# Taylor Series Expansion to Model the Dynamics of a Spacecraft in Convex Optimization: Application to Deep-Space Guidance

Tesi di Laurea Magistrale in
Space Engineering - Ingegneria Spaziale

Author: **Lucas Eddy Théo Baraton**

Student ID: 963751
Advisor: Prof. Francesco Topputo
Co-advisors: Christian Hofmann, Andrea Carlo Morelli
Academic Year: 2021-2022

# Abstract

The recent development of CubeSats equipped with low-thrust propulsion for deep-space exploration brought the need to improve their autonomy. In particular, an essential aspect is their capacity to compute their trajectory onboard (and thus solving an optimal control problem in real-time), with very few computational resources. Convex optimization is a good candidate to perform this task thanks to its robustness and computational efficiency. However, due to the linearization required by the method to model the dynamics, the latter is not precisely accounted for, causing reliability issues. This thesis proposes to use Taylor series expansion to include the dynamics in a more accurate fashion inside convex optimization algorithms, where the classic formulation of the former is modified to fit the convex frame. Two methods are developed. Both are able to solve the low-thrust trajectory problem but grant different performances. The first one improves the accuracy in most cases but is computationally heavier and has convergence issues. The second one solves the convergence issues of the former and is faster, but is very inaccurate with respect to state-of-the-art methods. Perspectives of improvement are presented and an alternative method involving difference of convex functions is given to extend the work.

**Keywords:** convex optimization, deep-space guidance, low-thrust trajectory, Taylor series expansion, nonlinear dynamics

# Abstract in lingua italiana

Il recente sviluppo di CubeSats dotati di propulsione a bassa spinta per l'esplorazione dello spazio profondo ha portato alla necessità di migliorarne l'autonomia. In particolare, un aspetto essenziale è la capacità di calcolare la traiettoria a bordo (e quindi di risolvere un problema di controllo ottimale in tempo reale), con pochissime risorse computazionali. L'ottimizzazione convessa è un buon candidato per svolgere questo compito grazie alla sua robustezza ed efficienza computazionale. Tuttavia, a causa della linearizzazione richiesta dal metodo per modellare la dinamica, quest'ultima non viene considerata con precisione, causando problemi di affidabilità. Questa tesi propone di utilizzare l'espansione in serie di Taylor per includere la dinamica in modo più accurato all'interno degli algoritmi di ottimizzazione convessi, dove la formulazione classica del primo viene modificata per adattarsi alla struttura convessa. Vengono sviluppati due metodi. Entrambi sono in grado di risolvere il problema della traiettoria a bassa spinta, ma garantiscono prestazioni diverse. Il primo migliora l'accuratezza nella maggior parte dei casi, ma è computazionalmente più pesante e presenta problemi di convergenza. Il secondo risolve i problemi di convergenza del primo ed è più veloce, ma è molto impreciso rispetto ai metodi classici. Vengono presentate prospettive di miglioramento e viene fornito un metodo alternativo che coinvolge la differenza di funzioni convesse per estendere il lavoro.

**Parole chiave:** ottimizzazione convessa, guida nello spazio profondo, traiettoria a bassa spinta, espansione in serie di Taylor, dinamiche non lineari

# Contents

# Introduction

In the last few years, a new trend arose in the space industry: the "New Space" [1]. Historical space actors (ESA, NASA, Arianespace, CNES, ASI, etc) as well as young companies are participating to develop this new economy by working on a growing number of innovative concepts. In the next decade, it is estimated that more and more objects (especially reduced-size spacecraft like NanoSats and CubeSats) will fly and thus the access to space will be broaden to more actors. Usually, CubeSats are launched as secondary payloads, assisting the main spacecraft. But recent developments in research and technology allowed the former to be more and more efficient and powerful, and ready to perform their own missions. CubeSats are interesting from many points of view: they are light, allowing one launch to bring several in space, and their production requires less resources with respect to state-of-the-art spacecraft. Thus, their use are necessary to improve the sustainability of the space industry. Furthermore, their short size and their (relative) simplicity make their use and manufacturing available to entities with less economic and technical resources than the usual actors but that have projects requiring access to space (modest research laboratories, start-ups, student associations, etc).

