\sqrt{-3\,\varepsilon-5}} & 0 & 0 \\ 0 & 0 & \frac{2\,\sqrt{3}}{3} & \frac{2\,\sqrt{3}}{3} & 0 & 0 \end{bmatrix} \tag{10.53}$$

Recall from Eq. (10.28) that the columns of $\mathbf{C}$ change if the corresponding eigenvalues are complex. Assuming that $\varepsilon > 0$, we observe that all eigenvalues except the first two are complex (see also Eq. (10.49)). Therefore, $\mathbf{C}$ is to be adjusted accordingly, and we obtain

$$\mathbf{C} = \begin{bmatrix} 0 & 0 & 0 & \mathrm{Im}\left(-\frac{2\sqrt{15}}{3\sqrt{-3\varepsilon-5}}\right) & 0 & 0 \\ 0 & 0 & \frac{2\sqrt{3}}{3} & 0 & 0 & 0 \end{bmatrix} \tag{10.54}$$

The Koopman canonical transform is given by the transformation $\mathbf{T}(\mathbf{x})$ in Eq. (10.19):

$$\mathbf{T}(\mathbf{x}) = [\phi_1(\mathbf{x}),\ \phi_2(\mathbf{x}),\ 2\,\mathrm{Re}(\phi_3(\mathbf{x})),\ -2\,\mathrm{Im}(\phi_3(\mathbf{x})),\ 2\,\mathrm{Re}(\phi_5(\mathbf{x})),\ -2\,\mathrm{Im}(\phi_5(\mathbf{x}))]^\top \tag{10.55}$$

Step 4: Computation of Koopman Matrix With Respect to Control

As the control u is scalar, there is only one control vector field $\mathbf{b} = [0, 1]^\top$ in Eq. (10.18). The Koopman matrix $\mathbf{K}^b$ with respect to the control is then obtained using Eq. (10.20):

$$K^b_{jk} = \int_{-1}^{1}\int_{-1}^{1} \nabla_x\,T_j(\mathbf{x})\,\mathbf{b}\,L_k(\mathbf{x})\,w(\mathbf{x})\,\mathrm{d}x_1\,\mathrm{d}x_2, \qquad j = 1, \ldots, 6,\ k = 1, \ldots, 6 \tag{10.56}$$

Assuming $\varepsilon = 10^{-3}$ so that we can properly calculate the double integrals, the nonzero elements of $\mathbf{K}^b$ are $K^b_{13} = 1.9351$, $K^b_{23} = 1.9367$, $K^b_{31} = 1.7321$, $K^b_{53} = 1.9362$, and $K^b_{62} = 1.9369$. Note that the entries are rounded to four significant digits due to the lengthy expressions.

The next step is to determine $\mathbf{B}$ that contains the Koopman modes of $\mathbf{K}^b$. According to Eq. (10.26), we need to multiply $\mathbf{K}^b$ with $\mathbf{V}_r$ of the unactuated system and update the columns of the obtained matrix using Eqs. (10.28a) and (10.28b). The nonzero elements are given by $B^b_{13} = 1.9351$, $B^b_{23} = 1.9367$, $B^b_{31} = -2.5842 \times 10^3$, $B^b_{32} = 2.5820 \times 10^3$, $B^b_{53} = 1.9362$, and $B^b_{64} = 1.9364$.

Finally, the only remaining quantity is $\mathbf{D}$ that is computed using Eq. (10.25). For $\varepsilon = 10^{-3}$, the nonzero elements read $D_{34} = -1.0003$, $D_{43} = 1.0003$, $D_{56} = -2.0007$, and $D_{65} = 2.0007$.

The bilinear system is then given by Eqs. (10.23) and (10.24), and the transformation $\mathbf{z} \equiv \mathbf{T}(\mathbf{x})$ lifts the original states $\mathbf{x} = [x_1, x_2]^\top$ into a six-dimensional space with states $\mathbf{z} = [z_1, \dots, z_6]^\top$.

10.4.2 Perturbed Keplerian Motion

With regard to orbital motion, we make use of MOE and KS coordinates of Chapter 7 because they result in linear unperturbed dynamics. As the control actions are small due to the low-thrust propulsion, the problem can often be considered a perturbation problem with a leading linear term. We present the required modifications so that both coordinate sets can be applied within the KOT. For demonstration purposes, we consider the planar case only. It is straightforward to extend the approach to three-dimensional problems.

Kustaanheimo–Stiefel Coordinates

Recalling the equations of motion in Eq. (7.117), it is apparent that they are polynomial except for the term $1/h$. In order to be able to compute the derivatives easily in an automated way, we define a new element $\tilde{h}$ as the inverse of the negative specific energy h:

$$\tilde{h} := \frac{1}{h} \tag{10.57}$$

The derivative of $\tilde{h}$ with respect to the independent variable E is then given by

$$\frac{\mathrm{d}\tilde{h}}{\mathrm{d}E} = -\tilde{h}^2 \frac{\mathrm{d}h}{\mathrm{d}E} \tag{10.58}$$

Therefore, the new differential equations are obtained by replacing $1/h$ with $\tilde{h}$ in Eq. (7.117). As a consequence, the differential equations become completely polynomial, which significantly eases the computation of derivatives and integrals.

Modified Orbital Elements

To the best of our knowledge, the Galerkin method has only been applied to problems with fully polynomial dynamics within the KOT in astrodynamics [164, 165]. In this section, we demonstrate that also non-polynomial representations can be used, although some modifications and more computational effort are required. According to Eq. (10.9), we need to project the Koopman operator onto the basis functions to obtain a finite-dimensional matrix. This is done by computing the inner product. Regardless of the chosen basis functions, it may happen that the integrals are not defined if the dynamics are not polynomial. For example, if we consider the equation of motion (7.62a) for the element Λ,

$$\frac{\mathrm{d}\Lambda}{\mathrm{d}\zeta} = -\eta + \frac{2\,\kappa + \Lambda}{\kappa\,(\kappa + \Lambda)^3}\,a_s \tag{10.59}$$

we notice the singularities at $\kappa = 0$ and $\kappa + \Lambda = 0$. Even though they are physically not relevant (e.g., κ as the inverse of the angular momentum is never zero), this causes the integrals over the interval $[-1, 1]$ to be undefined. Therefore, we propose a linear transformation and shift the values of each element such that the singularities are avoided. The standard linear mapping from the interval $[a, b]$ to $[c, d]$ is given by

$$f(x) = c + \frac{d - c}{b - a}\,(x - a) \tag{10.60}$$

As the Legendre polynomials are defined on $[-1, 1]$, this transformation is obtained by setting $a = x_{\text{min}}$, $a = x_{\text{max}}$, $c = -1$, and $d = 1$:

$$\tilde{x} = \frac{2\,x - x_{\text{max}} - x_{\text{min}}}{x_{\text{max}} - x_{\text{min}}} \tag{10.61}$$

x and $\tilde{x}$ are the original and transformed variables, respectively. x_{min} and x_{max} define the minimum and maximum values that element x can take. We found in our simulations that the selection of these values is not critical as long as the singularities can be avoided. For example, the values in Table 10.1 cover all transfers considered in this chapter, and allow us to compute the integrals. Although we restrict the domain of the integration this way, our simulations confirm that the accuracy does not deteriorate compared to integrating over the complete interval $[-1, 1]$. The reason is that the orbital elements vary within the range defined by x_{min} and x_{max}. The inverse transformation is

$$x = \frac{1}{2}\,\left[\tilde{x}\,\left(x_{\text{max}} - x_{\text{min}}\right) + x_{\text{max}} + x_{\text{min}}\right] \tag{10.62}$$

Taking the derivative with respect to ζ yields

$$\frac{\mathrm{d}\tilde{x}}{\mathrm{d}\zeta} = \frac{2}{x_{\text{max}} - x_{\text{min}}}\,\frac{\mathrm{d}x}{\mathrm{d}\zeta} \tag{10.63}$$

As an example, the differential equation

$$\frac{\mathrm{d}\Lambda}{\mathrm{d}\zeta} = -\eta \tag{10.64}$$

Table 10.1: Values for the linear transformation for MOE to avoid the singularity within KOT.

Element	Minimum value	Maximum value
Λ	-0.1	0.1
η	-0.1	0.1
γ	-0.7	0.7
s	-0.7	0.7
κ	0.7	0.99
β	-0.9	0.9

is transformed into

$$\begin{aligned}\frac{\mathrm{d}\tilde{\Lambda}}{\mathrm{d}\zeta} &= \frac{2}{\Lambda_{\max} - \Lambda_{\min}} \frac{\mathrm{d}\Lambda}{\mathrm{d}\zeta} = \frac{2}{\Lambda_{\max} - \Lambda_{\min}} (-\eta) \\ &= -\frac{\eta_{\max} - \eta_{\min}}{\Lambda_{\max} - \Lambda_{\min}} \tilde{\eta} - \frac{\eta_{\max} - \eta_{\min}}{\Lambda_{\max} - \Lambda_{\min}}\end{aligned} \tag{10.65}$$

The other equations and the control terms are mapped in a similar way, but omitted here due to the lengthy expressions.

For demonstration purposes, we considered the formulation of MOE that results in linear unperturbed dynamics so far. Yet, the same procedure applies to other problems where the Koopman operator can be applied, i.e., perturbation problems with a leading linear term and a small perturbation. Although we have already presented the Duffing oscillator that belongs to this class of problems, we briefly provide an example relevant for orbital motion. As shown in [165], the singularity of MOE can be eliminated by introducing a different time regularization defined as follows:

$$\frac{\mathrm{d}}{\mathrm{d}t} = \frac{p_h}{r^2 \cos^2\phi} \frac{\mathrm{d}}{\mathrm{d}\tilde{\zeta}} \tag{10.66}$$

where $\tilde{\zeta}$ is the new independent variable. Note the additional factor $\cos^2\phi$ in the denominator compared to the original regularization in Eq. (7.29). The relationship between ζ and $\tilde{\zeta}$ is then given by

$$\frac{\mathrm{d}}{\mathrm{d}\zeta} = \frac{1}{\cos^2\phi} \frac{\mathrm{d}}{\mathrm{d}\tilde{\zeta}} \tag{10.67}$$

This slightly changes the dynamics of the system. For example, the new differential equation for element Λ is

$$\frac{\mathrm{d}\Lambda}{\mathrm{d}\tilde{\zeta}} = -\eta + \eta\, s^2 + \frac{(2\,\kappa + \Lambda)\,\left(1 - s^2\right)}{\kappa\,(\kappa + \Lambda)^3}\, a_s \tag{10.68}$$

where we used $\cos^2\phi = 1 - s^2$ and omitted the normalization factor. The dynamics of the other elements are obtained similarly by multiplying the right-hand side with $1 - s^2$. Obviously, the unperturbed dynamics are not linear anymore, but become nonlinear due to the term $\eta\, s^2$. However, this product is small compared to the linear part $-\eta$, and therefore, the system can still be transformed accurately into bilinear form. The only difference is that higher-order basis functions are needed to achieve the same accuracy as in the completely linear case. The reason is that the total order of the system increases by two because of the factor s^2. Consequently, the degree of the basis functions should at least increase by two to capture the additional nonlinearity accurately. Naturally, other small perturbations can be included in the same way, for example, the J_2 term due to the oblateness of the Earth [165].

Remark 10.2. *In case of out-of-plane motion, we recall the differential equation* (7.62c) *for the element* γ*. Even though the integral of* $\sqrt{1 - s^2 - \gamma^2}$ *could be computed analytically using spherical coordinates, the coupling with the other terms makes it difficult to calculate it numerically in an automated way. Therefore, it is possible to define a new element:*

$$c := \cos i = \sqrt{1 - s^2 - \gamma^2} \tag{10.69}$$

where i *is the inclination of the orbit. The derivative with respect to the independent variable* ζ *is then*

$$\frac{\mathrm{d}c}{\mathrm{d}\zeta} = -\frac{\gamma}{\kappa\,(\kappa + \Lambda)^3}\,a_n \tag{10.70}$$

This way, all multi-dimensional integrals can be computed numerically.

The complete procedure to determine the bilinear representation is automated. To ensure a high flexibility, all calculations are performed symbolically due to the complex structure of the equations for MOE. The inner products (and hence, multi-dimensional integrals) in Eqs. (10.9) and (10.20) are computed using numerical quadrature on sparse grids [203].

10.5 Numerical Simulations

We perform numerical simulations to assess the accuracy of the bilinear and completely linear system when propagating the dynamics. In addition, the metric Ξ^x is determined to obtain an indication of the linearization performance. Scaling parameters for the Kepler problem are given in Table 10.2, and other relevant data in Tables 10.3 and 10.4. With regard to the orbital motion problem, we consider the data of the transfer from the Sun-Earth Lagrange point L_2 to asteroid 2000 SG344 [129]. Table 10.1 provides the parameters for the linear transformation for MOE. If not stated otherwise, all variables are scaled such that they stay within (or close) to $[-1, 1]$. Therefore, quantities without units refer to the normalized values throughout this section.

10.5.1 Bilinear and Linear System

We propagate the dynamics of the original, bilinear, and linear system and compare the results. With regard to the transformed systems, the orders $q = 3$ to $q = 7$ are considered for the basis functions.

Table 10.2: Physical constants for the Kepler problem.

Parameter	Value
Gravitational const. μ	$1.327\,124\,4 \times 10^{11}\,\mathrm{km^3\,s^{-2}}$
Length unit LU = AU	$1.495\,978\,707 \times 10^{8}\,\mathrm{km}$
Velocity unit VU	$\sqrt{\mu\,\mathrm{AU}^{-1}}$

Table 10.3: Simulation values for the Duffing oscillator.

Parameter	Value
Initial state x_1, x_2	1.0, 0.0
Small parameter ε	10^{-3}
Propagation period	15

Table 10.4: Simulation values for the orbital motion problem [129].

Parameter	Value
Initial position $\mathbf{r}_0$, AU	$[-0.70186065, 0.70623244, -3.51115 \times 10^{-5}]^\top$
Initial velocity $\mathbf{v}_0$, VU	$[-0.73296949, -0.71590485, 4.40245 \times 10^{-5}]^\top$
Final position $\mathbf{r}_f$, AU	$[0.41806795, 0.82897114, -0.00143382]^\top$
Final velocity $\mathbf{v}_f$, VU	$[-0.96990332, 0.43630220, -0.00123381]^\top$
Propagation period, days	700

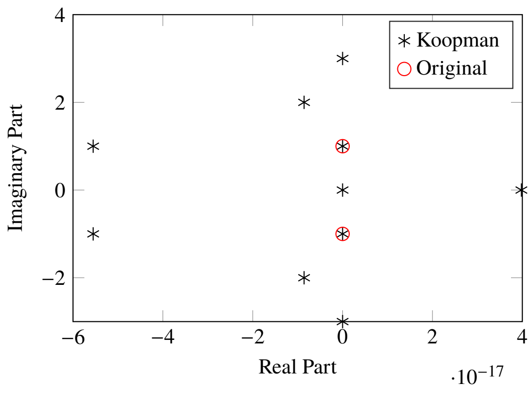
The resulting number of states n_z (or equivalently, number of eigenfunctions) for each order is given in Table 10.5. For higher polynomial degrees, the number of dimensions becomes relatively large, and consequently, the computational effort increases. To demonstrate that the KOT is able to determine the eigenvalues accurately, plots of the real and imaginary parts of the eigenvalues for $q = 3$ (Duffing) and $q = 7$ (Kepler) are shown in Fig. 10.2. Note that the eigenvalues refer to the linear part of the unperturbed dynamics, and the red circles are the analytically determined values. Apparently, the real parts are essentially zero, and the KOT finds the same eigenvalues. As expected, the total number of eigenvalues increases if more basis functions are taken into account.

Propagation Accuracy of the Bilinear System

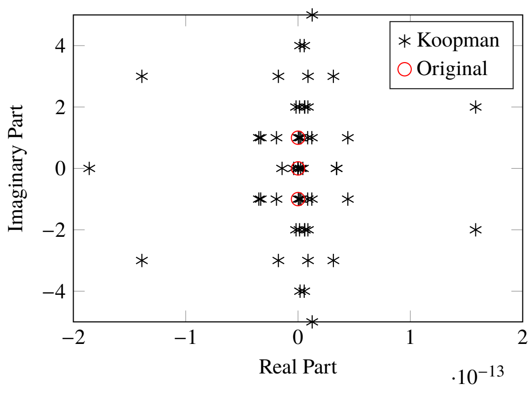
The original and the bilinear system are propagated using the control profiles shown in Fig. 10.3. The evolution of the states is shown in Figs. 10.4 and 10.5, and we observe that all curves seem to match.

Table 10.5: Number of states in the transformed system for different orders of the basis functions.

Problem	3rd order	4th order	5th order	6th order	7th order
Duffing ($n_x = 2$)	10	15	21	28	36
Kepler ($n_x = 3$)	20	35	56	84	120



(a) Duffing oscillator and 3rd order basis functions.



(b) Kepler problem and 5th order basis functions.

Figure 10.2: Eigenvalues of the unperturbed Duffing and Kepler problem.

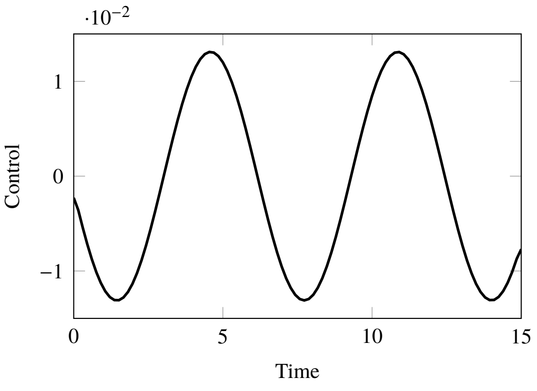
With regard to the results of the Duffing oscillator in Fig. 10.6, the obtained maximum errors of approximately 10^{-6} are small regardless of the order of the basis functions. Yet, the higher degrees 5 – 7 can achieve a two to three orders of magnitude better accuracy compared to $q = 3$ and $q = 4$. The negative peaks occur when x_2, and hence, the nonlinear term, becomes small.

For the Kepler problem, the accuracy improves by approximately one order of magnitude between q and $q+1$. Moreover, $q = 3$ performs worse compared to the Duffing oscillator. The reason is that 4th order terms occur in the denominator of the equations of motion. Consequently, the same degree of the polynomial is required to ensure an accurate representation. Regardless of the polynomial degree, the error is small at the beginning and increases over time. Still, the error can be reduced significantly if higher orders are used. For example, $q = 7$ results in an error of only 10^1 – 10^2 km. As we consider an interplanetary transfer where the length scales are of the order of an astronomical unit, errors of 10^3 km are often sufficient for the cruise phase. Moreover, if the bilinear system is to be used within a direct method to solve the corresponding optimal control problem, the trajectory is divided into segments, and the dynamics need to be accurate only for a short time period. Therefore, even lower orders can yield sufficiently accurate results for the example considered in this chapter. The behavior of the velocity error in Fig. 10.7b is similar with maximum errors of approximately 0.1 – $1\,\mathrm{m\,s^{-1}}$. Similar trends are observed for the KS coordinates.

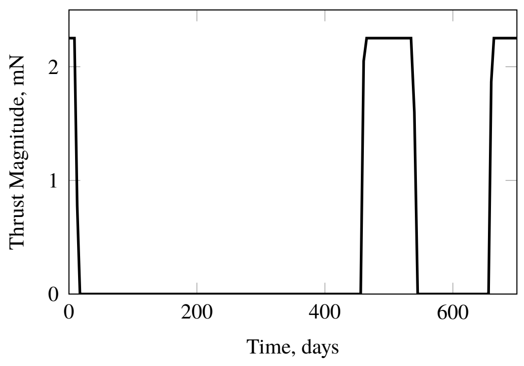
Note that Fig. 10.7 refers to the case without the additional perturbation term $1 - s^2$ on the right-hand side of Eq. (10.68). If it is included, the trend of the graphs is similar, but higher-order basis functions are needed to achieve the same accuracy.

Propagation Accuracy of the Linear System

A training set of 10^4 data points is generated by randomly adding noise to the nominal controls. In particular, each component of the control is varied by $10\,\%$ around the nominal one, and the set $\left[\mathbf{x}_j^+, \mathbf{x}_j, \mathbf{u}_j\right]$,

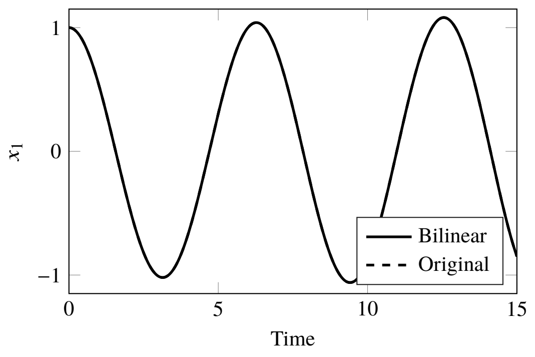


(a) Duffing oscillator.

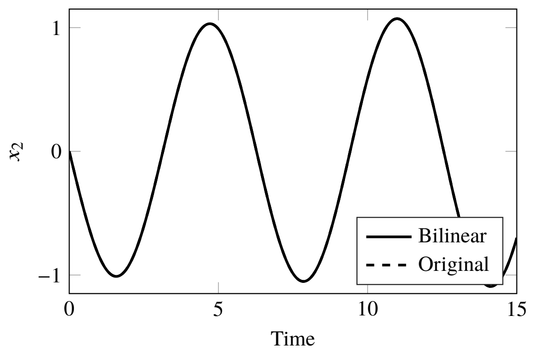


(b) Kepler problem.

