Enhanced Indirect and Convex Optimization Methods for Generating Minimum-Fuel Low-Thrust Trajectories

by

Saeid Tafazzol

A thesis submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Auburn, Alabama
Dec 14, 2024

Keywords: Optimal Control, Trajectory Optimization, Space Mission Design, Indirect Control, Convex Optimization

Abstract

The high costs of space missions necessitate optimization across all mission aspects, particularly fuel consumption, which impacts both economic feasibility and payload capacity. Electric propulsion systems, with higher fuel efficiency but lower thrust output (compared to their chemical counterparts), have been used for several decades as a promising alternative propulsion system. To optimize spacecraft equipped with electric propulsion systems, researchers employ two optimization paradigms: *indirect* and *direct* methods.

Indirect methods, based on the calculus of variations and Pontryagin's minimum principle, transform the task of solving an optimization problem into finding the roots of nonlinear boundary-value problems (BVPs). This is achieved by introducing time-varying Lagrange multipliers (a.k.a. costates) and constant Lagrange multipliers. While indirect methods offer high-precision (i.e., with respect to time and resolution) and the solutions are guaranteed to be extremal, the resulting BVPs exhibit high sensitivity to the unknown decision variables (typically costates) and can become challenging to form and solve, when they are applied to constrained nonlinear optimization problems.

Direct methods, on the other hand, discretize variables early, converting the problem into a nonlinear programming (NLP) problem. A convex optimization problem is a specific type of NLP problem where the objective function and constraint set are all "convex," meaning that any local minimum is also a global minimum, making it significantly easier to solve with guaranteed globally optimal solutions, while a general NLP problem can have multiple local minima, making it potentially much harder to find the true optimal solution; essentially, convex optimization is a subset of nonlinear programming with the added benefit of guaranteed global optimality due to its convex structure. By incorporating convexification techniques and leveraging convex optimization tools, direct methods offer a computationally feasible approach. This research proposes methods that alleviate some of the difficulties associated with solving minimum-fuel low-thrust trajectory optimization problems using both indirect and direct methods.

Acknowledgments

To the only love of my life Golnoosh, I dedicate this. You are the source of my will, and I am forever grateful to you.

Thank you, Maman, Baba, and Samira, for putting the love of science in my heart.

I am grateful for Dr. Taheri's support. He is the embodiment of a perfect supervisor, with a high passion for learning alongside his students and his ongoing support and dedication.

I would like to thank Dr. Davide Guzzetti and Dr. Hans Werner Van Wyk for being on my committee and providing insightful feedback.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

DACE	Differential Algebra Computational Engine
ECOS	Embedded Conic Solver
HTS	Hyperbolic Tangent Smoothing
IPM	Interior-Point Method
NLP	Nonlinear Program
ODE	Ordinary Differential Equation
SCVX	Successive Convex Optimization
SDP	Semi-Definite Programming
STM	State Transition Matrix
STT	State Transition Tensor
TPBVP	Two Point Boundary Value Problem
w.r.t.	With Respect To

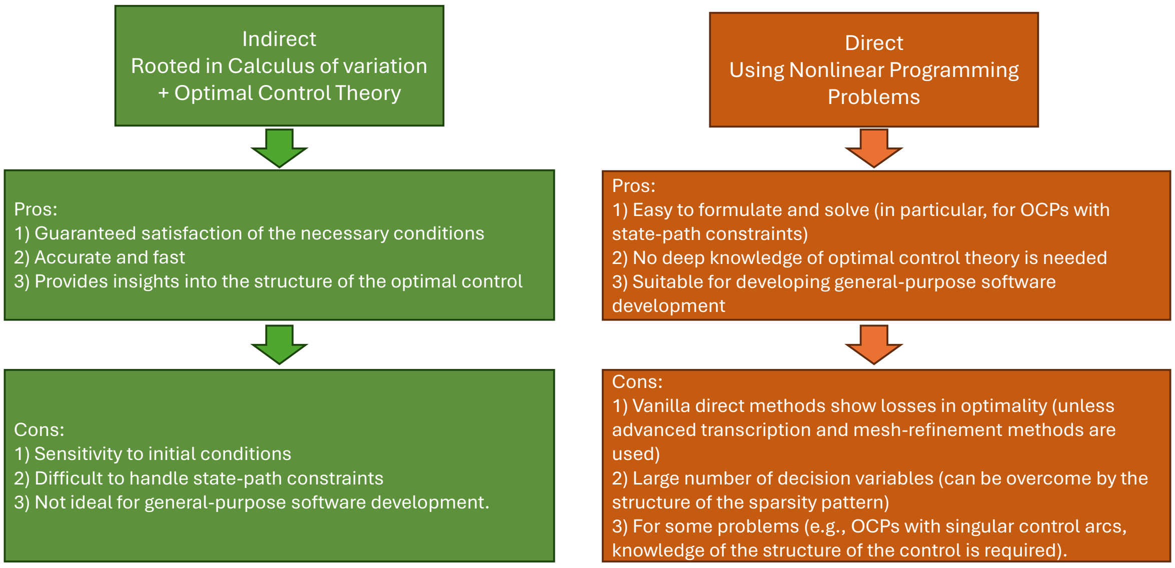
Chapter 1

Introduction

1.1 Introduction

Space exploration is arguably the pinnacle of human curiosity and is often correlated with a prosperous society. However, with great aspirations comes a great cost, which is attributed to the fact that space exploration is among the most costly of endeavors. This cost translates directly to not only sending payload to space, but also to maintaining the operation of the infrastructure needed to sustain the activities of space assets. Moreover, spacecraft have limited fuel capacity and there are no fuel stations in space for re-fueling spacecraft. Therefore, one of the most important metrics to minimize during mission design is the amount of fuel needed for completing space missions. Moreover, consuming less fuel corresponds to a larger payload capacity and consequently, can be used to carry more scientific payload to maximize science outcomes/objectives. As a consequence, electric propulsion systems, which offer great fuel efficiency, have become popular over the past several decades [1, 2].

Launched on October 24, 1998, Deep Space 1 was the first mission of NASA's New Millennium program [3], which among several advanced technologies demonstrated the use of electric propulsion for deep space missions. SMART-1 was the European Space Agency's first mission to the Moon that used electric propulsion system [4]. A similar technology was utilized for the Dawn mission, where a spacecraft was able (for the first time) to orbit a main-belt asteroid and visit multiple bodies during a single mission [5]. While being fuel-efficient with respect to their high specific impulse values [6], these propulsion systems generate low-thrust magnitudes, leading to trajectories that consist of many orbital revolutions around the central body [7, 8]. In very low-thrust cases, the number of orbital revolutions can even exceed a few hundred [9, 10]. Moreover, depending on the considered performance index (e.g., minimization of fuel consumption or time of flight), an optimal trajectory may consist of a finite, but unknown number of thrusting and coasting arcs. The sequence and duration of these arcs form the structure of the optimal control profile, where the duration of each arc is also unknown and must be determined as part of the solution procedure.



Pros:
1) Guaranteed satisfaction of the necessary conditions
2) Accurate and fast
3) Provides insights into the structure of the optimal control

Cons:
1) Sensitivity to initial conditions
2) Difficult to handle state-path constraints
3) Not ideal for general-purpose software development.

Pros:
1) Easy to formulate and solve (in particular, for OCPs with state-path constraints)
2) No deep knowledge of optimal control theory is needed
3) Suitable for developing general-purpose software development

Cons:
1) Vanilla direct methods show losses in optimality (unless advanced transcription and mesh-refinement methods are used)
2) Large number of decision variables (can be overcome by the structure of the sparsity pattern)
3) For some problems (e.g., OCPs with singular control arcs, knowledge of the structure of the control is required).

Figure 1.1: A comparison between indirect and direct methods advantages and disadvantages.

Numerical methods for solving practical optimal control problems (OCPs), which trajectory optimization problems constitute an important class thereof, are broadly classified into *direct* and *indirect* optimization methods [11, 12] (See Fig. 1.1). Both methods have been extensively used in spacecraft trajectory optimization. Indirect (variational) approaches to optimal control, are rooted in the calculus of variations, and result in (Hamiltonian) boundary-value problems (BVPs) using Lagrange multipliers (co-states) associated with states. However, in their simplest forms (i.e., for problems without control or state-path equality/inequality constraints), the BVPs reduce to two-point boundary value problems (TPBVPs). To solve BVPs/TPBVPs, quasi-Newton gradient-based root-solving methods are typically used, iterating on unknown costate values (typically at the initial time of maneuver) to generate state, costate, and control time histories that satisfy the necessary conditions of optimality.

In this research, we focus on both indirect and direct optimization methods. Indirect optimization methods are considered in Chapter 2 for the following reasons:

- **Astrodynamics Context:** In astrodynamics, spacecraft motion is primarily governed by Keplerian motion within the sphere of influence of the central body. The propulsive acceleration vector produced by low-thrust propulsion systems is small compared to central-body gravitational and third-body perturbing accelerations (e.g., due to Jupiter for a low-Earth orbit). Even with propulsion, the changes in orbital elements remain small over short time horizons. This makes low-thrust trajectory optimization problems easier to solve compared to the problems that arise in atmospheric flights, where significant aerodynamic forces are encountered and cannot be considered mere perturbations. Thus, indirect methods remain a viable choice for low-thrust trajectory optimization.

- **High-Resolution Solutions:** Indirect approaches provide high-resolution solutions that are highly accurate for the same local/global extremal solutions. High-resolution solutions can capture all events, such as thrust switches as well as entry to and exit from eclipses, with a high precision. This accuracy in capturing the occurrence of events is crucial, as solutions from indirect methods serve as baseline solutions for assessing the optimality gap, when compared to direct methods [13]. An example comparison between indirect and convex optimization-based methods is presented in [14] for minimum-fuel trajectories. Additionally, indirect methods, combined with homotopy and numerical continuation techniques, provide a more global mapping of the solution space [15].

- **Control Regularization:** The concept of control regularization, based on the principle of “invariant embedding,” is a key enabler for solving Hamiltonian BVPs, particularly when state, control, and mixed state-control inequality constraints are considered. Direct methods have historically been favored due to their ability to handle various constraints at a finite number of mesh points and their broader domain of convergence [16]. However, recent advances in indirect methods, such as Composited Smooth Control (CSC) [17], the Unified Trigonometrization Method (UTM) [18], Vectorized Trigonometric Regularization (VTR) [19], and Generalized Vectorized Trigonometric Regularization (GVTR) [20], have significantly alleviated many of the challenges associated with solving constrained optimal control problems.

However, when solving TPBVPs using indirect methods, it is necessary to determine the initial costates, often through quasi-Newton root-finding solvers. This process can be challenging and, depending on the problem, may be computationally expensive and time-consuming. In some cases, it may even be impossible to find a solution without resorting to techniques such as regularization and homotopy methods. Indirect methods also encounter situations where the control law does not have a closed-form solution during the minimization of the Hamiltonian. Given these challenges, it is worthwhile to consider direct optimization methods, where the continuous-time optimal control problem is approximated using a finite set of parameters, by discretizing the original problem.

Among direct optimization methods, convex optimization [21], which will be covered in Chapter 3, is considered among the most effective approaches for solving trajectory optimization problems for the following reasons [22, 23]:

- **High Performance for Large-Scale Problems:** Convex optimization provides efficient and fast solutions even for large-scale problems by utilizing interior-point methods (IPMs), which are guaranteed to converge in polynomial time. This makes convex optimization highly suitable for autonomous guidance and control applications.

- **Suitable for Embedded Applications:** Due to its efficiency, convex optimization has been adapted for use in embedded systems with limited computational resources [24]. It is expected that custom convex optimization codes will be implemented on different embedded computers to further enhance their application and utility for trajectory optimization in space missions.

Nonetheless, using convex optimization requires addressing the challenges of problem discretization and convexification, as detailed in Chapter 3.

1.2 Literature Review and Previous Contributions

1.2.1 Indirect Methods

In minimum-fuel trajectory optimization problems, the presence of thrusting and coasting arcs is a consequence of applying Pontryagin's minimum principle [25]. For minimum-time maneuvers, the optimality principle requires the thruster to operate at its maximum thrusting capability during the entire maneuver time [26]; however, it is still possible to have coasting arcs if the optimization problem has to take into account eclipses mostly during the planet-centric phases of flights [27, 28] and for cislunar trajectory optimization problems [29]. For minimum-fuel trajectory optimization problems, the optimal bang-off-bang thrust profile causes the Jacobian (used in gradient-based solvers for determining the search direction) to become singular [30, 25], which in turn deteriorates the selection of the descent direction (in typical gradient-based nonlinear root-finding methods) and renders the numerical solution of the resulting TPBVPs impossible. The existence of multiple throttle switches exacerbates the issue, and reduces the basin of attraction of the numerical methods [25, 31, 32].

To overcome the singularity in the Jacobian, the TPBVPs associated with minimum-fuel trajectory optimization problems are typically solved through various regularization methods, such as those that regularize the non-smooth thrust/throttle profiles. Control regularization methods such as logarithmic smoothing [30], extended logarithmic smoothing [33], hyperbolic tangent smoothing (HTS) [34] are among the popular and most efficient methods for attenuating the issues with non-smooth control profiles. Through regularization methods, the non-smooth optimal control problem is embedded into a one- or multiple-parameter family of smooth neighboring optimal control problems. Then, standard (or advanced) numerical continuation and/or homotopy methods are used to solve the resulting family of smooth TPBVPs until a solution to the original optimal control problem is obtained [12].

In terms of applications, the HTS method is already used to solve interplanetary minimum-fuel trajectory optimization and minimum-time rest-to-rest satellite reorientation problems [34]. One of the main advantages of the HTS method is its ease of implementation. One can proceed with the standard approach to the optimal control problem and determine the so-called thrust/throttle switching function. Then, the HTS method is applied as a filter to the thrust switching function to form a smooth approximation of the theoretically optimal bang-bang thrust profile. Thus, the HTS method is directly applied at the control level, which is an important feature because the standard procedure for formulating the necessary conditions (using indirect methods) remains unaffected. The HTS method is also a key component of a novel framework – Composite Smooth Control (CSC) – proposed for solving optimal control problems with discrete and multiple modes of operations [17, 35]. The CSC framework and the HTS method are also used for co-optimization of spacecraft propulsion system and its trajectory [36], optimal mode-selection and net payload mass optimization using indirect optimization methods [37], multimode trajectory optimization with low-thrust gravity-assist maneuvers [38], eclipse-conscious cislunar low-thrust trajectory design [39], and for minimum-fuel asteroid landing trajectory optimization problems [40].

In this research, we compare two state-of-the-art regularization methods, i.e., the HTS method and a recent L2-norm-based regularization, proposed by Taheri and Li [41]. In [41], the utility of the L2-norm-based regularization is demonstrated for solving three optimal control problems: 1) minimum-fuel low-thrust trajectories, 2) the standard Goddard rocket problem with its characteristic bang-singular-bang thrust profile and 3) a minimum-time spacecraft reorientation problem with both bang-bang and a second-order singular arc. The L2-norm-based regularization is already used for co-optimization of the spacecraft propulsion system and its trajectory using a direct optimization method [42]. The HTS method is already compared against the logarithmic smoothing in [34] and against the standard quadratic smoothing in [43] revealing its advantages by broadening the basin of attraction of the resulting TPBVPs. Application of the HTS combined with the State Transition Matrix (STM) is also studied in [44]. Similar to [44], we consider the impact of accurate calculation of sensitivities for the HTS and L2-norm-based regularization/smoothing methods, when finite-difference and the STM methods, are used. We compare the results of the two methods on two benchmark minimum-fuel trajectory optimization problems. Combining the advantages of the L2-norm-based regularization with the STM method is one of the contributions of this thesis.

1.2.2 Direct Optimization through Convex Programs

Optimal Control is a broad topic that deals mainly with two technical areas that are aimed to improve the overall efficiency of operation of dynamical systems: 1) (open-loop) path planning and guidance, and 2) closed-loop control design to regulate dynamical systems by leveraging optimization techniques. A well-known classical example is the Linear Quadratic Regulator (LQR) introduced by Kalman [45], where the control solution is derived in closed form, representing a special case of convex optimization. Over time, the connection between convex optimization and optimal control deepened, as it was recognized that numerous optimal control problems could be effectively modeled using convex optimization [46].

Even in scenarios where a problem could not be directly formulated as a convex problem, researchers exploited the efficiency of convex optimization by employing approximation techniques like convexification, enabling its application to a broader class of problems. The continuous advancement of convex optimization further influenced control theory in profound ways. For instance, the development of Semi-Definite Programming (SDP) [47] played a crucial role in formalizing the S-Lemma [48], thereby expanding the theoretical and practical horizons of optimization in control systems.

As a result, Convex Optimization has become a powerful numerical framework for solving complex optimal control problems. For example, the powered descent guidance for Mars pinpoint landing was effectively addressed using Second-Order Cone Programming (SOCP) in [49, 50, 51]. Similar techniques were applied to trajectory planning for near-field rendezvous and Proximity Operations (PRO) in [52, 53, 54], where the complex dynamics necessitated the use of successive convex optimization instead of direct convex methods. This approach has also been employed for spacecraft hypersonic reentry problems [55, 56, 57], where the vehicle re-enters Earth's atmosphere at speeds exceeding Mach 5, facing extreme aerodynamic heating and forces. Precise control is essential to maintain a safe trajectory while mitigating intense thermal loads and structural stresses. For further examples of successive convexification applications, please refer to [58, 59], which explore the application of convex optimization in rendezvous and docking scenarios.

One practical aspect of Convex Optimization is the possibility of making it highly efficient for onboard online trajectory optimization [24]. Tools like Embedded Conic Solver [60], CVX-GEN [61], and [62] enable exciting real-world applications such as Landing of Space Rockets [63], and autonomous obstacle avoidance in quad-rotors [64].

Chapter 2

Control Regularization within Indirect Optimization Methods

Trajectory optimization of space vehicles is a critical task in astrodynamics [65, 66], impacting both mission success and cost efficiency. Advances in electric propulsion systems, with their characteristic high specific impulse values, have led to more fuel-efficient low-thrust trajectories [6], often involving numerous orbital revolutions [7, 8]. These trajectories consist of thrusting and coasting arcs, determined by optimization methods [10].

Optimization approaches are broadly classified as direct or indirect [11]. Indirect methods, based on the calculus of variations, solve two-point boundary value problems (TPBVPs) using root-finding techniques, offering high-resolution solutions [13]. Despite numerical challenges, such as sensitivity of the TPBVPs to the unknown values of the costates, recent advances have enabled indirect methods to handle complex constraints effectively [18, 19].

In minimum-fuel space trajectory optimization, thrusting and coasting arcs arise from applying Pontryagin's principle [25]. However, the bang-off-bang thrust profile can cause singularities in the Jacobian, complicating the solution to the resulting Hamiltonian BVPs [30]. Regularization methods like Hyperbolic Tangent Smoothing (HTS) mitigate this issue, resulting in smooth control profiles that improve numerical stability [34].

This chapter compares the HTS method with a recent L2-norm-based regularization [41], analyzing their performance on two benchmark minimum-fuel low-thrust trajectory optimization problems. We evaluate the impact of accurate sensitivity calculations using finite-difference and State Transition Matrix (STM) methods, providing insights into optimal control problem solutions [44, 42].

2.1 Problem Formulation

To assess the impact of control regularization on the convergence performance of numerical methods, we consider two different parameterizations of the motion dynamics using Cartesian coordinates and the set of modified equinoctial elements (MEEs) [33, 67]. The choice of coordinates is shown to significantly impact the convergence of the BVPs associated with minimum-time and minimum-fuel space trajectory optimization problems. In addition, we consider optimizing the trajectory of spacecraft during heliocentric phases of flights with zero hyperbolic excess velocity relative to the departing and arrival bodies. We consider two-body dynamics assumptions and ignore third-body and other types of perturbations (e.g., solar-radiation pressure). The only perturbing acceleration (and at the same time control input) is due to the operation of the propulsion system during the thrusting arcs. These assumptions are consistent with the dynamical models and boundary conditions of the chosen benchmark problems [33].

2.1.1 Equations of motion

Following [67, 15], we adopt a unified control-affine representation of the dynamics. Let $\boldsymbol{x} \in \mathbb{R}^6$ denote part of the state vector (e.g., Cartesian coordinates consisting of components of the position and velocity vectors or the six modified equinoctial elements). Let $m \in \mathbb{R}^+$ denote the instantaneous mass of the spacecraft. We adopt a magnitude-vector parameterization of the control-acceleration vector. Let $\delta \in [0, 1]$ denote engine throttle input and let $\hat{\boldsymbol{\alpha}}$ denote the thrust steering unit vector (i.e., $\|\hat{\boldsymbol{\alpha}}\| = 1$), the acceleration vector produced by the propulsion system can be written as,

$$\boldsymbol{u} = \frac{T_{\text{max}}}{m}\hat{\boldsymbol{\alpha}}\delta, \tag{2.1}$$

where T_{max} denotes the maximum thrust magnitude and m is the mass of the spacecraft. Altogether, the control vector can be written as $\boldsymbol{u}^\top = [\hat{\boldsymbol{\alpha}}^\top, \delta] \in \mathbb{R}^4$. The time rate of change of $\boldsymbol{x}$

and m (with $\boldsymbol{x} = \boldsymbol{x}(t)$) can be written as,

$$\dot{\boldsymbol{x}} = \mathbb{A}(\boldsymbol{x}) + \mathbb{B}(\boldsymbol{x}) \left(\frac{T_{\text{max}}}{m} \hat{\boldsymbol{\alpha}} \delta \right), \quad \dot{m} = -\frac{T_{\text{max}}}{c} \delta, \tag{2.2}$$

where the entries of the $\mathbb{A}$ vector and the $\mathbb{B}$ matrix depend on the choice of coordinates and/or elements [33, 67]. In Eq.(2.2), $c = I_{\text{sp}} g_0$ denotes the effective exhaust velocity. In Eq. (2.1), I_{sp} is the thruster specific impulse value and g_0 is the Earth's sea-level gravitational constant. To simplify the problem formulation, it is assumed that the values of T_{max} and c remain constant throughout the trajectory. In reality, and for solar-powered low-thrust propulsion systems, these parameters depend on the power that is sent to the engine [36]. The main task, in spacecraft trajectory optimization problems, is to determine the time history of the optimal/extremal (typically denoted by superscript '*') control vector, $\boldsymbol{u}^*$, such that a performance index is minimized/maximized while equations of motions, given in Eq. (2.2), are satisfied along with additional boundary conditions and along-the-path state and control equality/inequality constraints. We proceed by defining the cost functional (a.k.a. the performance index or objective) of the minimum-fuel trajectory optimization problems.

2.1.2 Cost functional of minimum-fuel trajectory optimization problems

In this research, we focus on minimum-fuel trajectory optimization problems. In optimal control terminology, the cost functional, $J \in \mathbb{R}$, can be stated in various equivalent forms [68]. However, in this research, we consider the Lagrange form in which the cost functional is expressed in the form of an integral. For spacecraft with a fixed initial mass, minimum-fuel trajectories correspond to minimizing the propellant consumption, which can be stated mathematically as,

$$\underset{\delta \in \mathcal{U}_\delta \ \& \ \hat{\boldsymbol{\alpha}} \in \mathcal{U}_{\hat{\boldsymbol{\alpha}}}}{\text{Minimize}} \quad J = \int_{t_0}^{t_f} \frac{T_{\text{max}}}{c} \delta(t) \, dt, \tag{2.3}$$

where $\mathcal{U}_{\hat{\boldsymbol{\alpha}}} = \{\hat{\boldsymbol{\alpha}} : [t_0, +\infty) \rightarrow \boldsymbol{U} \mid \hat{\boldsymbol{\alpha}} \text{ is measurable}\}$ with $\boldsymbol{U} = \{\hat{\boldsymbol{\alpha}} \mid \|\hat{\boldsymbol{\alpha}}\| = 1\}$ and $\mathcal{U}_{\delta} = \{\delta : [t_0, +\infty) \rightarrow [0, 1] \mid \delta \text{ is measurable}\}$. The admissible set of throttle control is listed under the minimization operator.