In this new framework, several miniaturized probes equipped with low-thrust propulsion, instead of one big spacecraft, are the main trend of development regarding space activities. Indeed, low-thrust propulsion offers an advantage with respect to chemical engines in terms of fuel consumption, and thus of mission's economic cost. To minimize the required workforce on ground to operate a growing number of spacecraft, their autonomy is key. In particular, current spacecraft need to receive instructions from the guidance, navigation and control (GNC) engineers to maintain and follow a nominal trajectory. It is a complex task when considering a swarm of dozens of NanoSats, or a CubeSat exploring Deep-space and operating with very limited resources. Hence, the need for miniaturized spacecraft to compute their trajectory autonomously with few computational resources has never been higher. Nevertheless, the computation of the trajectory is an optimization problem, that can become difficult when no powerful computer is available. Hence, one has to develop methods that are accurate (the CubeSat shall follow the nominal path), robust (an algorithm failure isn't allowed in space) and efficient (for the reasons mentioned before).

Convex optimization is a suitable candidate to solve the aforementioned problem. It is a method that uses the convexity theory to solve the low-thrust trajectory optimization problem in a fast, efficient and robust fashion, and is thus interesting for onboard use. However, the classical way to deal with non-linearities induced by the spacecraft's dynamics relies on a local linearization with respect to a reference trajectory to comply with the convex programming framework. The error of the linearization process with respect to the true dynamics causes reliability concerns. A lot of work successfully dealing with this issue has been proposed in the literature, but as convex programming is a quite recent breakthrough (especially when applied to trajectory optimization in space), there are still some room for improvement concerning those algorithms: this is where this thesis fits.

The main goal of this thesis is to investigate novel methods to include non-linearities related to the dynamics in convex optimization algorithms. Instead of linearizing, the proposed solution to be developed is to employ Taylor Series Expansion (TSE) where the high-order terms are modified to fit the convex framework (we will use the word "convexify" from now on). To do so, methods will be developed theoretically based on the literature and implemented to get numerical evidences that they are valid.

The choice of exploring the convexified TSE has been made by selecting, adapting, implementing and comparing a few promising methods (not explicitly related to our aim), taken from the literature, that could be customized to better take into account the non-linear dynamics with respect to a pure linearization. The one that brought the most novelty and on which most expectations were placed was the TSE, because to the best of the team's knowledge, it has never been investigated before to reach our purpose.

This thesis raises expectations about its outcomes in short, mid and long terms. The former will be considered successful if, in a short-term, novel methods are developed which are able to solve the low-thrust trajectory optimization problem in particular cases (from the field of deep-space guidance). It would be a great achievement if, for two convex optimization algorithms of the same kind (meaning that their comparison is fair), the one that uses the methods developed in that project accounts better for the non-linear dynamics than the one using classical approaches, granting a better accuracy and thus a greater reliability. In a mid-term, if the work of this thesis is interesting and effort is put to deepen it, it could serve to make a step towards the successful demonstration of convex optimization in a realistic spacecraft simulator, or better, in a technological demonstration mission (in the ESA mission M-ARGO for instance). Finally, the long-term aim of this work is to participate (by an infinitesimal contribution) to enable autonomous miniaturized probes to explore our universe.

The research questions that are asked in this thesis are:

- Does the convexified Taylor Series Expansion methods to model the dynamics improve the reliability of the optimization solution ?
- How does it compare to state-of-the-art methods ?

This Master Thesis is part of the study at Politecnico di Milano, and is concluding the Master's degree "Space Engineering". It has been done within the Deep-space Astrodynamics Research & Technology (DART) Group and in particular in the frame of the ERC-funded project *EXTREMA* [2]. The thesis is supervised by Prof. Francesco Topputo and co-supervised by Christian Hofmann and Andrea Carlo Morelli.

# 1 | Literature Review

The low-thrust trajectory optimization problem has been historically studied as an Optimal Control Problem (OCP) [3]. To solve the former, two main classes of methods are available: the indirect and direct methods [4]. Indirect methods are based on Pontryagin's maximum principle [5] and the calculus of variations (in [6] an optimum low-thrust transfer between two elliptic orbits is computed using such method). Once the necessary conditions are derived, a two-point boundary value problem is solved. Although indirect methods are precise, they are complicated to initialize and thus not robust enough for onboard, real-time, implementation. Instead, direct methods discretize the trajectory to solve a finite parameter optimization problem using for instance Non-Linear Programming (NLP) techniques [7], but not exclusively. The main direct methods for low-thrust space trajectory optimization are gathered in [8].

Convex Optimization (CO) falls into the last category. An optimization problem can be solved by CO when minimizing convex functions over convex sets. Several other optimization categories are particular cases of convex programming problems (e.g. linear, quadratic, second-order cone, semi-definite and cone programming). CO theoretical [9] and practical [10] features are well known but, for aerospace applications, naturally convex functions are very rare. Considering the two-body problem's equations of motion for example, it can be seen that the term $1/r^3$ will be a source of non-convexity. Thus, scientists and engineers employ several techniques to convexify those functions, while keeping the solution of the convex problem feasible and optimal (or sub-optimal) for the original nonconvex one [11].