Figure 10.3: Control profiles for the Duffing oscillator and Kepler problem.

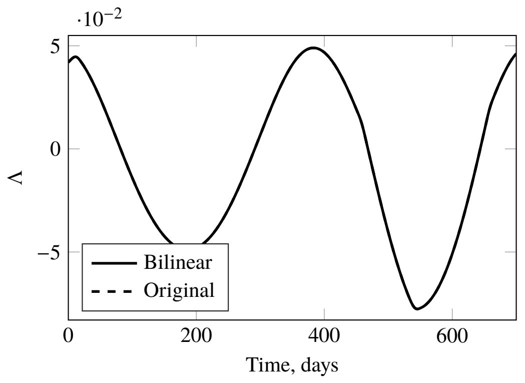


(a) Evolution of x_1.

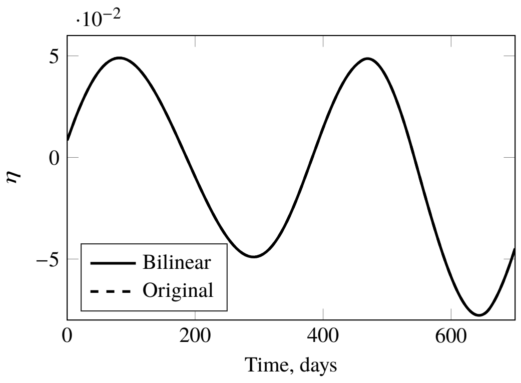


(b) Evolution of x_2.

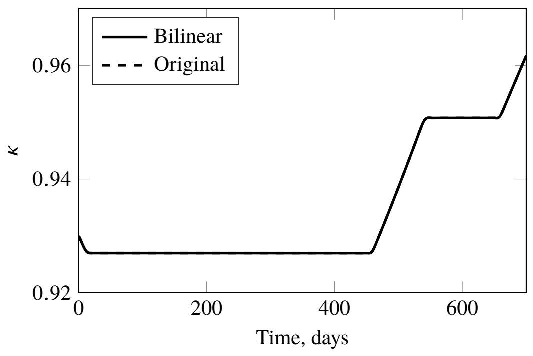
Figure 10.4: State trajectories for the Duffing oscillator.



(a) Evolution of Λ.

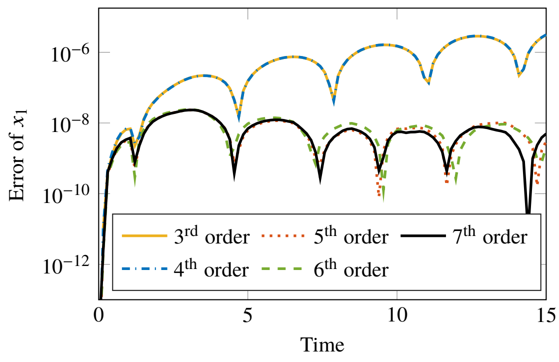


(b) Evolution of η.

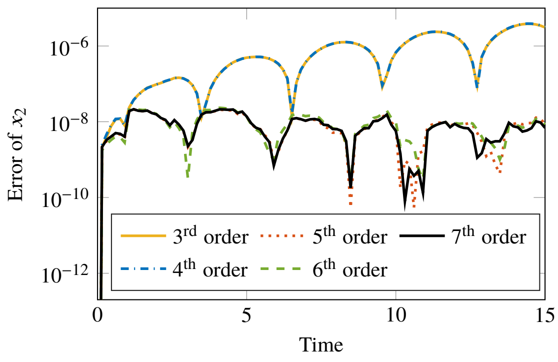


(c) Evolution of κ.

Figure 10.5: State trajectories for the Kepler problem.

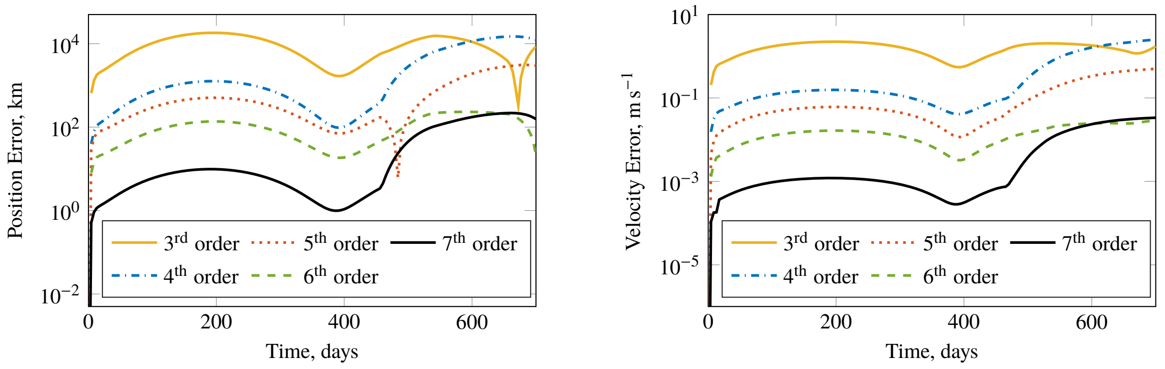


(a) Position error.



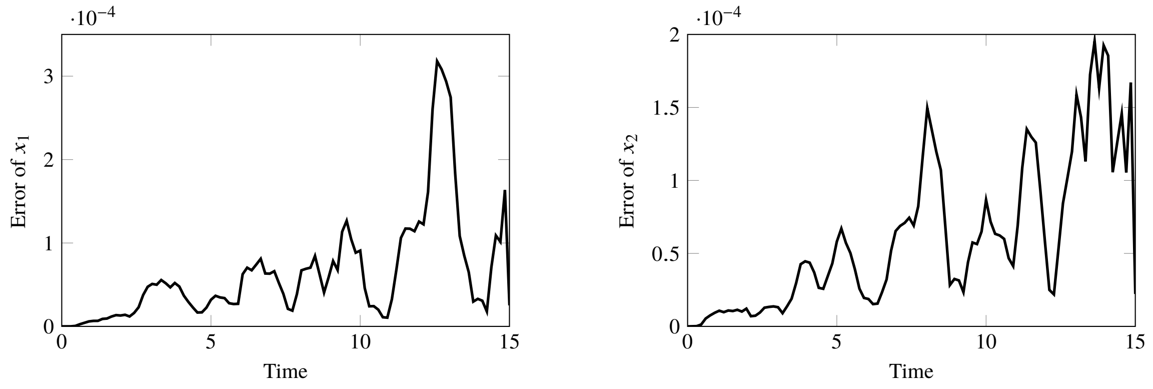
(b) Velocity error.

Figure 10.6: Position and velocity errors obtained with the bilinear system for basis function orders 3 to 7 for the Duffing oscillator.



(a) Position error. **(b)** Velocity error.

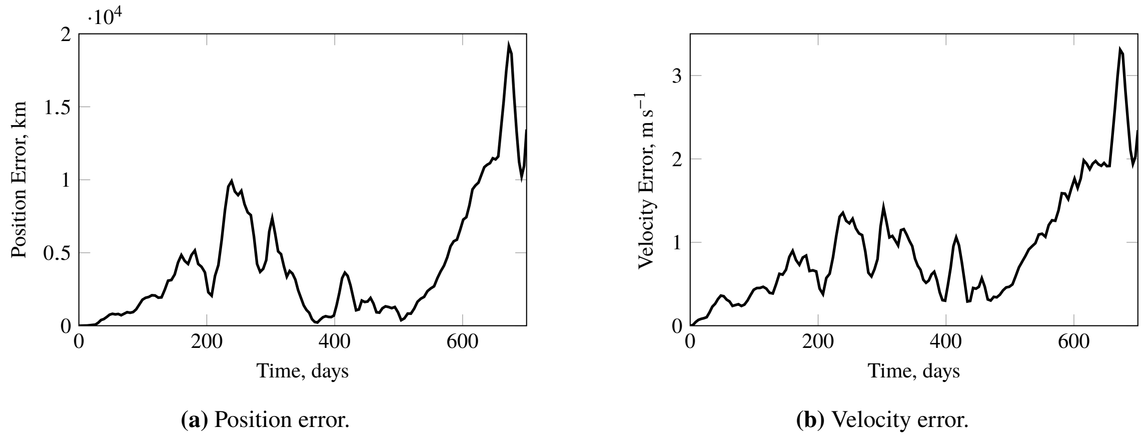
Figure 10.7: Position and velocity errors obtained with the bilinear system for basis function orders 3 to 7 for the Kepler problem.



(a) Position error. **(b)** Velocity error.

Figure 10.8: Position and velocity errors obtained with the linear system for the Duffing oscillator.

$j = 1, \ldots, 10^4$, is stored. Therefore, the evolution of the control profile is assumed to be approximately known. The constant control matrix $\mathbf{B}$ is computed, and the resulting linear dynamics are propagated. The position and velocity errors with respect to the integration of the original system is shown in Figs. 10.8 and 10.9. For both problems, the errors accumulate over time. However, this is expected because the EDMD optimizes over a single time step only. Despite the continuous control for the Duffing oscillator, the evolution of the states can be captured relatively accurately, especially during the beginning of the propagation. Regarding the orbital motion, a similar trend is observed. Even though a maximum error of approximately $2 \times 10^4\,\mathrm{km}$ and $3\,\mathrm{m\,s^{-1}}$ seems to be large, the propagation period is long, and such orders of magnitude may in fact be acceptable for deep-space applications. A disadvantage, however, is that the control structure needs to be known to some degree to achieve such an accuracy. If the variation of the controls is increased to $50\,\%$ or if the control profile is completely unknown, the order of magnitude of the errors increases by one (Kepler problem) or two (Duffing oscillator). In general, the accuracy can be enhanced if the system is re-lifted repeatedly to reset the error. Yet, this is a nonlinear operation, which may not be desirable.



(a) Position error. **(b)** Velocity error.

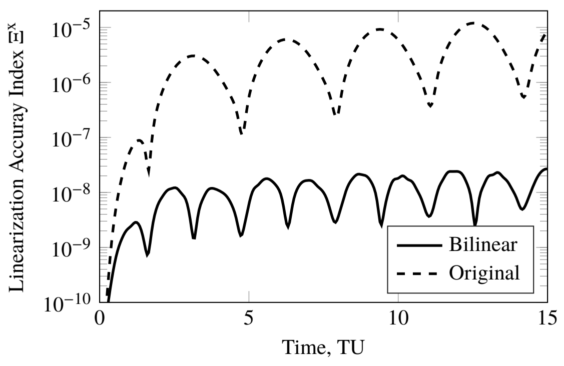
Figure 10.9: Position and velocity errors obtained with the linear system for the Kepler problem.

10.5.2 Linearization Accuracy Index

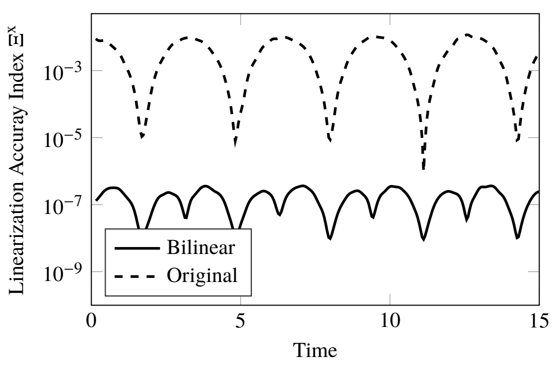
The perturbed trajectories for the Duffing oscillator are generated by multiplying the controls with random numbers between -10 and 10, and the original system is then propagated to obtain the states. The resulting indices Ξ^x for this case are shown in Fig. 10.10a. In an additional simulation, the states and controls are both multiplied by random numbers in the same range. For each case, 1000 samples are generated, and 100 discretization points are used to compute Ξ^x. The results are illustrated in Fig. 10.10b. Remarkably, the bilinear system achieves an index that is several orders of magnitude smaller than the one of the original system. Despite the larger number of dimensions, the accuracy of the linearization seems to be higher if a bilinear system is used.

With regard to the Kepler problem, we only consider the case where the unperturbed dynamics are linear. Even though it may seem unnecessary to apply the KOT to a system that is already linear in the unperturbed case, we want to investigate whether a highly nonlinear control term could benefit from a transformation into the form $\dot{\mathbf{z}} = \mathbf{D}\,\mathbf{z} + \mathbf{B}\,\mathbf{z}\,u$. Although this is still nonlinear, we expect it to behave better under linearization because $\mathbf{B}$ is constant, and $\mathbf{z}$ appears only linearly. The perturbed trajectories are obtained by multiplying the states and controls with random numbers between 0.9 and 1.1 (states), and 0.5 and 5 (controls). The resulting controls are then added to the nominal one. This ensures a sufficiently large perturbation even if the nominal control is zero. The number of discretization points is 150. The evolution of the index is shown in Fig. 10.11a. Although the difference becomes less significant, Ξ^x is still several times smaller in the bilinear case. A separate simulation is performed where the nominal controls are only multiplied by random numbers. This means that the perturbation of the control term is zero during coast arcs, and the linearization of the linear unperturbed dynamics is exact. Figure 10.11b shows the index over time. As expected, $\Xi^x = 0$ when there is no thrust. Interestingly, Ξ^x is again several times smaller for the bilinear representation. Therefore, even transforming a system that is already linear in the unperturbed case into bilinear form may be beneficial if the control terms are highly nonlinear.

Still, it is yet to be investigated whether the larger number of dimensions and the increased computational effort can be handled properly when solving more complex problems.

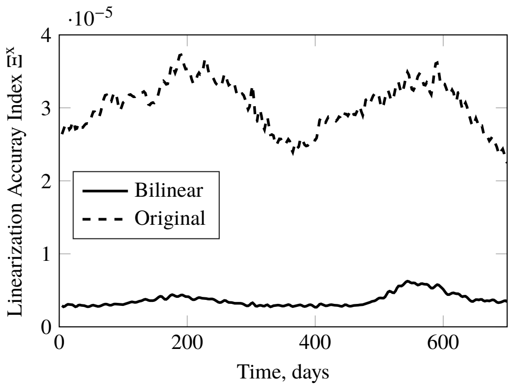


(a) Propagation with perturbed controls.

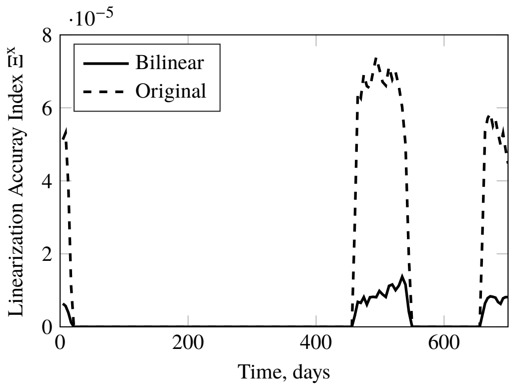


(b) Random perturbation of states and controls.

Figure 10.10: Linearization accuracy index for the Duffing oscillator.



(a) Adding randomly sampled perturbation to the control.



(b) Random perturbation of states and controls.

Figure 10.11: Linearization accuracy index for the Kepler problem.

11 Summary and Future Work

In this dissertation, a computationally efficient method based on convex programming was developed to solve the low-thrust trajectory optimization problem. A particular focus was on real-time guidance for interplanetary transfers. We conclude this work with a summary of the findings, and give recommendations for future work.

11.1 Summary

Based on the characteristics of the deep-space cruise, we define six high-level requirements for onboard guidance. A thorough assessment of state-of-the-art guidance methods is performed, and SCP is deemed most appropriate to generate optimal trajectories within the framework of an offline design tool, but also as a guidance approach where the trajectory is recomputed in real time when needed. We present a comprehensive overview of the sequential convex programming algorithm with different types of state vector representations, linearization techniques, and trust-region methods that we consider useful for a large variety of applications.

As none of the existing works has investigated how different discretization and trust-region methods and their parameters affect the performance of the SCP algorithm for complex interplanetary low-thrust fuel-optimal transfers, a thorough performance assessment is provided in Chapter 6. In that context, convex formulations of the differential and integral forms of the Legendre–Gauss–Radau pseudospectral method are developed, and their performance is compared with the first-order-hold and Legendre–Gauss–Lobatto method. The results show that FOH and a hard trust-region strategy often yield the best compromise in terms of convergence, onboard capability, accuracy, and optimality for the considered class of problems. The high percentage of converged cases has shown that SCP is very reliable despite the poor initial guesses. This is crucial for real-time guidance as a solution must be obtained at any time. The results show that a convex programming approach seems to be a viable method to compute low-thrust trajectories onboard. All of the fundamental requirements seem to be satisfied by almost all methods.

The choice of the coordinate system is crucial for nonlinear optimization methods, especially if linearization is involved. Therefore, different state vector representations are assessed for the low-thrust trajectory optimization problem in Chapter 7. In particular, three non-standard coordinates (two sets based on Kustaanheimo–Stiefel coordinates, and modified orbital elements) are introduced that result in linear dynamics in the unperturbed case. In addition, two nonlinearity-like metrics tailored to convex optimization are proposed to investigate how different state vector representations affect the performance of SCP when the same initial guess is provided. Our simulations suggest that MOE, MEE, spherical, and cylindrical coordinates outperform Cartesian coordinates in terms of success rate. Sets with an independent variable different from time that have linear unperturbed dynamics, such as MOE and KS$_E$, require significantly fewer iterations. Moreover, the different distribution of the discretization points of MOE and KS is often beneficial as more points are placed near periapsis where most thrust arcs occur. Despite the singularity for orbits with zero inclination, MOE are an excellent choice for preliminary studies due to the rapid speed and high robustness against poor initial guesses.

The homotopic approach in Chapter 8 is developed to improve convergence, accuracy, and the computational effort of SCP. The homotopy is embedded into the optimization process where the homotopic parameter is dynamically adjusted based on the constraint violation. This method is combined with a high-fidelity model where the complexity of the dynamics and constraints is successively increased. A novel trust-region update mechanism is presented to expand the feasible region when the homotopic parameter changes. The simulations show that increasing the complexity of the dynamical model only step by step increases convergence significantly. Despite the larger number of iterations that is required to reach convergence, the rapid speed of the proposed method makes it an excellent alternative to nonlinear programming solvers even for highly nonlinear problems. As no-thrust constraints can directly be included in the optimization process without a significant increase in computational effort and decrease in convergence, this approach can be considered another step towards computing more mission-compliant trajectories using convex programming techniques.

The closed-loop guidance simulation is presented in Chapter 9. A processor-in-the-loop experiment is carried out where a Raspberry Pi is used to assess the performance of the SCP algorithm on hardware comparable to flight qualified onboard processors. The deep-space cruise is simulated by repeatedly computing the reference trajectory and propagating the dynamics in a high-fidelity model. Various disturbances are considered, and a method is developed where the target is considered moving to allow for more flexibility in case of failure conditions. A Monte Carlo analysis is performed with a random sampling of the disturbances to demonstrate the high reliability of the proposed approach. The simulations show that the algorithm is able to provide feasible trajectories in all cases, and the maximum required CPU time of approximately one minute per optimization is acceptable for interplanetary transfers. In general, the CPU times to optimize a trajectory on a standard and underclocked Raspberry Pi range from few seconds to few minutes depending on the number of discretization points. This is considered acceptable for interplanetary transfers because they can last years. The excellent reliability and accuracy even in case of large, unexpected disturbances close to the target make the method a promising candidate for real space missions.

We provide a detailed procedure to transform a control-affine into a bilinear system, and we demonstrate how non-polynomial dynamical systems can be used within the Koopman operator framework. In particular, a step-by-step example of the Duffing oscillator is presented to make the reader familiar with the formalism. Besides this rather simple example, the bilinearization is also applied to the more complex perturbed Kepler problem. Although it requires some effort to transform the dynamical system into a higher-dimensional space, this needs to be done only once. The results show that such a global approximation can yield a sufficient level of accuracy even for lower-order basis functions. Yet, the computational effort is often considerably higher due to the larger number of dimensions. Moreover, the smaller linearization accuracy index suggests that using the bilinear form instead of the original system may be beneficial in terms of accuracy when computing a first-order Taylor series. Even though linear approximations of nonlinear systems are often preferable, we show that the accuracy becomes poor for larger prediction horizons. Still, the Koopman operator theory is an interesting approach to address nonlinear systems with a leading linear term and small perturbations.

Using the techniques described in each chapter, the reliability, accuracy, robustness, and computational efficiency of low-thrust trajectory optimization methods could be enhanced. The numerical simulations demonstrated that all defined high-level requirements for onboard guidance are fulfilled. The findings of this dissertation can therefore serve as a valuable contribution in the field of computational guidance for spacecraft equipped with low-thrust propulsion systems.

11.2 Future Work

Even though several advances could be made in this dissertation, there are various aspects that require further research:

SCP algorithm: Although the proposed SCP algorithm works well in practice, a more sophisticated mechanism for updating the trust region is desirable to avoid an excessive number of rejected steps. This may be advantageous in terms of iterations, CPU time, and also convergence. In addition, a general procedure on how to choose the initial trust-region radius is advisable. Moreover, it is yet to be investigated how the successive linearization approach compares with other methods, such as successive approximation or inner-convex approximations.

Discretization methods: As only a small selection of methods could be assessed in this dissertation, it is recommended to consider additional discretization techniques, for example based on Bernstein polynomials. Furthermore, a thorough analysis is desirable to investigate when a discretization method fails and when convergence can be guaranteed.