Please note that while optimization appears to be performed only over the control vector, $\boldsymbol{u}^\top(t) = [\hat{\boldsymbol{\alpha}}^\top(t), \delta(t)]$, the optimization is actually performed over all admissible sets of states and controls. The time derivative of states (consisting of $\boldsymbol{x}$ and m) is governed by the right-hand side of the differential equations that is given in Eq. (2.2). We note that one could have written the cost functional in Mayer form, as $J = -m(t_f)$ that considers the negative of the final mass of the spacecraft, minimization of which is equivalent to maximizing the final mass for a fixed initial mass. We now can proceed by writing the optimal control problem formulation associated with the minimum-fuel trajectory optimization problems, when Cartesian coordinates and MEEs are used.

2.1.3 Formulation of the minimum-fuel problem using Cartesian coordinates

Let $\boldsymbol{r} = [x, y, z]^\top$ and $\boldsymbol{v} = [v_x, v_y, v_z]^\top$ denote the position and velocity vectors of the center of mass of the spacecraft relative to the origin of the heliocentric frame of reference. Thus, we have $\boldsymbol{x}^\top = [\boldsymbol{r}^\top, \boldsymbol{v}^\top]$. The time rate of change of states can be written as, [67]

$$\begin{aligned}
\dot{\boldsymbol{r}} &= \boldsymbol{v} = \boldsymbol{f}_{\boldsymbol{r}}, \\
\dot{\boldsymbol{v}} &= -\frac{\mu}{r^3}\boldsymbol{r} + \frac{T_{\max}}{m}\hat{\boldsymbol{\alpha}}\delta = \boldsymbol{f}_{\boldsymbol{v}}, \\
\dot{m} &= -\frac{T_{\max}}{c}\delta = f_m,
\end{aligned} \tag{2.4}$$

where μ is the gravitational parameter for the central body (Sun in our problems) and $r = \|\boldsymbol{r}\|$. Let $\boldsymbol{f}^\top = [\boldsymbol{f}_{\boldsymbol{r}}^\top, \boldsymbol{f}_{\boldsymbol{v}}^\top]$, the time rate of change of position and velocity vectors, given in Eq. (2.4), can be written as $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), t)$. The minimum-fuel optimal control problem can be written as,

$$
\begin{aligned}
&\underset{\delta \in \mathcal{U}_\delta \ \& \ \hat{\boldsymbol{\alpha}} \in \mathcal{U}_{\hat{\boldsymbol{\alpha}}}}{\text{Minimize}} \ J = \int_{t_0}^{t_f} \frac{T_{\max}}{c} \delta(t)\, dt \\
&\text{s.t.:} \\
&\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), t), \quad \dot{m} = -\frac{T_{\max}}{c}\delta, \\
&\boldsymbol{r}(t_0) - \boldsymbol{r}_0 = \boldsymbol{0}, \quad \boldsymbol{v}(t_0) - \boldsymbol{v}_0 = \boldsymbol{0}, \\
&\boldsymbol{r}(t_f) - \boldsymbol{r}_f = \boldsymbol{0}, \quad \boldsymbol{v}(t_f) - \boldsymbol{v}_f = \boldsymbol{0}, \\
&m(t_0) - m_0 = 0,
\end{aligned}
\tag{2.5}
$$

where $\boldsymbol{x}_0^\top = [\boldsymbol{r}_0^\top, \boldsymbol{v}_0^\top]$ and m_0 denote the complete states of the spacecraft at the time of departure from the departing planet (Earth) and $\boldsymbol{r}_f$ and $\boldsymbol{v}_f$ denote the fixed position and velocity vectors of the target/arrival body. In addition, $\mathcal{U}_{\hat{\boldsymbol{\alpha}}} = \{\hat{\boldsymbol{\alpha}} : [t_0, +\infty) \rightarrow \boldsymbol{U} \mid \hat{\boldsymbol{\alpha}} \text{ is measurable}\}$ with $\boldsymbol{U} = \{\hat{\boldsymbol{\alpha}} \mid \|\hat{\boldsymbol{\alpha}}\| = 1\}$ and $\mathcal{U}_\delta = \{\delta : [t_0, +\infty) \rightarrow [0, 1] \mid \delta \text{ is measurable}\}$.

We can use the indirect formalism of optimal control theory, which treats the state differential equation as constraints [68]. We proceed by introducing the costate vector associated with Cartesian position and velocity vectors as $\boldsymbol{\lambda_r} = [\lambda_x, \lambda_y, \lambda_z]^\top$ and $\boldsymbol{\lambda_v} = [\lambda_{v_x}, \lambda_{v_y}, \lambda_{v_z}]^\top$. Let $\boldsymbol{\lambda}^\top = [\boldsymbol{\lambda}_{\boldsymbol{r}}^\top, \boldsymbol{\lambda}_{\boldsymbol{v}}^\top]$ denote the vector of Cartesian coordinates and let λ_m denote the mass costate. We have dropped the argument t from the terms to make the relations more concise (e.g., $\boldsymbol{\lambda_r} = \boldsymbol{\lambda_r}(t)$). We now proceed by forming the (variational) Hamiltonian associate with the Cartesian coordinates as,

$$
H_{\mathrm{C}} = \frac{T_{\max}}{c}\delta + \boldsymbol{\lambda}_{\boldsymbol{r}}^\top \boldsymbol{f_r} + \boldsymbol{\lambda}_{\boldsymbol{v}}^\top \boldsymbol{f_v} + \lambda_m f_m. \tag{2.6}
$$

The differential equations for the costates are determined using the Euler-Lagrange equation [68] as,

$$
\dot{\boldsymbol{\lambda}} = -\left[\frac{\partial H_{\mathrm{C}}}{\partial \boldsymbol{x}}\right]^\top, \qquad \dot{\lambda}_m = -\frac{\partial H_{\mathrm{C}}}{\partial m}. \tag{2.7}
$$

Following the primer vector theory of Lawden [31] and Pontryagin's minimum principle, the extremal control expressions [43, 44] can be derived and stated compactly as,

$$\hat{\boldsymbol{\alpha}}^* = -\frac{\boldsymbol{\lambda_v}}{\|\boldsymbol{\lambda_v}\|}, \qquad \delta^* \begin{cases} = 1 & \text{if } S > 0, \\ \in [0,1] & \text{if } S = 0, \\ = 0 & \text{if } S \leq 0, \end{cases} \qquad \text{with} \quad S = \frac{c\|\boldsymbol{\lambda_v}\|}{m} + \lambda_m - 1, \tag{2.8}$$

where S is the so-called throttle *switching function*. The primer vector is $\boldsymbol{p} = -\boldsymbol{\lambda_v}$. According to Lawden, and for problems in which the direction of the thrust vector is not constrained, the time history of $\boldsymbol{\lambda_v}$ is continuous, which results in a continuous profile of $\hat{\boldsymbol{\alpha}}^*$. However, the value of δ^* depends on the sign of the throttle switching function, which results in discontinuities in δ^*. Note that the number and time instants of the zero-crossings of S are not known *a priori*. The presence of instantaneous changes in the value of δ^* poses significant challenges to numerical solvers and integrators that are used for solving the resulting BVPs.

To overcome the control discontinuity issue, smoothing functions are used, which employ squeezing continuous functions such as the hyperbolic tangent function to map the switching function between 0 and 1. In this research, we consider and compare the following smoothing functions:

1. Hyperbolic-Tangent-Based Smoothing [34]:

$$\delta^*_{\text{tanh}} \approx \delta_{\text{tanh}}(S;\rho) = 0.5\left[1 + \tanh\left(\frac{S}{\rho}\right)\right], \tag{2.9}$$

2. L2-norm-Based Smoothing [41, 69]:

$$\delta^*_{\text{L2}} \approx \delta_{\text{L2}}(S;\rho) = 0.5\left[1 + \frac{S}{\sqrt{S^2+\rho^2}}\right], \tag{2.10}$$

where ρ denotes a smoothing parameter that allows us to control the sharpness of change in throttle. The smoothing parameter, as will be shown later, will be used within a standard numerical continuation method. The principal idea, in control regularization methods, is to

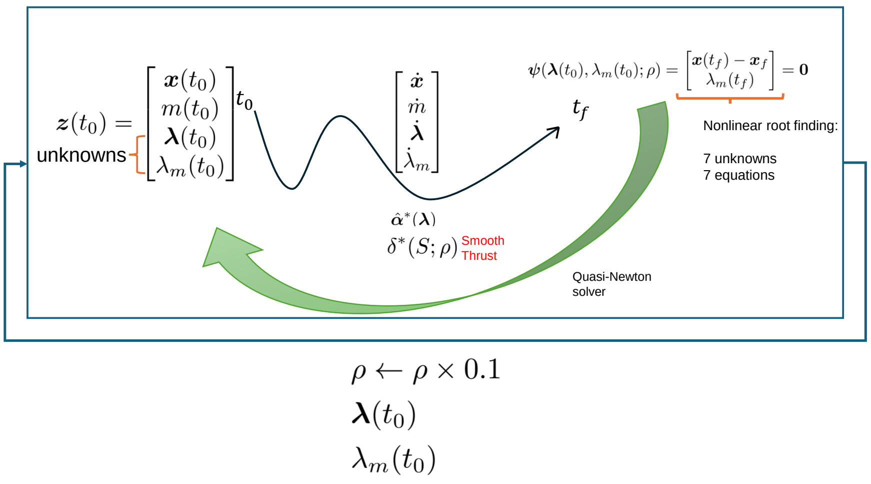

$\boldsymbol{z}(t_0) = \begin{bmatrix} \boldsymbol{x}(t_0) \\ m(t_0) \\ \boldsymbol{\lambda}(t_0) \\ \lambda_m(t_0) \end{bmatrix}$
unknowns
t_0
$\begin{bmatrix} \dot{\boldsymbol{x}} \\ \dot{m} \\ \dot{\boldsymbol{\lambda}} \\ \dot{\lambda}_m \end{bmatrix}$
t_f
$\boldsymbol{\psi}(\boldsymbol{\lambda}(t_0), \lambda_m(t_0); \rho) = \begin{bmatrix} \boldsymbol{x}(t_f) - \boldsymbol{x}_f \\ \lambda_m(t_f) \end{bmatrix} = \boldsymbol{0}$
Nonlinear root finding:
7 unknowns
7 equations
$\hat{\boldsymbol{\alpha}}^*(\boldsymbol{\lambda})$
$\delta^*(S;\rho)$
Smooth Thrust
Quasi-Newton solver
$\rho \leftarrow \rho \times 0.1$
$\boldsymbol{\lambda}(t_0)$
$\lambda_m(t_0)$

Figure 2.1: The overall idea behind the numerical continuation method. The nonlinear root finding problem is solved iteratively each time with a reduced ρ value until the optimal performance is achieved.

embed the non-smooth, piece-wise continuous optimal throttle logic, given in Eq. (2.8), into a one-parameter family of smooth curves. In the limit and as the value of ρ is decreased (from a relatively large value, e.g., $\rho = 1.0$) to small values (e.g., $\rho = 1.0 \times 10^{-3}$), the smooth throttle approaches the piece-wise continuous throttle, given in Eq. (2.8), i.e., $\delta(S;\rho) \rightarrow \delta^*$ as $\rho \rightarrow 0$(See Fig. 2.1).

The resulting one-parameter family of smooth TPBVPs can be summarized as follows: the state differential equations, given in Eq. (2.4), the costate differential equations, obtained from Eq. (2.7), the extremal thrust steering unit vector, $\hat{\boldsymbol{\alpha}}$ and the throttle switching function, given in Eq. (2.8), and the smooth throttle function (either HTS or L2-norm based), given in Eq. (2.9) or (2.10), and the transversality condition on the costate associated with mass at the final time, $\lambda_m(t_f) = 0$. The resulting TPBVP is written as a nonlinear shooting problem as,

$$\boldsymbol{\psi}(\boldsymbol{\eta}(t_0); \rho) = [\boldsymbol{r}^\top(t_f) - \boldsymbol{r}_f^\top, \boldsymbol{v}^\top(t_f) - \boldsymbol{v}_f^\top, \lambda_m(t_f)] = \boldsymbol{0}_{7\times 1}, \tag{2.11}$$

where $\boldsymbol{\eta}^\top(t_0) = [\boldsymbol{\lambda}_{\boldsymbol{r}}^\top(t_0), \boldsymbol{\lambda}_{\boldsymbol{v}}^\top(t_0), \lambda_m(t_0)] \in \mathbb{R}^7$ denotes the vector of unknown initial costates.

Typically, the resulting shooting problems are solved using single- or multiple-shooting solution schemes [29]. Note that the initial states at $t = t_0$ are known and these conditions are not added to the boundary conditions of the shooting problem. Numerical continuation methods are typically used for solving the problems by setting $\rho = 1$ and solving the shooting problem. Once a solution to $\boldsymbol{\eta}^\top(t_0)$ is obtained, the value of ρ is decreased, say by a factor of 0.1 or 0.5 (i.e., $\rho = \rho \times 0.1$). The previous value of $\boldsymbol{\eta}^\top(t_0)$ is used as an initial guess for the same shooting problem, but with a new value of ρ. These steps are repeated until the value of ρ is smaller than some prescribed user-defined value or when subsequent changes in the value of cost, J, become smaller than a threshold.

2.1.4 Formulation of the minimum-fuel problem using modified equinoctial elements (MEEs)

Let the state vector associated with the MEEs be denoted as $\boldsymbol{x} = [p,\ f,\ g,\ h,\ k,\ L]^\top$ (with an abuse of notation). The mappings between MEEs and the Keplerian classical orbital elements (COEs) are as follows [70]: $p = a(1 - e^2)$, $f = e\cos(\omega + \Omega)$, $g = e\sin(\omega + \Omega)$, $h = \tan\left(\frac{i}{2}\right)\cos(\Omega)$, $k = \tan\left(\frac{i}{2}\right)\sin(\Omega)$, $L = \theta + \omega + \Omega$, where a, e, i, Ω, ω and θ, denote the semi-major axis, eccentricity, inclination, right-ascension of the ascending note, argument of periapses, and true anomaly, respectively. The MEEs present an ideal parameterization of the evolution of low-thrust trajectories, avoiding the singularities (associated with zero inclination and eccentricity values) present in the classical orbital elements [67]. The time rate of change of MEEs and mass can be written as,

$$\dot{\boldsymbol{x}} = \mathbb{A}(\boldsymbol{x}) + \mathbb{B}(\boldsymbol{x})\left(\frac{T_{\max}}{m}\hat{\boldsymbol{\alpha}}\delta\right) \qquad \dot{m} = -\frac{T_{\max}}{c}\delta, \tag{2.12}$$

with $\mathbb{A}(\boldsymbol{x})$ and $\mathbb{B}(\boldsymbol{x})$ defined (under the canonical unit scaling with $\mu = 1.0$) as,

$$\mathbb{A}^\top(\boldsymbol{x}) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & \sqrt{p}\left(\frac{q}{p}\right)^2 \end{bmatrix}, \tag{2.13}$$

$$\mathbb{B}(\boldsymbol{x}) = \begin{bmatrix} 0 & \frac{2p}{q}\sqrt{p} & 0 \\ \sqrt{p}\sin L & \frac{\sqrt{p}}{q}\left((q+1)\cos L + f\right) & -\frac{\sqrt{p}g}{q}\left(h\sin L - k\cos L\right) \\ -\sqrt{p}\cos L & \frac{\sqrt{p}}{q}\left((q+1)\sin L + g\right) & \frac{\sqrt{p}f}{q}\left(h\sin L - k\cos L\right) \\ 0 & 0 & \frac{\sqrt{p}s^2\cos L}{2q} \\ 0 & 0 & \frac{\sqrt{p}s^2\sin L}{2q} \\ 0 & 0 & \frac{\sqrt{p}}{q}\left(h\sin L - k\cos L\right) \end{bmatrix}, \tag{2.14}$$

with $s^2 = 1 + h^2 + k^2$ and $q = 1 + f\cos L + g\sin L$. The minimum-fuel optimal control problem can be written as,

$$\begin{aligned} &\underset{\delta\in[0,1]\ \&\ \hat{\boldsymbol{\alpha}}\in\mathcal{U}_{\hat{\boldsymbol{\alpha}}}}{\text{Minimize}}\quad J = \int_{t_0}^{t_f} \frac{T_{\max}}{c}\delta(t)\,dt \\ &\text{s.t.:} \\ &\dot{\boldsymbol{x}}(t) = \mathbb{A}(\boldsymbol{x}) + \mathbb{B}(\boldsymbol{x})\left(\frac{T_{\max}}{m}\hat{\boldsymbol{\alpha}}\delta\right), \quad \dot{m}(t) = -\frac{T_{\max}}{c}\delta, \\ &\boldsymbol{x}(t_0) - \boldsymbol{x}_0 = \boldsymbol{0}, \\ &\boldsymbol{x}(t_f) - \boldsymbol{x}_f = \boldsymbol{0}, \\ &m(t_0) = m_0, \end{aligned} \tag{2.15}$$

where $\boldsymbol{x}_0^\top = [p_0, f_0, g_0, h_0, k_0, L_0]$ denotes the MEEs of the spacecraft at the time of departure from the departing planet (Earth) and $\boldsymbol{x}_f^\top = [p_f, f_f, g_f, h_f, k_f, L_f]$ denote the fixed vector of target MEE values associated with the target body at the end of the trajectory.

Let $\boldsymbol{\lambda}^\top = [\lambda_p, \lambda_f, \lambda_g, \lambda_h, \lambda_k, \lambda_L]$ (with an abuse of notation) denote the costate vector associated with the MEEs and let λ_m denote the costate associated with mass. We can proceed and form the Hamiltonian associated with the MEEs as,

$$H_{\text{MEE}} = \frac{T_{\max}}{c}\delta + \boldsymbol{\lambda}^\top\left(\mathbb{A} + \mathbb{B}\left[\frac{T_{\max}}{m}\delta\boldsymbol{\alpha}\right]\right) - \lambda_m\frac{T_{\max}}{c}\delta. \tag{2.16}$$

The differential equations for the costates associated with MEEs are determined using the Euler-Lagrange equation [68] as,

$$\dot{\boldsymbol{\lambda}} = -\left[\frac{\partial H_{\text{MEE}}}{\partial \boldsymbol{x}}\right]^\top, \qquad \dot{\lambda}_m = -\frac{\partial H_{\text{MEE}}}{\partial m}. \tag{2.17}$$

Following the steps that we followed, when we used the Cartesian coordinates, and upon using the primer vector theory of Lawden and the PMP, the extremal control expressions can be derived and written as

$$\hat{\boldsymbol{\alpha}}^* = -\frac{\mathbb{B}^\top \boldsymbol{\lambda}}{||\mathbb{B}^\top \boldsymbol{\lambda}||}, \quad \delta^* \begin{cases} = 1 & \text{if } S > 0, \\ \in [0,1] & \text{if } S = 0, \\ = 0 & \text{if } S \leq 0, \end{cases} \quad \text{with} \quad S = \frac{c||\mathbb{B}^\top \boldsymbol{\lambda}||}{m} + \lambda_m - 1, \tag{2.18}$$

where S is the so-called throttle *switching function*. It is important to notice that the primer vector, is obtained by modulating the costate vector associated with MEEs by the control influence matrix, i.e., $\boldsymbol{p} = -\mathbb{B}^\top \boldsymbol{\lambda}$. Please also note that the primer vector, when Cartesian coordiates are used, can be expressed in a similar form, i.e., $\boldsymbol{p} = -\mathbb{B}^\top \boldsymbol{\lambda}$ except that the control influence matrix for the set of Cartesian coordinates is $\mathbb{B} = [\mathbf{0}_{3\times 3}, I_{3\times 3}]$ and the costate vector is $\boldsymbol{\lambda} = [\boldsymbol{\lambda}_{\boldsymbol{r}}^\top, \boldsymbol{\lambda}_{\boldsymbol{v}}^\top]$. Thus, we have $\boldsymbol{p} = -\mathbb{B}^\top \boldsymbol{\lambda} = -\boldsymbol{\lambda}_{\boldsymbol{v}}$, which is identical to the primer vector relation given below Eq. (2.8).

The resulting smooth TPBVP is summarized as follows: the state differential equations, given in Eq. (2.12), the costate differential equations, obtained from Eq. (2.17), the extremal thrust steering unit vector, $\hat{\boldsymbol{\alpha}}$ and the throttle switching function, given in Eq. (2.18), and the smooth throttle function (either HTS or L2-norm based), follow the same relations given in Eq. (2.9) or (2.10) except that the switching function is calculated using the relation given in Eq. (2.18), and the transversality condition on the costate associated with mass at the final time, $\lambda_m(t_f) = 0$. The resulting TPBVP is written as a shooting problem as

$$\boldsymbol{\psi}(\boldsymbol{\eta}(t_0); \rho) = [\boldsymbol{x}^\top(t_f) - \boldsymbol{x}_f^\top, \lambda_m(t_f)] = \mathbf{0}_{7\times 1}, \tag{2.19}$$

where $\boldsymbol{\eta}^\top(t_0) = [\boldsymbol{\lambda}^\top(t_0), \lambda_m(t_0)] \in \mathbb{R}^7$ denotes the vector of unknown initial costates. Numerical continuation methods are typically used for solving the problems by setting $\rho = 1$ and solving the shooting problem. Once a solution to $\boldsymbol{\eta}^\top(t_0)$ is obtained. The value of ρ is decreased, say by a factor of 0.1 or 0.5 (i.e., $\rho \leftarrow \rho \times 0.1$). The previous value of $\boldsymbol{\eta}^\top(t_0)$ is used as an initial guess for this slightly-modified shooting problem. These steps are repeated until the value of ρ is smaller than some prescribed user-defined value or when subsequent changes in the value of cost, J, become smaller than a user-defined threshold.

2.2 Calculation of Sensitivities Using State Transition Matrix

The TPBVPs associated with minimum-fuel trajectory optimization problems, resulting in nonlinear shooting problems, can be written as,

$$\text{Cartesian:} \quad \boldsymbol{\psi}(\boldsymbol{\eta}(t_0); \rho) = \begin{bmatrix} \boldsymbol{r}(t_f) - \boldsymbol{r}_f \\ \boldsymbol{v}(t_f) - \boldsymbol{v}_f \\ \lambda_m(t_f) \end{bmatrix} = \boldsymbol{0}, \tag{2.20}$$

$$\text{MEE:} \quad \boldsymbol{\psi}(\boldsymbol{\eta}(t_0); \rho) = \begin{bmatrix} \boldsymbol{x}(t_f) - \boldsymbol{x}_f \\ \lambda_m(t_f) \end{bmatrix} = \boldsymbol{0}. \tag{2.21}$$

In these problems, the initial states, including the initial mass value, are known. Our objective is to determine the initial costate vector, denoted as $\boldsymbol{\eta}(t_0)$. While numerous off-the-shelf root-finding solvers are available, we utilize Powell's dog-leg method, a well-regarded choice for this category of problems. Powell's method is a gradient-based numerical solver that leverages the Jacobian to solve a nonlinear system of equations. If the Jacobian function is not provided, it approximates the Jacobian using a Finite Difference (FD) method. Although FD is often effective, it can introduce instability and inefficiency into the solver [71]. Therefore, it is often advantageous to provide Powell's method with accurate values for the Jacobian of the residual error vector with respect to the unknown decision values.