Before the solution process can occur, the original nonconvex problem has to be convexified. The first convexification technique implemented in this work is called lossless convexification. Its usual role is to handle nonconvex control constraints. It consists in relaxing the latter, switching from an equality to an inequality. Açıkmeşe and Blackmore proved in [12] that an optimal solution for the relaxed problem is also optimal for the original one, demonstrating that the former and the new constraints are exactly equivalent. Therefore, lossless convexification is a simple, yet very powerful technique. However, it can only be applied to particular constraints. In general, every constraint can not benefit from the performances of lossless convexification. The latter is nowadays systematically performed in the large majority of CO algorithm. The method is illustrated by solving a Mars landing problem in [13]. In our case, the process is required to convexify the constraint enforcing the thrust vector's norm to be equal to the thrust magnitude. It is illustrated for instance in [14–16]. For an application in the aeronautic military field, the reader can refer to [17] (aerodynamically controlled missile).

The second one is the Successive Convexification (SCvx) framework. Its principle is to linearize nonlinear constraints (that can not benefit from lossless convexification for instance) at a reference solution, which is usually the previous iteration's one, to convexify them. In [18], Mao et al. demonstrated strong convergence results regarding methods employing SCvx with respect to other numerical optimization methods. They showed that under some weak assumptions, the algorithm presents a superlinear convergence rate, making it faster than most of the NLP techniques. It is thus widely used in the aerospace field thanks to those properties. In such algorithms, the dynamical constraints are expressed as linear equalities and are thus affine (hence convex) with respect to the problem's variables.

The type of algorithm using SCvx is part of a large class called Sequential Convex Programming (SCP), which is the state-of-the-art way of dealing with nonconvex constraints due to nonlinear dynamics. SCP consists in solving a sequence of convex optimal control subproblems to converge towards a locally optimal solution for the nonconvex original one. Algorithms employing this process can have various features. Malyuta et al. [19] proposed a comprehensive tutorial on convex optimization in general and SCP in particular. SCP often employs SCvx but not exclusively. The *GuSTO* algorithm [20] possesses for example novel features with respect to previous SCP methods granting for some cases better convergence and optimality performances.

Once the convex sub-problem is properly stated, it can be solved using Interior-Point Methods (IPM) [21] that are, depending on the application, either generic or purposely designed. For example, the *Embedded Conic Solver (ECOS)* [22] is able to solve a wide variety of Second-Order Cone Program (SOCP) and is powerful for small to medium-sized problems. However, when the performances of generic solvers like *ECOS* are insufficient, one can develop their own IPM to better fit their particular problem. In [23], Dueri and Açıkmeşe developed a tuned method to fit their powered descent guidance algorithm for planetary pinpoint landing. The method was "flight-tested by NASA Jet Propulsion Laboratory and the NASA Flight Opportunities Program in 2013", proving that CO associated to an adapted IPM is a promising technology for the aerospace industry and, in our case, deep-space exploration.

Once the problem has been convexified, the continuous OCP shall be transformed into a finite parameter optimization (discretization) and the differential dynamical constraints into algebraic ones (collocation). To do so, the integral equations of the dynamics must be approximated (by numerical integration) and the points where the dynamical constraints are enforced (collocation points) must be chosen. Several methods exist to perform this discretization step [24] with various performances [25]. The convergence rate of a collocation method is for instance an important feature regarding the latter as it is directly linked with the speed and the computational efficiency of the algorithm. Several methods have been theoretically studied under that angle as the Gauss collocation method for respectively unconstrained [26] and constrained [27] OCPs, or the Radau collocation method for unconstrained OCP [28].