State vector representations: As we have seen that coordinate sets with linear unperturbed dynamics are often advantageous in terms of convergence and number of iterations within SCP, it is advisable to look into other coordinates that result in linear or weakly nonlinear constraints. Moreover, alternative ways to handle the time variable properly are recommended if the independent variable is not time. With regard to the modified orbital elements and Kustaanheimo–Stiefel coordinates, finding solutions to avoid the singularity and nonconvex boundary conditions, respectively, are also directions for future work.

High-fidelity models: Although the considered dynamical model yields accurate nominal results, it is an open question how the results change if uncertainty is included. For example, using initial and final probability distributions instead of fixed points is of utmost importance for real missions. How to convexify and solve the resulting stochastic optimal control problem requires further study.

Closed-loop guidance: Similarly, it is recommended to consider uncertainty also in the closed-loop guidance simulation. In particular, incorporating a high-fidelity model of the orbit determination process is important to make statements about the performance under more realistic conditions. Additional simulations to different targets and with different parameters would support the high reliability and robustness. This requires new methods, tools, and processes to eventually verify and validate autonomous guidance, navigation, and control algorithms.

Koopman operator theory: An important aspect not considered in this dissertation is the convergence of the Koopman operator. For example, it is to be investigated under what conditions convergence can be guaranteed. Moreover, other approaches to obtain a finite-dimensional representation of the Koopman matrix may yield more accurate solutions. If the matrix is not diagonalizable, robust methods are sought to obtain the eigenvalues reliably. Finally, the methods provided in this dissertation can be applied to optimization problems. This, however, merits further study as the original and transformed states are not independent.

Bibliography

[1] Slavinskis, A., Janhunen, P., Toivanen, P., Muinonen, K., Penttilä, A., Granvik, M., Kohout, T., Gritsevich, M., Slavinskis, A., Pajusalu, M., Sünter, I., Ehrpais, H., Dalbins, J., Iakubivskyi, I., Eenmäe, T., Pajusalu, M., Ilbis, E., Ehrpais, H., Muinonen, K., Gritsevich, M., Mauro, D., Stupl, J., Rivkin, A. S., and Bottke, W. F., "Nanospacecraft Fleet for Multi-Asteroid Touring With Electric Solar Wind Sails", *2018 IEEE Aerospace Conference*, 2018. DOI: 10.1109/AERO.2018.8396670.

[2] Hein, A., Saidani, M., and Tollu, H., "Exploring Potential Environmental Benefits of Asteroid Mining", *69th International Astronautical Congress*, 2018. Paper IAC-18,D4,5,11,x47396.

[3] Klesh, A., and Krajewski, J., "MarCO: Mars Cube One – Lessons Learned from Readying the First Interplanetary Cubesats for Flight", *Lunar and Planetary Science Conference*, 2018. Paper 2923.

[4] Poghosyan, A., and Golkar, A., "CubeSat Evolution: Analyzing CubeSat Capabilities for Conducting Science Missions", *Progress in Aerospace Sciences*, Vol. 88, 2017, pp. 59–83. DOI: 10.1016/j.paerosci.2016.11.002.

[5] Quadrelli, M. B., Wood, L. J., Riedel, J. E., McHenry, M. C., Aung, M., Cangahuala, L. A., Volpe, R. A., Beauchamp, P. M., and Cutts, J. A., "Guidance, Navigation, and Control Technology Assessment for Future Planetary Science Missions", *Journal of Guidance, Control, and Dynamics*, Vol. 38, No. 7, 2015, pp. 1165–1186. DOI: 10.2514/1.G000525.

[6] Dennehy, C. N., "Autonomous GN&C", *11th International ESA Conference on Guidance, Navigation & Control Systems*, 2021. Keynote talk.

[7] Lu, P., "Introducing Computational Guidance and Control", *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 2, 2017, pp. 193–193. DOI: 10.2514/1.G002745.

[8] Açıkmeşe, B., Aung, M., Casoliva, J., Mohan, S., Johnson, A., Scharf, D., Masten, Scotkin, D., Wolf, A., and Regehr, M., "Flight Testing of Trajectories Computed by G-FOLD: Fuel Optimal Large Divert Guidance Algorithm for Planetary Landing", *AAS/AIAA Space Flight Mechanics Meeting*, 2013. Paper AAS 13-386.

[9] Hofmann, C., and Topputo, F., "Rapid Low-Thrust Trajectory Optimization in Deep Space Based on Convex Programming", *Journal of Guidance, Control, and Dynamics*, Vol. 44, No. 7, 2021, pp. 1379–1388. DOI: 10.2514/1.G005839.

[10] Hofmann, C., and Topputo, F., "Pseudospectral Convex Low-Thrust Trajectory Optimization in a High-Fidelity Model", *AAS/AIAA Astrodynamics Specialist Conference*, 2021. Paper AAS 21-678.

[11] Hofmann, C., and Topputo, F., “Embedded Homotopy for Convex Low-Thrust Trajectory Optimization with Operational Constraints”, *AAS/AIAA Astrodynamics Specialist Conference*, 2022. Paper AAS 22-750.

[12] Hofmann, C., Morelli, A. C., and Topputo, F., “On the Performance of Discretization and Trust-Region Methods for On-Board Convex Low-Thrust Trajectory Optimization”, *AIAA SciTech Forum*, 2022. DOI: 10.2514/6.2022-1892.

[13] Hofmann, C., Morelli, A. C., and Topputo, F., “Performance Assessment of Convex Low-Thrust Trajectory Optimization Methods”, *Journal of Spacecraft and Rockets*, Vol. 60, No. 1, 2023, pp. 299–314. DOI: 10.2514/1.A35461.

[14] Hofmann, C., Morelli, A. C., and Topputo, F., “Impact of Different Coordinate Sets on the Performance of Convex Low-Thrust Trajectory Optimization”, *AAS/AIAA Space Flight Mechanics Meeting*, 2023. Paper AAS 23-291.

[15] Hofmann, C., and Topputo, F., “Closed-Loop Guidance for Low-Thrust Interplanetary Trajectories Using Convex Programming”, *11th International ESA Conference on Guidance, Navigation & Control Systems*, 2021. Paper 46.

[16] Hofmann, C., Servadio, S., Linares, R., and Topputo, F., “Advances in Koopman Operator Theory for Optimal Control Problems in Space Flight”, *AAS/AIAA Astrodynamics Specialist Conference*, 2022. Paper AAS 22-759.

[17] Bryson, A. E., and Ho, Y.-C., *Applied Optimal Control: Optimization, Estimation, and Control*, Blaisdell Publishing Company, 1969. pp. 42–127.

[18] Hull, D. G., *Optimal Control Theory for Applications*, Springer-Verlag New York, Inc., 2003. pp. 42–127.

[19] Betts, J. T., *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*, Society for Industrial and Applied Mathematics, 2010. pp. 1–49.

[20] Nocedal, J., and Wright, S. J., *Numerical Optimization*, Springer-Verlag New York, Inc., 1999. pp. 314–330.

[21] Boyd, S., and Vandenberghe, L., *Convex Optimization*, Cambridge University Press, 2004. pp. 127–184.

[22] Liu, X., Lu, P., and Pan, B., “Survey of Convex Optimization for Aerospace Applications”, *Astrodynamics*, Vol. 1, No. 1, 2017, pp. 23–40. DOI: 10.1007/s42064-017-0003-8.

[23] Zhang, C., Topputo, F., Bernelli-Zazzera, F., and Zhao, Y. S., “Low-Thrust Minimum-Fuel Optimization in the Circular Restricted Three-Body Model”, *Advances in the Astronautical Sciences*, Vol. 38, No. 8, 2015, pp. 1597–1615. DOI: 10.2514/1.G001080.

[24] Conway, B. A. (ed.), *Spacecraft Trajectory Optimization*, Cambridge Aerospace Series, Cambridge University Press, 2010. DOI: 10.1017/CBO9780511778025.

[25] Lantoine, G., and Russell, R. P., “Complete Closed-Form Solutions of the Stark Problem”, *Celestial Mechanics and Dynamical Astronomy*, Vol. 109, No. 4, 2011, pp. 333–366. DOI: 10.1007/s10569-010-9331-1.

[26] Edelbaum, T. N., "Propulsion Requirements for Controllable Satellites", *ARS Journal*, Vol. 31, No. 8, 1961, pp. 1079–1089. DOI: 10.2514/8.5723.

[27] Edelbaum, T. N., "An Asymptotic Solution for Optimum Power Limited Orbit Transfer", *AIAA Journal*, Vol. 4, No. 8, 1966, pp. 1491–1494. DOI: 10.2514/3.3725.

[28] Izzo, D., and Biscani, F., "Explicit Solution to the Constant Radial Acceleration Problem", *Journal of Guidance, Control, and Dynamics*, Vol. 38, No. 4, 2015, pp. 733–739. DOI: 10.2514/1.G000116.

[29] Bombardelli, C., Baù, G., and Pelàez, J., "Asymptotic Solution for the Two-Body Problem With Constant Tangential Thrust Acceleration", *Celestial Mechanics and Dynamical Astronomy*, Vol. 110, No. 3, 2011, pp. 239–256. DOI: 10.1007/s10569-011-9353-3.

[30] da Silva Fernandes, S., das Chagas Carvalho, F., and de Moraes, R., "Optimal Low-Thrust Transfers Between Coplanar Orbits With Small Eccentricities", *Computational and Applied Mathematics*, Vol. 35, No. 3, 2016, pp. 803–816. DOI: 10.1007/s40314-015-0249-9.

[31] Kechichian, J. A., "Orbit Raising With Low-Thrust Tangential Acceleration in Presence of Earth Shadow", *Journal of Spacecraft and Rockets*, Vol. 35, No. 4, 1998, pp. 516–525. DOI: 10.2514/2.3361.

[32] Kluever, C. A., "Using Edelbaum's Method to Compute Low-Thrust Transfers With Earth-Shadow Eclipses", *Journal of Guidance, Control, and Dynamics*, Vol. 34, No. 1, 2011, pp. 300–303. DOI: 10.2514/1.51024.

[33] Quarta, A. A., and Mengali, G., "Semi-Analytical Method for the Analysis of Solar Sail Heliocentric Orbit Raising", *Journal of Guidance, Control, and Dynamics*, Vol. 35, No. 1, 2012, pp. 330–335. DOI: 10.2514/1.55101.

[34] Betts, J. T., "Survey of Numerical Methods for Trajectory Optimization", *Journal of Guidance, Control, and Dynamics*, Vol. 21, No. 2, 1998, pp. 193 – 207. DOI: 10.2514/2.4231.

[35] Rao, A. V., "A Survey of Numerical Methods for Trajectory Optmization", *AAS/AIAA Astrodynamics Specialist Conference*, 2009. Paper AAS 09-334.

[36] Bertrand, R., and Epenoy, R., "New Smoothing Techniques for Solving Bang–Bang Optimal Control Problems – Numerical Results and Statistical Interpretation", *Optimal Control Applications and Methods*, Vol. 23, No. 4, 2002, pp. 171–197. DOI: 10.1002/oca.709.

[37] Haberkorn, T., Martinon, P., and Gergaud, J., "Low Thrust Minimum-Fuel Orbital Transfer: A Homotopic Approach", *Journal of Guidance, Control, and Dynamics*, Vol. 27, No. 6, 2008, pp. 1046–1060. DOI: 10.2514/1.4022.

[38] Guo, C., Zhang, J., Luo, Y., and Yang, L., "Phase-Matching Homotopic Method for Indirect Optimization of Long-Duration Low-Thrust Trajectories", *Advances in Space Research*, Vol. 62, No. 3, 2018, pp. 568–579. DOI: 10.1016/j.asr.2018.05.007.

[39] Taheri, E., Kolmanovsky, I., and Atkins, E., “Enhanced Smoothing Technique for Indirect Optimization of Minimum-Fuel Low-Thrust Trajectories”, *Journal of Guidance, Control, and Dynamics*, Vol. 39, No. 11, 2016, pp. 2500–2511. DOI: 10.2514/1.g000379.

[40] Guo, T., Jiang, F., and Li, J., “Homotopic Approach and Pseudospectral Method Applied Jointly to Low Thrust Trajectory Optimization”, *Acta Astronautica*, Vol. 71, 2012, pp. 38–50. DOI: 10.1016/j.actaastro.2011.08.008.

[41] Taheri, E., Li, N. I., and Kolmanovsky, I., “Co-State Initialization for the Minimum-Time Low-Thrust Trajectory Optimization”, *Advances in Space Research*, Vol. 59, No. 9, 2017, pp. 2360–2373. DOI: 10.1016/j.asr.2017.02.010.

[42] Patterson, M. A., and Rao, A. V., “Exploiting Sparsity in Direct Collocation Pseudospectral Methods for Solving Optimal Control Problems”, *Journal of Spacecraft and Rockets*, Vol. 49, No. 2, 2012, pp. 3627–3646. DOI: 10.2514/1.A32071.

[43] Darby, L., Hager, W. W., and Rao, A. V., “An hp-Adaptive Pseudospectral Method for Solving Optimal Control Problems”, *Optimal Control Applications and Methods*, Vol. 32, No. 4, 2011, pp. 476–502. DOI: 10.1002/oca.957.

[44] Patterson, M. A., Hager, W. W., and Rao, A. V., “A ph Mesh Refinement Method for Optimal Control”, *Optimal Control Applications and Methods*, Vol. 36, 2015, pp. 398–421. DOI: 10.1002/oca.2114.

[45] Bellman, R., *Dynamic Programming*, Princeton University Press, 1957. p. 83.

[46] Sassano, M., and Astolfi, A., “Dynamic Approximate Solutions of the HJ Inequality and of the HJB Equation for Input-Affine Nonlinear Systems”, *IEEE Transactions on Automatic Control*, Vol. 57, No. 10, 2012, pp. 2490–2503. DOI: 10.1109/TAC.2012.2186716.

[47] Colombo, C., Vasile, M., and Radice, G., “Optimal Low-Thrust Trajectories to Asteroids Through an Algorithm Based on Differential Dynamic Programming”, *Celestial Mechanics and Dynamical Astronomy*, Vol. 105, No. 1, 2009, pp. 75–112. DOI: 10.1007/s10569-009-9224-3.

[48] Lantoine, G., and Russell, R. P., “A Hybrid Differential Dynamic Programming Algorithm for Constrained Optimal Control Problems. Part 1: Theory”, *Journal of Optimization Theory and Applications*, Vol. 154, No. 2, 2012, pp. 382–417. DOI: 10.1007/s10957-012-0039-0.

[49] Lantoine, G., and Russell, R. P., “A Hybrid Differential Dynamic Programming Algorithm for Constrained Optimal Control Problems. Part 2: Application”, *Journal of Optimization Theory and Applications*, Vol. 154, No. 2, 2012, pp. 418–442. DOI: 10.1007/s10957-012-0038-1.

[50] Conway, B. A., “A Survey of Methods Available for the Numerical Optimization of Continuous Dynamic Systems”, *Journal of Optimization Theory and Applications*, Vol. 152, No. 2, 2012, pp. 271 – 306. DOI: 10.1007/s10957-011-9918-z.

[51] Yokoyama, N., and Suzuki, S., “Modified Genetic Algorithm for Constrained Trajectory Optimization”, *Journal of Guidance, Control, and Dynamics*, Vol. 28, No. 1, 2005, pp. 139–144. DOI: 10.2514/1.3042.

[52] Wall, B., and Conway, B. A., “Near-Optimal Low-Thrust Earth-Mars Trajectories via a Genetic Algorithm”, *Journal of Guidance, Control, and Dynamics*, Vol. 28, No. 5, 2005, pp. 1027–1031. DOI: 10.2514/1.11891.

[53] Olds, A. D., Kluever, C. A., and Cupples, M. L., “Interplanetary Mission Design Using Differential Evolution”, *Advances in the Astronautical Sciences*, Vol. 124 I, No. 5, 2006, pp. 837–856. DOI: 10.2514/1.27242.

[54] Vasile, M., Minisci, E., and Locatelli, M., “An Inflationary Differential Evolution Algorithm for Space Trajectory Optimization”, *IEEE Transactions on Evolutionary Computation*, Vol. 15, No. 2, 2011, pp. 267–281. DOI: 10.1109/TEVC.2010.2087026.

[55] Radice, G., and Olmo, G., “Ant Colony Algorithms for Two Impluse Interplanetary Trajectory Optimization”, *Journal of Guidance, Control, and Dynamics*, Vol. 29, No. 6, 2006, pp. 1440–1444. DOI: 10.2514/1.20828.

[56] Ceriotti, M., and Vasile, M., “MGA Trajectory Planning With an ACO-Inspired Algorithm”, *Acta Astronautica*, Vol. 67, No. 9, 2010, pp. 1202–1217. DOI: 10.1016/j.actaastro.2010.07.001.

[57] Pontani, M., and Conway, B. A., “Particle Swarm Optimization Applied to Space Trajectories”, *Journal of Guidance, Control, and Dynamics*, Vol. 33, No. 5, 2010, pp. 1429–1441. DOI: 10.2514/1.48475.

[58] Shan, J., and Ren, Y., “Low-Thrust Trajectory Design With Constrained Particle Swarm Optimization”, *Aerospace Science and Technology*, Vol. 36, 2014, pp. 114–124. DOI: 10.1016/j.ast.2014.04.004.

[59] Furfaro, R., Bloise, I., Orlandelli, M., Di, P., Topputo, F., and Linares, R., “A Recurrent Deep Architecture for Quasi-Optimal Feedback Guidance in Planetary Landing”, *IAA/AAS SciTech Forum on Space Flight Mechanics and Space Structures and Materials*, 2018. Paper AAS 18-813.

[60] Cheng, L., Wang, Z., Jiang, F., and Zhou, C., “Real-Time Optimal Control for Spacecraft Orbit Transfer via Multiscale Deep Neural Networks”, *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 55, No. 5, 2019, pp. 2436–2450. DOI: 10.1109/TAES.2018.2889571.

[61] Cheng, L., Wang, Z., Song, Y., and Jiang, F., “Real-Time Optimal Control for Irregular Asteroid Landings Using Deep Neural Networks”, *Acta Astronautica*, Vol. 170, 2020, pp. 66–79. DOI: 10.1016/j.actaastro.2019.11.039.

[62] Izzo, D., and Öztürk, E., “Real-Time Guidance for Low-Thrust Transfers Using Deep Neural Networks”, *Journal of Guidance, Control, and Dynamics*, Vol. 44, No. 2, 2021, pp. 315–327. DOI: 10.2514/1.G005254.

[63] Li, W., Huang, H., and Peng, F., “Trajectory Classification in Circular Restricted Three-Body Problem Using Support Vector Machine”, *Advances in Space Research*, Vol. 56, No. 2, 2015, pp. 273–280. DOI: 10.1016/j.asr.2015.04.017.

[64] Izzo, D., Sprague, C. I., and Tailor, D. V., “Machine Learning and Evolutionary Techniques in Interplanetary Trajectory Design”, *Modeling and Optimization in Space Engineering: State of the Art and New Challenges*, Springer International Publishing, 2019, pp. 191–210. DOI: 10.1007/978-3-030-10501-3_8.

[65] Gaudet, B., and Furfaro, R., "Robust Spacecraft Hovering Near Small Bodies in Environments with Unknown Dynamics Using Reinforcement Learning", *AIAA/AAS Astrodynamics Specialist Conference*, 2012. DOI: 10.2514/6.2012-5072.

[66] Hovell, K., and Ulrich, S., "On Deep Reinforcement Learning for Spacecraft Guidance", *AIAA SciTech Forum*, 2020. DOI: 10.2514/6.2020-1600.

[67] Scorsoglio, A., Furfaro, R., Linares, R., and Gaudet, B., "Image-based Deep Reinforcement Learning for Autonomous Lunar Landing", *AIAA SciTech Forum*, 2020. DOI: 10.2514/6.2020-1910.

[68] Raissi, M., Perdikaris, P., and Karniadakis, G., "Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations", *Journal of Computational Physics*, Vol. 378, 2019, pp. 686–707. DOI: https://doi.org/10.1016/j.jcp.2018.10.045.

[69] Petropoulos, A. E., "Simple Control Laws for Low-Thrust Orbit Transfers", *AAS/AIAA Astrodynamics Specialist Conference*, 2003. Paper AAS 03-630.

[70] Petropoulos, A. E., "Refinements to the Q-law for the Low-Thrust Orbit Transfers", *AAS/AIAA Space Flight Mechanics Conference*, 2005. Paper AAS 05-162.

[71] Ruggiero, A., Pergola, P., Marcuccio, S., and Andrenucci, M., "Low-Thrust Maneuvers for the Efficient Correction of Orbital Elements", *International Electric Propulsion Conference*, 2011. Paper IEPC-2011-102.

[72] Varga, G. I., and Sánchez Pérez, J. M., "Many-Revolution Low-Thrust Orbit Transfer Computation Using Equinoctial Q-Law Including J2 and Eclipse Effects", *AAS/AIAA Astrodynamics Specialist Conference*, 2015. Paper AAS 15-590.

[73] Hatten, N. A., "A Critical Evaluation of Modern Low-Thrust, Feedback-Driven Spacecraft Control Laws", Master's thesis, University of Texas at Austin, 2012.

[74] Gondelach, D. J., and Noomen, R., "Hodographic-Shaping Method for Low-Thrust Interplanetary Trajectory Design", *Journal of Spacecraft and Rockets*, Vol. 52, No. 3, 2015, pp. 728–738. DOI: 10.2514/1.A32991.