In our trajectory optimization problems, given the TPBVP, and for a fixed value of ρ, we intend to find the Jacobian of the final boundary conditions w.r.t. the initial costates, which can be written as,

$$\frac{\partial \boldsymbol{\psi}(\boldsymbol{x}(t_f), \lambda_m(t_f), t_f)}{\partial \boldsymbol{\eta}(t_0)}. \tag{2.22}$$

To find the Jacobian, we utilize the State Transition Matrix (STM) [72]. Let $\boldsymbol{z}$ denote the states of a continuous set of differential equations denoted as,

$$\dot{\boldsymbol{z}}(t) = \boldsymbol{\Gamma}(\boldsymbol{z}(t), t). \tag{2.23}$$

We intend to find the Jacobian of $\boldsymbol{z}(t)$ w.r.t. $\boldsymbol{z}(t_0)$, which can be written as,

$$\boldsymbol{\Phi}(t, t_0) = \frac{\partial \boldsymbol{z}(t)}{\partial \boldsymbol{z}(t_0)}. \tag{2.24}$$

The solution to the differential equation that is given in Eq. (2.23) can be written as,

$$\boldsymbol{z}(t) = \boldsymbol{z}(t_0) + \int_{t_0}^{t} \boldsymbol{\Gamma}(\boldsymbol{z}(t), t) dt. \tag{2.25}$$

We can take the derivative of both sides of Eq. (2.25), and using the definition given in Eq. (2.24), we can write

$$\frac{\partial \boldsymbol{z}(t)}{\partial \boldsymbol{z}(t_0)} = I + \int_{t_0}^{t} \frac{\partial \boldsymbol{\Gamma}(\boldsymbol{z}(t), t)}{\partial \boldsymbol{z}(t)} \frac{\partial \boldsymbol{z}(t)}{\partial \boldsymbol{z}(t_0)} dt, \tag{2.26}$$

which following the definition of the sensitivity matrix given in Eq. (2.24), can be written as,

$$\boldsymbol{\Phi}(t, t_0) = I + \int_{t_0}^{t} \frac{\partial \boldsymbol{\Gamma}(\boldsymbol{z}(t), t)}{\partial \boldsymbol{z}(t)} \boldsymbol{\Phi}(t, t_0) dt. \tag{2.27}$$

We proceed by taking the time derivative of both sides to obtain the differential equation that describes the time derivative of the sensitivity matrix as,

$$\dot{\boldsymbol{\Phi}}(t, t_0) = \frac{\partial \boldsymbol{\Gamma}(\boldsymbol{z}(t), t)}{\partial \boldsymbol{z}(t)} \boldsymbol{\Phi}(t, t_0), \quad \text{with} \quad \boldsymbol{\Phi}(t_0, t_0) = I. \tag{2.28}$$

In the context of the minimum-fuel trajectory optimization problems, $\boldsymbol{z}$ corresponds to the vector consisting of states $\boldsymbol{x}$, m, costates associated with $\boldsymbol{x}$, denoted as $\boldsymbol{\lambda}$, and the costate associated with mass, λ_m.

Let $\boldsymbol{z}^\top = [\boldsymbol{x}^\top, m, \boldsymbol{\lambda}^\top, \lambda_m] \in \mathbb{R}^{14}$. From Eq. (2.22) and using the chain rule, we have

$$\left[\frac{\partial \boldsymbol{\psi}(t_f)}{\partial \boldsymbol{\eta}(t_0)}\right]_{7\times 7} = \left[\frac{\partial \boldsymbol{\psi}(t_f)}{\partial \boldsymbol{z}(t_f)}\right]_{7\times 14} \underbrace{\left[\frac{\partial \boldsymbol{z}(t_f)}{\partial \boldsymbol{z}(t_0)}\right]_{14\times 14}}_{=\Phi(t_f,t_0)} \left[\frac{\partial \boldsymbol{z}(t_0)}{\partial \boldsymbol{\eta}(t_0)}\right]_{14\times 7}. \tag{2.29}$$

The elements of the $\partial \boldsymbol{\Gamma}(\boldsymbol{z}(t))/\partial \boldsymbol{z}(t)$ matrix for the Cartesian coordinates in Eq. (2.28), can be derived as follows:

$$\frac{\partial \boldsymbol{\Gamma}}{\partial \boldsymbol{z}} = \begin{bmatrix} \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times1} \\ \mathbf{F}_{21} & \mathbf{0}_{3\times3} & \mathbf{F}_{23} & \mathbf{0}_{3\times3} & \mathbf{F}_{25} & \mathbf{F}_{26} \\ \mathbf{0}_{1\times3} & \mathbf{0}_{1\times3} & F_{33} & \mathbf{0}_{1\times3} & \mathbf{F}_{35} & F_{36} \\ \mathbf{F}_{41} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times3} & \mathbf{F}_{45} & \mathbf{0}_{3\times1} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times1} & -\mathbf{I} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times1} \\ \mathbf{0}_{1\times3} & \mathbf{0}_{1\times3} & F_{63} & \mathbf{0}_{1\times3} & \mathbf{F}_{65} & F_{66} \end{bmatrix}. \tag{2.30}$$

$$\mathbf{F}_{21} = \frac{3\mu \boldsymbol{r}\boldsymbol{r}^\top}{||\boldsymbol{r}||^5} - \frac{\mu}{||\boldsymbol{r}||^3}\boldsymbol{I}_{3\times3}\ , \ \boldsymbol{F}_{23} = \frac{T_{\max}}{m^2||\boldsymbol{\lambda_v}||}\boldsymbol{\lambda_v}\delta(S) + \frac{T_{\max}c}{m^3}\boldsymbol{\lambda_v}\delta'(S)\ ,$$

$$\boldsymbol{F}_{25} = \frac{T_{\max}\boldsymbol{\lambda_v}\boldsymbol{\lambda_v}^\top}{m||\boldsymbol{\lambda_v}||^3}\delta(S) - \frac{T_{\max}}{m||\boldsymbol{\lambda_v}||}\boldsymbol{I}_{3\times3}\delta(S) - \frac{T_{\max}c}{(m||\boldsymbol{\lambda_v}||)^2}\boldsymbol{\lambda_v}\boldsymbol{\lambda_v}^\top\delta'(S)\ , \ \boldsymbol{F}_{26} = -\frac{T_{\max}}{m||\boldsymbol{\lambda_v}||}\boldsymbol{\lambda_v}\delta'(S),$$
$$F_{33} = \frac{T_{\max}}{m^2}||\boldsymbol{\lambda_v}||\delta'(S)\ , \ \boldsymbol{F}_{35} = \frac{T_{\max}}{m||\boldsymbol{\lambda_v}||}\boldsymbol{\lambda_v}^\top\delta'(S)\ , \ F_{36} = -\frac{T_{\max}}{c}\delta'(S),$$

$$\boldsymbol{F}_{41} = \frac{15\mu\boldsymbol{r}\boldsymbol{r}^\top\boldsymbol{\lambda_v}\boldsymbol{r}^\top}{||\boldsymbol{r}||^7} - \frac{3\mu\mathbf{1}_{3\times1}\boldsymbol{r}^\top\boldsymbol{\lambda_v}}{||\boldsymbol{r}||^5} - \frac{3\mu\boldsymbol{r}\mathbf{1}_{1\times3}\boldsymbol{\lambda_v}}{||\boldsymbol{r}||^5} - \frac{3\mu\boldsymbol{\lambda_v}\boldsymbol{r}^\top}{||\boldsymbol{r}||^5}\ , \ \boldsymbol{F}_{45} = -\frac{3\mu\boldsymbol{r}\boldsymbol{r}^\top}{||\boldsymbol{r}||^5} + \frac{\mu}{||\boldsymbol{r}||^3}\boldsymbol{I}_{3\times3}$$
$$F_{63} = \frac{2T_{\max}}{m^3}||\boldsymbol{\lambda_v}||\delta(S) + \frac{T_{\max}c}{m^4}||\boldsymbol{\lambda_v}||^2\delta'(S)\ , \ \boldsymbol{F}_{65} = -\frac{T_{\max}}{m^2||\boldsymbol{\lambda_v}||}\boldsymbol{\lambda_v}^\top\delta(S) - \frac{T_{\max}c}{m^3}\delta'(S)\boldsymbol{\lambda}^\top,$$
$$F_{66} = -\frac{T_{\max}}{m^2}||\boldsymbol{\lambda_v}||\delta'(S),$$

where S is the switching function and $\delta(S)$ is the smoothing function and $\delta'(S) = \dfrac{\partial \delta}{\partial S}$. For each smoothing method, δ and δ' are defined as,

1. Hyperbolic-Tangent-Based Smoothing:

$$\delta_{\text{tanh}}(S) = 0.5 \left[1 + \tanh\left(\frac{S}{\rho}\right)\right], \qquad \delta'_{\text{tanh}}(S) = \frac{0.5}{\rho}\text{sech}^2\left(\frac{S}{\rho}\right). \tag{2.31}$$

2. L2-norm-Based Smoothing [41]:

$$\delta_{\text{L2}}(S) = 0.5 \left[1 + \frac{S}{\sqrt{S^2 + \rho^2}}\right], \qquad \delta'_{\text{L2}}(S) = 0.5\frac{\rho^2}{(S^2 + \rho^2)^{\frac{3}{2}}}. \tag{2.32}$$

One of the key implementation steps in this derivation is that, the state transition matrix is obtained after substituting the smooth optimal throttle into the right-hand side of the differential equations to respect the *continuous* state assumption in the STM derivation [73]. The idea of substituting the smooth control is first introduced in [33] to avoid developing an event-detection logic as part of the solution of the resulting TPBVPs. However, the authors in [33] used logarithmic and extended logarithmic smoothing methods.

Please note that if the smooth control is not substituted in the right-hand side of the differential equations, an event-detection logic is required to determine the precise zero-crossings of the throttle switching function [74] to not only update the control value, but also to determine a transition matrix that is required to updating the value of the STM matrix after the control switch has occurred.

The details involved in deriving the relations for updating the state transition matrix are given in [75]. Finally, after forming the equations for both $\dot{\boldsymbol{z}}$ and $\dot{\boldsymbol{\Phi}}$, the system of differential equations consists of 210 differential equations (7 for states and mass, 7 for costates, and $14 \times 14 = 196$ for the Jacobian matrix $\boldsymbol{\Phi}$). For a randomly initialized costates, we obtain the Jacobian and boundary conditions and reiterate over initial costates using Powell's method to search for the solution that satisfies the boundary conditions. The solution procedure is similar to what is outlined in [44]. The results of applying the advanced indirect methods will be presented in Chapter 4.

Chapter 3

Optimizing Low-Thrust Trajectories Using Successive Convexification

In contrast with the indirect optimization methods, which utilize the optimality conditions to derive control in an indirect manner, direct optimization methods tackle the optimality problem through *direct manipulation of decision variables* [11]. To do so, one needs to transcribe the continuous-time optimization problem into a finite-dimensional parametric optimization problem through various transcription schemes. The resulting parameters constitute the elements of the entire decision vector, the decision vector is optimized/tuned to optimize the problem's objective. This transforms the continuous-time optimal control problem into a finite-dimensional Nonlinear programming (NLP) problem [76].

Significant progresses have been made in solving practical and challenging optimal control problems using direct methods [77]. Direct methods offer indispensable advantages compared to indirect methods [78, 79]. Most notably, a subclass of direct optimization methods, known as convex optimization problems can be formulated and solved efficiently using *Convex Optimization* [21], which offers to solve various trajectory optimization problems in a matter of milliseconds [23].

Convex optimization represents one of the state-of-the-art in rapid path-planning/guidance algorithms [80, 81]. These methods make up for the computational inefficiencies that come with direct methods by casting and solving the problem (if possible) in a convex form. Because of the efficiency of the state-of-the-art interior-point methods (IPMs), convex programming problems can be solved extremely fast and are guaranteed to converge in polynomial time, making them highly desirable for autonomous operations.

More recently, convex optimization methods have been applied to low-thrust spacecraft trajectory optimization. Wang and Grant present the minimum-time [82] and minimum-fuel [83] convex optimization solutions. Tang et al. [84] show that the adjoined variables recovered from convex optimization low-thrust solutions can be used to initialize the adjoint variables in an indirect-based shooting algorithm. Kayama et al. [85] present convex optimization solutions to low-thrust trajectory problems under three-body dynamics. Hofmann and Topputo [86] have also investigated convex optimization methods for solving low-thrust trajectory optimization problems.

Nurre and Taheri have conducted the first study to compare the application of convex optimization using the set of MEEs and Cartesian coordinates and compared the results of a convex optimization method against an indirect method for minimum-fuel low-thrust trajectories [14]. Inspired by the work in [14], this chapter covers the process of solving the inherently nonlinear optimization of minimum-fuel low-thrust trajectories by converting the original problem into a convex one and presents an advanced convex optimization method, which uses a trust-region based algorithm. We also show that solutions obtained by the proposed convex optimization methods have a smaller optimality gap compared to the theoretically optimal solutions obtained using indirect methods.

3.1 Convexifying the Continuous-time Original Control Problem

Let $\boldsymbol{T} \in \mathbb{R}^3$ denote the thrust vector produced by the spacecraft propulsion system. As in the previous chapter, consider the following minimum-fuel trajectory optimization problem:

$$\begin{aligned}
&\underset{\boldsymbol{T}}{\text{Minimize}} \quad J = \int_{t_0}^{t_f} \|\boldsymbol{T}(t)\|_2 \, dt \\
&\text{s.t.:} \\
&\dot{\boldsymbol{x}}(t) = \mathbb{A}(\boldsymbol{x}) + \mathbb{B}(\boldsymbol{x})\frac{\boldsymbol{T}}{m}, \quad \dot{m} = -\frac{\|\boldsymbol{T}\|_2}{c}, \\
&\|\boldsymbol{T}(t)\|_2 \le T_{\max}, \\
&\boldsymbol{x}(t_0) - \boldsymbol{x}_0 = \boldsymbol{0}, \qquad \boldsymbol{x}(t_f) - \boldsymbol{x}_f = \boldsymbol{0}, \\
&m(t_0) - m_0 = 0.
\end{aligned} \tag{3.1}$$

The primary difference, in this formulation, is that the magnitude of the thrust vector and its direction are not decoupled. Aside from this, we continue to address the same TP-BVP, with the state and mass represented by $\boldsymbol{x} \in \mathbb{R}^6$ and $m \in \mathbb{R}^+$, respectively. Several steps are required to enable convex optimization, since the dynamics are continuous and highly nonlinear. Although the derivations are based on the MEEs, the general control-affine form $\dot{\boldsymbol{x}}(t) = \mathbb{A}(\boldsymbol{x}) + \mathbb{B}(\boldsymbol{x})\dfrac{\boldsymbol{T}}{m}$ can be applied to any choice of coordinate system and/or element sets, as it is demonstrated in [14] for the set of spherical coordinates.

3.1.1 Introducing Logarithm of Mass

To achieve effective convexification, it is preferable to apply transformations that minimize any potential loss of optimality. One such method was previously introduced in [49], which was also adopted in [87]. This method utilizes the logarithm of mass as an alternative to its original counterpart. In the dynamics, the term $\mathbb{B}(\boldsymbol{x})\dfrac{\boldsymbol{T}}{m}$ couples, mass, thrust, and other states (through the $\mathbb{B}(\boldsymbol{x})$ matrix). Let $\boldsymbol{\tau} \in \mathbb{R}^3$ denote the acceleration vector defined as $\boldsymbol{\tau} := \dfrac{\boldsymbol{T}}{m}$, and logarithm of mass $(z := \ln m)$, the dynamics can be re-written as:

$$\dot{\boldsymbol{x}}(t) = \mathbb{A}(\boldsymbol{x}) + \mathbb{B}(\boldsymbol{x})\boldsymbol{\tau}, \quad \dot{z} = \frac{\dot{m}}{m} = -\frac{\|\boldsymbol{T}\|_2}{cm} = -\frac{1}{c}\frac{\|\boldsymbol{T}\|_2}{m} = -\frac{\|\boldsymbol{\tau}\|_2}{c}, \tag{3.2}$$

in which the thrust and mass are decoupled causing more stability in numerical solutions later. However, the original second-order cone constraint on thrust (i.e., $\|\boldsymbol{T}\|_2 \leq T_{\max}$), has taken a new form:

$$\|\boldsymbol{\tau}\|_2 = \frac{\|\boldsymbol{T}\|_2}{m} \leq \frac{T_{\max}}{m} = T_{\max}e^{-z}, \tag{3.3}$$

which is not a convex constraint and requires linearization. Given a reference pseudo mass $\hat{z}$, we can approximate the right-hand side of Eq. (3.3) as follows:

$$\|\boldsymbol{\tau}\|_2 \leq T_{\max}e^{-\hat{z}}[1 - (z - \hat{z})]. \tag{3.4}$$

A notable property of the relation in Eq. (3.4) is that it imposes a stronger constraint on the upper bound than it is defined in the original problem. To be more specific, any feasible solution to the linearized problem is also feasible for the original one. This can be demonstrated using Taylor's Theorem:

$$e^{-z} = e^{-\hat{z}}\left[1 - (z - \hat{z})\right] + e^{z_*}\frac{(z - \hat{z})^2}{2}, \tag{3.5}$$

for some $z_* \in [\hat{z}, z]$. Given that the term $e^{z_*}\frac{(z - \hat{z})^2}{2} \geq 0$, it follows that:

$$e^{-\hat{z}}\left[1 - (z - \hat{z})\right] \leq e^{-z}. \tag{3.6}$$

3.1.2 Relaxing Control Constraint

Another challenge in the formulation of a convex optimization problem is the presence of the 2-norm ($\|\boldsymbol{\tau}\|_2$) in Eq. (3.2) as well as in the objective function of Eq. (3.1). This corresponds to a constraint on the surface of the cone, which is not convex. To enable convexification, the "epigraph approach" approach is used [21], which refers to a method where a convex optimization problem is reformulated by using the epigraph of the objective function. To achieve the goal, a slack variable is introduced in place of the magnitude, and the constraint is relaxed:

$$\|\boldsymbol{\tau}\|_2 \leq \tau, \tag{3.7}$$

where τ now represents the magnitude of the thrust vector and the constraint can be solved as part of a second-order cone program (SOCP) with the following constraints:

$$\dot{z} = -\frac{\tau}{c}, \qquad\qquad \|\boldsymbol{\tau}\|_2 \leq \tau. \tag{3.8}$$

Note that $\dot{z} = -\tau/c$, is now a linear constraint n terms of the decision variables, z and τ. A feasible solution to the original problem remains feasible in this reformulated problem; however, the reverse is not necessarily true. Nevertheless, it is shown in [49] that the Hamiltonian for this problem (verified also in Chapter 4) assumes a linear form, which ensures that the optimal control follows a bang-off-bang structure. Consequently, the actual thrust magnitude resides on the surface of the constraint in Eq. (3.7).

3.1.3 Linearization of the Dynamics

Addressing the nonlinearities in the dynamics of Eq. (3.2) is still necessary. To accomplish this, we use the idea of successive convexification [16]. More specifically, we linearize the equations of motion around a reference state and thrust, denoted by $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{\tau}})$. These references are initially set and are subsequently updated using the solution from the previous optimization step in each successive iteration and is repeated until certain convergence criteria are satisfied.

In this work, the state reference is initialized by a linear interpolation between $\boldsymbol{x}(t_0)$ and $\boldsymbol{x}(t_f)$; the control reference is initially set to $\mathbf{0}$ (although other values could have been considered). For any well-behaved dynamics, it is possible to perform linearization around these reference values using a Taylor series expansion:

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \approx A_L(t)\boldsymbol{x}(t) + B_L(t)\boldsymbol{u}(t) + \boldsymbol{d}(t), \tag{3.9a}$$

$$A_L(t) := \frac{\partial}{\partial \boldsymbol{x}} \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) \bigg|_{\hat{\boldsymbol{x}}(t), \hat{\boldsymbol{u}}(t)}, \tag{3.9b}$$

$$B_L(t) := \frac{\partial}{\partial \boldsymbol{u}} \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) \bigg|_{\hat{\boldsymbol{x}}(t), \hat{\boldsymbol{u}}(t)}, \tag{3.9c}$$

$$\boldsymbol{d}(t) := \boldsymbol{f}(\hat{\boldsymbol{x}}(t), \hat{\boldsymbol{u}}(t)) - A_L(t)\hat{\boldsymbol{x}}(t) - B_L(t)\hat{\boldsymbol{u}}(t). \tag{3.9d}$$

Upon applying the linearization to the governing equations of spacecraft motion, this will result in the following approximation:

$$\dot{\boldsymbol{x}}(t) \approx \mathbb{A}(\hat{\boldsymbol{x}}) + \frac{\partial \mathbb{A}}{\partial \boldsymbol{x}} \bigg|_{\hat{\boldsymbol{x}}} (\boldsymbol{x} - \hat{\boldsymbol{x}}) + \left[\frac{\partial \mathbb{B}}{\partial \boldsymbol{x}} \bigg|_{\hat{\boldsymbol{x}}_{IJK}} \hat{\boldsymbol{\tau}}_J (\boldsymbol{x} - \hat{\boldsymbol{x}})_K \right]_I + \mathbb{B}(\hat{\boldsymbol{x}})\boldsymbol{\tau}. \tag{3.10}$$

The third term, on the right-hand side, is computed using a tensor contraction method (over the involved dimensions I, J, and K, where I and K correspond to the dimension of $\boldsymbol{x}$, and J corresponds to the dimension of $\boldsymbol{\tau}$) known as the Einstein summation, which is explained in Appendix A. For the dynamics under consideration, the matrices defined in Eq. (3.9) are given by:

$$A_L(t) = \left.\frac{\partial \mathbb{A}}{\partial \boldsymbol{x}}\right|_{\hat{\boldsymbol{x}}} + \left[\left.\frac{\partial \mathbb{B}}{\partial \boldsymbol{x}}\right|_{\hat{\boldsymbol{x}}_{IJK}} \hat{\boldsymbol{\tau}}_J\right]_{IK}, \tag{3.11a}$$

$$B_L(t) = \mathbb{B}(\hat{\boldsymbol{x}}), \tag{3.11b}$$

$$\boldsymbol{d}(t) = \mathbb{A}(\hat{\boldsymbol{x}}) - \left.\frac{\partial \mathbb{A}}{\partial \boldsymbol{x}}\right|_{\hat{\boldsymbol{x}}} \hat{\boldsymbol{x}}. \tag{3.11c}$$

Note that the term $\left.\frac{\partial \mathbb{A}}{\partial \boldsymbol{x}}\right|_{\hat{\boldsymbol{x}}} (\boldsymbol{x} - \hat{\boldsymbol{x}})$ does not need a tensor contraction since $\mathbb{A}(\boldsymbol{x})$ is a vector (see Eq. (2.13)) and its derivative results in a matrix. Finally, the original nonlinear dynamics are linearized and expressed in an affine form, and we have reduced the original non-convex optimization problem into an approximated continuous-time convex one:

$$\begin{aligned}
&\underset{\boldsymbol{\tau}}{\text{Minimize}}\ J = \int_{t_0}^{t_f} \tau(t)\, dt, \\
&\text{s.t.:} \\
&\dot{\boldsymbol{x}}(t) = A_L(t)\boldsymbol{x}(t) + B_L(t)\boldsymbol{\tau}(t) + \boldsymbol{d}(t), \quad \dot{z} = -\frac{\tau}{c}, \\
&\|\boldsymbol{\tau}\|_2 \le \tau, \quad \tau \le T_{\max} e^{-\hat{z}}[1 - (z - \hat{z})], \\
&\boldsymbol{x}(t_0) - \boldsymbol{x}_0 = \boldsymbol{0}, \\
&\boldsymbol{x}(t_f) - \boldsymbol{x}_f = \boldsymbol{0}, \\
&z(t_0) - \ln(m_0) = 0.
\end{aligned} \tag{3.12}$$

Before proceeding with the discretization step, it is important to clarify a subtle difference between Problem (3.12) and Problem (3.1): the objectives in these two problems are not exactly identical. To demonstrate that the objective of both problems are equivalent, we can rewrite the objective in (3.12) as shown in [87] as:

$$J = \int_{t_0}^{t_f} \tau(t)\, dt = -c \int_{t_0}^{t_f} \dot{z}(t)\, dt = -c \int_{t_0}^{t_f} \frac{\dot{m}(t)}{m(t)}\, dt = -c \ln \frac{m(t_f)}{m(t_0)}, \tag{3.13}$$

and thus,

$$m(t_f) = m(t_0) \exp\left[-\frac{1}{c}\int_{t_0}^{t_f} \tau(t)\, dt\right]. \tag{3.14}$$

Equation (3.14) shows that minimizing the magnitude of the new acceleration vector, τ, along the entire time horizon, is equivalent to maximizing the final mass, and ultimately, minimizing the original thrust (note that $c > 0$).