In this paragraph, examples of collocation and discretization methods for astrodynamics will be under focus. The most simple one is the trapezoidal discretization. It relies on the trapezoidal rule to approximate integrals. Its simplicity makes it easy to manipulate in order to try new approaches or for relatively low-complexity problems such as the Earth-Mars transfer [16]. However, advanced algorithms don't use this method due to its poor accuracy. A good solution to improve precision without highly increasing the complexity is the Hermite-Simpson discretization scheme [8]. It consists in using Hermite interpolation [29] to approximate the dynamics, and to evaluate the integral with the Simpson's rule [30]. The advantage of the method with respect to trapezoidal discretization is, for a given time interval, to impose the dynamical constraints at both boundaries (called nodes) and at the collocation point that lies in the center of the interval, whereas the former only imposes the dynamics at the nodes. In this work, the two previous techniques are employed, but the end of the paragraph will mention two other methods that are tremendously important in the field of astrodynamics. For advanced algorithm, where high-accuracy is needed, one can extend the Hermite-Simpson scheme to arbitrary order interpolating polynomials. It is called arbitrary-order Gauss-Lobatto collocation method [31]. The collocation points are computed by finding the roots of the derivative of a well-chosen Legendre polynomial and are thus named Legendre-Gauss-Lobatto (LGL) collocation points [32]. In [15], Morelli et al. solved an Earth-Dionysus low-thrust trajectory problem using LGL. The method shows a higher convergence rate and accuracy of the results with respect to Hermite-Simpson. An other method worth to mention is the adaptive Flipped Radau Pseudospectral Discretization (FRPD) studied in [33]. In [34], the differences with the LGL method is explained and the two methods are compared. FRPD also showed good performances for space-based practical applications [35, 36].

In the last decade, the interests for CO algorithms (and SCP in particular) have quickly grown within the aerospace engineering community for its broad set of potential applications. First, both minimum-time [37] and minimum-fuel [14] problems have been solved. Furthermore, the method dealt with a wide set of aerospace problems [38]. Indeed, CO can be used for both military or civil, aeronautic and space fields. In [17] Liu et al. proposed an algorithm to deal with the terminal guidance of aerodynamically controlled missiles, where the constraints on the final angle and dynamical pressure are taken into account (and thus convexified). Multi-rotor aerial vehicles motion planning problems have also been treated by Mao et al. [39], and the performances of their practical onboard implementation have been assessed. Concerning the space field, every stage of the mission is covered by CO algorithms. From the launchers point of view, a multistage launcher ascent trajectory optimization is solved in [40] and the descent of a reusable launcher (reusability being one of the most studied topic in the field nowadays) with a convex approach is discussed in [41]. The planetary entry [42] and landing [13] are also one of many applications. For our topic in particular, deep-space guidance, a lot of work have already been cited in the previous lines. However, two more applications are worth mentioning because of their novelty and the high expectations they raise. First, planetary ballistic capture is a way of being captured in orbit without having to spend fuel (or with minor orbit corrections) making possible an orbit insertion with low-thrust technology. It has been first performed by the Japanese spacecraft Hiten in 1991 [43], and the BepiColombo mission is also supposed to use this orbit insertion strategy around Mercury in the next years [44]. In [45], Morelli et al. developed a convex guidance approach to target the corridors where this capture is feasible, which is a big challenge because of the very narrow conditions to meet in order to perform such a manoeuvre. Then, with the development of new missions targeting asteroids (for science, planetary protection or mining for instance), guidance and navigation around these bodies are key. For instance, the NASA mission OSIRIS-REx [46] launched in 2016 will return asteroid samples on Earth in 2023. Thanks to their exponential development, CubeSats will soon be able to perform those missions, in a more and more autonomous fashion. The ESA mission M-ARGO (that could be launched between 2023 and 2025) has for objective to demonstrate autonomous navigation around its target [47]. Following this trend, autonomous guidance around asteroids is an important topic as it would represent a leap towards missions' full autonomy. CO algorithm have also been proposed to cover the latter, and in particular, the landing problem have been addressed for irregularly-shaped asteroids [48].

Nevertheless, when developing a CO algorithm, one has to comply with a set of conditions. The one followed classically is the *Disciplined Convex Programming (DCP)* framework

[49]. *DCP* is a list of rules and guidelines to obey when writing a CO algorithm, which ensures that the compiling of the code will go smoothly. Following those ensures that all the functions called are convex, as well as all the constraints. In this thesis, a *MatLab* software called *CVX* [50, 51] will be used to perform numerical simulations to check that the *DCP* framework is respected. However, in the general case, it is the operator's role to verify that the right conditions for convex programming are satisfied.