[75] Taheri, E., and Abdelkhalik, O., "Shape Based Approximation of Constrained Low-Thrust Space Trajectories using Fourier Series", *Journal of Spacecraft and Rockets*, Vol. 49, No. 3, 2012, pp. 535–546. DOI: 10.2514/1.58789.

[76] Taheri, E., and Abdelkhalik, O., "Initial Three-Dimensional Low-Thrust Trajectory Design", *Advances in Space Research*, Vol. 57, No. 3, 2016, pp. 889 – 903. DOI: 10.1016/j.asr.2015.11.034.

[77] Vavrina, M., and Howell, K., "Global Low-Thrust Trajectory Optimization Through Hybridization of a Genetic Algorithm and a Direct Method", *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, 2008, pp. 1–27. DOI: 10.2514/6.2008-6614.

[78] Yang, D.-L., Xu, B., and Zhang, L., “Optimal Low-Thrust Spiral Trajectories Using Lyapunov-Based Guidance”, *Acta Astronautica*, Vol. 126, 2016, pp. 275–285. DOI: 10.1016/j.actaastro.2016.04.028.

[79] Franzese, V., and Topputo, F., “Optimal Beacons Selection for Deep-Space Optical Navigation”, *The Journal of the Astronautical Sciences*, Vol. 67, No. 4, 2020, p. 1775–1792. DOI: 10.1007/s40295-020-00242-z.

[80] Andreis, E., Franzese, V., and Topputo, F., “Onboard Orbit Determination for Deep-Space CubeSats”, *Journal of Guidance, Control, and Dynamics*, Vol. 45, No. 8, 2022, pp. 1466–1480. DOI: 10.2514/1.G006294.

[81] Açıkmeşe, B., Aung, M., Casoliva, J., Mohan, S., Johnson, A., Scharf, D., Masten, Scotkin, D., Wolf, A., and Regehr, M., “Flight Testing of Trajectories Computed by G-FOLD: Fuel Optimal Large Divert Guidance Algorithm for Planetary Landing”, *AAS/AIAA Space Flight Mechanics Meeting*, 2013. Paper AAS 13-386.

[82] Liu, C., Lin, C.-Y., and Tomizuka, M., “The Convex Feasible Set Algorithm for Real-Time Optimization in Motion Planning”, *SIAM Journal on Control and Optimization*, Vol. 56, No. 4, 2018. DOI: 10.1137/16M1091460.

[83] Szmuk, M., Pascucci, C. A., and Açıkmeşe, B., “Real-Time Quad-Rotor Path Planning for Mobile Obstacle Avoidance Using Convex Optimization”, *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 1–9. DOI: 10.1109/IROS.2018.8594351.

[84] Açıkmeşe, B., and Blackmore, L., “Lossless Convexification of a Class of Optimal Control Problems With Non-Convex Control Constraints”, *Automatica*, Vol. 47, No. 2, 2011, pp. 341–347. DOI: 10.1016/j.automatica.2010.10.037.

[85] Harris, M. W., and Açıkmeşe, B., “Lossless Convexification of Non-Convex Optimal Control Problems for State Constrained Linear Systems”, *Automatica*, Vol. 50, No. 9, 2014, p. 2304–2311. DOI: 10.1016/j.automatica.2014.06.008.

[86] Domahidi, A., Chu, E., and Boyd, S., “ECOS: An SOCP Solver for Embedded Systems”, *European Control Conference*, 2013, pp. 3071–3076. DOI: 10.23919/ECC.2013.6669541.

[87] Mao, Y., Szmuk, M., Xu, X., and Açıkmeşe, B., “Successive Convexification: A Superlinearly Convergent Algorithm for Non-convex Optimal Control Problems”, 2019. Preprint, `https://arxiv.org/abs/1804.06539`.

[88] Alonso-Mora, J., Baker, S., and Rus, D., “Multi-robot navigation in formation via sequential convex programming”, *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 4634–4641. DOI: 10.1109/IROS.2015.7354037.

[89] Chen, Y., Cutler, M., and How, J. P., “Decoupled multiagent path planning via incremental sequential convex programming”, *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 5954–5961. DOI: 10.1109/ICRA.2015.7140034.

[90] Lu, P., and Liu, X., "Autonomous Trajectory Planning for Rendezvous and Proximity Operations by Conic Optimization", *Journal of Guidance, Control, and Dynamics*, Vol. 36, No. 2, 2013, pp. 375–389. DOI: 10.2514/1.58436.

[91] Liu, X., and Lu, P., "Solving Nonconvex Optimal Control Problems by Convex Optimization", *Journal of Guidance, Control, and Dynamics*, Vol. 37, No. 3, 2014. DOI: 10.2514/1.62110.

[92] Açıkmeşe, B., and Ploen, S. R., "Convex Programming Approach to Powered Descent Guidance for Mars Landing", *Journal of Guidance, Control, and Dynamics*, Vol. 30, No. 5, 2007, pp. 1353–1366. DOI: 10.2514/1.27553.

[93] Blackmore, L., Açıkmeşe, B., and Scharf, D. P., "Minimum-Landing-Error Powered-Descent Guidance for Mars Landing Using Convex Optimization", *Journal of Guidance, Control, and Dynamics*, Vol. 33, No. 4, 2010, pp. 1161–1171. DOI: 10.2514/1.47202.

[94] Sagliano, M., "Pseudospectral Convex Optimization for Powered Descent and Landing", *Journal of Guidance, Control, and Dynamics*, Vol. 41, No. 2, 2018. DOI: 10.2514/1.G002818.

[95] Yang, H., Bai, X., and Baoyin, H., "Rapid Generation of Time-Optimal Trajectories for Asteroid Landing via Convex Optimization", *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 3, 2018. DOI: 10.2514/1.G002170.

[96] Liu, X., and Shen, Z., "Entry Trajectory Optimization by Second-Order Cone Programming", *Journal of Guidance, Control, and Dynamics*, Vol. 39, No. 2, 2016. DOI: 10.2514/1.G001210.

[97] Wang, Z., and Grant, M. J., "Constrained Trajectory Optimization for Planetary Entry via Sequential Convex Programming", *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 10, 2017. DOI: 10.2514/1.G002150.

[98] Sagliano, M., "Generalized hp Pseudospectral-Convex Programming for Powered Descent and Landing", *Journal of Guidance, Control, and Dynamics*, Vol. 42, No. 7, 2019, pp. 1562–1570. DOI: 10.2514/1.G003731.

[99] Grant, M., and Boyd, S., "CVX: Matlab Software for Disciplined Convex Programming", `http://cvxr.com/cvx`, Mar. 2014.

[100] Reynolds, T., Malyuta, D., Mesbahi, M., Açıkmeşe, B., and Carson, J. M., "A Real-Time Algorithm for Non-Convex Powered Descent Guidance", *AIAA SciTech Forum*, 2020. DOI: 10.2514/6.2020-0844.

[101] Dueri, D., Açıkmeşe, B., Scharf, D. P., and Harris, M. W., "Customized Real-Time Interior-Point Methods for Onboard Powered-Descent Guidance", *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 2, 2017, pp. 197–212. DOI: 10.2514/1.G001480.

[102] Mao, Y., Szmuk, M., and Açıkmeşe, B., "A Tutorial on Real-time Convex Optimization Based Guidance and Control for Aerospace Applications", *2018 Annual American Control Conference (ACC)*, 2018, pp. 2410–2416. DOI: 10.23919/ACC.2018.8430984.

[103] Yu, Y., Elango, P., Topcu, U., and Açıkmeşe, B., "Proportional–Integral Projected Gradient Method for Conic Optimization", *Automatica*, Vol. 142, 2022, p. 110359. DOI: 10.1016/j.automatica.2022.110359.

[104] Wang, Z., and Grant, M. J., "Optimization of Minimum-Time Low-Thrust Transfers Using Convex Programming", *Journal of Spacecraft and Rockets*, Vol. 55, No. 3, 2018, pp. 586–598. DOI: 10.2514/1.A33995.

[105] Wang, Z., and Grant, M. J., "Minimum-Fuel Low-Thrust Transfers for Spacecraft: A Convex Approach", *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 54, No. 5, 2018, pp. 2274–2290. DOI: 10.1109/TAES.2018.2812558.

[106] Tang, G., Jiang, F., and Li, J., "Fuel-Optimal Low-Thrust Trajectory Optimization Using Indirect Method and Successive Convex Programming", *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 54, No. 4, 2018, pp. 2053–2066. DOI: 10.1109/TAES.2018.2803558.

[107] Huang, R. C., Hwang, I., and Corless, M. J., "Nonlinear Algorithm for Tracking Interplanetary Low-Thrust Trajectories", *Journal of Guidance, Control, and Dynamics*, Vol. 35, No. 2, 2012, pp. 696–700. DOI: 10.2514/1.53001.

[108] Oestreich, C. E., Linares, R., and Gondhalekar, R., "Tube-Based Model Predictive Control with Uncertainty Identification for Autonomous Spacecraft Maneuvers", *Journal of Guidance, Control, and Dynamics*, Vol. 46, No. 1, 2023, pp. 6–20. DOI: 10.2514/1.G006438.

[109] Lee, U., and Mesbahi, M., "Constrained Autonomous Precision Landing via Dual Quaternions and Model Predictive Control", *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 2, 2017, pp. 292–308. DOI: 10.2514/1.G001879.

[110] Kabzan, J., Hewing, L., Liniger, A., and Zeilinger, M. N., "Learning-Based Model Predictive Control for Autonomous Racing", *IEEE Robotics and Automation Letters*, Vol. 4, No. 4, 2019, pp. 3363–3370. DOI: 10.1109/LRA.2019.2926677.

[111] Scharf, D. P., Regehr, M. W., Vaughan, G. M., Benito, J., Ansari, H., Aung, M., Johnson, A., Casoliva, J., Mohan, S., Dueri, D., Açıkmeşe, B., Masten, D., and Nietfeld, S., "ADAPT Demonstrations of Onboard Large-Divert Guidance with a VTVL Rocket", *IEEE Aerospace Conference*, 2014. DOI: 10.1109/AERO.2014.6836462.

[112] Szmuk, M., Pascucci, C. A., Dueri, D., and Açıkmeşe, B., "Convexification and Real-Time On-Board Optimization for Agile Quad-Rotor Maneuvering and Obstacle Avoidance", *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 4862–4868. DOI: 10.1109/IROS.2017.8206363.

[113] Szmuk, M., Malyuta, D., Reynolds, T. P., Mceowen, M. S., and Açıkmeşe, B., "Real-Time Quad-Rotor Path Planning Using Convex Optimization and Compound State-Triggered Constraints", *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 7666–7673. DOI: 10.1109/IROS40897.2019.8967706.

[114] Reynolds, T., Malyuta, D., Mesbahi, M., Açıkmeşe, B., and Carson, J. M., "Funnel Synthesis for the 6-DOF Powered Descent Guidance Problem", *AIAA SciTech Forum*, 2021. DOI: 10.2514/6.2021-0504.

[115] Wang, Z., and Lu, Y., "Improved Sequential Convex Programming Algorithms for Entry Trajectory Optimization", *Journal of Spacecraft and Rockets*, Vol. 57, No. 6, 2020, pp. 1373–1386. DOI: 10.2514/1.A34640.

[116] Topputo, F., and Zhang, C., "Survey of Direct Transcription for Low-Thrust Space Trajectory Optimization with Applications", *Abstract and Applied Analysis*, Vol. 2014, 2014, pp. 1–15. DOI: 10.1155/2014/851720.

[117] Williams, P., "Hermite–Legendre–Gauss–Lobatto Direct Transcription in Trajectory Optimization", *Journal of Guidance, Navigation, and Control*, Vol. 32, No. 4, 2009, pp. 1392–1395. DOI: 10.2514/1.42731.

[118] Benson, D. A., Huntington, G. T., Thorvaldsen, T. P., and Rao, A. V., "Direct Trajectory Optimization and Costate Estimation via an Orthogonal Collocation Method", *Journal of Guidance, Control, and Dynamics*, Vol. 29, No. 6, 2006, pp. 1435–1440. DOI: 10.2514/1.20478.

[119] Garg, D., Patterson, M. . A., Francolin, C., L. Darby, C. L., Huntington, G. T., Hager, W. W., and Rao, A. V., "Direct Trajectory Optimization and Costate Estimation of Finite-Horizon and Infinite-Horizon Optimal Control Problems Using a Radau Pseudospectral Method", *Computational Optimization and Applications*, Vol. 49, No. 2, 2011, pp. 335–358. DOI: 10.2514/1.20478.

[120] Garg, D., Patterson, M., Hager, W., Rao, A., Benson, D., and Huntington, G., "An Overview of Three Pseudospectral Methods for the Numerical Solution of Optimal Control Problems", *Advances in the Astronautical Sciences*, Vol. 135, 2009, pp. 475–487.

[121] Yu, C., and Zhao, Y., D. Yang, "Efficient Convex Optimization of Reentry Trajectory via the Chebyshev Pseudospectral Method", *International Journal of Aerospace Engineering*, Vol. 2019, 2019. DOI: 10.1155/2019/1414279, article 1414279.

[122] Reynolds, T. P., Szmuk, M., Malyuta, D., Mesbahi, M., Açıkmeşe, B., and Carson, J. M., "Dual Quaternion-Based Powered Descent Guidance with State-Triggered Constraints", *Journal of Guidance, Control, and Dynamics*, Vol. 43, No. 9, 2020, pp. 1584–1599. DOI: 10.2514/1.G004536.

[123] Kayama, Y., Howell, K. C., Bando, M., and Hokamoto, S., "Low-Thrust Trajectory Design with Successive Convex Optimization for Libration Point Orbits", *Journal of Guidance, Control, and Dynamics*, Vol. 45, No. 4, 2022, pp. 623–637. DOI: 10.2514/1.G005916.

[124] Malyuta, D., Reynolds, T., Szmuk, M., Mesbahi, M., Açıkmeşe, B., and Carson, J. M., "Discretization Performance and Accuracy Analysis for the Rocket Powered Descent Guidance Problem", *AIAA SciTech Forum*, 2019. DOI: 10.2514/6.2019-0925.

[125] Saghamanesh, M., Taheri, E., and Baoyin, H., "Systematic Low-Thrust Trajectory Design to Mars Based on a Full Ephemeris Modeling", *Advances in Space Research*, Vol. 64, No. 11, 2019, pp. 2356–2378. DOI: 10.1016/j.asr.2019.08.013.

[126] Taheri, E., Junkins, J. L., Kolmanovsky, I., and Girard, A., “A Novel Approach for Optimal Trajectory Design With Multiple Operation Modes of Propulsion System, Part 1”, *Acta Astronautica*, Vol. 172, 2020, pp. 151–165. DOI: 10.1016/j.actaastro.2020.02.042.

[127] Whiffen, G. J., “Mystic: Implementation of the Static Dynamic Optimal Control Algorithm for High-Fidelity, Low-Thrust Trajectory Design”, *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, 2006. DOI: 10.2514/6.2006-6741.

[128] Englander, J., Knittel, J. M., Williams, K., Stanbridge, D., and Ellison, D. H., “Validation of a Low-Thrust Mission Design Tool Using Operational Navigation Software”, *AAS/AIAA Space Flight Mechanics Meeting*, 2017. Paper AAS 17-204.

[129] Topputo, F., Wang, Y., Giordano, G., Franzese, V., Goldberg, H., Perez-Lissi, F., and Walker, R., “Envelop of Reachable Asteroids by M-ARGO CubeSat”, *Advances in Space Research*, Vol. 67, No. 12, 2021, pp. 4193–4221. DOI: 10.1016/j.asr.2021.02.031.

[130] Pan, B., Lu, P., Pan, X., and Ma, Y., “Double-Homotopy Method for Solving Optimal Control Problems”, *Journal of Guidance, Control, and Dynamics*, Vol. 39, No. 8, 2016, pp. 1706–1720. DOI: 10.2514/1.G001553.

[131] Jiang, F., Baoyin, H., and Li, J., “Practical Techniques for Low-Thrust Trajectory Optimization with Homotopic Approach”, *Journal of Guidance, Control, and Dynamics*, Vol. 35, No. 1, 2012, pp. 245–258. DOI: 10.2514/1.52476.

[132] Morelli, A. C., Hofmann, C., and Topputo, F., “Robust Low-Thrust Trajectory Optimization Using Convex Programming and a Homotopic Approach”, *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 58, No. 3, 2021, pp. 2103–2116. DOI: 10.1109/TAES.2021.3128869.

[133] Taheri, E., Atkins, E. M., and Kolmanovsky, I., “Performance Comparison of Smoothing Functions for Indirect Optimization of Minimum-Fuel Low-thrust Trajectories”, *2018 Space Flight Mechanics Meeting*, 2018. DOI: 10.2514/6.2018-0214.

[134] Malyuta, D., and Açıkmeşe, B., “Fast Homotopy for Spacecraft Rendezvous Trajectory Optimization with Discrete Logic”, 2021. Preprint, `https://arxiv.org/abs/2107.07001`.

[135] Jia, F., Qiao, D., Han, H., and Li, X., “Efficient Optimization Method for Variable-Specific-Impulse Low-Thrust Trajectories With Shutdown Constraint”, *Science China Technological Sciences*, Vol. 65, No. 3, 2022, p. 581–594. DOI: 10.1007/s11431-021-1949-0.

[136] Carson, J. M., and Açıkmeşe, B., “A Model Predictive Control Technique with Guaranteed Resolvability and Required Thruster Silent Times for Small-Body Proximity Operations”, *AIAA Guidance, Navigation and Control Conference and Exhibit*, 2006. DOI: 10.2514/6.2006-6780.

[137] Mannocchi, A., Giordano, C., and Topputo, F., “A Homotopic Direct Collocation Approach for Operational-Compliant Trajectory Design”, *The Journal of the Astronautical Sciences*, 2022. DOI: 10.1007/s40295-022-00351-x.

[138] Liu, F., Hager, W. W., and Rao, A. V., "Adaptive Mesh Refinement Method for Optimal Control Using Decay Rates of Legendre Polynomial Coefficients", *IEEE Transactions on Control Systems Technology*, Vol. 26, No. 4, 2018, pp. 1475–1483. DOI: 10.1109/TCST.2017.2702122.

[139] Zhao, J., and Li, S., "Mars Atmospheric Entry Trajectory Optimization With Maximum Parachute Deployment Altitude Using Adaptive Mesh Refinement", *Acta Astronautica*, Vol. 160, 2019, pp. 401–413. DOI: 10.1016/j.actaastro.2019.03.027.

[140] Zhou, X., He, R.-Z., Zhang, H.-B., Tang, G.-J., and Bao, W.-W., "Sequential Convex Programming Method Using Adaptive Mesh Refinement for Entry Trajectory Planning Problem", *Aerospace Science and Technology*, Vol. 109, 2021, p. 106374. DOI: 10.1016/j.ast.2020.106374.

[141] Eide, J. D., Hager, W. W., and Rao, A. V., "Modified Legendre–Gauss–Radau Collocation Method for Optimal Control Problems with Nonsmooth Solutions", *Journal of Optimization Theory and Applications*, Vol. 191, No. 2, 2021, pp. 600–633. DOI: 10.1007/s10957-021-01810-5.

[142] Agamawi, Y. M., Hager, W. W., and Rao, A. V., "Mesh Refinement Method for Optimal Control Problems with Discontinuous Control Profiles", *AIAA Guidance, Navigation, and Control Conference*, 2017. DOI: 10.2514/6.2017-1506.

[143] Agamawi, Y. M., Hager, W. W., and Rao, A. V., "Mesh Refinement Method for Solving Bang-Bang Optimal Control Problems using Direct Collocation", *AIAA SciTech Forum*, 2018. DOI: 10.2514/6.2020-0378.

[144] Junkins, J. L., and Taheri, E., "Exploration of Alternative State Vector Choices for Low-Thrust Trajectory Optimization", *Journal of Guidance, Control, and Dynamics*, Vol. 42, No. 1, 2019, pp. 47–64. DOI: 10.2514/1.G003686.

[145] Lara, M., San-Juan, J. F., and López-Ochoa, L. M., "Delaunay Variables Approach to the Elimination of the Perigee in Artificial Satellite Theory", *Celestial Mechanics and Dynamical Astronomy*, Vol. 120, No. 1, 2014, pp. 39–56. DOI: 10.1007/s10569-014-9559-2.

[146] Vallado, D. A., *Fundamentals of Astrodynamics and Applications*, 4th ed., Microcosm Press, 2013. pp. 95–111.

[147] Broucke, R. A., and Cefola, P. J., "On the Equinoctial Orbit Elements", *Celestial Mechanics*, Vol. 5, No. 3, 1972, pp. 303–310. DOI: 10.1007/BF01228432.

[148] Betts, J. T., "Optimal Low–Thrust Orbit Transfers With Eclipsing", *Optimal Control Applications and Methods*, Vol. 36, No. 2, 2015, pp. 218–240. DOI: 10.1002/oca.2111.

[149] Roa, J., and Kasdin, N. J., "Alternative Set of Nonsingular Quaternionic Orbital Elements", *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 11, 2017, pp. 2737–2751. DOI: 10.2514/1.G002753.

[150] Sreesawet, S., and Dutta, A., "Fast and Robust Computation of Low-Thrust Orbit-Raising Trajectories", *Journal of Guidance, Control, and Dynamics*, Vol. 41, No. 9, 2018, pp. 1888–1905. DOI: 10.2514/1.G003319.

[151] Gondelach, D. J., and Armellin, R., “Element Sets for High-Order Poincaré Mapping of Perturbed Keplerian Motion”, *Celestial Mechanics and Dynamical Astronomy*, Vol. 130, No. 10, 2018, p. 65. DOI: 10.1007/s10569-018-9859-z.