3.2 Discretization of Continuous-time Convex Problems

To solve Problem (3.12) using convex optimization, it is necessary to discretize both the dynamics and the objective and the constraints. This step turns the infinite-dimensional optimization problem into a finite-dimensional parametric optimization problem. Over the past decades, efficient interior-point algorithms are developed for solving convex optimization problems.

Following the approach in [88], the state transition matrix (STM), introduced in Chapter 2, along with convolution operations, is used to obtain an accurate discretized formulation suitable for convex optimization. The trajectory is discretized into K evenly distributed discretization points. Let us define the following sets for convenience:

$$\begin{aligned} \mathcal{K} &:= \{0, 1, \dots, K-2, K-1\}, \\ \bar{\mathcal{K}} &:= \{0, 1, \dots, K-3, K-2\}. \end{aligned} \tag{3.15}$$

Previous studies assume a first-order hold on control over each time step to preserve more feasibility. Thus, over any time segment $t \in [t_k, t_{k+1}]$, the acceleration vector, $\boldsymbol{\tau}(t)$, can be expressed in terms of $\boldsymbol{\tau}_k := \boldsymbol{\tau}(t_k)$ and $\boldsymbol{\tau}_{k+1} := \boldsymbol{\tau}(t_{k+1})$ as follows:

$$\boldsymbol{\tau}(t) := \alpha_k(t)\boldsymbol{\tau}_k + \beta_k(t)\boldsymbol{\tau}_{k+1}, \quad t \in [t_k, t_{k+1}], \quad \forall k \in \bar{\mathcal{K}}, \tag{3.16a}$$

$$\alpha_k(t) := \left(\frac{t_{k+1} - t}{t_{k+1} - t_k}\right), \tag{3.16b}$$

$$\beta_k(t) := \left(\frac{t - t_k}{t_{k+1} - t_k}\right). \tag{3.16c}$$

Following the details of the derivation of the STM in Chapter 2, $\Phi_A(t_{k+1}, t_k)$ is donated as the STM and describes the zero-input solution from $\boldsymbol{x}_k := \boldsymbol{x}(t_k)$ to $\boldsymbol{x}_{k+1} := \boldsymbol{x}(t_{k+1})$. The same

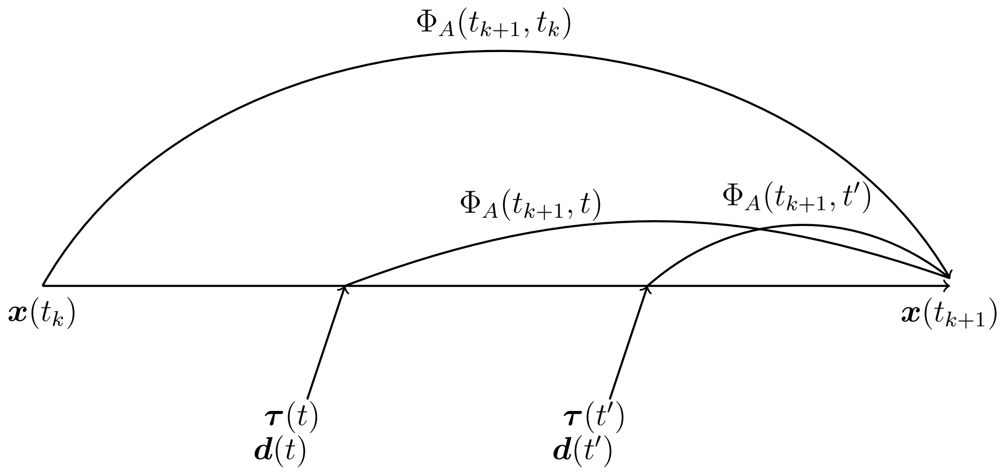


Figure 3.1: Diagram showing how $\boldsymbol{x}(t_{k+1})$ is constructed. While $\Phi_A(t_{k+1}, t_k)$ maps $\boldsymbol{x}(t_k)$ to $\boldsymbol{x}(t_{k+1})$, $\Phi(t_{k+1}, t)$ and $\Phi(t_{k+1}, t')$ maps two instances of control and independent to the final state through convolution.

differential equations are used:

$$\dot{\Phi}_A(t, t_k) = A_L(t)\Phi_A(t, t_k), \quad \Phi_A(t_k, t_k) = I, \quad \forall k \in \bar{\mathcal{K}}, \tag{3.17}$$

The discrete-time dynamics can be expressed as:

$$\boldsymbol{x}_{k+1} = \bar{A}_k \boldsymbol{x}_k + \bar{B}_k \boldsymbol{\tau}_k + \bar{C}_k \boldsymbol{\tau}_{k+1} + \bar{\boldsymbol{d}}_k, \quad \forall k \in \bar{\mathcal{K}}, \tag{3.18a}$$

$$\bar{A}_k := \Phi_A(t_{k+1}, t_k), \tag{3.18b}$$

$$\bar{B}_k := \int_{t_k}^{t_{k+1}} \Phi_A(t_{k+1}, t) B_L(t) \alpha_k(t) dt, \tag{3.18c}$$

$$\bar{C}_k := \int_{t_k}^{t_{k+1}} \Phi_A(t_{k+1}, t) B_L(t) \beta_k(t) dt, \tag{3.18d}$$

$$\bar{\boldsymbol{d}}_k := \int_{t_k}^{t_{k+1}} \Phi_A(t_{k+1}, t) \boldsymbol{d}(t) dt, \tag{3.18e}$$

where the convolution operator was applied to account for the effect of each control input and constant perturbation on $\boldsymbol{x}(t_{k+1})$ (see Fig. 3.1). In these equations, the integrands are functions of t, while the references (that we monitor) are discrete and finite. To approximate the continuous solution, the discrete references for each segment are used to construct a continuous trajectory by incorporating the trajectory itself as part of the integration, resulting in a more accurate representation of continuous dynamics in the discrete formulation.

Note that for practical purposes, it is not efficient to calculate $\Phi(t_{k+1}, t)$ for all values of t. Also note that, for Φ itself, we only monitor $\Phi(t, t_k)$ for all values of t, which is in the opposite direction of the sense of increase in the dummy variable of integration. To address this, [89] utilizes the following identity:

$$\Phi(t_{k+1}, t) = \Phi(t_{k+1}, t_k)\Phi^{-1}(t, t_k), \tag{3.19}$$

which, for instance, converts Eq. (3.18c) into:

$$\bar{B}_k \coloneqq \Phi_A(t_{k+1}, t_k) \int_{t_k}^{t_{k+1}} \Phi_A^{-1}(t, t_k) B_L(t) \alpha_k(t)\, dt. \tag{3.20}$$

Finally, the constraints in our problem are applied at each discrete point along the trajectory and control, making the problem ready to be solved using a convex optimization solver.

3.3 Successive Convex Optimization & Trust Region

Due to the convexification process, particularly the linearization of dynamics, the original problem is reduced to a subproblem that is not an entirely accurate representation of the initial formulation. Similar to Newton's method and other iterative approaches, this convex subproblem serves as an intermediate solution and is solved iteratively.

Through repeated iterations of Successive Convex Optimization (SCVX), the convex subproblem converges to a solution that, due to the proximity of the linear approximation, accurately represents a solution to the original problem. To effectively manage deviation from the reference solution, trust-region constraints are commonly applied to restrict the update step sizes (within each iteration of the SDCVX) to within a neighborhood of the reference solution. In this research, we focus primarily on adaptive trust-region methods.

3.3.1 Convex Subproblem

One practical concern is the issue of *artificial infeasibility* [89, 22, 90, 14]. The artificial infeasibility occurs when, despite the existence of a feasible solution to the original nonlinear dynamics, the problem becomes infeasible after linearization is applied. To address this, slack variables are introduced to help the convex optimization process find a solution, because the slack variables play the role of "virtual" controls and are meant to prevent the potential infeasibility caused by the linearization.

The use of slack variables is then penalized (with the use of a properly chosen norm) to ensure that, when a feasible trajectory exists, the slack variables approach zero. Eventually and on a converged solution, the penalty term will be effectively negligible, meaning that the obtained states and controls are accurate enough to satisfy the equations of motion, and therefore the solution is a dynamically feasible one.

Let $\boldsymbol{h}_k \in \mathbb{R}^6 \; \forall k \in \bar{\mathcal{K}}$ denote the vector of slack variables. Finally, constructing the convex subproblem:

$$\underset{\boldsymbol{\tau}_k, \tau_k, \boldsymbol{x}_k, z_k, \boldsymbol{h}_k}{\text{Minimize}} \; L = \sum_{k \in \bar{\mathcal{K}}} \frac{\tau_k + \tau_{k+1}}{2}(t_{k+1} - t_k) + C \sum_{k \in \bar{\mathcal{K}}} \sum_{i=0}^{5} |h_{k,i}|, \tag{3.21a}$$

$$\text{s.t.:} \tag{3.21b}$$

$$\boldsymbol{x}_{k+1} = \bar{A}_k \boldsymbol{x}_k + \bar{B}_k \boldsymbol{\tau}_k + \bar{C}_k \boldsymbol{\tau}_{k+1} + \bar{\boldsymbol{d}}_k + \boldsymbol{h}_k, \quad \forall k \in \bar{\mathcal{K}}, \tag{3.21c}$$

$$z_{k+1} = z_k - \frac{\tau_k + \tau_{k+1}}{2c}(t_{k+1} - t_k), \tag{3.21d}$$

$$\|\boldsymbol{\tau}_{k'}\|_2 \le \tau_{k'}, \quad \forall k' \in \mathcal{K}, \tag{3.21e}$$

$$\tau_{k'} \le T_{\max} e^{-\hat{z}_{k'}} [1 - (z_{k'} - \hat{z}_{k'})], \tag{3.21f}$$

$$\boldsymbol{x}_0 = \boldsymbol{x}(t_0), \quad \boldsymbol{x}_{K-1} = \boldsymbol{x}(t_f), \quad z_0 = \ln(m_0), \tag{3.21g}$$

$$|\boldsymbol{x}_{k'} - \hat{\boldsymbol{x}}_{k'}| \le \boldsymbol{r}_{k'}. \tag{3.21h}$$

In Eq. (3.21a), C represents a coefficient for penalizing the introduction of the slack variables. For the trust region in Eq. (3.21h), the absolute operator is used. Since this research utilizes CVXPY [91], there is no need to handle the absolute operator present in the objective manually. However, the presence of an absolute value function could be managed by introducing an auxiliary slack variable $\boldsymbol{h}_2$ with the inequality constraints $-\boldsymbol{h}_2 \leq \boldsymbol{h} \leq \boldsymbol{h}_2$, then incorporating $\boldsymbol{h}_2$ in the objective to bypass the absolute operator. Eq. (3.21d) represents the discrete form of the continuous integral, taking a trapezoidal structure. Eq. (3.21h) is known as the trust-region constraint, which is discussed in detail in the following subsections.

3.3.2 Trust-Region Constraint

Another common issue in applying SCVX is due to linearization, in which the previously bounded dynamics (with finite thrust) may now become relatively unbounded and, in some cases, ill-defined [90]. For these reasons, it is desirable to limit the deviation from the reference state variables (within each iteration). This approach, rooted in optimization theory [92], is known as the trust-region method. The idea is simply enforced in Eq. (3.21h), meaning that the deviation from state trajectory cannot exceed a threshold defined as $\boldsymbol{r}$. While some approaches [14, 87] consider a constant threshold $\boldsymbol{r}$, in this chapter, we focus on an adaptive approach introduced in [90].

First, we must distinguish between the actual cost, J, from the cost returned after optimizing Eq. (3.21a). By "actual cost," we mean the result when the obtained thrust profile is applied to each segment of the trajectory (divided by discrete starting points throughout the entire trajectory). While the thrust cost remains identical, the discrepancy between the achieved states and the desired ones differs from the slack variables (which are introduced to track this difference in the linearized version). Thus, for each segment, we can define:

$$\boldsymbol{E}_k = \left| \underbrace{\boldsymbol{x}_k + \int_{t_k}^{t_{k+1}} \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{\tau}(t))\, dt}_{\text{solution of the nonlinear dynamics}} - \underbrace{\boldsymbol{x}_{k+1}}_{\text{solution of the convex optimization}} \right|, \quad \forall k \in \bar{\mathcal{K}}, \tag{3.22}$$

where $\boldsymbol{E}_k$ represents the discrepancy between the state produced by convex optimization and the actual state obtained by propagating the nonlinear dynamics, $\boldsymbol{f}$, from the point $\boldsymbol{x}_k$. Consequently, the nonlinear cost is defined as:

$$J = \int_{t_0}^{t_f} \tau(t)\, dt + C \sum_{k \in \bar{\mathcal{K}}} \sum_{i=0}^{5} E_{k,i}. \tag{3.23}$$

By monitoring the linear cost L (given in Eq. (3.21a)), and the nonlinear cost, J, the SCVX algorithm is defined in Algorithm 1. After each iteration of optimization, the algorithm monitors the predicted improvement (i.e., the difference between the previous nonlinear cost and the obtained convex cost) and the actual nonlinear improvement.

If the improvement in the nonlinear cost is significantly lower than predicted, it indicates an overly optimistic estimate, suggesting that the trust region should be reduced. Conversely, if the improvement exceeds expectations, the trust region can be expanded. These adjustments are guided by three thresholds, ρ_0, ρ_1, and ρ_2.

Additionally, α and β represent the factors by which the trust region is reduced and expanded, respectively. The iterations continue until the algorithm predicts a minimal improvement. Moreover, the high-level logic for successive optimization is depicted in Fig. 3.2. It is important to note that in the trust region, Eq. (3.21h), $\boldsymbol{r}_k$ has a subscript k, indicating that the trust region does not need to remain constant along the trajectory. Although this is not explored in the SCVX algorithm, it will be explored in the next subsection.

Algorithm 1 The SCVX Algorithm

```
1: procedure SCVX($\hat{\boldsymbol{x}}, \hat{\boldsymbol{\tau}}, \hat{\boldsymbol{z}}, \epsilon_{\text{tol}}$)
2:     input Select initial state $\hat{\boldsymbol{x}} \in \mathbb{R}^{(K-1)\times 6}$, control $\hat{\boldsymbol{u}} \in \mathbb{R}^{(K-1)\times 3}$, and logarithm of mass
       $\hat{\boldsymbol{z}} \in \mathbb{R}^{N-1}$. Initialize trust region vector $\boldsymbol{r} > 0$. Select parameters $0 < \rho_0 < \rho_1 < \rho_2 < 1$,
       $\boldsymbol{r}_l > 0$ and $\alpha > 1$, $\beta > 1$.
3:     while not converged, i.e., $\Delta L > \epsilon_{\text{tol}}$ do
4:         step 1 Solve Problem (3.21) at $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{\tau}}, \hat{\boldsymbol{z}}, \boldsymbol{r})$ to get an optimal solution $(\boldsymbol{x}, \boldsymbol{\tau}, \boldsymbol{z})$.
5:         step 2 Compute the actual change in the penalty cost Eq. (3.23):

               $$\Delta J = J(\hat{\boldsymbol{x}}, \hat{\boldsymbol{\tau}}) - J(\boldsymbol{x}, \boldsymbol{\tau}), \quad (3.24)$$

       and the predicted change by the convex cost (3.21a):

               $$\Delta L = J(\hat{\boldsymbol{x}}, \hat{\boldsymbol{\tau}}) - L(\boldsymbol{x}, \boldsymbol{\tau}). \quad (3.25)$$

6:         compute the ratio
               $$\rho := \frac{\Delta J}{\Delta L}. \quad (3.26)$$
7:         step 3
8:         if $\rho < \rho_0$ then
9:             reject this step, contract the trust region radius, i.e., $\boldsymbol{r} \leftarrow \boldsymbol{r}/\alpha$ and go back to
       step 1;
10:        else
11:            accept this step, i.e., $\hat{\boldsymbol{x}} \leftarrow \boldsymbol{x}$, $\hat{\boldsymbol{\tau}} \leftarrow \boldsymbol{\tau}$, $\hat{\boldsymbol{z}} \leftarrow \boldsymbol{z}$, and update the trust region radius
       $\boldsymbol{r}^{k+1}$ by
               $$\boldsymbol{r} \leftarrow \begin{cases} \boldsymbol{r}/\alpha, & \text{if } \rho < \rho_1; \\ \boldsymbol{r}, & \text{if } \rho_1 \le \rho < \rho_2; \\ \beta\boldsymbol{r}, & \text{if } \rho_2 \le \rho. \end{cases} \quad (3.27)$$
12:        end if
13:        $\boldsymbol{r} \leftarrow \max\{\boldsymbol{r}, \boldsymbol{r}_{\min}\}$, and go back to step 1.
14:    end while
15:    return $(\boldsymbol{x}, \boldsymbol{\tau}, \boldsymbol{z})$.
16: end procedure
```

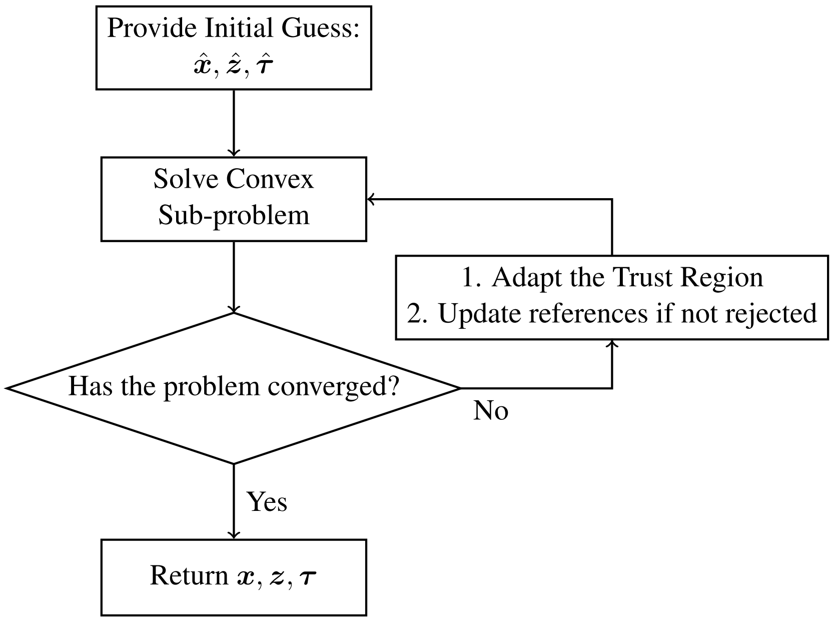

Provide Initial Guess: $\hat{\boldsymbol{x}}, \hat{\boldsymbol{z}}, \hat{\boldsymbol{\tau}}$
Solve Convex Sub-problem
1. Adapt the Trust Region
2. Update references if not rejected
Has the problem converged?
No
Yes
Return $\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\tau}$

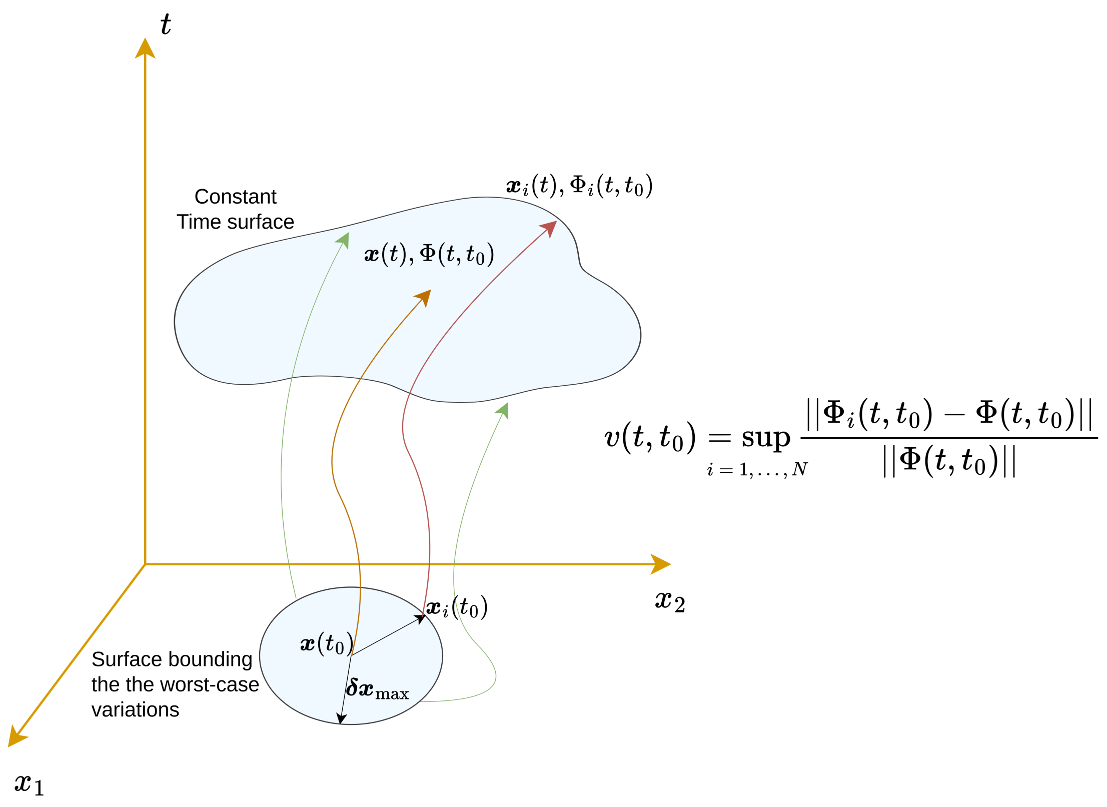
Figure 3.2: High-level overview of the SCVX algorithm.