Consequently, in state-of-the-art CO algorithms for aerospace applications, the standard way of dealing with nonlinear (and nonconvex) dynamics is to linearize it (to get affine equality constraints, which is one of the *DCP* rules). The linearization process leads to an error with respect to the true dynamics that increases fast when deviating from the reference solution, which causes reliability issues. Precise collocation methods to account for the dynamics (LGL for instance) improve this aspect but don't prevent the deviation from the reference. This problem is usually dealt with by employing a trust region to force the algorithm to stay close to it [35, 52]. It is a powerful method that possesses a lot of variations [25] and also have some weaknesses, like the oscillation phenomenon occurring when the current solution is close to a local optimum [53]. Previous work attempted other methods in addition of the trust-region to take a better care of non-linearities: in [54], Foust, Chung and Hadaegh proposed a nonlinear dynamics correction to the classical SCP algorithm using Lipschitz coefficients to correct the constraints, applied to Unmanned Aerial Vehicles' (UAV) motion planning. In [55], non-linearities (not the dynamical ones but those induced by physical inequality constraints, like obstacles avoidance) are coped with using convexified Taylor Series Expansion (TSE) to obtain a convex approximation of the constraints. In this last paper, a method is given to convexify the terms up to an arbitrary order, and thus obtaining an Inner-Convex Approximation (ICA) [55]. The latter is a very interesting way of approximating a function because first, as the name suggests, it is convex and thus can be integrated into CO algorithms by definition. However, adapting it to our purpose requires efforts. Additionally, the original function and its ICA coincide when evaluated on the reference solution. It is also the case for their gradients. Finally, ICAs are enjoying some very interesting mathematical properties of convergence and recursive feasibility [56], that could be exploited to reach the goal of this thesis by adapting the method presented in [55] to model the dynamical equality constraint.

# 2 Classic Problem Statement

In this chapter, the OCP is stated and convexified with state-of-the-art methods and a simple algorithm is proposed to solve it.

## 2.1. Equations of Motion

### 2.1.1. Two-Body Problem

We consider a spacecraft of mass $m$, powered by a low-thrust engine, moving in a two-body environment with the Sun as primary body and no other perturbation. We denote:

- $\mu$ the Sun's planetary constant
- $\mathbf{r} = [r_x,\ r_y,\ r_z]^\top$ the spacecraft's position vector in cartesian coordinates
- $\mathbf{v} = [v_x,\ v_y,\ v_z]^\top$ the spacecraft's velocity vector in cartesian coordinates
- $\mathbf{T} = [\mathrm{T}_x,\ \mathrm{T}_y,\ \mathrm{T}_z]^\top$ the thrust's vector
- $\mathrm{T} = ||\mathbf{T}||_2$ the thrust's magnitude
- $v_e = I_{sp}\ g_0$ the engine's exhaust velocity, with $I_{sp}$ the engine's specific impulse and $g_0$ the gravitational acceleration at sea level
- $\mathbf{x} = [\mathbf{r},\ \mathbf{v},\ m]^\top$ the state variable

The equations of motion (EoM) are:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{T}) = \begin{cases} \dot{\mathbf{r}} = \mathbf{v} \\ \dot{\mathbf{v}} = \dfrac{-\mu}{r^3}\mathbf{r} + \dfrac{\mathbf{T}}{m} \\ \dot{m} = \dfrac{-\mathrm{T}}{v_e} \end{cases} \tag{2.1}$$

### 2.1.2. Normalization

For numerical reasons, all the quantities that appear in the equations of motion are normalized. The reference values are gathered in table 2.1.

Table 2.1: Normalization quantities

| **Quantity** | **Value** | **Description** |
|---|---|---|
| Distance $R_0$ | $1AU = 1.49597870.10^8\ km$ | Astronomical unit |
| Velocity $V_0$ | $\sqrt{\mu/R_0} = 29.78469190\ km.s^{-1}$ | Earth's circular velocity |
| Time $T_0$ | $R_0/V_0 = 5.02264286.10^6\ s$ | Ratio between distance and velocity |
| Mass $m_0$ | *Problem-dependent* | Spacecraft's initial mass |
| Thrust $\mathrm{T}_{max}$ | *Problem-dependent* | Maximum thrust available onboard |

The normalized EoM are:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{T}) = \begin{cases} \dot{\mathbf{r}} = \mathbf{v} \\ \dot{\mathbf{v}} = \dfrac{-\mathbf{r}}{r^3} + \dfrac{c\mathbf{T}}{m} \qquad \text{where} \quad c = \dfrac{\mathrm{T}_{max} R_0}{m_0 V_0^2} \\ \dot{m} = \dfrac{-c\mathrm{T}}{v_e} \end{cases} \tag{2.2}$$

*Remark 1: $v_e$ has been normalized with $V_0$.*

*Remark 2: For conciseness, the quantities in eq.* (2.1) *and their normalized versions in eq.* (2.2) *have been given the same name. From now on, we will always refer to the adimensional variables. If not, it will be mentioned.*