[152] Nurre, N. P., and Taheri, E., “Comparison of Indirect and Convex-Based Methods for Low-Thrust Minimum-Fuel Trajectory Optimization”, *AAS/AIAA Astrodynamics Specialist Conference*, 2022. Paper AAS 22-782.

[153] Junkins, J. L., and Singla, P., “How Nonlinear is it? A Tutorial on Nonlinearity of Orbit and Attitude Dynamics”, *The Journal of the Astronautical Sciences*, Vol. 52, No. 1, 2004, pp. 7–60. DOI: 10.1007/BF03546420.

[154] Omran, A., and Newman, B., “Nonlinearity Index Theory for Aircraft Dynamic Assessment”, *Journal of Guidance, Control, and Dynamics*, Vol. 36, No. 1, 2013, pp. 293–303. DOI: 10.2514/1.53906.

[155] Kelly, P., Arya, V., Junkins, J. L., and Majii, M., “Nonlinearity Index for State-Costate Dynamics of Optimal Control Problems”, *AAS/AIAA Astrodynamics Specialist Conference*, 2022. Paper AAS 22-830.

[156] Fossà, A., Armellin, R., Delande, E., Losacco, M., and Sanfedino, F., “Multifidelity Orbit Uncertainty Propagation using Taylor Polynomials”, *AIAA SciTech Forum*, 2022. DOI: 10.2514/6.2022-0859.

[157] Malyuta, D., Yu, Y., Elango, P., and Açıkmeşe, B., “Advances in Trajectory Optimization for Space Vehicle Control”, *Annual Reviews in Control*, Vol. 52, 2021, pp. 282–315. DOI: 10.1016/j.arcontrol.2021.04.013.

[158] Asada, H. H., and Sotiropoulos, F. E., “Dual Faceted Linearization of Nonlinear Dynamical Systems Based on Physical Modeling Theory”, *Journal of Dynamic Systems, Measurement, and Control*, Vol. 141, No. 2, 2018. DOI: 10.1115/1.4041448.

[159] Brunton, S. L., Budišić, M., Kaiser, E., and Kutz, J. N., “Modern Koopman Theory for Dynamical Systems”, 2021. Preprint, `https://arxiv.org/abs/1804.06539`.

[160] Budišić, M., Mohr, R., and Mezić, I., “Applied Koopmanism”, *Chaos: An Interdisciplinary Journal of Nonlinear Science*, Vol. 22, No. 4, 2012, p. 047510. DOI: 10.1063/1.4772195.

[161] Otto, S. E., and Rowley, C. W., “Koopman Operators for Estimation and Control of Dynamical Systems”, *Annual Review of Control, Robotics, and Autonomous Systems*, Vol. 4, No. 1, 2021, pp. 59–87. DOI: 10.1146/annurev-control-071020-010108.

[162] Castaño, M. L., Hess, A., Mamakoukas, G., Gao, T., Murphey, T., and Tan, X., “Control-Oriented Modeling of Soft Robotic Swimmer with Koopman Operators”, *2020 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, 2020, pp. 1679–1685. DOI: 10.1109/AIM43001.2020.9159033.

[163] Mezić, I., “Analysis of Fluid Flows via Spectral Properties of the Koopman Operator”, *Annual Review of Fluid Mechanics*, Vol. 45, No. 1, 2013, pp. 357–378. DOI: 10.1146/annurev-fluid-011212-140652.

[164] Servadio, S., Arnas, D., and Linares, R., “Dynamics Near the Three-Body Libration Points via Koopman Operator Theory”, *Journal of Guidance, Control, and Dynamics*, Vol. 45, No. 10, 2022, pp. 1800–1814. DOI: 10.2514/1.G006519.

[165] Arnas, D., and Linares, R., "Approximate Analytical Solution to the Zonal Harmonics Problem Using Koopman Operator Theory", *Journal of Guidance, Control, and Dynamics*, Vol. 44, No. 11, 2021, pp. 1909–1923. DOI: 10.2514/1.G005864.

[166] Chen, T., and Shan, J., "Koopman-Operator-Based Attitude Dynamics and Control on SO(3)", *Journal of Guidance, Control, and Dynamics*, Vol. 43, No. 11, 2020, pp. 2112–2126. DOI: 10.2514/1.G005006.

[167] Linares, R., "Koopman Operator Theory Applied to the Motion of Satellites", *AAS/AIAA Astrodynamics Specialist Conference*, 2019. Paper AAS 19-821.

[168] Jenson, E. L., Bando, M., Sato, K., and Scheeres, D. J., "Robust Nonlinear Optimal Control Using Koopman Operator Theory", *2022 AAS/AIAA Astrodynamics Specialist Conference*, 2022. Paper AAS 22-647.

[169] Hofmann, C., and Topputo, F., "Toward On-Board Guidance of Low-Thrust Spacecraft in Deep Space Using Sequential Convex Programming", *Proceedings of AAS/AIAA Space Flight Mechanics Meeting*, 2021, pp. 1–19. Paper AAS 21-350.

[170] Malyuta, D., Reynolds, T. P., Szmuk, M., Lew, T., Bonalli, R., Pavone, M., and Açıkmeşe, B., "Convex Optimization for Trajectory Generation: A Tutorial on Generating Dynamically Feasible Trajectories Reliably and Efficiently", *IEEE Control Systems Magazine*, Vol. 42, No. 5, 2022, pp. 40–113. DOI: 10.1109/MCS.2022.3187542.

[171] Liu, X., Shen, Z., and Lu, P., "Exact Convex Relaxation for Optimal Flight of Aerodynamically Controlled Missiles", *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 52, No. 4, 2016. DOI: 10.1109/TAES.2016.150741.

[172] Szmuk, M., Açıkmeşe, B., and Berning, A. W., "Successive Convexification for Fuel-Optimal Powered Landing with Aerodynamic Drag and Non-Convex Constraints", *AIAA Guidance, Navigation, and Control Conference*, 2017. DOI: 10.2514/6.2016-0378.

[173] Bonalli, R., Cauligi, A., Bylard, A., and Pavone, M., "GuSTO: Guaranteed Sequential Trajectory optimization via Sequential Convex Programming", *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 6741–6747. DOI: 10.1109/ICRA.2019.8794205.

[174] Sagliano, M., Theil, S., Bergsma, M., D'Onofrio, V., Whittle, L., and Viavattene, G., "On the Radau pseudospectral Method: Theoretical and Implementation Advances", *CEAS Space Journal*, Vol. 9, 2017. DOI: 10.1007/s12567-017-0165-5.

[175] Patterson, M. A., and Rao, A. V., "GPOPS-II: A MATLAB Software for Solving Multiple-Phase Optimal Control Problems using Hp-Adaptive Gaussian Quadrature Collocation Methods and Sparse Nonlinear Programming", *ACM Trans. Math. Softw.*, Vol. 41, No. 1, 2014. DOI: 10.1145/2558904.

[176] Sagliano, M., Theil, S., D'Onofrio, V., and Bergsma, M., "SPARTAN: A Novel Pseudospectral Algorithm for Entry, Descent, and Landing Analysis", *Advances in Aerospace Guidance, Navigation and Control*, Springer International Publishing, 2018, pp. 669–688. DOI: 10.1007/978-3-319-65283-2_36.

[177] Fahroo, F., and Ross, I. M., “Advances in Pseudospectral Methods for Optimal Control”, *AIAA Guidance, Navigation and Control Conference and Exhibit*, 2008. DOI: 10.2514/6.2008-7309.

[178] Abramowitz, M., and Stegun, I. A., *Handbook of Mathematical Functions*, United States Department of Commerce, 1972. pp. 877 – 897.

[179] Berrut, J.-P., and Trefethen, L. N., “Barycentric Lagrange Interpolation”, *SIAM Review*, Vol. 46, No. 3, 2004. DOI: 10.1137/S0036144502417715.

[180] Françolin, C. C., Benson, D. A., Hager, W. W., and Rao, A. V., “Costate Approximation in Optimal Control Using Integral Gaussian Quadrature Orthogonal Collocation Methods”, *Optimal Control Applications and Methods*, Vol. 36, No. 4, 2015, pp. 381–397. DOI: 10.1002/oca.2112.

[181] Garg, D., Hager, W. W., and Rao, A. V., “Pseudospectral Methods for Solving Infinite-Horizon Optimal Control Problems”, *Automatica*, Vol. 47, No. 4, 2011, pp. 829–837. DOI: 10.1016/j.automatica.2011.01.085.

[182] Rugh, W. J., *Linear System Theory*, 2$^{\text{nd}}$ ed., Prentice-Hall, 1996. pp. 58–73.

[183] Szmuk, M., Reynolds, T. P., and Açıkmeşe, B., “Successive Convexification for Real-Time Six-Degree-of-Freedom Powered Descent Guidance with State-Triggered Constraints”, *Journal of Guidance, Control, and Dynamics*, Vol. 43, No. 8, 2020, pp. 1399–1413. DOI: 10.2514/1.G004549.

[184] Gill, P. E., Murray, W., and Saunders, M. A., “SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization”, *SIAM Journal on Optimization*, Vol. 12, No. 4, 2002, pp. 979–1006. DOI: 10.1137/S1052623499350013.

[185] Morelli, A. C., Merisio, G., Hofmann, C., and Topputo, F., “A Convex Guidance Approach to Target Ballistic Capture Corridors at Mars”, *AAS Guidance, Navigation and Control Conference*, 2022. Paper AAS 22-083.

[186] Yakimenko, O., Yakimenko, O., and Romano, M., “Real-Time 6DoF Guidance For of Spacecraft Proximity Maneuvering and Close Approach with a Tumbling Object”, *AIAA/AAS Astrodynamics Specialist Conference*, 2010. DOI: 10.2514/6.2010-7666.

[187] Zhou, H., Wang, X., Bai, Y., and Cui, N., “Ascent Phase Trajectory Optimization for Vehicle With Multi-Combined Cycle Engine Based on Improved Particle Swarm Optimization”, *Acta Astronautica*, Vol. 140, 2017, pp. 156–165. DOI: 10.1016/j.actaastro.2017.08.024.

[188] Massari, M., Lizia, P. D., Cavenago, F., and Wittig, A., “Differential Algebra Software Library With Automatic Code Generation for Space Embedded Applications”, *2018 AIAA Information Systems-AIAA Infotech Aerospace*, 2018. DOI: 10.2514/6.2018-0398.

[189] Stiefel, E. L., and Scheifele, G., *Linear And Regular Celestial Mechanics*, Springer Berlin, 1971. pp. 19–35.

[190] Battin, R., *An Introduction to the Mathematics and Methods of Astrodynamics*, AIAA Education Series, American Institute of Aeronautics & Astronautics, 1999. pp. 471–515.

[191] Lubey, D. P., and Scheeres, D. J., “Identifying and Estimating Mismodeled Dynamics via Optimal Control Policies and Distance Metrics”, *Journal of Guidance, Control, and Dynamics*, Vol. 37, No. 5, 2014, pp. 1512–1523. DOI: 10.2514/1.G000369.

[192] Acton, C., Bachman, N., Semenov, B., and Wright, E., “A Look Towards the Future in the Handling of Space Science Mission Geometry”, *Planetary and Space Science*, Vol. 150, 2018, pp. 9–12. DOI: 10.1016/j.pss.2017.02.013.

[193] Taheri, E., and Junkins, J. L., “Generic Smoothing for Optimal Bang-Off-Bang Spacecraft Maneuvers”, *Journal of Guidance, Control, and Dynamics*, Vol. 41, No. 11, 2018, pp. 2470–2475. DOI: 10.2514/1.G003604.

[194] Ltd, R. P., “Raspberry Pi 3 Model B+”, URL `https://www.raspberrypi.com/products/raspberry-pi-3-model-b-plus/`, [Online; accessed January 26, 2023].

[195] Dei Tos, D. A., Rasotto, M., Renk, F., and Topputo, F., “LISA Pathfinder Mission Extension: A feasibility Analysis”, *Advances in Space Research*, Vol. 63, No. 12, 2019, pp. 3863–3883. DOI: 10.1016/j.asr.2019.02.035.

[196] Tapley, B. D., Schutz, B. E., and Born, G. H., *Statistical Orbit Determination*, Elsevier Academic Press, 2004. pp. 230–233.

[197] Holzmann, G., “The power of 10: rules for developing safety-critical code”, *Computer*, Vol. 39, No. 6, 2006, pp. 95–99. DOI: 10.1109/MC.2006.212.

[198] Hughes, S., Jun, L., and Shoan, W., “C++ Coding Standards and Style Guide”, Tech. Rep. 20080039927, NASA, 2005.

[199] Guennebaud, G., Jacob, B., et al., “Eigen v3”, 2010. URL `http://eigen.tuxfamily.org`.

[200] Arnas, D., “Solving Perturbed Dynamic Systems Using Schur Decomposition”, *Journal of Guidance, Control, and Dynamics*, Vol. 45, No. 12, 2022, pp. 2211–2228. DOI: 10.2514/1.G006726.

[201] Goswami, D., and Paley, D. A., “Global Bilinearization and Controllability of Control-Affine Nonlinear Systems: A Koopman Spectral Approach”, *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, 2017, pp. 6107–6112. DOI: 10.1109/CDC.2017.8264582.

[202] Korda, M., and Mezić, I., “Linear Predictors for Nonlinear Dynamical Systems: Koopman Operator Meets Model Predictive Control”, *Automatica*, Vol. 93, 2018, pp. 149–160. DOI: 10.1016/j.automatica.2018.03.046.

[203] Heiss, F., and Winschel, V., “Likelihood Approximation by Numerical Integration on Sparse Grids”, *Journal of Econometrics*, Vol. 144, No. 1, 2008, pp. 62–80. DOI: 10.1016/j.jeconom.2007.12.004.

A Appendix: Shape-Based Method for Initial Guess Generation

Based on [76], the initial guess is generated with a shape-based method. Assuming cylindrical coordinates r (radial distance), θ (azimuth) and z (axial coordinate), each component of the state is approximated by a cubic polynomial:

$$\mathbf{r}(t) = \mathbf{a}\,t^3 + \mathbf{b}\,t^2 + \mathbf{c}\,t + \mathbf{d} \tag{A.1}$$

$$\mathbf{v}(t) = 3\,\mathbf{a}\,t^2 + 2\,\mathbf{b}t + \mathbf{c}\,t \tag{A.2}$$

The vectors of coefficients $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$, and $\mathbf{d}$ are calculated using the known boundary conditions to obtain:

$$\mathbf{d} = \mathbf{r}_0 \tag{A.3}$$

$$\mathbf{c} = \mathbf{v}_0 \tag{A.4}$$

$$\mathbf{a} = 2\,\frac{(\mathbf{v}_f - \mathbf{v}_0)\frac{t_f}{2} + \mathbf{v}_0\,t_f + \mathbf{r}_0 - \mathbf{r}_f}{t_f^3} \tag{A.5}$$

$$\mathbf{b} = \frac{\mathbf{v}_f - \mathbf{v}_0 - 3\,\mathbf{a}\,t_f^2}{2\,t_f} \tag{A.6}$$

with $\mathbf{r} = [r, \theta, z]^\top$ and $\mathbf{v} = \dot{\mathbf{r}}$. The subscripts 0 and f refer to the corresponding initial and target values, respectively. Different numbers of revolutions are incorporated by simply adding multiples of 2π to the azimuthal component.

B Appendix: Coordinate Transformations

B.1 Hamiltonian in Cartesian and Spherical Coordinates

Given the kinetic energy $T = \frac{1}{2}\, m\, \mathbf{v}^\top \mathbf{v}$ and the potential energy $V = -\mu\, m/r$ of a satellite about the primary body, the normalized Hamiltonian function $\mathcal{H}$ reads in Cartesian coordinates

$$\mathcal{H} = T + V = \frac{\mathbf{v}^\top \mathbf{v}}{2} - \frac{\mu}{r} \tag{B.1}$$

As the velocity can be expressed as (see also Eq. (C.10))

$$\mathbf{v} = \dot{r}\, \mathbf{i}_r + r\, \dot{\theta}\, \cos\phi\, \mathbf{i}_\theta + r\, \dot{\phi}\, \mathbf{i}_\phi \tag{B.2}$$

the Hamiltonian in spherical coordinates is given by

$$\mathcal{H} = \frac{1}{2}\left(\dot{r}^2 + r^2\, \dot{\theta}^2\, \cos^2\phi + r^2\, \dot{\phi}^2\right) - \frac{\mu}{r} \tag{B.3}$$

B.2 Classical Orbital Elements

The relationship between classical orbital elements and Cartesian coordinates is given by [190]:

$$a = \left(\frac{2}{r} - \frac{\mathbf{v}^\top \mathbf{v}}{\mu}\right)^{-1} \tag{B.4a}$$

$$\mathbf{h} = \mathbf{r} \times \mathbf{v} \tag{B.4b}$$

$$p = \frac{\|\mathbf{h}\|^2}{\mu} \tag{B.4c}$$

$$e = \left\| \frac{\mathbf{v} \times \mathbf{h}}{\mu} - \frac{\mathbf{r}}{r} \right\| \tag{B.4d}$$

B.3 Spherical Coordinates

The transformation from Cartesian (x, y, z) to spherical (r, θ, ϕ) coordinates is:

$$r = \sqrt{x^2 + y^2 + z^2} \tag{B.5a}$$

$$\theta = \arctan \frac{y}{x} \tag{B.5b}$$

$$\phi = \arcsin \frac{z}{\sqrt{x^2 + y^2 + z^2}} \tag{B.5c}$$

The inverse transformation is:

$$x = r \, \cos\theta \, \cos\phi \tag{B.6a}$$

$$y = r \, \sin\theta \, \cos\phi \tag{B.6b}$$

$$z = r \, \sin\phi \tag{B.6c}$$

The velocities are transformed using the rotation matrix in Eq. (7.50).

B.4 Cylindrical Coordinates

The transformation from Cartesian (x, y, z) to cylindrical (ρ, θ, z) coordinates is:

$$\rho = \sqrt{x^2 + y^2} \tag{B.7a}$$

$$\theta = \arctan \frac{y}{x} \tag{B.7b}$$

$$z = z \tag{B.7c}$$

Transforming cylindrical to Cartesian coordinates is done using the following equations:

$$x = \rho \, \cos\theta \tag{B.8a}$$

$$y = \rho \, \sin\theta \tag{B.8b}$$

$$z = z \tag{B.8c}$$

Velocities are transformed using the rotation matrix about the third axis by θ:

$$\begin{bmatrix} v_\rho \\ v_\theta \\ v_z \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} \tag{B.9}$$

B.5 Modified Equinoctial Elements

The relationship between MEE and Cartesian coordinates is given by [148]:

$$\mathbf{r} = \begin{bmatrix} \frac{p}{s^2\,\sigma}\,\big(\cos l + \Xi^2\,\cos l + 2\,h_x\,h_y\,\sin l\big) \\ \frac{p}{s^2\,\sigma}\,\big(\sin l - \Xi^2\,\sin l + 2\,h_x\,h_y\,\cos l\big) \\ \frac{2\,p}{s^2\,\sigma}\,(h_x\,\sin l - h_y\,\cos l) \end{bmatrix} \tag{B.10a}$$

$$\mathbf{v} = \begin{bmatrix} -\frac{1}{s^2}\sqrt{\frac{\mu}{p}}\,\big(\sin l + \Xi^2\,\sin l - 2\,h_x\,h_y\,\cos l + e_y - 2\,e_x\,h_x\,h_y + \Xi^2\,e_y\big) \\ -\frac{1}{s^2}\sqrt{\frac{\mu}{p}}\,\big(-\cos l + \Xi^2\,\cos l + 2\,h_x\,h_y\,\sin l - e_x + 2\,e_y\,h_x\,h_y + \Xi^2\,e_x\big) \\ \frac{2}{s^2}\sqrt{\frac{\mu}{p}}\,(h_x\,\cos l + h_y\,\sin l + e_x\,h_x + e_y\,h_y) \end{bmatrix} \tag{B.10b}$$

with $\Xi^2 = h_x^2 - h_y^2$. MEE can be transformed to classical orbital elements as follows:

$$a = \frac{p}{1 - e_x^2 - e_y^2} \tag{B.11a}$$

$$e = \sqrt{e_x^2 + e_y^2} \tag{B.11b}$$

$$i = 2\,\arctan\sqrt{h_x^2 + h_y^2} \tag{B.11c}$$

$$\Omega = \arctan\frac{h_y}{h_x} \tag{B.11d}$$

$$\omega = \arctan\frac{e_y}{e_x} - \arctan\frac{h_y}{h_x} \tag{B.11e}$$

$$\vartheta = l - \arctan\frac{e_y}{e_x} \tag{B.11f}$$

C Appendix: Derivation of Equations of Motion

C.1 Spherical Coordinates

The equation of motion can be written as

$$\ddot{\mathbf{r}} + \frac{\mu}{r^2}\,\mathbf{i}_r = \mathbf{a}_p \tag{C.1}$$

where

$$\mathbf{a}_p = a_r\,\mathbf{i}_r + a_\theta\,\mathbf{i}_\theta + a_\phi\,\mathbf{i}_\phi \tag{C.2}$$