3.3.3 Enhancing Trust Region Using Nonlinearity Index

In [93], it is proposed that, since inaccuracies from linearization clearly arise from nonlinearities, measuring these nonlinearities could potentially allow for adjusting the trust region accordingly—making it more conservative in highly nonlinear areas and more generous in more linear regions. Therefore, the study suggests using the successful nonlinearity index introduced by Junkins in [94, 95]. As used numerously in this research, the STM, $\Phi_A(t, t_0)$, contains the information of zero-input evolution of trajectory, relating the initial state to a final state at time t.

Clearly, if the dynamics are linear, the STM will be constant given any initial state. On the other hand, the higher the deviation, the higher the nonlinearity. Given this intuition, one can write the nonlinearity index as:

$$v(t, t_0) := \sup_{i=1,\dots,N} \frac{\|\Phi_i(t, t_0) - \Phi(t, t_0)\|}{\|\Phi(t, t_0)\|}, \tag{3.28}$$



t
Constant Time surface
$\boldsymbol{x}_i(t), \Phi_i(t, t_0)$
$\boldsymbol{x}(t), \Phi(t, t_0)$
$v(t, t_0) = \sup_{i=1,\dots,N} \frac{||\Phi_i(t, t_0) - \Phi(t, t_0)||}{||\Phi(t, t_0)||}$
x_2
$\boldsymbol{x}_i(t_0)$
$\boldsymbol{x}(t_0)$
Surface bounding the the worst-case variations
$\boldsymbol{\delta x}_{\max}$
x_1

Figure 3.3: An illustration showcasing nonlinearity index.

where given an initial state, x_0, and using Eq. (3.17), the STM $\Phi(t, t_0)$ is propagated until time t. As for Φ_i, the initial state is perturbed (see Fig. 3.3) over N sample perturbations:

$$\boldsymbol{x}_i(t_0) = \boldsymbol{x}(t_0) + \delta \boldsymbol{x}_i(t_0), \quad \text{with} \quad \|\delta \boldsymbol{x}_i(t_0)\| = \delta x_{\max}, \text{for } i = 1, \cdots, N. \tag{3.29}$$

The same set of equations is then propagated forward. The two resulting matrices are subtracted, and the norm of this difference serves as the nonlinearity index, after being normalized by the magnitude of Φ to eliminate the influence of the first-order linear transformation. Finally, we sample from these perturbations and select the maximum of the collected nonlinearity indices. Although sampling can be used to compute the nonlinearity matrix, [96] suggests that this issue can be avoided by approximating the STM using State Transition Tensor (STT) Models [97, 98, 99]. While the STM provides a first-order approximation of zero-input state evolution, higher-order state transition tensors can also be employed. Define:

$$\Lambda_{IJK}(t,t_0) := \frac{\partial^2 \boldsymbol{x}_I(t)}{\partial \boldsymbol{x}_J(t_0)\partial \boldsymbol{x}_K(t_0)} = \frac{\partial \Phi_{IJ}(t,t_0)}{\partial \boldsymbol{x}_K(t_0)}, \tag{3.30}$$

where $\Lambda_{IJK} \in \mathbb{R}^{IJK}$ denotes the second-order STT. The method used to derive the ODE for the STM can similarly be applied here to compute the state transition tensor. The only distinction is using tensor derivative conventions introduced in Appendix A.

Starting from the original ODE:

$$\boldsymbol{x}_I(t_1) = \boldsymbol{x}_I(t_0) + \int_{t_0}^{t_1} \boldsymbol{f}_I(\boldsymbol{x}(t), \boldsymbol{u}(t))\, dt, \tag{3.31a}$$

STM ODE:

$$\implies \left(\frac{\partial \boldsymbol{x}_I(t_1)}{\partial \boldsymbol{x}_J(t_0)}\right) = I_{IJ} + \int_{t_0}^{t_1} \frac{\partial \boldsymbol{f}_I(\boldsymbol{x}(t), \boldsymbol{u}(t))}{\partial \boldsymbol{x}_D(t)} \frac{\partial \boldsymbol{x}_D(t)}{\partial \boldsymbol{x}_J(t_0)}\, dt, \tag{3.31b}$$

State Transition Tensor ODE:

$$\implies \frac{\partial^2 \boldsymbol{x}_I(t_1)}{\partial \boldsymbol{x}_J(t_0)\partial \boldsymbol{x}_K(t_0)} = \frac{\partial}{\partial \boldsymbol{x}_K(t_0)}\left(\frac{\partial \boldsymbol{x}_I(t_1)}{\partial \boldsymbol{x}_J(t_0)}\right) \tag{3.31c}$$

$$= \int_{t_0}^{t_1} \left[\frac{\partial}{\partial \boldsymbol{x}_K(t_0)}\left(\frac{\partial \boldsymbol{f}_I(\boldsymbol{x}(t), \boldsymbol{u}(t))}{\partial \boldsymbol{x}_D(t)}\right) \frac{\partial \boldsymbol{x}_D(t)}{\partial \boldsymbol{x}_J(t_0)} + \frac{\partial \boldsymbol{f}_I(\boldsymbol{x}(t), \boldsymbol{u}(t))}{\partial \boldsymbol{x}_D(t)} \frac{\partial}{\partial \boldsymbol{x}_K(t_0)}\left(\frac{\partial \boldsymbol{x}_D(t)}{\partial \boldsymbol{x}_J(t_0)}\right)\right] dt, \tag{3.31d}$$

where:

$$\frac{\partial}{\partial \boldsymbol{x}_K(t_0)}\left(\frac{\partial \boldsymbol{f}_I(\boldsymbol{x}(t),\boldsymbol{u}(t))}{\partial \boldsymbol{x}_D(t)}\right)=\frac{\partial^2 \boldsymbol{f}_I(\boldsymbol{x}(t),\boldsymbol{u}(t))}{\partial \boldsymbol{x}_D(t)\partial \boldsymbol{x}_E(t)}\frac{\partial \boldsymbol{x}_E(t)}{\partial \boldsymbol{x}_K(t_0)}, \tag{3.31e}$$

$$\frac{\partial}{\partial \boldsymbol{x}_K(t_0)}\left(\frac{\partial \boldsymbol{x}_D(t)}{\partial \boldsymbol{x}_J(t_0)}\right)=\frac{\partial^2 \boldsymbol{x}_D(t)}{\partial \boldsymbol{x}_J(t_0)\partial \boldsymbol{x}_K(t_0)} \implies \frac{\partial^2 \boldsymbol{x}_I(t_1)}{\partial \boldsymbol{x}_J(t_0)\partial \boldsymbol{x}_K(t_0)}= \tag{3.31f}$$

$$\int_{t_0}^{t_1}\left[\frac{\partial^2 \boldsymbol{f}_I(\boldsymbol{x}(t),\boldsymbol{u}(t))}{\partial \boldsymbol{x}_D(t)\partial \boldsymbol{x}_E(t)}\frac{\partial \boldsymbol{x}_E(t)}{\partial \boldsymbol{x}_K(t_0)}\frac{\partial \boldsymbol{x}_D(t)}{\partial \boldsymbol{x}_J(t_0)}+\frac{\partial \boldsymbol{f}_I(\boldsymbol{x}(t),\boldsymbol{u}(t))}{\partial \boldsymbol{x}_D(t)}\frac{\partial^2 \boldsymbol{x}_D(t)}{\partial \boldsymbol{x}_J(t_0)\partial \boldsymbol{x}_K(t_0)}\right]dt, \tag{3.31g}$$

Here, each term represents:

$$\Lambda_{IJK}(t_1)=\int_{t_0}^{t_1}\left[H_{IDE}(t)\Phi_{EK}(t,t_0)\Phi_{DJ}(t,t_0)+A_{ID}(t)\Lambda_{DJK}(t,t_0)\right]dt. \tag{3.31h}$$

Starting from the system dynamics in Eq. (3.31a), by taking the partial derivative w.r.t. $\boldsymbol{x}_J(t_0)$, we obtain the already mentioned ODE for the State Transition Matrix (STM), this time with specified dimensions. In Eq. (3.31b), the resulting STM has dimensions IJ, hence the two factors in the integral—having dimensions ID and DJ—should yield IJ through matrix multiplication.

To derive the STT, we take another derivative in Eq. (3.31c) w.r.t. $\boldsymbol{x}_K(t_0)$. Note that the subscript K is used only to distinguish dimensions and does not imply that $\boldsymbol{x}(t_0)$ is a different variable. In Eq. (3.31d), the product rule is applied, while in Eq. (3.31e), the chain rule is utilized. Eq. (3.31f) reverts to the definition of the second-order STT. Substituting in these results yields Eq. (3.31g). Finally, by replacing each term with its respective representation, we arrive at Eq. (3.31h), where H is the Hessian of the dynamics ($\dot{\boldsymbol{x}}$) w.r.t. the state variables, A is the Jacobian, Φ is the STM, and Λ is the STT.

The integral in Eq. (3.31h) contains two terms as integrands. In the first term, the factors have dimensions IDE, EK, and DJ, while in the second term, they have dimensions ID and DJK. Both yield the final dimension of IJK. To compute these operations, the Einstein summation convention (Einsum) can be conveniently utilized using Python's NumPy:

```
Lambda_dot = np.einsum('IDE,EK,DJ->IJK', H, Phi, Phi) + np.einsum('ID,DJK->IJK', A, Lambda).
```

Like the STM, the STT can also be computed for each trajectory segment during the convex optimization process, as explained in Section 3.2. In [96], it is proposed that the perturbed STM in the numerator of Eq. (3.28) can be approximated using the STT as follows:

$$\frac{\|\Phi_i(t,t_0) - \Phi(t,t_0)\|}{\|\Phi(t,t_0)\|} \approx \frac{\|\cancel{\Phi(t,t_0)} + [\Lambda_{IJK}(t,t_0)\delta x_{i,K}]_{IJ} - \cancel{\Phi(t,t_0)}\|}{\|\Phi(t,t_0)\|}, \tag{3.32}$$

where the tensor approximation replaces the need for directly computing the perturbed STM. The same authors in [100] assume that each entry of δx is either $+1$ or -1. Under this assumption, they use the entrywise L_1 norm for the matrix, leading to the following expression for the nonlinearity matrix:

$$v(t,t_0) = \sup_{i=1,\dots,N} \frac{\|[\Lambda_{IJK}(t,t_0)\delta \boldsymbol{x}_{i,K}]_{IJ}\|}{\|\Phi(t,t_0)\|} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}|\Lambda_{ijk}(t,t_0)|}{\sum_{i=1}^{n}\sum_{j=1}^{n}|\Phi_{ij}(t,t_0)|}. \tag{3.33}$$

Furthermore, since we are interested in the contribution of each state variable to the nonlinearity index, the directional nonlinearity index can be computed by considering the specific variable of interest in the last dimension:

$$v_e(t,t_0) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n}|\Lambda_{ije}(t,t_0)|}{\sum_{i=1}^{n}\sum_{j=1}^{n}|\Phi_{ij}(t,t_0)|}. \tag{3.34}$$

This would be the case if the perturbation vector $\delta \boldsymbol{x}_e$ takes the form of a one hot vector: $\begin{bmatrix} 0 & \dots & 0 & \delta x & 0\dots & 0 \end{bmatrix}$. In the previous subsection, the trust region, $\boldsymbol{r}$ in (3.21h) remained constant throughout the trajectory segments. Now, given this nonlinearity index for each segment and along each direction, we will arrive at $\boldsymbol{v}_{\mathcal{K}E}$, where $\mathcal{K}$ is the dimension for segments and E is the dimension for each direction of the state vector. Then the trust region can be modified using the nonlinearity index with:

$$\tilde{\boldsymbol{r}}_{\mathcal{K}E} \leftarrow (\frac{s}{\boldsymbol{v}})_{\mathcal{K}E}\boldsymbol{r}_E, \tag{3.35}$$

where s is a scaling parameter. Replacing $\tilde{\boldsymbol{r}}$ in Eq. (3.21h) with the previous trust region allows for penalizing nonlinearity while maintaining a larger trust region in more linear regions. For stability, $\dfrac{s}{\boldsymbol{v}}$ is clipped between two values. Note that if the system is almost linear, this value can become enormous.

Chapter 4

Numerical Simulations and Results

We investigate two benchmark fixed-time minimum-fuel trajectory optimization problems: the Earth-to-Mars and Earth-to-Dionysus problems [33]. Given the bang-off-bang nature of optimal control in these scenarios, numerical solvers and integrators may face significant challenges in solving these problems. To address and overcome issues in solving these problems, we compare two smoothing functions, HTS and L2 (short for L2-norm-based) methods, solving both using Cartesian and MEEs as outlined in earlier chapters. Additionally, we examine the impact of calculating the sensitivities using the STM for both regularization methods and for both problems. Furthermore, the problems are solved using the proposed SCVC method and compared against indirect results, with a detailed examination of iterations, especially when the trust region is enhanced with a nonlinearity index.

The time and distance units used to solve the problems, respectively, are TU $= 3.1536 \times 10^7$ s, AU $= 1.496 \times 10^8$ km. The unknown initial values for the costates, in the Cartesian coordinate system, are randomly generated in the range $[0, 1]$, and for the MEEs, they are randomly generated in the range $[0, 0.1]$ except for λ_m, which is randomly generated in the range $[0, 1.0]$. Moreover, the relative and absolute tolerances for the numerical integrator are both set to 1.0×10^{-13}.

The code was developed using Python leveraging Sympy [101] and Casadi libraries [102]. Casadi is employed for modeling the problem, while SciPy [103] is used for numerical integration and root finding. It is noteworthy that the default nonlinear root finder in SciPy utilizes the original Minpack [104] hybrid process, which is an implementation of a modified version of Powell's dog leg method [92]. Finally, CVXPY [91] was utilized for solving the convex optimization subproblems, where the SOCPs were solved using ECOS [60].

4.1 Minimum-Fuel Earth-to-Mars Problem

In this problem, the spacecraft should leave the Earth and rendezvous with Mars over a fixed time of flight of TOF $= 348.795$ days. Only the heliocentric phase of flight is considered, and the hyperbolic excess velocity vectors with respect to Earth and Mars are assumed to be zero. We used subscript 'E' for denoting position and velocity vectors of the Earth. We also used 'M' for position and velocity vectors of Mars. The parameters and boundary conditions that are used for this problem are summarized in Table 4.1. Note that MEEs can be obtained from the Cartesian boundary conditions [67]. Furthermore, the hyperparameters for the SCVX algorithm are given in Table 4.2.

Table 4.1: Earth-to-Mars problem: spacecraft mission parameters and initial/final conditions.

Symbol	**Value** (unit)
μ	132712440018 (km^3/s^2)
m_0	1000 (kg)
I_{sp}	2000 (s)
$T_{\max}$	$0.5\,\text{N}$
$(\mathbf{r}_E, \mathbf{v}_E)$	$\mathbf{r}_E = [-140699693, -51614428, 980]^\top$ (km), $\mathbf{v}_E = [9.774596, -28.07828, 4.337725 \times 10^{-4}]^\top$ (km/s)
$(\mathbf{r}_M, \mathbf{v}_M)$	$\mathbf{r}_M = [-172682023, 176959469, 7948912]^\top$ (km), $\mathbf{v}_M = [-16.427384, -14.860506, 9.21486 \times 10^{-2}]^\top$ (km/s)
TOF	348.795 (days)

Table 4.2: Parameters for successive convex optimization (Earth-to-Mars).

Parameter	**Value**
Number of discrete points (N)	1000
Penalty coefficient (C)	30
Tolerance (ϵ_{tol})	1×10^{-3}
Estimated final mass	500
Trust region vector	$[10, 0.1, 10, 1, 1, 10]$
Step size increase factor (α)	1.5
Step size decrease factor (β)	1.5
Lower threshold (ρ_0)	0.04
Middle threshold (ρ_1)	0.2
Upper threshold (ρ_2)	0.7

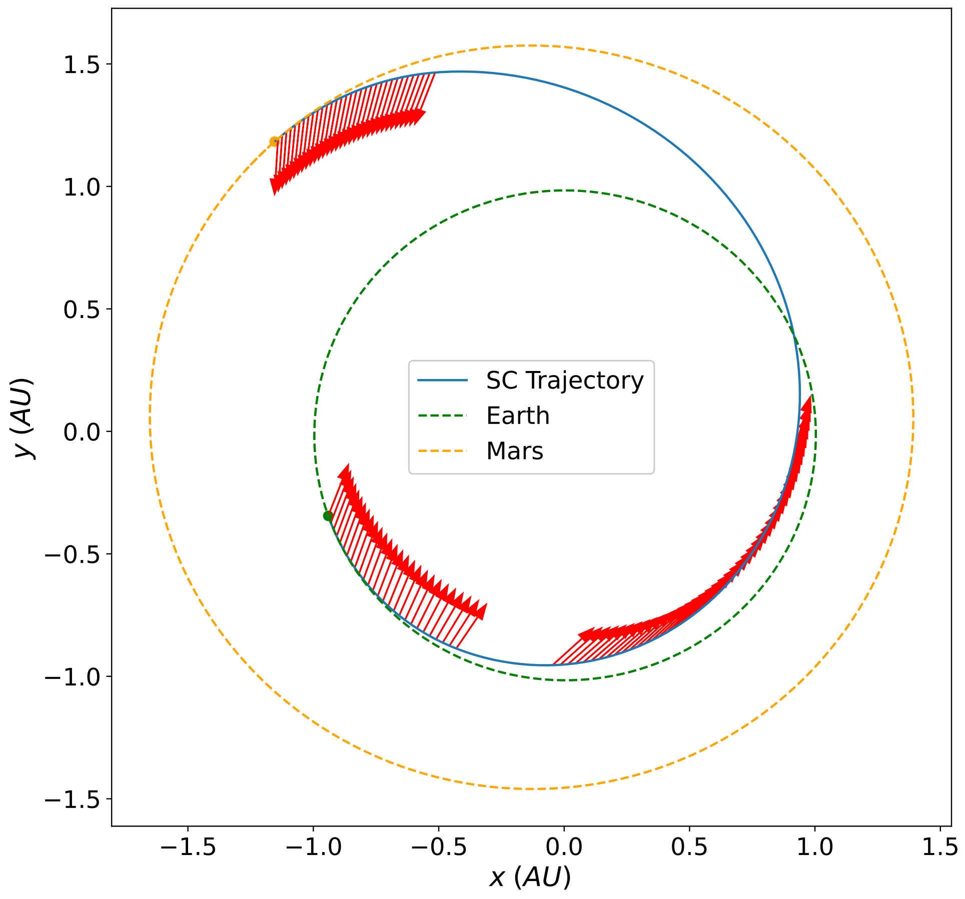


Figure 4.1: Earth-to-Mars problem: $x - y$ view of the optimal trajectory for $\rho = 1.0 \times 10^{-5}$. The red arrows show the thrust direction.

The indirect method results, for 100 randomly generated sets of costates, are summarized in Table 4.3. The last column refers to the average time it took for the solver to find a solution. Note that this average time includes the time for all values of ρ from 1.0 to 1.0×10^{-5}. From the results, it is observed that L2 smoothing has a slight advantage over the HTS method. Moreover, confirming the results of previous studies [33, 44], using the STM method greatly improves the convergence rate of the nonlinear root-finding algorithms. In Table 4.3, the optimal final mass for this problem is $m(t_f) = m_f = 603.935$ kg, which is consistent with the globally optimal solution for this benchmark problem [33]. The minimum-fuel trajectory is shown in Fig. 4.1. Figure 4.2 shows the throttle and switching function values for L2, HTS, and SCVX methods.

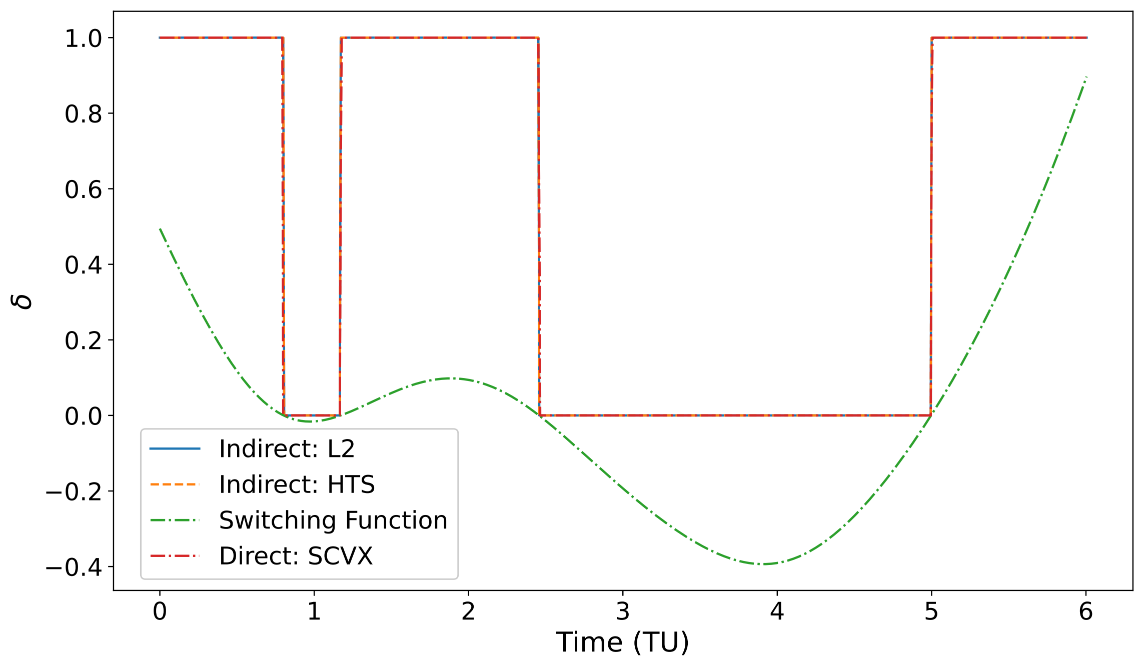


Figure 4.2: Earth-to-Mars problem: throttle and switching function vs. time for $\rho = 1.0 \times 10^{-5}$.

Table 4.3: Earth-to-Mars problem: comparison of convergence rate of the Indirect method for HTS and L2 smoothing and for Cartesian and MEEs with and without the STM.

Smoothing Function	Coordinates	STM	Convergence %	Time (s)
Hyperbolic Tangent	Cartesian	True	85	1.60
Hyperbolic Tangent	Cartesian	False	76	1.62
L2	Cartesian	True	89	1.60
L2	Cartesian	False	78	1.47
Hyperbolic Tangent	MEE	True	76	1.25
Hyperbolic Tangent	MEE	False	75	0.98
L2	MEE	True	77	1.37
L2	MEE	False	66	1.29

4.2 Minimum-Fuel Earth-to-Dionysus Problem

In this problem, the spacecraft departs from Earth and rendezvous with the asteroid Dionysus over a fixed time of flight of 3534 days. Only the heliocentric phase of flight is considered, and the hyperbolic excess velocities with respect to Earth (subscript 'E') and Dionysus (subscript 'D') are assumed to be zero. The parameters and boundary conditions used for this problem are summarized in Table 4.4. Additionally, the hyperparameters for the SCVX algorithm are listed in Table 4.5.

Table 4.4: Earth-to-Dionysus problem: spacecraft/mission parameters and initial/final conditions.