### 2.1.3. Decoupling States and Controls

We define: $\boldsymbol{\tau} = \mathbf{T}/m$, $\tau = \mathrm{T}/m$ and $z = \log(m)$. The new state variable is thus $\mathbf{x} = [\mathbf{r},\ \mathbf{v},\ z]^\top$ (we keep the same name for conciseness and we will refer to this definition from now on) and the control variable is: $\mathbf{u} = [\boldsymbol{\tau},\ \tau]^\top$. The decoupled EoM are:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) = \begin{bmatrix} \mathbf{v} \\ \frac{-\mathbf{r}}{r^3} \\ 0 \end{bmatrix} + \mathbf{B}\mathbf{u} \quad \text{where} \quad \mathbf{B} = \begin{bmatrix} & \mathbf{0_{3\times4}} & & \\ c & 0 & 0 & 0 \\ 0 & c & 0 & 0 \\ 0 & 0 & c & 0 \\ 0 & 0 & 0 & -c/v_e \end{bmatrix} \tag{2.3}$$

## 2.2. Formulation and Convexification

In this section, the OCP is stated and convexified. Then, the convex sub-problem to be solved iteratively [18] is formulated.

### 2.2.1. Original Problem

The low-thrust trajectory optimization problem is formulated as the following OCP:

$$\min_{\mathbf{u}} J_0 \coloneqq -m_f \tag{2.4}$$

Such that:

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) & (2.5a) \\ \tau_x^2 + \tau_y^2 + \tau_z^2 = \tau^2 & (2.5b) \\ 0 \leq \tau \leq e^{-z} & (2.5c) \\ \mathbf{x}(t_0) = \mathbf{x_0}, \quad \mathbf{x}(t_f) = \mathbf{x_f} & (2.5d) \\ \mathbf{x_l} \leq \mathbf{x} \leq \mathbf{x_u}, \quad \mathbf{u_l} \leq \mathbf{u} \leq \mathbf{u_u} & (2.5e) \end{cases}$$

where $m_f$ is the spacecraft's final mass ($J_0$ is called Mayer performance index), $t_0$ and $t_f$ are respectively the initial and final times of the problem, $\mathbf{x_0}$ and $\mathbf{x_f}$ are the boundary conditions to be satisfied and parameters with $(\cdot)_l$ and $(\cdot)_u$ subscripts are respectively lower and upper boundaries. The constraints 2.5b and 2.5c translate respectively the relations $||\mathbf{T}||_2 = \mathrm{T}$ and $0 \leq \mathrm{T} \leq 1$ (normalized) with the decoupling variables $\boldsymbol{\tau}$, $\tau$ and $z$ defined before.

In the frame of this thesis, only fixed final times will be considered, thus the time of flight $t_f$ is constant in all the following.

### 2.2.2. Lossless Convexification

The first convexification step is to relax eq. (2.5b) into a second-order cone constraint by lossless convexification as studied in [11] and done in [13]. The relaxed constraint is:

$$\tau_x^2 + \tau_y^2 + \tau_z^2 \leq \tau^2 \tag{2.6}$$

By doing so, the feasible set is extended but the original equality constraint will be respected once the solution found [12].

### 2.2.3. SCvx Framework

#### Linearization

In the SCvx framework, the dynamics and the nonlinear constraints are linearized around a reference solution $(\mathbf{x}^*, \mathbf{u}^*)$. Let $\mathbf{p}(\mathbf{x})$ be the natural part of the dynamics' function $\mathbf{f}(\mathbf{x}, \mathbf{u})$ :

$$\mathbf{p}(\mathbf{x}) = \begin{bmatrix} \mathbf{v}, & \frac{-\mathbf{r}}{r^3}, & 0 \end{bmatrix}^\top \tag{2.7}$$

The (partially) linearized dynamics is obtained by taking the first order Taylor Series Expansion (TSE) of $\mathbf{p}(\mathbf{x})$ with respect to $\mathbf{x}$:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \simeq \mathbf{p}(\mathbf{x}^*, \mathbf{u}^*) + \mathbf{A}(\mathbf{x}^*)\,(\mathbf{x} - \mathbf{x}^*) + \mathbf{B}\mathbf{u} \tag{2.8}$$

where $\mathbf{A}$ is the Jacobian of $\mathbf{p}$. For the normalized two-body problem:

$$\mathbf{A}(\mathbf{x}) = \frac{\partial \mathbf{p}(\mathbf{x})}{\partial \mathbf{x}} = \begin{bmatrix} \mathbf{0_{3\times3}} & \mathbf{I_3} \\ \frac{3}{r^5}\begin{bmatrix} r_x^2 & r_x r_y & r_x r_z \\ r_y r_x & r_y^2 & r_y r_z \\ r_z r_x & r_z r_y & r_z^2 \end{bmatrix} - \frac{\mathbf{I_3}}{r^3} & \mathbf{0_{3\times3}} \end{bmatrix} \tag{2.9}$$

Furthermore, eq. (2.5c) is linearized to become:

$$0 \leq \tau \leq e^{-z^*}\left(1 - (z - z^*)\right) \tag{2.10}$$

## Trust Region

An additional constraint called "trust-region" can be enforced (depending on the problem) to keep the solution of the current iteration close to the reference one [25, 52]. It is formulated as:

$$||\mathbf{x} - \mathbf{x}^*|| \leq \mathrm{R}_{tr} \tag{2.11}$$

The type of norm and $\mathrm{R}_{tr}$ are set according to the problem.