The goal is to express $\ddot{\mathbf{r}}$ in terms of the spherical unit vectors $\mathbf{i}_r$, $\mathbf{i}_\theta$, and $\mathbf{i}_\phi$, where

$$\mathbf{i}_r = \begin{bmatrix} \cos\theta\,\cos\phi \\ \sin\theta\,\cos\phi \\ \sin\phi \end{bmatrix}, \quad \mathbf{i}_\theta = \begin{bmatrix} -\sin\theta \\ \cos\theta \\ 0 \end{bmatrix}, \quad \mathbf{i}_\phi = \begin{bmatrix} -\cos\theta\,\sin\phi \\ -\sin\theta\,\sin\phi \\ \cos\phi \end{bmatrix} \tag{C.3}$$

Given the position in spherical coordinates

$$\mathbf{r} = r\,\mathbf{i}_r \tag{C.4}$$

the velocity and acceleration can be expressed as

$$\mathbf{v} = \dot{\mathbf{r}} = \dot{r}\,\mathbf{i}_r + r\,\dot{\mathbf{i}}_r \tag{C.5}$$

$$\dot{\mathbf{v}} = \ddot{\mathbf{r}} = \ddot{r}\,\mathbf{i}_r + 2\,\dot{r}\,\dot{\mathbf{i}}_r + r\,\ddot{\mathbf{i}}_r \tag{C.6}$$

Hence, the expressions for $\dot{\mathbf{i}}_r$ and $\ddot{\mathbf{i}}_r$ are needed. The time derivative of the rotation matrix $\mathbf{R}_{\text{XYZ}\to\text{SPH}}$ in Eq. (7.50) is

$$\begin{bmatrix} \dot{\mathbf{i}}_r \\ \dot{\mathbf{i}}_\theta \\ \dot{\mathbf{i}}_\phi \end{bmatrix} = \underbrace{\begin{bmatrix} -\dot{\theta}\,\sin\theta\,\cos\phi - \dot{\phi}\,\cos\theta\,\sin\phi & \dot{\theta}\cos\theta\,\cos\phi - \dot{\phi}\,\sin\theta\,\sin\phi & \dot{\phi}\,\cos\phi \\ -\dot{\theta}\cos\theta & -\dot{\theta}\sin\theta & 0 \\ \dot{\theta}\,\sin\theta\,\sin\phi - \dot{\phi}\,\cos\theta\,\cos\phi & -\dot{\theta}\cos\theta\,\sin\phi - \dot{\phi}\,\sin\theta\,\cos\phi & -\dot{\phi}\,\sin\phi \end{bmatrix}}_{=:\,\dot{\mathbf{R}}_{\text{XYZ}\to\text{SPH}}} \begin{bmatrix} \mathbf{i}_x \\ \mathbf{i}_y \\ \mathbf{i}_z \end{bmatrix} \tag{C.7}$$

Therefore,

$$\begin{bmatrix} \dot{\mathbf{i}}_r \\ \dot{\mathbf{i}}_\theta \\ \dot{\mathbf{i}}_\phi \end{bmatrix} = \dot{\mathbf{R}}_{\mathrm{XYZ}\to\mathrm{SPH}}\,\mathbf{R}^\top_{\mathrm{XYZ}\to\mathrm{SPH}} \begin{bmatrix} \mathbf{i}_r \\ \mathbf{i}_\theta \\ \mathbf{i}_\phi \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & \dot\theta\,\cos\phi & \dot\phi \\ -\dot\theta\,\cos\phi & 0 & \dot\theta\,\sin\phi \\ -\dot\phi & -\dot\theta\,\sin\phi & 0 \end{bmatrix}}_{=:\,\dot{\mathbf{R}}_{\mathrm{SPH}}} \begin{bmatrix} \mathbf{i}_r \\ \mathbf{i}_\theta \\ \mathbf{i}_\phi \end{bmatrix} \tag{C.8}$$

Taking the derivative of Eq. (C.8) with respect to time gives

$$\begin{aligned} \begin{bmatrix} \ddot{\mathbf{i}}_r \\ \ddot{\mathbf{i}}_\theta \\ \ddot{\mathbf{i}}_\phi \end{bmatrix} &= \ddot{\mathbf{R}}_{\mathrm{SPH}} \begin{bmatrix} \mathbf{i}_r \\ \mathbf{i}_\theta \\ \mathbf{i}_\phi \end{bmatrix} + \dot{\mathbf{R}}_{\mathrm{SPH}} \begin{bmatrix} \dot{\mathbf{i}}_r \\ \dot{\mathbf{i}}_\theta \\ \dot{\mathbf{i}}_\phi \end{bmatrix} \\ &= \begin{bmatrix} -\dot\phi^2 - \dot\theta^2\,\cos^2\phi & \ddot\theta\,\cos\phi - 2\,\dot\theta\,\dot\phi\sin\phi & \frac{1}{2}\,\dot\theta^2\,\sin 2\phi + \ddot\phi \\ -\ddot\theta\,\cos\phi & -\dot\theta^2 & \ddot\theta\,\sin\phi \\ \frac{1}{2}\,\dot\theta^2\sin 2\phi - \ddot\phi & -\ddot\theta\,\sin\phi - 2\,\dot\theta\,\dot\phi\,\cos\phi & -\dot\phi^2 - \dot\theta^2\,\sin^2\phi \end{bmatrix} \begin{bmatrix} \mathbf{i}_r \\ \mathbf{i}_\theta \\ \mathbf{i}_\phi \end{bmatrix} \end{aligned} \tag{C.9}$$

Using Eq. (C.5), the velocity can be expressed as

$$\mathbf{v} = \dot r\,\mathbf{i}_r + r\,\dot\theta\,\cos\phi\,\mathbf{i}_\theta + r\,\dot\phi\,\mathbf{i}_\phi \tag{C.10}$$

Therefore,

$$v_r = \dot r \tag{C.11}$$

$$v_\theta = r\,\dot\theta\,\cos\phi \tag{C.12}$$

$$v_\phi = r\,\dot\phi \tag{C.13}$$

It follows that

$$\dot v_r = \ddot r \tag{C.14}$$

$$\dot v_\theta = \dot r\,\dot\theta\,\cos\phi + r\,\ddot\theta\,\cos\phi - r\,\dot\theta\,\dot\phi\sin\phi \tag{C.15}$$

$$\dot v_\phi = \dot r\,\dot\phi + r\,\ddot\phi \tag{C.16}$$

Using Eqs. (C.6), (C.8) and (C.9), the acceleration can be written as follows:

$$\begin{aligned} \ddot{\mathbf{r}} = &\left(\ddot r - r\,\dot\phi^2 - r\,\dot\theta^2\cos^2\phi\right)\,\mathbf{i}_r + \left(2\,\dot r\,\dot\theta\,\cos\phi + r\,\ddot\theta\,\cos\phi - 2\,r\,\dot\theta\,\dot\phi\,\sin\phi\right)\,\mathbf{i}_\theta \\ &+ \left(2\,\dot r\,\dot\phi + \frac{1}{2}\,r\,\dot\theta^2\,\sin 2\phi + r\,\ddot\phi\right)\,\mathbf{i}_\phi \end{aligned} \tag{C.17}$$

Substituting the expressions for $\dot{r}$, $\dot{\theta}$, $\dot{\phi}$, $\ddot{r}$, $\ddot{\theta}$, $\ddot{\phi}$ of Eqs. (C.11)–(C.16) into Eq. (C.17) and using Eq. (C.1), the equations of motion can be formulated in spherical coordinates:

$$\dot{r} = v_r \tag{C.18a}$$

$$\dot{\theta} = \frac{v_\theta}{r\,\cos\phi} \tag{C.18b}$$

$$\dot{\phi} = \frac{v_\phi}{r} \tag{C.18c}$$

$$\dot{v}_r = \frac{v_\theta^2 + v_\phi^2}{r} - \frac{\mu}{r^2} + a_r \tag{C.18d}$$

$$\dot{v}_\theta = \frac{v_\theta\, v_\phi}{r}\,\tan\phi - \frac{v_r\, v_\theta}{r} + a_\theta \tag{C.18e}$$

$$\dot{v}_\phi = -\frac{v_r\, v_\phi}{r} - \frac{v_\theta^2}{r}\,\tan\phi + a_\phi \tag{C.18f}$$

C.2 Cylindrical Coordinates

The equation of motion in cylindrical coordinates reads

$$\ddot{\mathbf{r}} + \frac{\mu}{r^3}\,(\rho\,\mathbf{i}_\rho + z\,\mathbf{i}_z) = \mathbf{a}_p \tag{C.19}$$

where the position in cylindrical coordinates is given by

$$\mathbf{r} = \rho\,\mathbf{i}_\rho + z\,\mathbf{i}_z \tag{C.20}$$

The perturbing acceleration $\mathbf{a}_p$ is

$$\mathbf{a}_p = a_\rho\,\mathbf{i}_\rho + a_\theta\,\mathbf{i}_\theta + a_z\,\mathbf{i}_z \tag{C.21}$$

The first and second derivative of the position vector are:

$$\dot{\mathbf{r}} = \mathbf{v} = \dot{\rho}\,\mathbf{i}_\rho + \rho\,\dot{\mathbf{i}}_\rho + \dot{z}\,\mathbf{i}_z + z\,\dot{\mathbf{i}}_z \tag{C.22}$$

$$\ddot{\mathbf{r}} = \dot{\mathbf{v}} = \ddot{\rho}\,\mathbf{i}_\rho + 2\,\dot{\rho}\,\dot{\mathbf{i}}_\rho + \rho\,\ddot{\mathbf{i}}_\rho + \ddot{z}\,\mathbf{i}_z + 2\,\dot{z}\,\dot{\mathbf{i}}_z + z\,\ddot{\mathbf{i}}_z \tag{C.23}$$

The rotation matrix $\mathbf{R}_{\text{XYZ}\to\text{SPH}}$ to transform Cartesian to cylindrical coordinates is given by a rotation about the third axis by θ:

$$\begin{bmatrix}\mathbf{i}_\rho \\ \mathbf{i}_\theta \\ \mathbf{i}_z\end{bmatrix} = \underbrace{\begin{bmatrix}\cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1\end{bmatrix}}_{=:\,\mathbf{R}_{\text{XYZ}\to\text{CYL}}} \begin{bmatrix}\mathbf{i}_x \\ \mathbf{i}_y \\ \mathbf{i}_z\end{bmatrix} \tag{C.24}$$

Using its derivative

$$\begin{bmatrix} \dot{\mathbf{i}}_\rho \\ \dot{\mathbf{i}}_\theta \\ \dot{\mathbf{i}}_z \end{bmatrix} = \underbrace{\begin{bmatrix} -\dot\theta\,\sin\theta & \dot\theta\,\cos\theta & 0 \\ -\dot\theta\,\cos\theta & -\dot\theta\,\sin\theta & 0 \\ 0 & 0 & 0 \end{bmatrix}}_{=:\,\dot{\mathbf{R}}_{\text{XYZ}\to\text{CYL}}} \begin{bmatrix} \mathbf{i}_x \\ \mathbf{i}_y \\ \mathbf{i}_z \end{bmatrix} \tag{C.25}$$

the first derivative of the unit vectors can be expressed in terms of the unit vectors themselves:

$$\begin{bmatrix} \dot{\mathbf{i}}_\rho \\ \dot{\mathbf{i}}_\theta \\ \dot{\mathbf{i}}_z \end{bmatrix} = \dot{\mathbf{R}}_{\text{XYZ}\to\text{CYL}}\,\mathbf{R}^\top_{\text{XYZ}\to\text{CYL}} \begin{bmatrix} \mathbf{i}_\rho \\ \mathbf{i}_\theta \\ \mathbf{i}_z \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & \dot\theta & 0 \\ -\dot\theta & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}}_{=:\,\dot{\mathbf{R}}_{\text{CYL}}} \begin{bmatrix} \mathbf{i}_\rho \\ \mathbf{i}_\theta \\ \mathbf{i}_z \end{bmatrix} \tag{C.26}$$

The second derivative is computed as

$$\begin{aligned} \begin{bmatrix} \ddot{\mathbf{i}}_\rho \\ \ddot{\mathbf{i}}_\theta \\ \ddot{\mathbf{i}}_z \end{bmatrix} &= \ddot{\mathbf{R}}_{\text{CYL}} \begin{bmatrix} \mathbf{i}_\rho \\ \mathbf{i}_\theta \\ \mathbf{i}_z \end{bmatrix} + \dot{\mathbf{R}}_{\text{CYL}} \begin{bmatrix} \dot{\mathbf{i}}_\rho \\ \dot{\mathbf{i}}_\theta \\ \dot{\mathbf{i}}_z \end{bmatrix} \\ &= \begin{bmatrix} -\dot\theta^2 & \ddot\theta & 0 \\ -\ddot\theta & -\dot\theta^2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{i}_\rho \\ \mathbf{i}_\theta \\ \mathbf{i}_\phi \end{bmatrix} \end{aligned} \tag{C.27}$$

Using Eqs. (C.22) and (C.26), the velocity reads

$$\dot{\mathbf{r}} = \dot\rho\,\mathbf{i}_\rho + \rho\,\dot\theta\,\mathbf{i}_\theta + \dot z\,\mathbf{i}_z \tag{C.28}$$

Therefore,

$$v_\rho = \dot\rho \tag{C.29}$$

$$v_\theta = \rho\,\dot\theta \tag{C.30}$$

$$v_z = \dot z \tag{C.31}$$

and

$$\dot v_\rho = \ddot\rho \tag{C.32}$$

$$\dot v_\theta = \dot\rho\,\dot\theta + \rho\,\ddot\theta \tag{C.33}$$

$$\dot v_z = \ddot z \tag{C.34}$$

The acceleration is then determined using Eqs. (C.23) and (C.27):

$$\ddot{\mathbf{r}} = \left(\ddot{\rho} - \rho\,\dot{\theta}^2\right)\,\mathbf{i}_\rho + \left(2\,\dot{\rho}\,\dot{\theta} + \rho\,\ddot{\theta}\right)\,\mathbf{i}_\theta + \ddot{z}\,\mathbf{i}_z \tag{C.35}$$

The equations of motion are found by substituting $\dot{\rho}$, $\dot{\theta}$, $\dot{z}$, $\ddot{\rho}$, $\ddot{\theta}$, $\ddot{z}$ of Eqs. (C.29)–(C.34) into Eq. (C.35) and comparing coefficients with Eq. (C.19):

$$\dot{\rho} = v_\rho \tag{C.36a}$$

$$\dot{\theta} = \frac{v_\theta}{\rho} \tag{C.36b}$$

$$\dot{z} = v_z \tag{C.36c}$$

$$\dot{v}_\rho = \frac{v_\theta^2}{\rho} - \frac{\mu}{r^3}\,\rho + a_\rho \tag{C.36d}$$

$$\dot{v}_\theta = -\frac{v_\rho\,v_\theta}{\rho} + a_\theta \tag{C.36e}$$

$$\dot{v}_z = -\frac{\mu}{r^3}\,z + a_z \tag{C.36f}$$

C.3 Partial Derivatives for Modified Orbital Elements

We report the partial derivatives of the modified orbital elements with respect to the Cartesian velocity that are required to determine the equations of motion when a perturbing acceleration is present. Recalling the definition of the elements in Eqs. (7.31)–(7.36), the following auxiliary partial derivatives are needed:

$$\frac{\partial r}{\partial \mathbf{v}},\ \frac{\partial \phi}{\partial \mathbf{v}},\ \frac{\partial \dot{r}}{\partial \mathbf{v}},\ \frac{\partial \dot{\theta}}{\partial \mathbf{v}},\ \frac{\partial \dot{\phi}}{\partial \mathbf{v}},\ \frac{\partial p_\theta}{\partial \mathbf{v}},\ \frac{\partial p_\phi}{\partial \mathbf{v}},\ \frac{\partial p_h}{\partial \mathbf{v}} \tag{C.37}$$

The partial derivatives of the position with respect to the velocity are zero due to the independence of state vector components:

$$\frac{\partial r}{\partial \mathbf{v}} = \mathbf{0}^\top \tag{C.38}$$

$$\frac{\partial \theta}{\partial \mathbf{v}} = \mathbf{0}^\top \tag{C.39}$$

$$\frac{\partial \phi}{\partial \mathbf{v}} = \mathbf{0}^\top \tag{C.40}$$

We need to compute the partial derivatives of the spherical elements with respect to the velocity. Given the spherical unit vectors $\mathbf{i}_r$, $\mathbf{i}_\theta$, $\mathbf{i}_\phi$, the position and velocity can be expressed in spherical coordinates as follows:

$$\mathbf{r} = r\,\mathbf{i}_r \tag{C.41}$$

$$\mathbf{v} = \dot{r}\,\mathbf{i}_r + r\,\dot{\theta}\,\cos\phi\,\mathbf{i}_\theta + r\,\dot{\phi}\,\mathbf{i}_\phi \tag{C.42}$$

Taking the partial derivative of Eq. (C.42) with respect to $\mathbf{v}$ yields

$$\begin{aligned}
\frac{\partial \mathbf{v}}{\partial \mathbf{v}} &= \frac{\partial}{\partial \mathbf{v}} \begin{bmatrix} \dot{r} \\ r\,\dot{\theta}\,\cos\phi \\ r\,\dot{\phi} \end{bmatrix} \\
\iff \quad \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} &= \begin{bmatrix} \frac{\partial \dot{r}}{\partial \mathbf{v}} \\ r\,\cos\phi\,\frac{\partial \dot{\theta}}{\partial \mathbf{v}} + \dot{\theta}\,\cos\phi\,\frac{\partial r}{\partial \mathbf{v}} - \dot{\theta}\,r\,\sin\phi\,\frac{\partial \phi}{\partial \mathbf{v}} \\ \dot{\phi}\,\frac{\partial r}{\partial \mathbf{v}} + r\,\frac{\partial \dot{\phi}}{\partial \mathbf{v}} \end{bmatrix} = \begin{bmatrix} \frac{\partial \dot{r}}{\partial \mathbf{v}} \\ r\cos\phi\frac{\partial \dot{\theta}}{\partial \mathbf{v}} \\ r\frac{\partial \dot{\phi}}{\partial \mathbf{v}} \end{bmatrix}
\end{aligned} \tag{C.43}$$

because $\partial r/\partial \mathbf{v} = \partial \phi/\partial \mathbf{v} = \mathbf{0}^\top$ (independence of state vector components). Solving for the partial derivatives results in

$$\frac{\partial \dot{r}}{\partial \mathbf{v}} = \mathbf{i}_r^\top \tag{C.44}$$

$$\frac{\partial \dot{\theta}}{\partial \mathbf{v}} = \frac{1}{r\cos\phi}\,\mathbf{i}_\theta^\top \tag{C.45}$$

$$\frac{\partial \dot{\phi}}{\partial \mathbf{v}} = \frac{1}{r}\,\mathbf{i}_\phi^\top \tag{C.46}$$

Computing the partial derivative of p_h with respect to $\mathbf{v}$ gives

$$\begin{aligned}
\frac{\partial}{\partial \mathbf{v}} p_h^2 &= \frac{\partial}{\partial \mathbf{v}} p_\phi^2 + \frac{p_\theta^2}{\cos^2\phi} \\
\iff \quad 2\,p_h\,\frac{\partial p_h}{\partial \mathbf{v}} &= 2\left(p_\phi\,\frac{\partial p_\phi}{\partial \mathbf{v}} + \frac{p_\theta}{\cos^2\phi}\,\frac{\partial p_\theta}{\partial \mathbf{v}} + p_\theta\,\tan\phi\,\frac{\partial \phi}{\partial \mathbf{v}}\right) \\
\iff \quad \frac{\partial p_h}{\partial \mathbf{v}} &= \frac{p_\phi}{p_h}\frac{\partial p_\phi}{\partial \mathbf{v}} + \frac{p_\theta}{p_h\cos^2\phi}\frac{\partial p_\theta}{\partial \mathbf{v}}
\end{aligned} \tag{C.47}$$

It follows from

$$\begin{aligned}
\frac{\partial p_\theta}{\partial \mathbf{v}} &= \frac{\partial}{\partial \mathbf{v}} r^2\,\dot{\theta}\,\cos^2\phi = r\,\cos\phi\left(2\,\dot{\theta}\,\cos\phi\,\frac{\partial r}{\partial \mathbf{v}} + r\,\cos\phi\,\frac{\partial \dot{\theta}}{\partial \mathbf{v}} - 2\,r\,\dot{\theta}\,\sin\phi\,\frac{\partial \phi}{\partial \mathbf{v}}\right) \\
&= r\,\cos\phi\,\mathbf{i}_\theta^\top
\end{aligned} \tag{C.48}$$

and

$$\begin{aligned}
\frac{\partial p_\phi}{\partial \mathbf{v}} &= \frac{\partial}{\partial \mathbf{v}} r^2\,\dot{\phi} = 2\,r\,\dot{\phi}\,\frac{\partial r}{\partial \mathbf{v}} + r^2\,\frac{\partial \dot{\phi}}{\partial \mathbf{v}} \\
&= r\,\mathbf{i}_\phi^\top
\end{aligned} \tag{C.49}$$

that

$$\frac{\partial p_h}{\partial \mathbf{v}} = \frac{p_\phi}{p_h}\,r\,\mathbf{i}_\phi^\top + \frac{p_\theta}{p_h\cos\phi}\,r\,\mathbf{i}_\theta^\top \tag{C.50}$$