Symbol	**Value** (unit)
μ	132712440018 (km^3/s^2)
m_0	4000 (kg)
I_{sp}	3000 (s)
$T_{\max}$	0.32 (N)
$(\mathbf{r}_E, \mathbf{v}_E)$	$\mathbf{r}_E = [-3637871.081, 147099798.784, -2261.441]^\top$ (km), $\mathbf{v}_E = [-30.265097, -0.8486854, 0.505 \times 10^{-4}]^\top$ (km/s)
$(\mathbf{r}_D, \mathbf{v}_D)$	$\mathbf{r}_D = [-302452014.884, 316097179.632, 82872290.0755]^\top$ (km), $\mathbf{v}_D = [-4.533473, -13.110309, 0.656163]^\top$ (km/s)
TOF	3534 days

Table 4.5: Parameters for successive convex optimization (Earth-to-Dionysus)

Parameter	**Value**
Number of discrete points (N)	1000
Penalty coefficient (C)	5
Tolerance (ϵ_{tol})	5×10^{-3}
Estimated final mass	2500
Initial trust region vector	$[10, 0.1, 10, 1, 1, 10, 30]^\top$
Step size increase factor (α)	1.5
Step size decrease factor (β)	1.5
Lower threshold (ρ_0)	0.04
Middle threshold (ρ_1)	0.2
Upper threshold (ρ_2)	0.7

The problem features multiple extremal points [15], corresponding to different numbers of revolutions around the Sun. The most optimal solution involves five orbital revolutions, as shown in the three-dimensional (3D) view of the trajectory in Fig. 4.3. The projection onto the $x - y$ plane is depicted in Fig. 4.4. The time histories of the throttle and switching functions for L2, HTS, and SCVX methods are shown in Fig. 4.5.

The results of the simulations are presented in Table 4.6. In total, 100 randomly generated initial costates are considered. The L2 smoothing method has a slight advantage over HTS, except when the STM is not used with MEEs. The optimal final mass is $m(t_f) = m_f = 2718.33$ kg, which aligns with the globally optimal solution for this problem [33].

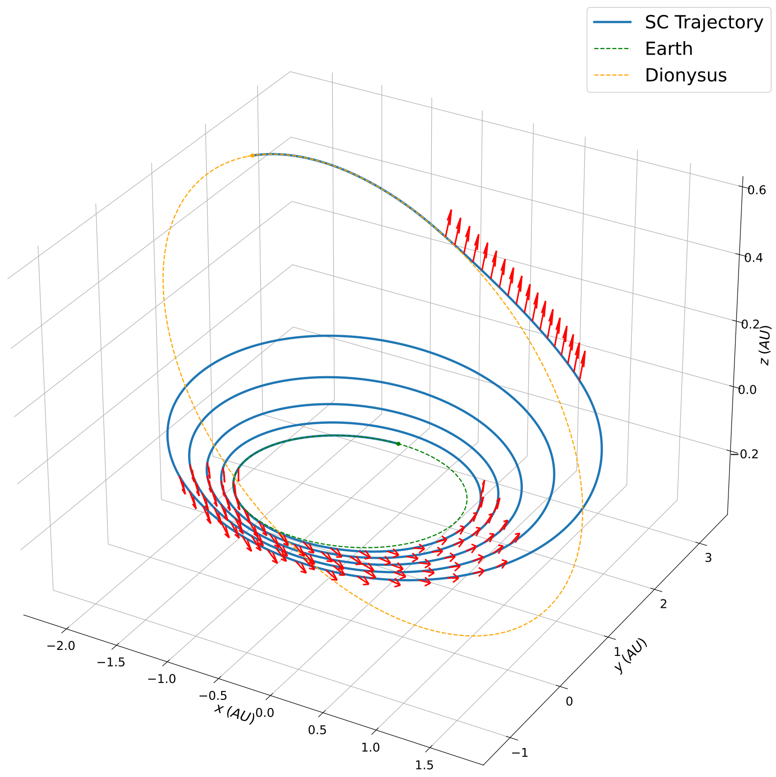


Figure 4.3: Earth-to-Dionysus problem: 3D view of the minimum-fuel trajectory for $\rho = 1.0 \times 10^{-5}$. The red arrows indicate the thrust vector direction.

4.3 Smoothing Parameter in Regularization of Indirect Methods

For indirect methods, we initially set the value of ρ (the smoothing parameter) to 1 to find an initial solution. The obtained solution is then fed back into the problem with a reduced value of $\rho \leftarrow 0.1\rho$ and solved again. This iterative process continues until ρ reaches 1.0×10^{-5}.

For the Earth-to-Mars problem, the throttle profile and its sensitivity to different values of ρ are shown in Fig. 4.6, which depicts the throttle behavior for five distinct values of ρ. This analysis demonstrates the gradual transition from a smooth control profile at higher ρ to a sharper, bang-bang type control as the value of ρ decreases. The impact of ρ on the final mass is illustrated in Fig. 4.7, highlighting the trade-off between regularization smoothness and fuel efficiency.

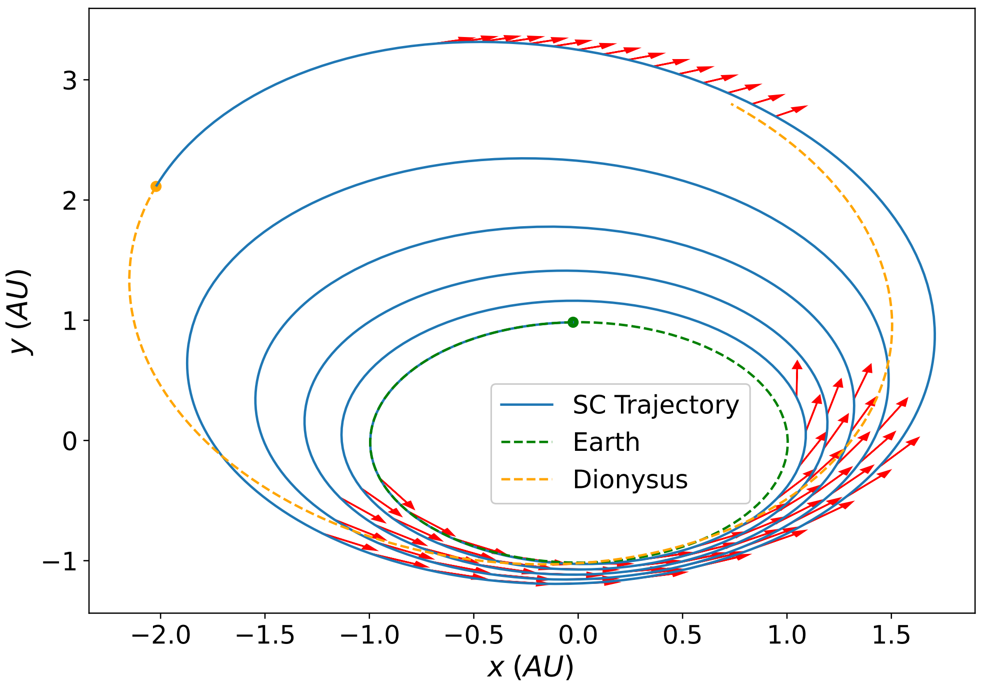


Figure 4.4: Earth-to-Dionysus problem: $x - y$ view of the optimal trajectory for $\rho = 1.0 \times 10^{-5}$.

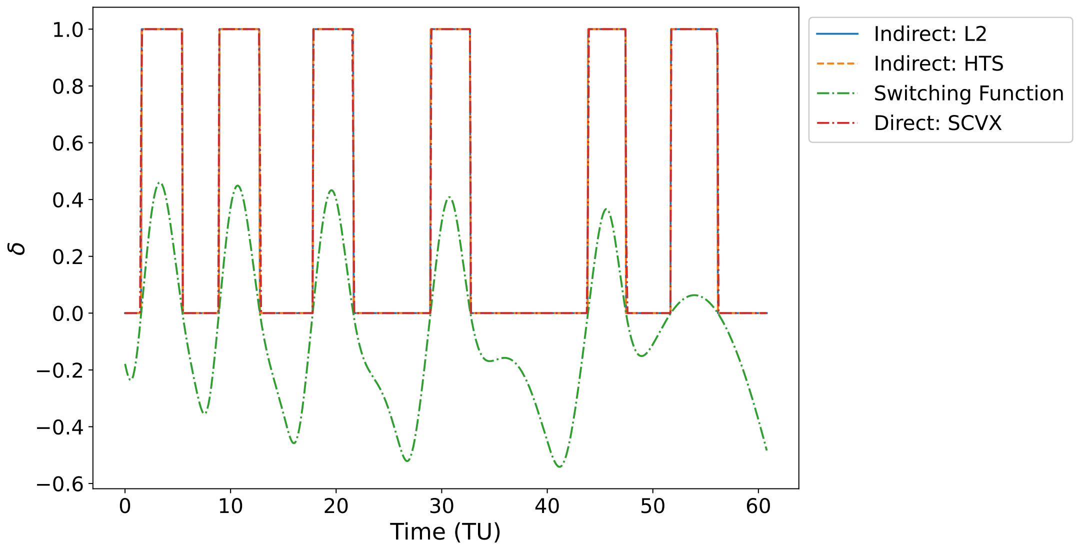


Figure 4.5: Earth-to-Dionysus problem: throttle and switching function vs. time for $\rho = 1.0 \times 10^{-5}$.

Similarly, for the Earth-to-Dionysus problem, the variation in throttle profiles with changing ρ values is depicted in Fig. 4.8. The results reveal a similar trend, where a smoother control profile is observed at higher ρ values, transitioning to a more step-like throttle behavior at lower

Table 4.6: Earth-to-Dionysus problem: comparison of results using Cartesian and MEEs for L2 and HTS methods with and without STM.

Smoothing Function	Coordinates	STM	Convergence %	Time (s)
Hyperbolic Tangent	Cartesian	True	34	20.48
Hyperbolic Tangent	Cartesian	False	3	9.33
L2	Cartesian	True	40	22.34
L2	Cartesian	False	3	19.87
Hyperbolic Tangent	MEE	True	70	5.05
Hyperbolic Tangent	MEE	False	36	4.23
L2	MEE	True	72	5.35
L2	MEE	False	34	4.01

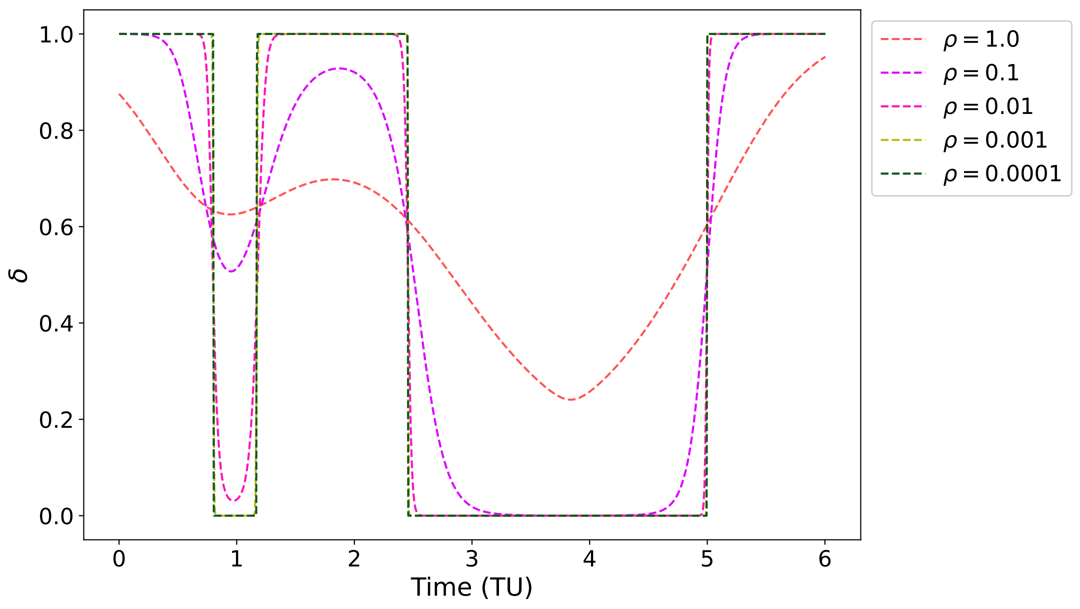


Figure 4.6: Earth-to-Mars problem: throttle profile vs. time for different values of ρ.

ρ values. The corresponding changes in final mass are shown in Fig. 4.9, indicating a consistent mass efficiency improvement as ρ is reduced, aligning with the expected behavior of the solutions of minimum-fuel trajectory optimization problems.

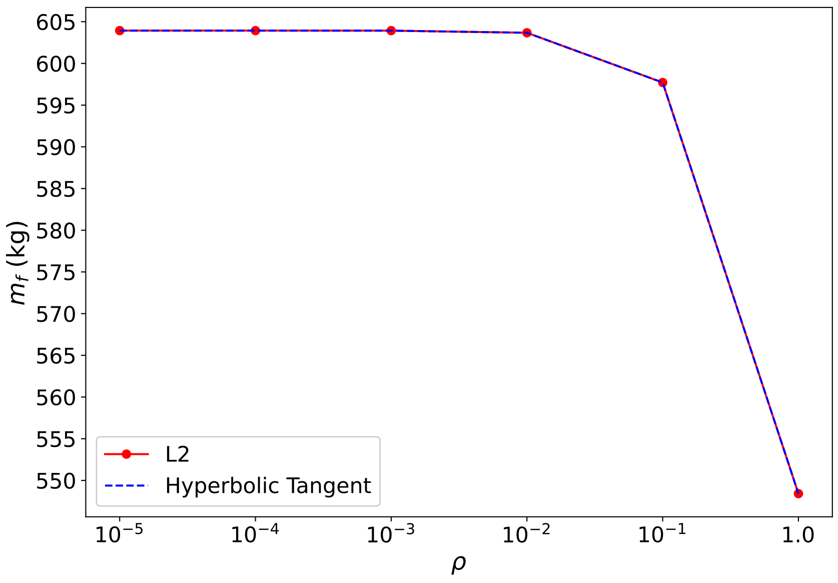


Figure 4.7: Earth-to-Mars problem: final mass vs. ρ.

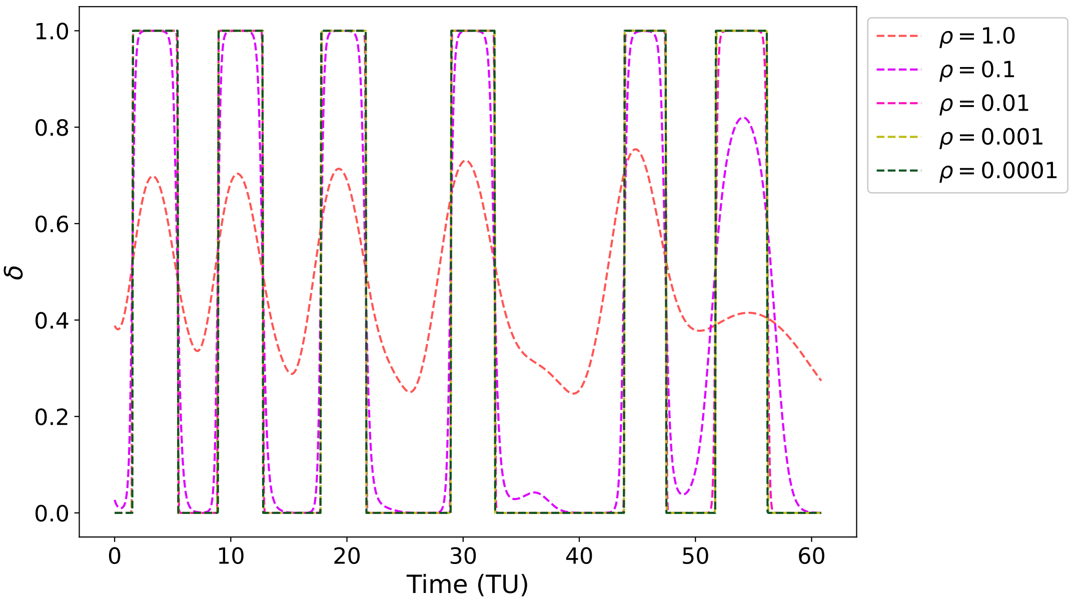


Figure 4.8: Earth-to-Dionysus problem: throttle vs. time for different values of ρ.

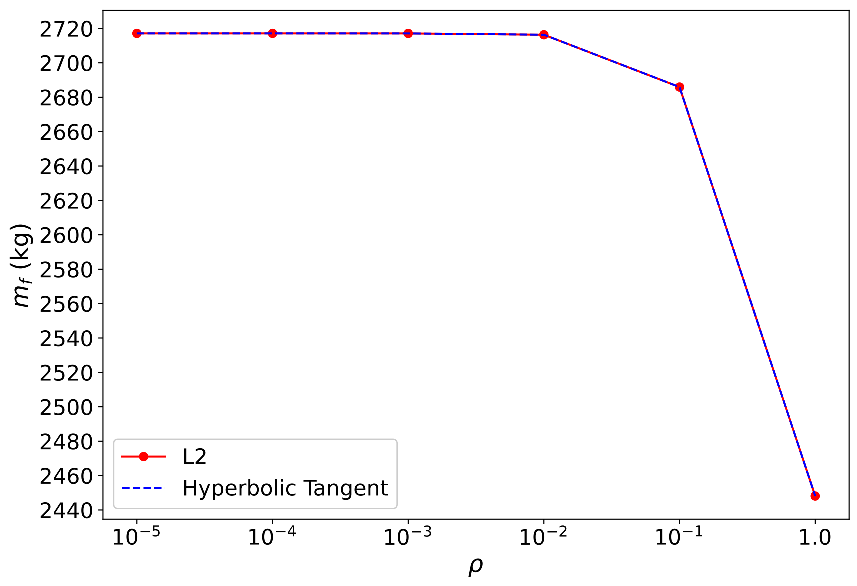


Figure 4.9: Earth-to-Dionysus problem: final mass vs. ρ.

4.4 Analyzing the solutions of the SCVX and its comparison when nonlinearity index is used

4.4.1 Number of Discrete Points

As SCVX relies heavily on discretization, the number of discrete points directly impacts its performance. This effect is illustrated in Figs. 4.10 and 4.11 for the Earth-to-Mars and Earth-to-Dionysus problems, respectively. In these figures, it can be observed that increasing the number of discrete points leads to a higher final mass. For the Earth-to-Dionysus problem, this outcome aligns with expectations, as a finer discretization typically yields a solution closer to the true optimal final mass. In contrast, for the Earth-to-Mars problem, the final mass obtained by the optimization is higher than the known optimal value. This discrepancy warrants further investigation; however, it is likely attributable to the linearization of the original system dynamics within the convex optimization framework.

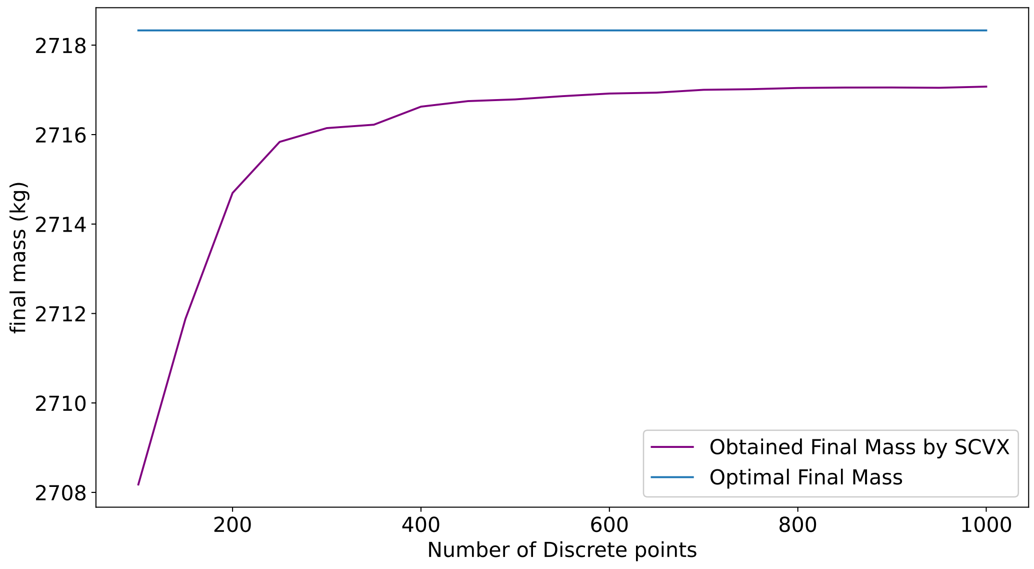


Figure 4.10: Earth-to-Dionysus problem: final mass vs. number of discrete points.

4.4.2 SCVX process and effectiveness of using the nonlinearity index

The number of iterations required for convergence is summarized in Table 4.7. The effect of the nonlinearity index is constrained between 1 and 3, allowing for a more relaxed trust region in linear regions. Additionally, the scaling parameter was set to $\frac{1}{30}$ (as discussed in Section 3.3).

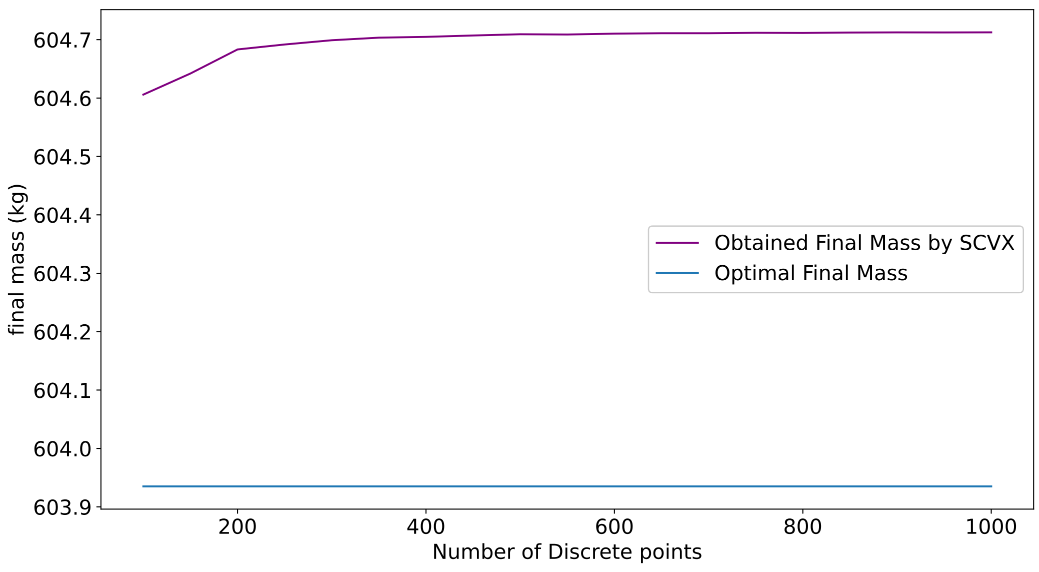


Figure 4.11: Earth-to-Mars problem: final mass vs. number of discrete points.

The figures illustrate various aspects of the optimization process for both problems. Specifically, Figs. 4.12, 4.13, and 4.14 show the evolution of final mass, nonlinear cost, and trust region ratio against the number of iterations for the Earth-to-Mars problem. For the Earth-to-Dionysus problem, similar analyses are presented in Figs. 4.15, 4.16, and 4.17.

Table 4.7: Number of iterations for the SCVX algorithm with and without the nonlinearity index augmentation.