## Artificial Infeasibility

Within the framework described before, one may encounter additional issues such as *artificial infeasibility* [52]. This phenomenon appears when a feasible non-convex problem is linearized, leading to an unfeasible convex sub-problem. The standard way of solving the issue is to add an unconstrained control variable $\boldsymbol{\nu}$ to the linearized constraint in order to always reach feasibility [14]:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}^*, \mathbf{u}^*) + \mathbf{A}(\mathbf{x}^*)\,(\mathbf{x} - \mathbf{x}^*) + \mathbf{B}\mathbf{u} + \boldsymbol{\nu} \tag{2.12}$$

To make sure that artificial infeasibility is tackled only when strictly necessary, a penalty is added in the cost function. The new convex sub-problem is:

$$\min_{\mathbf{u}} J = J_0 + \mathrm{w}||\boldsymbol{\nu}||_1 \tag{2.13}$$

Such that:

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}^*, \mathbf{u}^*) + \mathbf{A}(\mathbf{x}^*)\,(\mathbf{x} - \mathbf{x}^*) + \mathbf{B}\mathbf{u} + \boldsymbol{\nu} & (2.14\text{a}) \\ \tau_x^2 + \tau_y^2 + \tau_z^2 \leq \tau^2 & (2.14\text{b}) \\ 0 \leq \tau \leq e^{-z^*}\,(1 - (z - z^*)) & (2.14\text{c}) \\ ||\mathbf{x} - \mathbf{x}^*|| \leq \mathrm{R}_{tr} & (2.14\text{d}) \\ \mathbf{x}(t_0) = \mathbf{x_0}, \quad \mathbf{x}(t_f) = \mathbf{x_f} & (2.14\text{e}) \\ \mathbf{x_l} \leq \mathbf{x} \leq \mathbf{x_u}, \quad \mathbf{u_l} \leq \mathbf{u} \leq \mathbf{u_u} & (2.14\text{f}) \end{cases}$$

where w is a sufficiently large penalty weight. Artificial infeasibility could have been tackled also in eq. (2.14c) but it wasn't necessary in our frame. Another issue of the same type exists, called *artificial unboundedness* [52]. It is also linked with the linearization but isn't adressed here.

## 2.3. Summary

Finally we can change the objective function into an integrated control cost to highlight better the dependence from the control variables:

$$J_0 = \int_{t_0}^{t_f} \tau(t)dt \tag{2.15}$$

The two cost functions eq. (2.4) and eq. (2.15) are proved to be equivalent by Wang and Grant in [16]. To sum up, the convexified sub-problem is:

$$\min_{\mathbf{u}} J = J_0 + \mathrm{w}||\boldsymbol{\nu}||_1 \tag{2.16}$$

Such that:

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}^*, \mathbf{u}^*) + \mathbf{A}(\mathbf{x}^*)(\mathbf{x} - \mathbf{x}^*) + \mathbf{B}\mathbf{u} + \boldsymbol{\nu} & \text{(2.17a)} \\ \tau_x^2 + \tau_y^2 + \tau_z^2 \leq \tau^2 & \text{(2.17b)} \\ 0 \leq \tau \leq e^{-z^*}(1 - (z - z^*)) & \text{(2.17c)} \\ ||\mathbf{x} - \mathbf{x}^*|| \leq \mathrm{R}_{tr} & \text{(2.17d)} \\ \mathbf{x}(t_0) = \mathbf{x_0}, \quad \mathbf{x}(t_f) = \mathbf{x_f} & \text{(2.17e)} \\ \mathbf{x_l} \leq \mathbf{x} \leq \mathbf{x_u}, \quad \mathbf{u_l} \leq \mathbf{u} \leq \mathbf{u_u} & \text{(2.17f)} \end{cases}$$

## 2.4. The SCP Algorithm

Various versions of the SCP algorithm have been abundantly used and studied in the literature, thus the reader is invited to refer to chapter 1 to have a wider overview of the topic. This section will briefly describe a simple version of the SCP algorithm that will be used as a comparison for the methods that are developed in this work.