The partial derivatives in Eqs. (C.38)–(C.40), (C.44)–(C.46) and (C.48)–(C.50) are now used to determine the partial derivatives of the modified orbital elements with respect to the velocity:

$$\begin{aligned}\frac{\partial \Lambda}{\partial \mathbf{v}} &= \sqrt{\frac{C}{\mu}}\,\frac{\partial}{\partial \mathbf{v}}\left(\frac{p_h}{r} - \frac{\mu}{p_h}\right) = \sqrt{\frac{C}{\mu}}\left(-\frac{p_h}{r^2}\frac{\partial r}{\partial \mathbf{v}} + \frac{1}{r}\frac{\partial p_h}{\partial \mathbf{v}} + \frac{\mu}{p_h^2}\frac{\partial p_h}{\partial \mathbf{v}}\right) \\ &= \sqrt{\frac{C}{\mu}}\left[\frac{p_\theta\,\left(p_h^2 + r\,\mu\right)}{p_h^3\,\cos\phi}\,\mathbf{i}_\theta^\top + \frac{\gamma}{\cos\phi}\left(1 + \frac{r\,\mu}{p_h^2}\right)\mathbf{i}_\phi^\top\right]\end{aligned} \tag{C.51}$$

$$\begin{aligned}\frac{\partial \eta}{\partial \mathbf{v}} &= \sqrt{\frac{C}{\mu}}\,\frac{\partial p_r}{\partial \mathbf{v}} = \sqrt{\frac{C}{\mu}}\,\frac{\partial \dot{r}}{\partial \mathbf{v}} \\ &= \sqrt{\frac{C}{\mu}}\,\mathbf{i}_r^\top\end{aligned} \tag{C.52}$$

$$\begin{aligned}\frac{\partial s}{\partial \mathbf{v}} &= \sqrt{\frac{C}{\mu}}\,\frac{\partial \sin\phi}{\partial \mathbf{v}} = \sqrt{\frac{C}{\mu}}\,\cos\phi\,\frac{\partial \phi}{\partial \mathbf{v}} \\ &= \mathbf{0}^\top\end{aligned} \tag{C.53}$$

$$\begin{aligned}\frac{\partial \gamma}{\partial \mathbf{v}} &= \frac{\partial}{\partial \mathbf{v}}\left(\frac{p_\phi}{p_h}\cos\phi\right) = \frac{\cos\phi}{p_h}\,\frac{\partial p_h}{\partial \mathbf{v}} + p_\phi\left(-\frac{\cos\phi}{p_h^2}\,\frac{\partial p_h}{\partial \mathbf{v}} - \frac{\sin\phi}{p_h}\,\frac{\partial \phi}{\partial \mathbf{v}}\right) \\ &= -\frac{r\,\gamma\,p_\theta}{p_h^2\,\cos\phi}\,\mathbf{i}_\theta^\top + \frac{r\,\cos\phi - r\,\gamma^2\,\cos^{-1}\phi}{p_h}\,\mathbf{i}_\phi^\top\end{aligned} \tag{C.54}$$

$$\begin{aligned}\frac{\partial \kappa}{\partial \mathbf{v}} &= \sqrt{C\,\mu}\,\frac{\partial}{\partial \mathbf{v}}\frac{1}{p_h} = -\sqrt{C\,\mu}\,\frac{1}{p_h^2}\,\frac{\partial p_h}{\partial \mathbf{v}} \\ &= -\sqrt{C\,\mu}\,\frac{r}{p_h^3}\left(\frac{p_\theta}{\cos\phi}\,\mathbf{i}_\theta^\top + p_\phi\,\mathbf{i}_\phi^\top\right)\end{aligned} \tag{C.55}$$

$$\begin{aligned}\frac{\partial \beta}{\partial \mathbf{v}} &= \frac{\partial}{\partial \mathbf{v}}\left[\theta - \arcsin\left(\tan\phi\sqrt{\frac{p_\theta^2}{p_h^2 - p_\theta^2}}\right)\right] \\ &= \frac{r\,p_\theta\,\left(-p_h^2\,\sin\phi + p_\theta^2\,\tan\phi\,\cos^{-1}\phi\right)}{\sqrt{\frac{p_\theta^2}{p_h^2 - p_\theta^2}}\,\left(p_\theta^2 - p_h^2\right)^2\,\sqrt{\frac{p_\theta^2 - p_h^2 + p_\theta^2\,\tan^2\phi}{p_\theta^2 - p_h^2}}}\,\mathbf{i}_\theta^\top + \frac{r\,p_\theta\,p_\phi\,\tan\phi}{\left(p_h^2 - p_\theta^2\right)\,\sqrt{p_h^2 - p_\theta^2}\sqrt{\frac{p_\theta^2 - p_h^2 + p_\theta^2\,\tan^2\phi}{p_\theta^2 - p_h^2}}}\,\mathbf{i}_\phi^\top\end{aligned} \tag{C.56}$$

D Appendix: Parsing into Standard Form

D.1 Parsing of General Operators and Constraints

The parsing of the relevant operators and constraints into equivalent problems is based on [21, Chapters 4, 6]. If not stated otherwise, $\mathbf{x}$ and $\mathbf{s}$ (or s) denote the decision and slack variables, respectively.

Absolute Value Operator

A constraint of the form $|\mathbf{x} - \mathbf{c}| \leq \mathbf{t}$ can be rewritten as

$$\mathbf{x} \leq \mathbf{t} + \mathbf{c} \tag{D.1a}$$

$$-\mathbf{x} \leq \mathbf{t} - \mathbf{c} \tag{D.1b}$$

1-Norm Operator

Recalling the definition of the 1-norm

$$\|\mathbf{x}\|_1 := \sum_{i=1}^{n} |x_i|, \tag{D.2}$$

the constraint $\|\mathbf{x} - \mathbf{c}\|_1 \leq t$ is rewritten using the epigraph form to obtain

$$|\mathbf{x} - \mathbf{c}| \leq \mathbf{s} \tag{D.3a}$$

$$\mathbf{1}^\top \mathbf{s} = t \tag{D.3b}$$

Using Eq. (D.1) yields

$$\mathbf{x} \leq \mathbf{s} + \mathbf{c} \tag{D.4a}$$

$$-\mathbf{x} \leq \mathbf{s} - \mathbf{c} \tag{D.4b}$$

$$\mathbf{1}^\top \mathbf{s} = t \tag{D.4c}$$

Minimizing the 1-norm, i.e., $\min \mu \,\|\mathbf{x}\|_1$, is equivalent to

$$\begin{aligned}\text{minimize} \quad & \mu \,\mathbf{1}^\top \mathbf{s} && \text{(D.5a)}\\ \text{subject to:} \quad & \mathbf{x} \leq \mathbf{s} && \text{(D.5b)}\\ & -\mathbf{x} \leq \mathbf{s} && \text{(D.5c)}\end{aligned}$$

Maximum Value Operator

Minimizing the maximum value operator, i.e., $\min\ \mu\ \max(x_1, x_2)$, is equivalent to

$$\begin{aligned}\text{minimize} \quad & \mu \, s && \text{(D.6a)}\\ \text{subject to:} \quad & s \geq x_1 && \text{(D.6b)}\\ & s \geq x_2 && \text{(D.6c)}\end{aligned}$$

Second-Order Cone Constraints

The general second-order cone constraint

$$\|\mathbf{A}_k \, \mathbf{x}_k + \mathbf{b}_k\|_2 \leq \mathbf{c}_k^\top \mathbf{x}_k + d_k \tag{D.7}$$

can be transformed into

$$\begin{aligned}\mathbf{s}_k &= \mathbf{A}_k \, \mathbf{x}_k + \mathbf{b}_k && \text{(D.8a)}\\ t &= \mathbf{c}_k^\top \, \mathbf{x}_k + d_k && \text{(D.8b)}\\ \|\mathbf{s}_k\|_2 &\leq t && \text{(D.8c)}\end{aligned}$$

Quadratic Constraints

Quadratic constraints of the form

$$\mathbf{x}^\top \mathbf{x} \leq y \, z \tag{D.9}$$

with $\mathbf{x} \in \mathbb{R}^n$, $y, z \in \mathbb{R}$ and $y \geq 0$, $z \geq 0$, and $\mathbf{x}$ and y being decision variables. This can be rewritten as the following second-order cone constraint:

$$\left\| \begin{bmatrix} 2\,\mathbf{x} \\ y - z \end{bmatrix} \right\|_2 \leq y + z \tag{D.10}$$

In addition, minimizing $\mathbf{x}^\top \mathbf{Q}\,\mathbf{x}$ is equivalent to the epigraph form

$$\begin{aligned} &\text{minimize} && s && \text{(D.11a)}\\ &\text{subject to:} && \mathbf{x}^\top \mathbf{Q}\,\mathbf{x} \leq s && \text{(D.11b)} \end{aligned}$$

Assuming that the symmetric matrix $\mathbf{Q}$ is positive definite, its Cholesky decomposition $\mathbf{Q} = \mathbf{L}^\top \mathbf{L}$ is used to obtain

$$\begin{aligned} &\text{minimize} && s && \text{(D.12a)}\\ &\text{subject to:} && \mathbf{u}^\top \mathbf{u} \leq s && \text{(D.12b)} \end{aligned}$$

where $\mathbf{u} = \mathbf{L}\,\mathbf{x}$. This problem can be transformed into a second-order cone constraints using Eq. (D.10):

$$\begin{aligned} &\text{minimize} && s && \text{(D.13a)}\\ &\text{subject to:} && \left\| \begin{bmatrix} 2\,\mathbf{L}\,\mathbf{x} \\ s-1 \end{bmatrix} \right\|_2 \leq s+1 && \text{(D.13b)} \end{aligned}$$

Composite Functions

Recalling the soft trust-region constraint from Eq. (5.40) where the expression

$$p(\mathbf{x}) = \lambda_{\mathrm{TR}} \left[\max(0, \|\mathbf{x} - \bar{\mathbf{x}}\|_1 - R)\right]^2 \tag{D.14}$$

is to be minimized, the epigraph form is

$$\begin{aligned} &\text{minimize} && s && \text{(D.15a)}\\ &\text{subject to:} && \lambda_{\mathrm{TR}} \left[\max(0, \|\mathbf{x} - \bar{\mathbf{x}}\|_1 - R)\right]^2 \leq s && \text{(D.15b)} \end{aligned}$$

Rewriting $p(\mathbf{x})$ as $\tilde{p}(\mathbf{x}) = \left[\sqrt{\lambda_{\mathrm{TR}}}\, \max(0, \|\mathbf{x} - \bar{\mathbf{x}}\|_1 - R)\right]^2$, we consider $\tilde{p}(\mathbf{x})$ as a composition of two functions f and g, i.e., $\tilde{p}(\mathbf{x}) = f(g(\mathbf{x}))$, where

$$f(y) = y^2 \tag{D.16}$$

$$g(\mathbf{x}) = \sqrt{\lambda_{\mathrm{TR}}}\, \max(0, \|\mathbf{x} - \bar{\mathbf{x}}\|_1 - R) \tag{D.17}$$

As $\lambda_{\text{TR}} > 0$ and the $\max$ function is nondecreasing, g is nondecreasing in $[0, \infty)$. Therefore, we introduce another slack variable z, and the expressions $f(g(\mathbf{x})) \leq s$ and $f(z) \leq s$, $g(\mathbf{x}) \leq z$ are equivalent. The problem in Eq. (D.15) can then be reformulated as

$$\begin{aligned} &\text{minimize} && s && \text{(D.18a)} \\ &\text{subject to:} && z^2 \leq s && \text{(D.18b)} \\ &&& \sqrt{\lambda_{\text{TR}}}\,[\max(0, \|\mathbf{x} - \bar{\mathbf{x}}\|_1 - R)] \leq z && \text{(D.18c)} \end{aligned}$$

Using the results from previous subsections, this is transformed into standard form and we obtain

$$\begin{aligned} &\text{minimize} && s && \text{(D.19a)} \\ &\text{subject to:} && \left\| \begin{bmatrix} 2\,z \\ s - 1 \end{bmatrix} \right\|_2 \leq s + 1 && \text{(D.19b)} \\ &&& -z \leq 0 && \text{(D.19c)} \\ &&& \sqrt{\lambda_{\text{TR}}}\,\mathbf{1}^\top \mathbf{s}_{\text{TR}} = z + \sqrt{\lambda_{\text{TR}}}\,R && \text{(D.19d)} \\ &&& \sqrt{\lambda_{\text{TR}}}\,(-\mathbf{s}_{\text{TR}} - \mathbf{x} + \bar{\mathbf{x}}) \leq \mathbf{0} && \text{(D.19e)} \\ &&& \sqrt{\lambda_{\text{TR}}}\,(-\mathbf{s}_{\text{TR}} + \mathbf{x} - \bar{\mathbf{x}}) \leq \mathbf{0} && \text{(D.19f)} \end{aligned}$$

where $\mathbf{s}_{\text{TR}}$ are the concatenated slack variables when rewriting the 1-norm of the trust-region constraint.

D.2 Parsing of Low-Thrust Trajectory Optimization Problem

For the sake of completeness, we present the parsing of the fixed final time problem with general nonconvex boundary conditions, and of the energy-optimal problem.

Fixed Final Time, Nonconvex Final Boundary Conditions

Without loss of generality, we assume that the final boundary condition is a nonconvex function $\boldsymbol{\psi}$ of the final states $\mathbf{r}_N, \mathbf{v}_N$, i.e., the states at t_f:

$$\left| \boldsymbol{\psi}(\mathbf{r}_N, \mathbf{v}_N) - \begin{bmatrix} \mathbf{r}_f \\ \mathbf{v}_f \end{bmatrix} \right| \leq \begin{bmatrix} \Delta\mathbf{r} \\ \Delta\mathbf{v} \end{bmatrix} \tag{D.20}$$

The linearized formulation is

$$\left| \boldsymbol{\psi}(\bar{\mathbf{r}}_N, \bar{\mathbf{v}}_N) + \nabla\boldsymbol{\psi}(\bar{\mathbf{r}}_N, \bar{\mathbf{v}}_N) \begin{bmatrix} \mathbf{r}_N - \bar{\mathbf{r}}_N \\ \mathbf{v}_N - \bar{\mathbf{v}}_N \end{bmatrix} - \begin{bmatrix} \mathbf{r}_f \\ \mathbf{v}_f \end{bmatrix} \right| \leq \begin{bmatrix} \Delta\mathbf{r} \\ \Delta\mathbf{v} \end{bmatrix} + \boldsymbol{\zeta} \tag{D.21}$$

where $\boldsymbol{\zeta} \in \mathbb{R}^m_{\geq 0}$ (m being the number of final boundary conditions) is another virtual control that is to be penalized in the objective function. The resulting optimization problem reads

$$\underset{\mathbf{x},\mathbf{u},\boldsymbol{\nu},\boldsymbol{\eta},\boldsymbol{\zeta}}{\text{minimize}} \quad -w_N + \lambda_\nu \sum_{i=1}^{N-1} \|\boldsymbol{\nu}_i\|_1 + \lambda_\eta \sum_{i=1}^{N} \max(0,\eta_i) + \lambda_\zeta \sum_{i=1}^{m} \max(0,\zeta_i) \tag{D.22a}$$

$$\text{subject to:} \quad \text{Eqs. (9.11b)–(9.11f)} \tag{D.22b}$$

$$\left| \boldsymbol{\psi}(\bar{\mathbf{r}}_N, \bar{\mathbf{v}}_N) + \nabla\boldsymbol{\psi}(\bar{\mathbf{r}}_N, \bar{\mathbf{v}}_N) \begin{bmatrix} \mathbf{r}_N - \bar{\mathbf{r}}_N \\ \mathbf{v}_N - \bar{\mathbf{v}}_N \end{bmatrix} - \begin{bmatrix} \mathbf{r}_f \\ \mathbf{v}_f \end{bmatrix} \right| \leq \begin{bmatrix} \Delta\mathbf{r} \\ \Delta\mathbf{v} \end{bmatrix} + \boldsymbol{\zeta} \tag{D.22c}$$

Reformulating yields

$$\underset{\mathbf{x},\mathbf{u},\boldsymbol{\nu},\boldsymbol{\eta},\boldsymbol{\zeta},\mathbf{s}_\nu,\mathbf{s}_\eta,\mathbf{s}_\zeta}{\text{minimize}} \quad -w_N + \lambda_\nu \mathbf{1}^\top \mathbf{s}_\nu + \lambda_\eta \mathbf{1}^\top \mathbf{s}_\eta + \lambda_\zeta \mathbf{1}^\top \mathbf{s}_\zeta \tag{D.23a}$$

$$\text{subject to:} \quad \text{Eqs. (9.12b)–(9.12h) and (9.13d)} \tag{D.23b}$$

$$-\mathbf{s}_\zeta \leq \mathbf{0},\ \boldsymbol{\zeta} \leq \mathbf{s}_\zeta,\ -\boldsymbol{\zeta} \leq \mathbf{0} \tag{D.23c}$$

$$\boldsymbol{\psi}(\bar{\mathbf{r}}_N, \bar{\mathbf{v}}_N) + \nabla\boldsymbol{\psi}(\bar{\mathbf{r}}_N, \bar{\mathbf{v}}_N) \begin{bmatrix} \mathbf{r}_N - \bar{\mathbf{r}}_N \\ \mathbf{v}_N - \bar{\mathbf{v}}_N \end{bmatrix} - \begin{bmatrix} \mathbf{r}_f \\ \mathbf{v}_f \end{bmatrix} \leq \begin{bmatrix} \Delta\mathbf{r} \\ \Delta\mathbf{v} \end{bmatrix} + \boldsymbol{\zeta} \tag{D.23d}$$

$$-\boldsymbol{\psi}(\bar{\mathbf{r}}_N, \bar{\mathbf{v}}_N) - \nabla\boldsymbol{\psi}(\bar{\mathbf{r}}_N, \bar{\mathbf{v}}_N) \begin{bmatrix} \mathbf{r}_N - \bar{\mathbf{r}}_N \\ \mathbf{v}_N - \bar{\mathbf{v}}_N \end{bmatrix} + \begin{bmatrix} \mathbf{r}_f \\ \mathbf{v}_f \end{bmatrix} \leq \begin{bmatrix} \Delta\mathbf{r} \\ \Delta\mathbf{v} \end{bmatrix} + \boldsymbol{\zeta} \tag{D.23e}$$

$\mathbf{s}_\zeta$ is the slack variable associated with the virtual control $\boldsymbol{\zeta}$.

Energy-Optimal Problem

We consider a homotopy from the energy-optimal to the fuel-optimal problem with control magnitude u and the performance index

$$J = \int_{t_0}^{t_f} [u - \varepsilon u(1-u)]\,\mathrm{d}t = \int_{t_0}^{t_f} u(1-\varepsilon)\,\mathrm{d}t + \int_{t_0}^{t_f} \varepsilon\, u^2\,\mathrm{d}t \tag{D.24}$$

The quadratic part can be rewritten in standard form using the relations of the previous section. The discretized objective function takes the general form

$$J = (1-\varepsilon) \sum_{i}^{N} C_i\, u_i\, d_i + \varepsilon \sum_{i}^{N} C_i\, u_i^2\, d_i \tag{D.25}$$

with factors C_i and quadrature weights d_i that depend on the discretization and quadrature method. The expression $\sum_i^N C_i\, u_i^2\, d_i$ can be rewritten in matrix form as $\mathbf{u}^\top \mathbf{Q}\, \mathbf{u}$ with the concatenated controls $\mathbf{u}$. The diagonal matrix $\mathbf{Q}$ is positive definite with entries

$$Q_{ii} = \varepsilon\, C_i\, d_i \tag{D.26}$$

We transform the quadratic part of J into a second-order cone constraint to obtain:

$$\begin{aligned} &\text{minimize} && (1-\varepsilon)\sum_{i}^{N} C_i\,u_i\,d_i + s_\varepsilon && \text{(D.27a)} \\ &\text{subject to:} && \left\| \begin{bmatrix} 2\,\mathbf{L}\,\mathbf{u} \\ s_\varepsilon - 1 \end{bmatrix} \right\|_2 \leq s_\varepsilon + 1 && \text{(D.27b)} \end{aligned}$$

where s_ε is a slack variable, and $\mathbf{Q} = \mathbf{L}^\top\mathbf{L}$.

D.3 Structure of Matrices and Vectors

Throughout this section, $\mathbf{0}_{n\times m}$ denotes a zero matrix of size $n \times m$. A $n \times n$ identity matrix is represented by $\mathbf{1}_{n\times n}$, and $\mathbf{1}_n$ indicates a row vector of n ones.

Equality constraints

The matrix $\mathbf{A}$ and vector $\mathbf{b}$ are defined as follows

$$\mathbf{A} := \begin{bmatrix} \mathbf{A}_{\text{dyn}} \\ \mathbf{A}_{x0} \\ \mathbf{A}_{\text{TR}} \end{bmatrix}, \qquad \mathbf{b} := \begin{bmatrix} \mathbf{b}_{\text{dyn}} \\ \mathbf{b}_{x0} \\ \mathbf{b}_{\text{TR}} \end{bmatrix} \tag{D.28}$$

and contain all equality constraints:

1) $\mathbf{A}_{\text{dyn}}$ refers to the dynamics in Eq. (9.12b).
2) $\mathbf{A}_{x_0}$ is comprised of the initial boundary condition in Eq. (9.12h).
3) $\mathbf{A}_{\text{TR}}$ contains the trust-region constraint $\mathbf{1}^\top\,\mathbf{s}_{\text{TR}} = R$ in Eq. (9.12g).

In the following, the elements of each component are addressed in more detail. The braces over the matrices and vectors refer to the corresponding elements of the solution vector $\mathbf{y}$.