Problem	**With Augmentation**	**Without Augmentation**
Earth-to-Mars	5	5
Earth-to-Dionysus	25	31

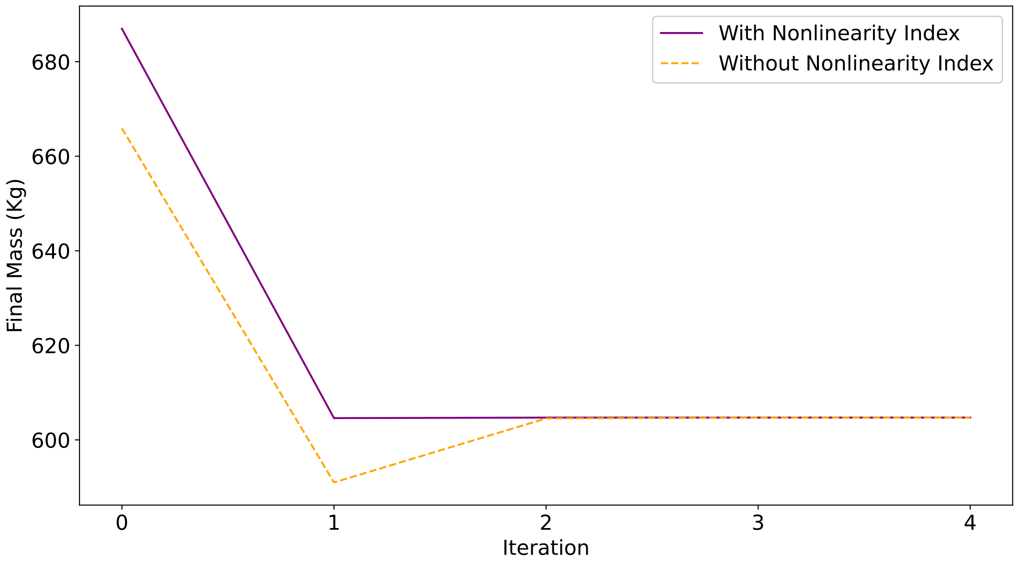


Figure 4.12: Earth-to-Mars problem: final mass vs. iteration #.

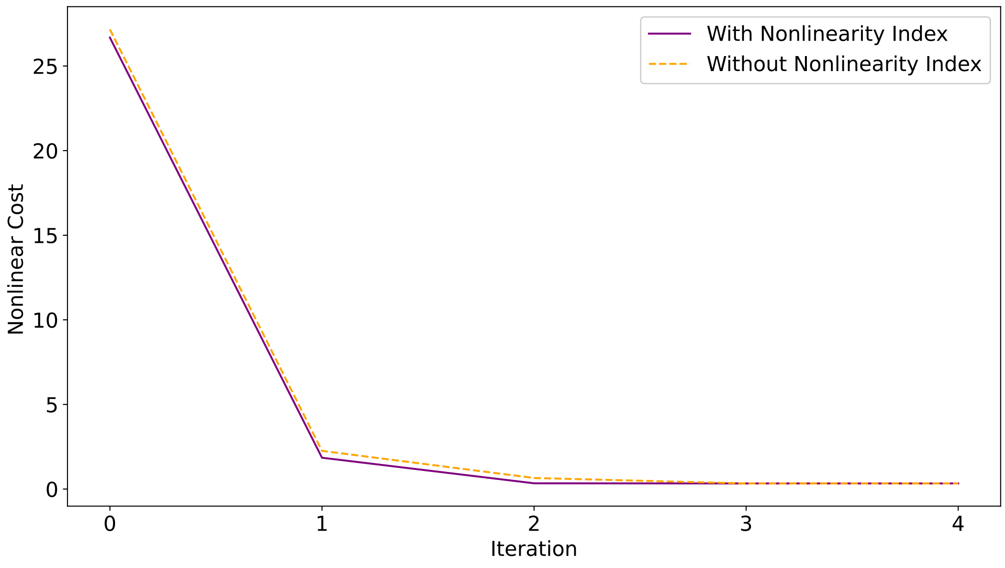


Figure 4.13: Earth-to-Mars problem: nonlinear cost vs. iteration #

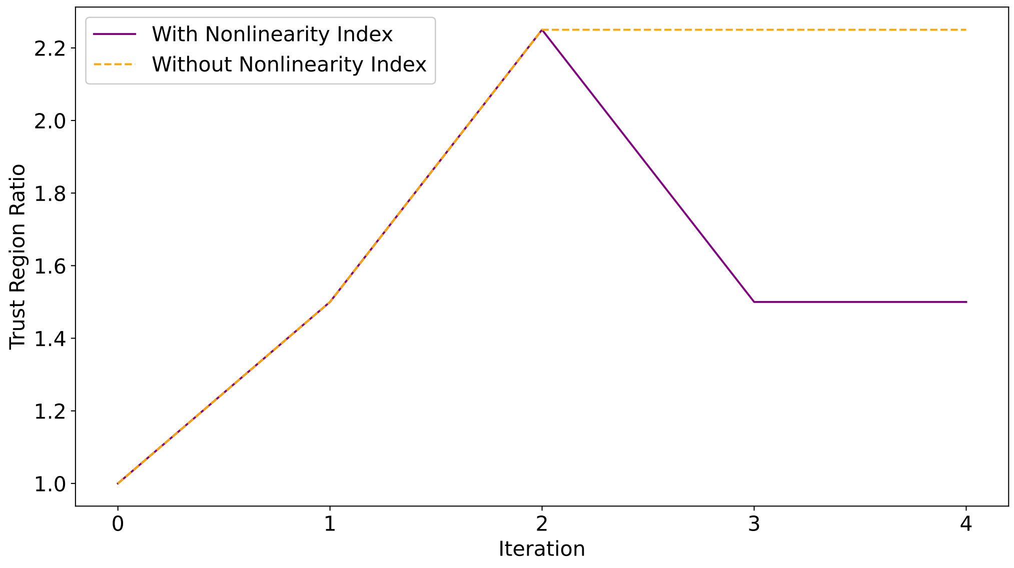


Figure 4.14: Earth-to-Mars problem: trust region ratio vs. iteration #.

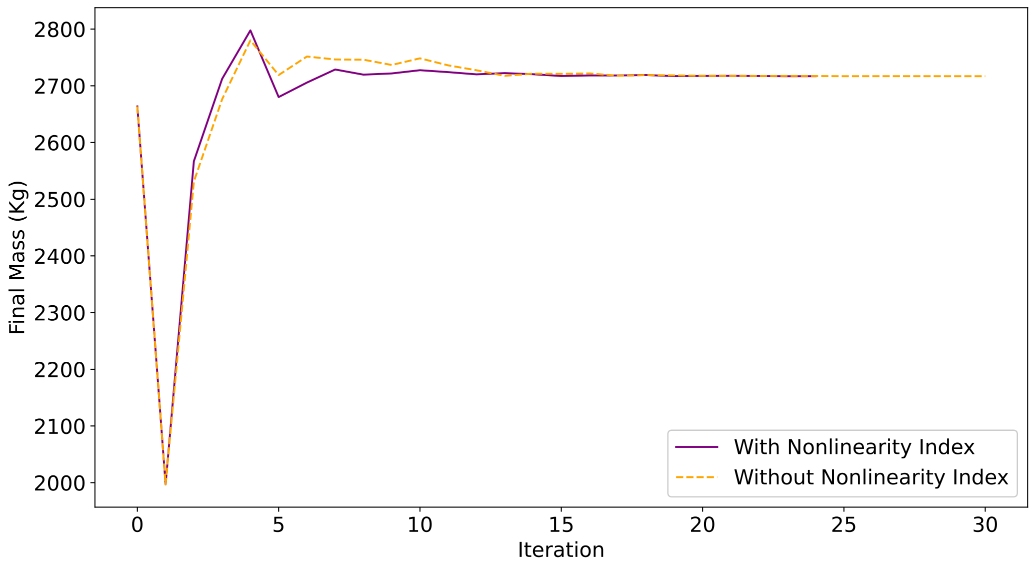


Figure 4.15: Earth-to-Dionysus problem: final mass vs. iteration #.

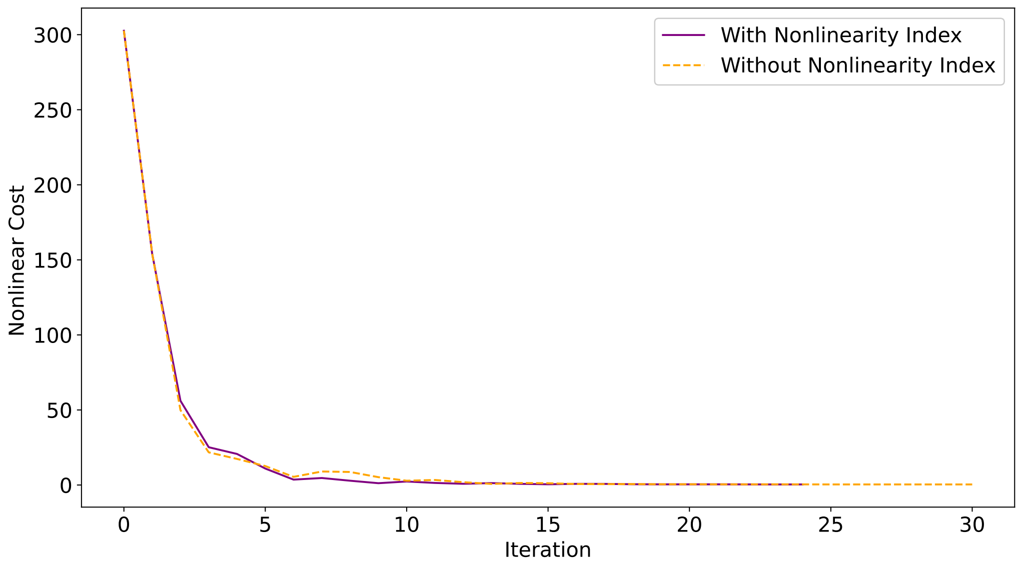


Figure 4.16: Earth-to-Dionysus problem: nonlinear cost vs. iteration #.

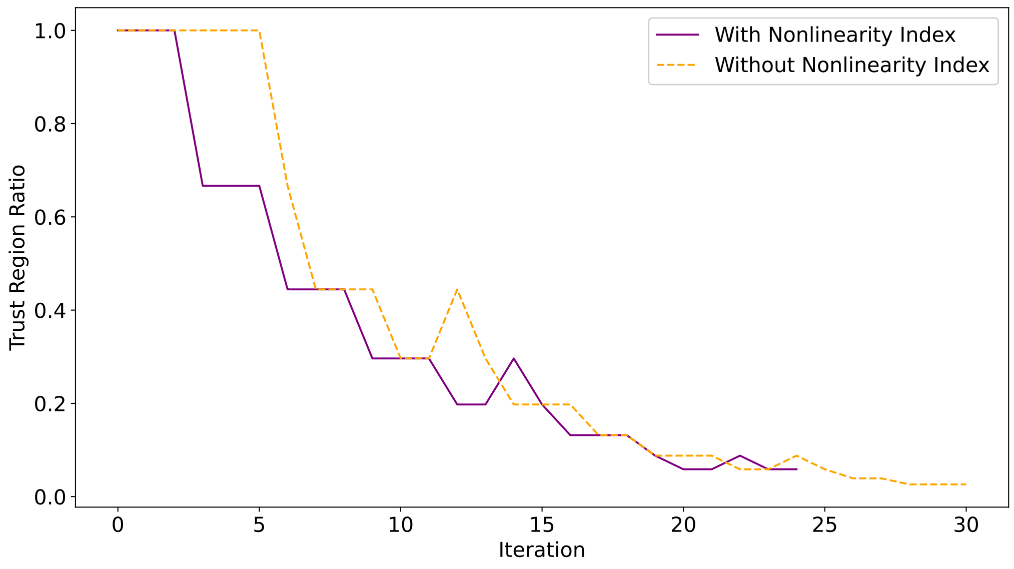


Figure 4.17: Earth-to-Dionysus problem: trust region vs. iteration #.

Chapter 5

Concluding Remarks

5.1 Summary

This thesis explored the challenges and methodologies associated with solving the class of fixed-time minimum-fuel low-thrust trajectory optimization problems, focusing on two benchmark missions: Earth-to-Mars and Earth-to-Dionysus. Through a comparative analysis of indirect methods and convex optimization techniques, this work demonstrated the effectiveness and limitations of each approach.

Indirect optimization methods, rooted in the calculus of variations, were examined due to their high accuracy and ability to capture fine-scale dynamics. However, the inherent difficulties in solving two-point boundary value problems (TPBVPs), highlighted the need for a regularization technique. This study emphasized the role of control regularization in stabilizing the numerical solvers and ensuring convergence for challenging low-thrust scenarios with discontinuous control structures. The comparative analysis between different smoothing techniques (L2 and HTS) and the use of Cartesian and Modified Equinoctial Elements (MEEs) provided deeper insights into the performance trade-offs. The inclusion of the State Transition Matrix (STM) significantly enhanced the convergence rate of both indirect and direct methods, underscoring its importance in sensitivity analysis.

The thesis introduced Successive Convex Optimization as a robust alternative to traditional indirect methods by discretizing the problem and employing convexification. The augmentation with a nonlinearity index slightly improved the number of iterations, particularly for the multi-revolution Earth-to-Dionysus trajectory. The results demonstrated that while indirect methods offer higher-resolution solutions, Successive Convex Optimization also provides a scalable and reliable framework for practical mission scenarios.

In conclusion, this thesis contributes to the field of trajectory optimization by presenting a comprehensive framework that leverages both indirect methods and modern convex optimization techniques.

5.2 Future Work

Several potential research directions could further enhance the methods presented in this thesis:

1. In this thesis, the costate initialization process was performed manually without leveraging specific problem insights. A promising direction for improvement involves exploring data-driven techniques for costate initialization [105]. An interesting approach would be to employ a reinforcement learning-based framework that iteratively learns the costate values with minimal supervision and without the need for labeled data.

2. The nonlinearity index, employed in this research, was based on the zero-input evolution of the states. Future studies could benefit from redefining the nonlinearity index using a perturbation mapping that accounts for the transition between zero or first-order hold control inputs and the final state. This could provide a more accurate measure of the system's nonlinear characteristics.

3. In this work, the nonlinearity index was approximated using a second-order state transition tensor. Future research could investigate the use of higher-order state transition models to enhance the accuracy of the nonlinearity index estimation.

4. The Differential Algebra Computational Engine (DACE) [106] is a powerful tool that can automatically compute higher-order state transition tensors and time-discrete mappings. Incorporating DACE into future research could be especially valuable for tackling more complex problems, where detailed higher-order analysis is required.

References

[1] Edgar Y Choueiri. New dawn for electric rockets. *Scientific American*, 300(2):58–65, 2009.

[2] Marc D Rayman and Steven N Williams. Design of the first interplanetary solar electric propulsion mission. *Journal of Spacecraft and Rockets*, 39(4):589–595, 2002.

[3] Marc D Rayman, Philip Varghese, David H Lehman, and Leslie L Livesay. Results from the deep space 1 technology validation mission. *Acta Astronautica*, 47(2-9):475–487, 2000.

[4] Joakim Kugelberg, Per Bodin, Staffan Persson, and Peter Rathsman. Accommodating electric propulsion on smart-1. *Acta Astronautica*, 55(2):121–130, 2004.

[5] Paul D Fieseler, Jim Taylor, and Roger W Klemm. Dawn spacecraft performance: resource utilization and environmental effects during an 11-year mission. *Journal of Spacecraft and Rockets*, 57(1):147–159, 2020.

[6] Dan Lev, Roger M Myers, Kristina M Lemmer, Jonathan Kolbeck, Hiroyuki Koizumi, and Kurt Polzin. The technological and commercial expansion of electric propulsion. *Acta Astronautica*, 159:213–227, 2019.

[7] VG Petukhov. Application of the angular independent variable and its regularizing transformation in the problems of optimizing low-thrust trajectories. *Cosmic research*, 57(5):351–363, 2019.

[8] Yazhe Meng, Hao Zhang, and Yang Gao. Low-thrust minimum-fuel trajectory optimization using multiple shooting augmented by analytical derivatives. *Journal of Guidance, Control, and Dynamics*, 42(3):662–677, 2019.

[9] Jonathan D Aziz, Jeffrey S Parker, Daniel J Scheeres, and Jacob A Englander. Low-thrust many-revolution trajectory optimization via differential dynamic programming and a sundman transformation. *The Journal of the Astronautical Sciences*, 65:205–228, 2018.

[10] Ehsan Taheri. Optimization of many-revolution minimum-time low-thrust trajectories using sundman transformation. In *AIAA Scitech 2021 Forum*, page 1343, 2021.

[11] John T Betts. Survey of numerical methods for trajectory optimization. *Journal of guidance, control, and dynamics*, 21(2):193–207, 1998.

[12] Emmanuel Trélat. Optimal control and applications to aerospace: some results and challenges. *Journal of Optimization Theory and Applications*, 154:713–758, 2012.

[13] I Michael Ross, Qi Gong, and Pooya Sekhavat. Low-thrust, high-accuracy trajectory optimization. *Journal of Guidance, Control, and Dynamics*, 30(4):921–933, 2007.

[14] Nicholas P Nurre and Ehsan Taheri. Comparison of indirect and convex-based methods for low-thrust minimum-fuel trajectory optimization. In *2022 AAS/AIAA Astrodynamics Specialist Conference, AAS 22*, volume 782, page 21, 2022.

[15] Ehsan Taheri and John L Junkins. How many impulses redux. *The Journal of the Astronautical Sciences*, 67(2):257–334, 2020.

[16] Danylo Malyuta, Yue Yu, Purnanand Elango, and Behçet Açıkmeşe. Advances in trajectory optimization for space vehicle control. *Annual Reviews in Control*, 52:282–315, 2021.

[17] Ehsan Taheri, John L Junkins, Ilya Kolmanovsky, and Anouck Girard. A novel approach for optimal trajectory design with multiple operation modes of propulsion system, part 1. *Acta Astronautica*, 172:151–165, 2020.

[18] Ehsan Taheri and Kshitij Mall. Minimum-fuel low-thrust trajectory optimization using trigonometric-based regularization. In *Proceedings of the 2020 AAS/AIAA Astrodynamics Specialist Conference, Virtually*, pages 9–13, 2020.

[19] Yevhenii Kovryzhenko and Ehsan Taheri. Vectorized trigonometric regularization for singular control problems with multiple state path constraints. *The Journal of the Astronautical Sciences*, 71(1):1, 2023.

[20] Yevhenii Kovryzhenko, Nicholas P Nurre, and Ehsan Taheri. Generalized vectorized trigonometric regularization for solving optimal control problems with complex solution structures. In *AIAA Scitech 2024 Forum*, page 2208, 2024.

[21] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[22] Xinfu Liu, Ping Lu, and Binfeng Pan. Survey of convex optimization for aerospace applications. *Astrodynamics*, 1:23–40, 2017.

[23] Zhenbo Wang. A survey on convex optimization for guidance and control of vehicular systems. *Annual Reviews in Control*, 57:100957, 2024.

[24] Daniel Dueri, Behçet Açıkmeşe, Daniel P Scharf, and Matthew W Harris. Customized real-time interior-point methods for onboard powered-descent guidance. *Journal of Guidance, Control, and Dynamics*, 40(2):197–212, 2017.

[25] Thomas Haberkorn, Pierre Martinon, and Joseph Gergaud. Low thrust minimum-fuel orbital transfer: a homotopic approach. *Journal of Guidance, Control, and Dynamics*, 27(6):1046–1060, 2004.

[26] Ehsan Taheri, Nan I Li, and Ilya Kolmanovsky. Co-state initialization for the minimum-time low-thrust trajectory optimization. *Advances in Space Research*, 59(9):2360–2373, 2017.

[27] Max Cerf. Fast solution of minimum-time low-thrust transfer with eclipses. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 233(7):2699–2714, 2019.

[28] Samuel Sowell and Ehsan Taheri. Eclipse-conscious low-thrust trajectory optimization using pseudospectral methods and control smoothing techniques. *Journal of Spacecraft and Rockets*, 61(3):900–907, 2024.

[29] Nicholas P Nurre and Ehsan Taheri. End-to-end operationally-constrained low-thrust transfers to gateway's near-rectilinear halo orbit. In *AIAA SCITECH 2024 Forum*, page 0838, 2024.

[30] Régis Bertrand and Richard Epenoy. New smoothing techniques for solving bang–bang optimal control problems—numerical results and statistical interpretation. *Optimal Control Applications and Methods*, 23(4):171–197, 2002.

[31] John E Prussing. Primer vector theory and applications. *Spacecraft trajectory optimization*, 29:16, 2010.

[32] Fanghua Jiang, Hexi Baoyin, and Junfeng Li. Practical techniques for low-thrust trajectory optimization with homotopic approach. *Journal of guidance, control, and dynamics*, 35(1):245–258, 2012.

[33] Ehsan Taheri, Ilya Kolmanovsky, and Ella Atkins. Enhanced smoothing technique for indirect optimization of minimum-fuel low-thrust trajectories. *Journal of Guidance, Control, and Dynamics*, 39(11):2500–2511, 2016.

[34] Ehsan Taheri and John L Junkins. Generic smoothing for optimal bang-off-bang spacecraft maneuvers. *Journal of Guidance, Control, and Dynamics*, 41(11):2470–2475, 2018.

[35] Ehsan Taheri, John L. Junkins, Ilya Kolmanovsky, and Anouck Girard. A novel approach for optimal trajectory design with multiple operation modes of propulsion system, part 2. *Acta Astronautica*, 172:166–179, 2020.

[36] Vishala Arya, Ehsan Taheri, and John L Junkins. A composite framework for co-optimization of spacecraft trajectory and propulsion system. *Acta Astronautica*, 178:773–782, 2021.

[37] Vishala Arya, Ehsan Taheri, and John Junkins. Electric thruster mode-pruning strategies for trajectory-propulsion co-optimization. *Aerospace Science and Technology*, 116:106828, 2021.

[38] Vishala Arya, Ehsan Taheri, and John L Junkins. Low-thrust gravity-assist trajectory design using optimal multimode propulsion models. *Journal of Guidance, Control, and Dynamics*, 44(7):1280–1294, 2021.

[39] Sandeep Singh, John Junkins, Brian Anderson, and Ehsan Taheri. Eclipse-conscious transfer to lunar gateway using ephemeris-driven terminal coast arcs. *Journal of Guidance, Control, and Dynamics*, 44(11):1972–1988, 2021.

[40] Ryota Nakano, Ehsan Taheri, and Masatoshi Hirabayashi. Time-optimal and fuel-optimal trajectories for asteroid landing via indirect optimization. In *AIAA SCITECH 2022 Forum*, page 1128, 2022.

[41] Ehsan Taheri and Nan Li. L2 norm-based control regularization for solving optimal control problems. *IEEE Access*, 11:125959–125971, 2023.

[42] Keziban Saloglu and Ehsan Taheri. Co-optimization of spacecraft and low-thrust trajectory with direct methods. *arXiv preprint arXiv:2408.15943*, 2024.

[43] Ehsan Taheri, Ella M Atkins, and Ilya Kolmanovsky. Performance comparison of smoothing functions for indirect optimization of minimum-fuel low-thrust trajectories. In *2018 Space Flight Mechanics Meeting*, page 0214, 2018.

[44] Vishala Arya, Ehsan Taheri, and J Junkins. Hyperbolic tangent-based smoothing with state transition matrix implementation for generating fuel-optimal trajectories. In *29th AAS/AIAA Space Flight Mechanics Meeting*, 2019.

[45] Rudolf Emil Kalman et al. Contributions to the theory of optimal control. *Bol. soc. mat. mexicana*, 5(2):102–119, 1960.

[46] Stephen Boyd, Laurent El Ghaoui, Eric Feron, and Venkataramanan Balakrishnan. *Linear matrix inequalities in system and control theory*. SIAM, 1994.

[47] Lieven Vandenberghe and Stephen Boyd. Semidefinite programming. *SIAM review*, 38(1):49–95, 1996.

[48] Imre Pólik and Tamás Terlaky. A survey of the s-lemma. *SIAM review*, 49(3):371–418, 2007.

[49] Behcet Acikmese and Scott R Ploen. Convex programming approach to powered descent guidance for mars landing. *Journal of Guidance, Control, and Dynamics*, 30(5):1353–1366, 2007.

[50] Lars Blackmore, Behçet Açikmeşe, and Daniel P Scharf. Minimum-landing-error powered-descent guidance for mars landing using convex optimization. *Journal of guidance, control, and dynamics*, 33(4):1161–1171, 2010.

[51] Behçet Açıkmeşe, John M Carson, and Lars Blackmore. Lossless convexification of nonconvex control bound and pointing constraints of the soft landing optimal control problem. *IEEE transactions on control systems technology*, 21(6):2104–2113, 2013.

[52] Ping Lu and Xinfu Liu. Autonomous trajectory planning for rendezvous and proximity operations by conic optimization. *Journal of Guidance, Control, and Dynamics*, 36(2):375–389, 2013.

[53] Xinfu Liu and Ping Lu. Robust trajectory optimization for highly constrained rendezvous and proximity operations. In *AIAA Guidance, Navigation, and Control (GNC) Conference*, page 4720, 2013.

[54] Xinfu Liu and Ping Lu. Solving nonconvex optimal control problems by convex optimization. *Journal of Guidance, Control, and Dynamics*, 37(3):750–765, 2014.

[55] Xinfu Liu, Zuojun Shen, and Ping Lu. Entry trajectory optimization by second-order cone programming. *Journal of Guidance, Control, and Dynamics*, 39(2):227–241, 2016.

[56] Zhenbo Wang and Michael J Grant. Constrained trajectory optimization for planetary entry via sequential convex programming. *Journal of Guidance, Control, and Dynamics*, 40(10):2603–2615, 2017.

[57] Xinfu Liu. Fuel-optimal rocket landing with aerodynamic controls. *Journal of Guidance, Control, and Dynamics*, 42(1):65–77, 2019.

[58] Matthew W Harris and Behçet Açıkmeşe. Minimum time rendezvous of multiple spacecraft using differential drag. *Journal of Guidance, Control, and Dynamics*, 37(2):365–373, 2014.

[59] Danylo Malyuta, Michael Szmuk, and Behcet Acikmese. Lossless convexification of non-convex optimal control problems with disjoint semi-continuous inputs. *arXiv preprint arXiv:1902.02726*, 2019.

[60] A. Domahidi, E. Chu, and S. Boyd. ECOS: An SOCP solver for embedded systems. In *European Control Conference (ECC)*, pages 3071–3076, 2013.

[61] Jacob Mattingley and Stephen Boyd. Cvxgen: A code generator for embedded convex optimization. *Optimization and Engineering*, 13:1–27, 2012.

[62] Daniel Dueri, Jing Zhang, and Behçet Açikmese. Automated custom code generation for embedded, real-time second order cone programming. *IFAC Proceedings Volumes*, 47(3):1605–1612, 2014.

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

[64] Michael Szmuk, Carlo Alberto Pascucci, Daniel Dueri, and Behcet Açikmeşe. Convexification and real-time on-board optimization for agile quad-rotor maneuvering and

obstacle avoidance. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4862–4868. IEEE, 2017.

[65] Gregory Whiffen. Mystic: Implementation of the static dynamic optimal control algorithm for high-fidelity, low-thrust trajectory design. In *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, page 6741, 2006.

[66] Donald H Ellison, Bruce A Conway, Jacob A Englander, and Martin T Ozimek. Application and analysis of bounded-impulse trajectory models with analytic gradients. *Journal of Guidance, Control, and Dynamics*, 41(8):1700–1714, 2018.

[67] John L Junkins and Ehsan Taheri. Exploration of alternative state vector choices for low-thrust trajectory optimization. *Journal of Guidance, Control, and Dynamics*, 42(1):47–64, 2019.

[68] Arthur Earl Bryson. *Applied optimal control: optimization, estimation and control*. Routledge, 2018.

[69] Saeid Tafazzol, Ehsan Taheri, and Nan Li. Comparison of control regularization techniques for minimum-fuel low-thrust trajectory design using indirect methods. *arXiv preprint arXiv:2409.01490*, 2024.

[70] MJH Walker, B Ireland, and Joyce Owens. A set modified equinoctial orbit elements. *Celestial mechanics*, 36(4):409–419, 1985.

[71] Joaquim RRA Martins and John T Hwang. Review and unification of methods for computing derivatives of multidisciplinary computational models. *AIAA journal*, 51(11):2582–2599, 2013.

[72] Eric A. Butcher. Analytical mechanics of space systems: Fourth edition [bookshelf]. *IEEE Control Systems*, 39(5):110–111, October 2019.

[73] Hanspeter Schaub and John L Junkins. *Analytical mechanics of space systems*. Aiaa, 2003.

[74] Mohammadreza Saghamanesh, Ehsan Taheri, and Hexi Baoyin. Interplanetary gravity-assist fuel-optimal trajectory optimization with planetary and solar radiation pressure perturbations. *Celestial Mechanics and Dynamical Astronomy*, 132:1–21, 2020.

[75] Ryan P Russell. Primer vector theory applied to global low-thrust trade studies. *Journal of Guidance, Control, and Dynamics*, 30(2):460–472, 2007.

[76] Bruce A Conway. A survey of methods available for the numerical optimization of continuous dynamic systems. *Journal of Optimization Theory and Applications*, 152:271–306, 2012.

[77] Bruce A Conway and Stephen W Paris. Spacecraft trajectory optimization using direct transcription and nonlinear programming. *Spacecraft trajectory optimization*, 29:37, 2010.

[78] Prasun N Desai and Bruce A Conway. Six-degree-of-freedom trajectory optimization using a two-timescale collocation architecture. *Journal of guidance, control, and dynamics*, 31(5):1308–1315, 2008.

[79] Jacob A Englander and Bruce A Conway. Automated solution of the low-thrust interplanetary trajectory problem. *Journal of Guidance, Control, and Dynamics*, 40(1):15–27, 2017.

[80] Xinfu Liu, Ping Lu, and Binfeng Pan. Survey of convex optimization for aerospace applications. *Astrodynamics*, 1(1):23–40, September 2017.

[81] Danylo Malyuta, Taylor P. Reynolds, Michael Szmuk, Thomas Lew, Riccardo Bonalli, Marco Pavone, and Behcet Acikmese. Convex Optimization for Trajectory Generation, June 2021. arXiv:2106.09125 [cs, eess, math].

[82] Zhenbo Wang and Michael J. Grant. Optimization of Minimum-Time Low-Thrust Transfers Using Convex Programming. *Journal of Spacecraft and Rockets*, 55(3):586–598, May 2018.

[83] Zhenbo Wang and Michael J. Grant. Minimum-Fuel Low-Thrust Transfers for Spacecraft: A Convex Approach. *IEEE Transactions on Aerospace and Electronic Systems*, 54(5):2274–2290, October 2018.

[84] Gao Tang, Fanghua Jiang, and Junfeng Li. Fuel-Optimal Low-Thrust Trajectory Optimization Using Indirect Method and Successive Convex Programming. *IEEE Transactions on Aerospace and Electronic Systems*, 54(4):2053–2066, August 2018.

[85] Yuki Kayama, Kathleen C. Howell, Mai Bando, and Shinji Hokamoto. Low-Thrust Trajectory Design with Successive Convex Optimization for Libration Point Orbits. *Journal of Guidance, Control, and Dynamics*, 45(4):623–637, April 2022.

[86] Christian Hofmann and Francesco Topputo. Toward On-Board Guidance of Low-Thrust Spacecraft in Deep Space Using Sequential Convex Programming. page 19, 2021.

[87] Zhenbo Wang and Michael J Grant. Minimum-fuel low-thrust transfers for spacecraft: A convex approach. *IEEE Transactions on Aerospace and Electronic Systems*, 54(5):2274–2290, 2018.

[88] Michael Szmuk and Behcet Acikmese. Successive convexification for 6-dof mars rocket powered landing with free-final-time. In *2018 AIAA Guidance, Navigation, and Control Conference*, page 0617, 2018.

[89] Danylo Malyuta, Taylor P Reynolds, Michael Szmuk, Thomas Lew, Riccardo Bonalli, Marco Pavone, and Behcet Acikmese. Convex optimization for trajectory generation. *arXiv preprint arXiv:2106.09125*, 2021.

[90] Yuanqi Mao, Michael Szmuk, Xiangru Xu, and Behçet Açikmese. Successive convexification: A superlinearly convergent algorithm for non-convex optimal control problems. *arXiv preprint arXiv:1804.06539*, 2018.

[91] Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

[92] Michael JD Powell. A hybrid method for nonlinear equations. *Numerical methods for nonlinear algebraic equations*, pages 87–161, 1970.

[93] Nicolò Bernardini, Nicola Baresi, and Roberto Armellin. State-dependent trust region for successive convex programming for autonomous spacecraft. *Astrodynamics*, pages 1–23, 2024.

[94] John L Junkins. von karman lecture: Adventures on the interface of dynamics and control. *Journal of Guidance, Control, and Dynamics*, 20(6):1058–1071, 1997.

[95] John L Junkins and Puneet Singla. How nonlinear is it? a tutorial on nonlinearity of orbit and attitude dynamics. *The Journal of the Astronautical Sciences*, 52:7–60, 2004.

[96] Alberto Fossà, Roberto Armellin, Emmanuel Delande, Matteo Losacco, and Francesco Sanfedino. Multifidelity orbit uncertainty propagation using taylor polynomials. In *AIAA SciTech 2022 Forum*, page 0859, 2022.

[97] Ahmad Bani Younes, James Turner, Manoranjan Majji, and John Junkins. High-order uncertainty propagation enabled by computational differentiation. In *Recent Advances in Algorithmic Differentiation*, pages 251–260. Springer, 2012.

[98] A Bani Younes and J Turner. High order state transition tensors of perturbed orbital motion using computational differentiation. In *Proceedings of the 26th AAS/AIAA Space Flight Mechanics Meeting, Napa, CA, USA, AAS*, pages 16–342, 2016.

[99] Ahmad Bani Younes. Exact computation of high-order state transition tensors for perturbed orbital motion. *Journal of Guidance, Control, and Dynamics*, 42(6):1365–1371, 2019.

[100] Matteo Losacco, Alberto Fossà, and Roberto Armellin. Low-order automatic domain splitting approach for nonlinear uncertainty mapping. *Journal of Guidance, Control, and Dynamics*, 47(2):291–310, 2024.

[101] Aaron Meurer, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K. Moore, Sartaj Singh,

Thilina Rathnayake, Sean Vig, Brian E. Granger, Richard P. Muller, Francesco Bonazzi, Harsh Gupta, Shivam Vats, Fredrik Johansson, Fabian Pedregosa, Matthew J. Curry, Andy R. Terrel, Štěpán Roučka, Ashutosh Saboo, Isuru Fernando, Sumith Kulal, Robert Cimrman, and Anthony Scopatz. Sympy: symbolic computing in python. *PeerJ Computer Science*, 3:e103, January 2017.

[102] Joel A E Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36, 2019.

[103] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[104] Jorge J Moré, Burton S Garbow, and Kenneth E Hillstrom. User guide for minpack-1. Technical report, CM-P00068642, 1980.

[105] Zhijun Chen, Jiaxiang Luo, Quan Chen, Yong Zhao, Yuzhu Bai, and Xiaoqian Chen. Fast estimation of initial costate for time-optimal trajectory based on surrogate model. *Journal of Aerospace Engineering*, 36(6):04023078, 2023.

[106] Mirco Rasotto, Alessandro Morselli, Alexander Wittig, Mauro Massari, Pierluigi Di Lizia, Roberto Armellin, Celia Valles, and Guillermo Ortega. Differential algebra space toolbox for nonlinear uncertainty propagation in space dynamics. 2016.

[107] Albert Einstein. Die grundlagen der allgemeinen. *Relativitats- teorie, Annale der Physic*, 49:769, 1916.

[108] Nicolas Vasilache, Oleksandr Zinenko, Theodoros Theodoridis, Priya Goyal, Zachary DeVito, William S Moses, Sven Verdoolaege, Andrew Adams, and Albert Cohen. Tensor comprehensions: Framework-agnostic high-performance machine learning abstractions. *arXiv preprint arXiv:1802.04730*, 2018.

[109] Tim Rocktäschel. Einsum is all you need - einstein summation in deep learning. `https://rockt.github.io/2018/04/30/einsum`, 2018.

Appendices

Appendix A

Einstein Summation

Einstein notation [107] was introduced as a mathematical convention to simplify tensor contractions in physics. While it offers a concise and efficient way to represent summation over indices, it is often overlooked by alternative methods like explicit index-based notation and the Kronecker product. Explicit index-based notation, which writes out each index explicitly, provides clarity but can become verbose and cumbersome for high-dimensional tensor operations. In the original notation, considerations of covariant and contravariant indices were included. However, for the purposes of our applications, these distinctions are not relevant.

Despite the growing popularity of Einstein notation under the term "Einsum," particularly in the Deep Learning community [108, 109], there remains a lack of a standardized set of conventions for manipulating tensors using Einsum notation. This Appendix introduces a series of conventional rules for tensor contraction and computing derivatives involving tensors using Einsum notation, aiming to partially address this gap and streamline the mathematical handling of tensors.

A.1 Tensor Contraction

In Einsum notation, the first aspect to consider is the dimensions of the involved tensors. For instance, $\boldsymbol{x}_I$ has a single dimension I, making it a vector; A_{IJ} has two dimensions I and J, making it a matrix; and a third-rank tensor, B_{IJK}, has three dimensions I, J, and K. Now, suppose we intend to compute the dot product between two vectors, $\boldsymbol{x}_I$ and $\boldsymbol{y}_I$. Using explicit index-based notation, this operation is expressed as:

$$r = \boldsymbol{x} \cdot \boldsymbol{y} = \sum_{i \in I} x_i y_i, \tag{A.1}$$

where x_i, y_i are shorthand notations for the i-th entry of $\boldsymbol{x}$, and $\boldsymbol{y}$ tensors along their I dimension. However, in the Einsum notation, the same operation can be written as:

$$r = x_I y_I. \tag{A.2}$$

This indicates that we started with two one-dimensional tensors sharing the same dimension and contracted them to a scalar. The general case for any tensor contraction operation will be explained and become clearer shortly. Now, consider multiplying a matrix A_{IJ} with a vector x_J, which yields a vector y_I. Using the standard matrix multiplication notation, we have:

$$\boldsymbol{y} = A\boldsymbol{x}, \tag{A.3}$$

while in explicit index notation, the same operation can be written as:

$$y_i = \sum_{j \in J} A_{ij} x_j, \quad \forall i \in I, \tag{A.4}$$

and in Einsum notation, it becomes:

$$y_I = A_{IJ} x_J. \tag{A.5}$$

From these two examples, we can introduce the general rule for contraction between two arbitrary tensors:

$$C_{\Gamma^r} = A_{\Gamma^1} B_{\Gamma^2}, \tag{A.6}$$

where $\Gamma^{(\cdot)} := (\Gamma^{(\cdot)}_1, \ldots, \Gamma^{(\cdot)}_N)$ represents the N-tuple of dimensions for each tensor, with each $\Gamma^{(\cdot)}_a$ (for $a = 1, \cdots, N$) indicating a specific dimension.

For example, in the matrix multiplication given in Eq. (A.5), we have $\Gamma^1 = (I, J)$ for A. In explicit index notation, at each iteration, the tuples are reduced to a specific order of indices.

In this example: $A_{ij};\ i \in I,\ j \in J$. To formalize this, we define $\gamma^{(.)} = (\gamma_1^{(.)}, \ldots, \gamma_N^{(.)})$, where $\gamma_a^{(.)} \in \Gamma_a^{(.)}$ represents an individual index. For instance, in Eq. (A.6), if $\Gamma^1 = (I, J, K)$, where $I = \{1, 2, 3, 4\}, J = \{1, 2, 3\}, K = \{1, 2, 3\}$ then $\gamma^1 = (i, j, k)$ where $i \in I,\ j \in J,\ k \in K$.

Next, we introduce $\tilde{\Gamma}$ as the set form of Γ, and $\tilde{\gamma}$ as its reduced counterpart. This conversion from tuples to sets is helpful because the order of indices does not matter during tensor contraction. Finally, the contraction in Eq. (A.6) can be defined using explicit index-based notation as:

$$C_{\gamma^r} = \sum_{\tilde{\gamma}_u \in (\tilde{\Gamma}^1 \cup \tilde{\Gamma}^2) \setminus \tilde{\Gamma}^r} A_{\gamma^1} B_{\gamma^2}, \tag{A.7}$$

where the indices that do not appear in the result, are summed over (i.e., contracted). It is also possible, though less common, for the resulting tensor to have indices not present in the input arguments. In this case, the resulting tensor is repeated along the absent indices. The contraction generalizes to:

$$C_{\Gamma^r} = A^1_{\Gamma^1} A^2_{\Gamma^2} \cdots A^N_{\Gamma^N} \implies C_{\gamma^r} = \sum_{\tilde{\gamma}_u \in \left(\bigcup_i \tilde{\Gamma}^i\right) \setminus \tilde{\Gamma}^r} A^1_{\gamma^1} A^2_{\gamma^2} \cdots A^N_{\gamma^N}, \tag{A.8}$$

where A^i corresponds to an arbitrary i-th tensor and the superscript denotes the index for each tensor. While this notation might initially seem complex, the following sections demonstrate that it improves the conciseness of both implementation and the representation of equations involving tensors.

A.2 Derivative Rules

When three scalars are multiplied together, such as $d = abc$, determining the derivative $\frac{\partial d}{\partial b} = ac$ is straightforward. Now, consider a similar setup involving the product of three matrices, $D = ABC$. In this case, computing the derivative $\frac{\partial D}{\partial B}$ is not as trivial and certainly $\neq AC$. This expression would not only be incorrect, but also invalid, as the dimensions of A and C may not align for multiplication.

To leap into it, let us represent the matrix multiplication using Einsum notation:

$$D_{IL} = A_{IJ}B_{JK}C_{KL}. \tag{A.9}$$

In this form, the derivative can be computed as:

$$\frac{\partial D_{IL}}{\partial B_{JK}} = [A_{IJ}C_{KL}]_{ILJK}, \tag{A.10}$$

which closely resembles its scalar counterpart in simplicity and notation. To show why that is the case, first note that $\frac{\partial D_{IL}}{\partial B_{JK}}$ means taking the derivative of every component of D w.r.t. every component of B, which results in a 4D tensor. Looking at explicit indices:

$$D_{il} = \sum_{j\in J}\sum_{k\in K} A_{ij}B_{jk}C_{kl}, \tag{A.11}$$

and taking the derivative (with $\land$ denoting logical 'and'), we have:

$$\frac{\partial D_{il}}{\partial B_{j'k'}} = \sum_{j\in J}\sum_{k\in K} \mathbb{1}(j=j' \land k=k')A_{ij}C_{kl} = A_{ij'}C_{k'l}, \tag{A.12}$$

which leads to the same conclusion. In general, the derivative rule from Eq. (A.8) can be expressed as:

$$\frac{\partial C_{\Gamma^r}}{\partial A^e_{\Gamma^{e'}}} = [A^1_{\Gamma^1}A^2_{\Gamma^2}\cdots \underbrace{\delta_{\Gamma^e\Gamma^{e'}}}_{\frac{\partial A^e_{\Gamma^e}}{\partial A^e_{\Gamma^{e'}}}} \cdots A^N_{\Gamma^N}]_{\Gamma^r\Gamma^{e'}}, \tag{A.13}$$

where $\delta_{\Gamma^e\Gamma^{e'}}$ is the Kronecker delta tensor:

$$\delta_{\gamma\gamma'} = \begin{cases} 1, & \gamma = \gamma', \\ 0, & \text{otherwise.} \end{cases} \tag{A.14}$$

In other words, entries with identical indices are assigned a value of 1, and all other entries are assigned 0. By $\Gamma^e\Gamma^{e'}$ or $\Gamma^r\Gamma^{e'}$, we denote the concatenation of dimensions.

A subtlety arises in Eq. (A.10), where we did not introduce these prime (') dimensions and computed the derivative without adding new dimensions. This approach works without issue if the resulting tensor does not share any dimensions with the differentiation variable. However, when dimensions are shared, taking the partial derivative yields 0 for distinct indices within the same dimension, as they are independent of each other. By including only one pair for each shared dimension, no information is lost, but the notation becomes less systematic. To illustrate the difference between these conventions, consider the derivative of a vector $\boldsymbol{x}$ w.r.t. itself. For consistency, the standard convention is to write:

$$\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{x}} = I, \tag{A.15}$$

rather than yielding a vector of ones ($\mathbf{1}$), which could also preserve the information but lacks systematic notation. Setting the groundwork for derivatives, we will next examine the Product Rule and Chain Rule within the context of Einsum.

1. **Product Rule**: Starting from Eq. (A.8):

$$C_{\Gamma^r}(x_\zeta) = A^1_{\Gamma^1}(x_\zeta) A^2_{\Gamma^2}(x_\zeta) \cdots A^N_{\Gamma^N}(x_\zeta), \tag{A.16}$$

where the tensors now depend on the tensor variable x. To compute the derivative with respect to x, we have:

$$\begin{aligned} \frac{\partial C_{\Gamma^r}(x_\zeta)}{\partial x_\zeta} = & \\ & \left[\frac{\partial}{\partial x_\zeta}(A^1_{\Gamma^1}(x_\zeta)) A^2_{\Gamma^2}(x_\zeta) \cdots A^N_{\Gamma^N}(x_\zeta)\right]_{\Gamma^r \zeta} \\ & + \left[A^1_{\Gamma^1}(x_\zeta) \frac{\partial}{\partial x_\zeta}(A^2_{\Gamma^2}(x_\zeta)) \cdots A^N_{\Gamma^N}(x_\zeta)\right]_{\Gamma^r \zeta} \\ & \cdots \\ & + \left[A^1_{\Gamma^1}(x_\zeta) A^2_{\Gamma^2}(x_\zeta) \cdots \frac{\partial}{\partial x_\zeta}(A^N_{\Gamma^N}(x_\zeta))\right]_{\Gamma^r \zeta} . \end{aligned} \tag{A.17}$$

For an avid user of Einsum, the summation in the product rule can be changed by first concatenating each product term and then contracting the resulting dimensions:

$$\frac{\partial C_{\Gamma^r}(x_\zeta)}{\partial x_\zeta} = [R_{\Gamma^r \zeta N}]_{\Gamma^r \zeta}\,, \tag{A.18}$$

where Einsum is used as a unary operator to perform summation along the dimension N.

2. **Chain Rule**:

   Consider a chain of tensor functions:

$$A_\alpha(B_\beta(C_\gamma)), \tag{A.19}$$

   then:

$$\left[\frac{\partial A_\alpha(B_\beta(C_\gamma))}{\partial C_\gamma}\right]_{\alpha\gamma} = \left[\left[\frac{\partial A_\alpha(B_\beta)}{\partial B_\beta}\right]_{\alpha\beta}\left[\frac{\partial B_\beta(C_\gamma)}{\partial C_\gamma}\right]_{\beta\gamma}\right]_{\alpha\gamma}, \tag{A.20}$$

where α,β, and γ are tuple of dimensions as before.