Dynamics

The $n_x\,(N-1)$ equality constraints are similar to the ones in Eq. (6.70), but with an additional block of zeros $\mathbf{0}_* \in \mathbb{R}^{n_x\,(N-1)\times N + n_x\,N}$ to account for the remaining slack variables:

$$\mathbf{A}_{\text{dyn}} = \left[\begin{array}{cc:cc:c:c} \overbrace{\hat{\mathbf{A}}_1 \quad\quad \mathbf{0}}^{\mathbf{x}} & & \overbrace{\hat{\mathbf{B}}_1 \quad\quad \mathbf{0}}^{\mathbf{u}} & & \overbrace{\phantom{\mathbf{1}}}^{\boldsymbol{\nu}} & \overbrace{\phantom{\mathbf{0}}}^{\mathbf{s}_\eta,\,\mathbf{s}_{\text{TR}}} \\ \ddots & & \ddots & & \mathbf{1}_{n_\nu\times n_\nu} & \mathbf{0}_* \\ \mathbf{0} & \hat{\mathbf{A}}_{N-1} & \mathbf{0} & \hat{\mathbf{B}}_{N-1} & & \end{array}\right] \tag{D.29}$$

where $n_\nu = n_x\,(N-1)$ and $k = 1,\ldots,N-1$. The matrices $\hat{\mathbf{A}}_k$ and $\hat{\mathbf{B}}_k$ have $(n_x-1)\,n_x + 1 + n_x$ and 2 $[(n_x-1)\,n_u+1]$ nonzero elements, respectively. They are to be updated in each iteration except for the entries that correspond to the identity matrices $-\mathbf{1}$ in $\hat{\mathbf{A}}_k$. The $(N-1)\,n_x$ entries for $\boldsymbol{\nu}$ do not change.

The vector $\mathbf{b}_{\text{dyn}}$ reads

$$\mathbf{b}_{\text{dyn}} = \begin{bmatrix} -\mathbf{q}_1 \\ \vdots \\ -\mathbf{q}_{N-1} \end{bmatrix} \tag{D.30}$$

With regard to the moving target problem, the matrix changes to

$$\mathbf{A}_{\text{dyn}} = \left[\begin{array}{cc:cc:c:c:c:c} \overbrace{\hat{\mathbf{A}}_1 \quad\quad \mathbf{0}}^{\mathbf{x}} & & \overbrace{\hat{\mathbf{B}}_1 \quad\quad \mathbf{0}}^{\mathbf{u}} & & \overbrace{\phantom{\mathbf{1}}}^{\boldsymbol{\nu}} & \overbrace{\phantom{\mathbf{0}}}^{\mathbf{s}_\eta,\,\mathbf{s}_{\text{TR}}} & \overbrace{\mathbf{S}_1}^{t_f} & \overbrace{\phantom{\mathbf{0}}}^{\boldsymbol{\zeta},\,\mathbf{s}_\zeta} \\ \ddots & & \ddots & & \mathbf{1}_{n_\nu\times n_\nu} & \mathbf{0}_* & \vdots & \mathbf{0}_{n_x\,(N-1)\times 2\,(n_x-1)} \\ \mathbf{0} & \hat{\mathbf{A}}_{N-1} & \mathbf{0} & \hat{\mathbf{B}}_{N-1} & & & \mathbf{S}_{N-1} & \end{array}\right] \tag{D.31}$$

Initial Boundary Conditions

The initial boundary conditions are n_x linear constraints:

$$\mathbf{A}_{x0} = \left[\begin{array}{cc:c} \overbrace{\mathbf{1}_{n_x\times n_x}}^{\mathbf{x}_1} & \overbrace{\mathbf{0}_{n_x\times(N-1)n_x}}^{\mathbf{x}_{2,\ldots,}\mathbf{x}_N} & \overbrace{\quad\mathbf{0}_*\quad}^{\mathbf{u},\,\boldsymbol{\nu},\,\mathbf{s}_\nu,\boldsymbol{\eta},\,\mathbf{s}_\eta,\,\mathbf{s}_{\text{TR}}} \end{array}\right] \tag{D.32}$$

where $\mathbf{b}_{x0} = \mathbf{x}_0$, and $\mathbf{0}_* \in \mathbb{R}^{n_x\times n_u\,N+2\,n_x\,(N-1)+2\,N+n_x\,N}$. $\mathbf{A}_{x0}$ consists of n_x nonzero elements that remain constant.

Trust Region

The trust-region constraint is comprised of a single row of the following form:

$$\mathbf{A}_{\text{TR}} = \left[\begin{array}{c:c} \overbrace{\quad\mathbf{0}_*\quad}^{\mathbf{x},\,\mathbf{u},\,\boldsymbol{\nu},\,\mathbf{s}_\nu,\,\boldsymbol{\eta},\,\mathbf{s}_\eta} & \overbrace{\mathbf{1}_{n_x\,N}}^{\mathbf{s}_{\text{TR}}} \end{array}\right] \tag{D.33}$$

There are $n_x\,N$ nonzero, constant elements, and $b_{\text{TR}} = R$ is to be updated when the trust-region radius changes. The block of zeros is $\mathbf{0}_* \in \mathbb{R}^{1\times n_u\,N+2\,n_x\,(N-1)+2\,N+n_x\,N}$.

For the moving target case, there is an additional 1 due to the imposed trust-region constraint on t_f.

Inequality constraints

With regard to the inequality constraints, the matrix $\mathbf{G}$ and vector $\mathbf{h}$ are defined as follows

$$\mathbf{G} \coloneqq \begin{bmatrix} \mathbf{G}_{\text{thrust}} \\ \mathbf{G}_{\text{TR}} \\ \mathbf{G}_{x_f} \\ \mathbf{G}_{\nu} \\ \mathbf{G}_{\eta} \\ \mathbf{G}_{\text{socc}} \end{bmatrix}, \qquad \mathbf{h} \coloneqq \begin{bmatrix} \mathbf{h}_{\text{thrust}} \\ \mathbf{h}_{\text{TR}} \\ \mathbf{h}_{x_f} \\ \mathbf{h}_{\nu} \\ \mathbf{h}_{\eta} \\ \mathbf{h}_{\text{socc}} \end{bmatrix} \tag{D.34}$$

where:

1) $\mathbf{G}_{\text{thrust}}$ refers to the linearized upper bounds of the thrust magnitude in Eq. (9.12e).
2) $\mathbf{G}_{\text{TR}}$ contains the the trust-region constraints in Eq. (9.12g).
3) $\mathbf{G}_{x_f}$ defines the final boundary conditions in Eqs. (9.12i) and (9.12j).
4) $\mathbf{G}_{\nu}$ and $\mathbf{G}_{\eta}$ refer to the constraints on the slack variables $\boldsymbol{\nu}$ and $\boldsymbol{\eta}$, respectively, in Eqs. (9.12c) and (9.12d).
5) $\mathbf{G}_{\text{socc}}$ is comprised of the second-order cone constraints in Eq. (9.12f).

We proceed in a similar way and present details about the implementation of the matrices.

Thrust Magnitude

There are N constraints for the upper bounds of the thrust magnitude. The corresponding matrix reads

$$\mathbf{G}_{\text{thrust}} = \left[\begin{array}{ccc:ccc:c:c:c} \multicolumn{3}{c:}{\overbrace{\hspace{10em}}^{\mathbf{x}}} & \multicolumn{3}{c:}{\overbrace{\hspace{10em}}^{\mathbf{u}}} & \overbrace{\hspace{4em}}^{\boldsymbol{\nu},\, \mathbf{s}_{\nu}} & \overbrace{\hspace{2em}}^{\boldsymbol{\eta}} & \overbrace{\hspace{1em}}^{\mathbf{s}_{\eta},\, \mathbf{s}_{\text{TR}}} \\ \mathbf{0}_{1\times n_x-1} & \Gamma_{\max,1} & \mathbf{0}_{1\times n_x (N-1)} & \mathbf{0}_{1\times n_u-1} & 1 & \mathbf{0}_{1\times n_u (N-1)} & & & \\ \mathbf{0}_{1\times 2\,n_x-1} & \Gamma_{\max,2} & \mathbf{0}_{1\times n_x (N-2)} & \mathbf{0}_{1\times 2\,n_u-1} & 1 & \mathbf{0}_{1\times n_u (N-2)} & \mathbf{0}_{N\times n_\nu+n_{s_\nu}} & \mathbf{1}_{N\times N} & \mathbf{0}_{*} \\ & \vdots & & & \vdots & & & & \\ \mathbf{0}_{1\times n_x N-1} & & \Gamma_{\max,N} & \mathbf{0}_{1\times n_u N-1} & & 1 & & & \end{array}\right] \tag{D.35}$$

where n_ν and n_{s_ν} denote the lengths of the vectors $\boldsymbol{\nu}$ and s_ν, respectively. Furthermore,

$$\Gamma_{\max,k} = T_{\max}(\bar{r}_k)\, \mathrm{e}^{-\bar{w}_k}, \qquad k = 1, \ldots, N \tag{D.36}$$

$\mathbf{0}_* \in \mathbb{R}^{N \times N + n_x N}$ is a block of zeros. The number of nonzero elements is $3\,N$, and the only changing elements are $\Gamma_{\max,k}$, $k = 1, \ldots, N$. The vector h_{thrust} is given by

$$h_{\text{thrust}} = \begin{pmatrix} \Gamma_{\max,1}\,(1 + \bar{w}_1) \\ \vdots \\ \Gamma_{\max,N}\,(1 + \bar{w}_N) \end{pmatrix} \tag{D.37}$$

and is to be updated when the reference trajectory changes.

Trust Region

The $2\,n_x\,N$ trust-region constraints are given by:

$$\mathbf{G}_{\text{TR}} = \left[\begin{array}{c|c|c} \overbrace{\phantom{-\mathbf{1}_{n_x N \times n_x N}}}^{\mathbf{x}} & \overbrace{\phantom{\mathbf{u},\boldsymbol{\nu},\mathbf{s}_\nu,\boldsymbol{\eta},\mathbf{s}_\eta,\mathbf{s}_{\text{TR}}}}^{\mathbf{u},\boldsymbol{\nu},\mathbf{s}_\nu,\boldsymbol{\eta},\mathbf{s}_\eta,\mathbf{s}_{\text{TR}}} & \overbrace{\phantom{-\mathbf{1}_{n_x N \times n_x N}}}^{\mathbf{s}_{\text{TR}}} \\ \mathbf{1}_{n_x N \times n_x N} & & -\mathbf{1}_{n_x N \times n_x N} \\ & \mathbf{0}_* & \\ -\mathbf{1}_{n_x N \times n_x N} & & -\mathbf{1}_{n_x N \times n_x N} \end{array}\right] \tag{D.38}$$

All $4\,n_x\,N$ nonzero entries are constant, and $\mathbf{0}_* \in \mathbb{R}^{2\,n_x\,N \times n_u\,N + 2\,n_x\,(N-1) + 2\,N}$. The vector $\mathbf{h}_{\text{TR}}$ is

$$\mathbf{h}_{\text{TR}} = \begin{bmatrix} \bar{\mathbf{x}} \\ -\bar{\mathbf{x}} \end{bmatrix} \tag{D.39}$$

and must be updated when the reference changes.

In case of a moving target where the trust-region constraint is also imposed on the additional time variable t_f, $\mathbf{G}_{\text{TR}}$ has 4 additional nonzero elements, and $\mathbf{h}_{\text{TR}} = [\bar{\mathbf{x}}^\top,\, -\bar{\mathbf{x}}^\top,\, \bar{t}_f,\, -\bar{t}_f]^\top$.

Final Boundary Conditions

In case of a rendezvous problem with free final mass, the final boundary conditions are linear inequality constraints:

$$\mathbf{G}_{x_f} = \left[\begin{array}{c|cc|c} \overbrace{\phantom{\mathbf{0}_{n_x-1 \times n_x (N-1)}}}^{\mathbf{x}_1,\ldots,\mathbf{x}_{N-1}} & \multicolumn{2}{c|}{\overbrace{\phantom{-\mathbf{1}_{n_x-1 \times n_x-1} \quad \mathbf{0}_{n_x-1 \times 1}}}^{\mathbf{x}_N}} & \overbrace{\phantom{\mathbf{u},\boldsymbol{\nu},\mathbf{s}_\nu,\boldsymbol{\eta},\mathbf{s}_\eta,\mathbf{s}_{\text{TR}}}}^{\mathbf{u},\boldsymbol{\nu},\mathbf{s}_\nu,\boldsymbol{\eta},\mathbf{s}_\eta,\mathbf{s}_{\text{TR}}} \\ \mathbf{0}_{n_x-1 \times n_x (N-1)} & \mathbf{1}_{n_x-1 \times n_x-1} & \mathbf{0}_{n_x-1 \times 1} & \\ & & & \mathbf{0}_* \\ \mathbf{0}_{n_x-1 \times n_x (N-1)} & -\mathbf{1}_{n_x-1 \times n_x-1} & \mathbf{0}_{n_x-1 \times 1} & \end{array}\right] \tag{D.40}$$

with $\mathbf{0}_* \in \mathbb{R}^{n_x-1 \times n_u\,N + 2\,n_x\,(N-1) + 2\,N + n_x\,N}$. There are $n_x - 1$ nonzero elements that remain constant. Moreover,

$$\mathbf{h}_{x_f} = \begin{bmatrix} \Delta\mathbf{x}_f + \mathbf{x}_f \\ \Delta\mathbf{x}_f - \mathbf{x}_f \end{bmatrix} \tag{D.41}$$

where $\mathbf{x}_f := [\mathbf{r}_f^\top, \mathbf{v}_f^\top]^\top$ and $\Delta\mathbf{x} := [\Delta\mathbf{r}_f^\top, \Delta\mathbf{v}_f^\top]^\top$.

With regard to the moving target problem, the number of nonzero elements of $\mathbf{G}_{x_f}$ is $6\,(n_x - 1)$, where $n_x - 1$ entries change in each iteration according to Eqs. (9.13e) and (9.13f). In addition, all elements of $\mathbf{h}_{x_f}$ are to be updated.

Slack Variables $\boldsymbol{\nu}$ and $\mathbf{s}_\nu$

The matrix $\mathbf{G}_\nu$ consists of two blocks with a total of $4\,n_x\,(N-1)$ non-changing elements:

$$\mathbf{G}_\nu = \left[\begin{array}{c:c:c:c} \overbrace{\qquad\qquad}^{\mathbf{x},\mathbf{u}} & \overbrace{\qquad\qquad}^{\boldsymbol{\nu}} & \overbrace{\qquad\qquad}^{\mathbf{s}_\nu} & \overbrace{\qquad\quad}^{\boldsymbol{\eta},\mathbf{s}_\eta,\mathbf{s}_{\mathrm{TR}}} \\ \mathbf{0}_{n_x\,(N-1)\times(n_x+n_u)\,N} & \mathbf{1}_{n_x\,(N-1)\times n_x\,(N-1)} & -\mathbf{1}_{n_x\,(N-1)\times n_x\,(N-1)} & \\ & & & \mathbf{0}_* \\ \mathbf{0}_{n_x\,(N-1)\times(n_x+n_u)\,N} & -\mathbf{1}_{n_x\,(N-1)\times n_x\,(N-1)} & -\mathbf{1}_{n_x\,(N-1)\times n_x\,(N-1)} & \end{array}\right] \tag{D.42}$$

The dimensions of $\mathbf{0}_*$ are $2\,n_x\,(N-1)\times 2\,N + n_x\,N$. The corresponding vector is $\mathbf{h}_\nu = \mathbf{0}_{2\,n_x\,(N-1)\times 1}$.

Slack Variables η and s_η

$\mathbf{G}_\eta$ is comprised of three blocks and $4\,N$ constant nonzero elements:

$$\mathbf{G}_\eta = \left[\begin{array}{c:c:c:c} \overbrace{\qquad\qquad}^{\mathbf{x},\mathbf{u},\boldsymbol{\nu},\mathbf{s}_\nu} & \overbrace{\qquad\quad}^{\boldsymbol{\eta}} & \overbrace{\qquad\quad}^{\mathbf{s}_\eta} & \overbrace{\qquad\qquad}^{\mathbf{s}_{\mathrm{TR}}} \\ & \mathbf{0}_{N\times N} & -\mathbf{1}_{N\times N} & \\ \mathbf{0}_* & \mathbf{1}_{N\times N} & -\mathbf{1}_{N\times N} & \mathbf{0}_{3\,N\times n_x\,N} \\ & -\mathbf{1}_{N\times N} & \mathbf{0}_{N\times N} & \end{array}\right] \tag{D.43}$$

where $\mathbf{0}_* \in \mathbb{R}^{3\,N\times n_x\,N+n_u\,N+2\,n_x\,(N-1)}$. Furthermore, $\mathbf{h}_\eta = \mathbf{0}_{3\,N\times 1}$.

Moving Target: Slack Variables $\boldsymbol{\zeta}$ and s_ζ

The moving target problem requires an additional matrix $\mathbf{G}_\zeta$ due to Eq. (9.13d):

$$\mathbf{G}_\zeta = \left[\begin{array}{c:c:c} \overbrace{\qquad\qquad}^{\mathbf{y},t_f} & \overbrace{\qquad\qquad}^{\boldsymbol{\zeta}} & \overbrace{\qquad\qquad}^{\mathbf{s}_\zeta} \\ & \mathbf{0}_{n_x-1\times n_x-1} & -\mathbf{1}_{n_x-1\times n_x-1} \\ \mathbf{0}_* & \mathbf{1}_{n_x-1\times n_x-1} & -\mathbf{1}_{n_x-1\times n_x-1} \\ & -\mathbf{1}_{n_x-1\times n_x-1} & \mathbf{0}_{n_x-1\times n_x-1} \end{array}\right] \tag{D.44}$$

There are $4\,(n_x-1)$ nonzero, constant elements, and $\mathbf{0}_* \in \mathbb{R}^{3\,(n_x-1)\times n_y+1}$. The vector on the right-hand side is $\mathbf{h}_\zeta = \mathbf{0}_{3\,(n_x-1)\times 1}$.

Moving Target: Bounds on t_f

If lower and upper bounds are imposed on t_f, two constraints are required with two constant, nonzero entries:

$$\mathbf{G}_{\text{bounds}} = \left[\begin{array}{c:c:c} \overbrace{\phantom{\mathbf{0}_*}}^{\mathbf{y}} & \overbrace{}^{t_f} & \overbrace{\phantom{\mathbf{0}_{2\times 2\,(n_x-1)}}}^{\boldsymbol{\zeta},\mathbf{s}_\zeta} \\ & -1 & \\ \mathbf{0}_* & & \mathbf{0}_{2\times 2\,(n_x-1)} \\ & 1 & \end{array}\right] \tag{D.45}$$

where $\mathbf{0}_* \in \mathbb{R}^{2\times n_y}$. The corresponding vector on the right-hand side is $\mathbf{h}_{\text{bounds}} = [-t_{f,\text{lb}},\ t_{f,\text{ub}}]^\top$.

Second-Order Cone Constraints

The structure of the matrix that contains the second-order cone constraints may depend on the chosen solver. In case of ECOS [86] that is used throughout this dissertation, $\mathbf{G}_{\text{socc}}$ has $n_u\,N$ constant nonzero elements and takes the following form:

$$\mathbf{G}_{\text{socc}} = \left[\begin{array}{c:ccc:c} \overbrace{\phantom{\mathbf{0}_{n_u N\times n_x N}}}^{\mathbf{x}} & & \overbrace{\phantom{\mathbf{M}\quad\mathbf{0}}}^{\mathbf{u}} & & \overbrace{\phantom{\mathbf{0}_*\quad\quad}}^{\boldsymbol{\nu},\mathbf{s}_\nu,\boldsymbol{\eta},\mathbf{s}_\eta,\mathbf{s}_{\text{TR}}} \\ & \mathbf{M} & & \mathbf{0} & \\ \mathbf{0}_{n_u\,N\times n_x\,N} & & \ddots & & \mathbf{0}_* \\ & \mathbf{0} & & \mathbf{M} & \end{array}\right] \tag{D.46}$$

where

$$\mathbf{M} = \begin{bmatrix} 0 & 0 & 0 & -1 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix} \tag{D.47}$$

and $\mathbf{0}_* \in \mathbb{R}^{n_u\,N\times 2\,n_x\,(N-1)+2\,N+n_x\,N}$. The corresponding vector on the right-hand side is $\mathbf{h}_{\text{socc}} = \mathbf{0}_{n_u\,N\times 1}$.

Objective Function

All $n_c = n_\nu + n_\eta + 1 = n_x\,(N-1) + N + 1$ nonzero elements of the objective function vector $\mathbf{c}$ remain constant:

$$\mathbf{c} = \left[\begin{array}{ccc:c:c:c:c:c} \overbrace{\mathbf{0}_{1\times n_x\,(N-1)}}^{\mathbf{x}_1,\ldots,\mathbf{x}_{N-1}} & \overbrace{\mathbf{0}_{1\times n_x-1}\quad -1}^{\mathbf{x}_N} & & \overbrace{\mathbf{0}_{1\times n_u\,N+n_\nu}}^{\mathbf{u},\boldsymbol{\nu}} & \overbrace{\lambda_\nu\,\mathbf{1}_{n_\nu}}^{\mathbf{s}_\nu} & \overbrace{\mathbf{0}_{1\times n_\eta}}^{\boldsymbol{\eta}} & \overbrace{\lambda_\eta\,\mathbf{1}_{n_\eta}}^{\mathbf{s}_\eta} & \overbrace{\mathbf{0}_{1\times n_x\,N}}^{\mathbf{s}_{\text{TR}}} \end{array}\right]^\top \tag{D.48}$$

In case of a moving target, the elements $\lambda_\zeta\,\mathbf{1}_{n_\zeta}^\top$, $n_\zeta = n_x - 1$, are to be included in $\mathbf{c}$.