Let $J^{(i)}$, $(\mathbf{x}^{(\mathbf{i})}, \mathbf{u}^{(\mathbf{i})})$ and $\mathrm{R}_{tr,i}$ be, respectively, at iteration $i$, the cost function, the solution of the convex sub-problem 2.16 and the trust-region radius. Let $\delta$ be the tolerance value to check the convergence criterion, which is:

$$\Delta J = |J^{(i)} - J^{(i-1)}| < \delta \tag{2.18}$$

Algorithm 2.1 SCP Algorithm for comparison purposes

```
1: Generate initial guess $(\mathbf{x^{(0)}}, \mathbf{u^{(0)}})$ and initial trust-region radius $\mathrm{R}_{tr,1}$
2: $(\mathbf{x}^*, \mathbf{u}^*) = (\mathbf{x^{(0)}}, \mathbf{u^{(0)}})$
3: $i = 1,\ J^{(0)} = 100,\ J^{(1)} = 10$
4: while $|J^{(i)} - J^{(i-1)}| \geq \delta$ do
5:     Evaluate the required natural dynamics and the Jacobians with $(\mathbf{x}^*, \mathbf{u}^*)$
6:     Solve problem 2.16 for $(\mathbf{x^{(i)}}, \mathbf{u^{(i)}})$
7:     $(\mathbf{x}^*, \mathbf{u}^*) = (\mathbf{x^{(i)}}, \mathbf{u^{(i)}})$
8:     $J^{(i-1)} = J^{(i)}$
9:     $J^{(i)} = J(\mathbf{x^{(i)}}, \mathbf{u^{(i)}})$
10:    $i = i + 1$
11:    Update trust-region radius $\mathrm{R}_{tr,i}$ with the chosen strategy
12: end while
13: The solution is $(\mathbf{x}^*, \mathbf{u}^*)$
```

## 2.5. Initial Guess Generation

When possible, the initial guess is an integrated trajectory with a given control. The advantage of this method is that it is dynamically accurate. However, when several revolutions around the primary body are involved, integrating a prescribed control is inefficient because there are few chances that the final boundary conditions are met (even roughly), making the search of a feasible solution while respecting the trust-region unsuccessful. To solve this issue, first, the cartesian boundary conditions are converted into cylindrical coordinates $\left(r_0, \theta_0, \mathrm{z}_0, \dot{r}_0, \dot{\theta}_0, \dot{\mathrm{z}}_0\right)^\top$ and $\left(r_f, \theta_f, \mathrm{z}_f, \dot{r}_f, \dot{\theta}_f, \dot{\mathrm{z}}_f\right)^\top$. To the final azimuthal coordinate $\theta_f$ is added the desired number of revolutions $N_{rev}$. From [57] (appendix C), a cubic polynomial function can be used to approximate each state:

$$f(t) = at^3 + bt^2 + ct + d \tag{2.19}$$

The boundary conditions $\left(r_0, \theta_0, \mathrm{z}_0, \dot{r}_0, \dot{\theta}_0, \dot{\mathrm{z}}_0\right)^\top$ and $\left(r_f, \theta_f + 2\pi N_{rev}, \mathrm{z}_f, \dot{r}_f, \dot{\theta}_f, \dot{\mathrm{z}}_f\right)^\top$ are then used to get $(a, b, c, d)$ for each state (see [57]), and thus the full polynomial approximation for all time instants. Converting back into cartesian coordinates, we have our multi-revolution initial guess that is now respecting exactly the boundary conditions. However, the guess is dynamically inaccurate and thus poor in terms of quality with respect to other methods as the Finite Fourier Series also studied in [57]. It is a good opportunity to test the robustness of our algorithm though.

# 3 | Convexified Taylor Series Expansion Method

The main theoretical development and novelties of this work are explained through this chapter, where two new methods to include non-linear dynamics are developed.

## 3.1. Introduction

This section gathers useful definitions and builds the basis of the theoretical development for the following sections. The definition of an inner-convex approximation is recalled, and a convexification method for a multi-variate TSE of a scalar function is presented.

Let $f : \mathcal{E} \longrightarrow \mathbb{R}$ be a function of a variable $x = (x_1, x_2, x_3, \ldots, x_d)^\top$ with $\mathcal{E} \subset \mathbb{R}^d$ ($d$ represents the length of the states vector. It is arbitrary for now but will be equal to 7 in practice). Let $x^* \in \mathcal{E}$, $x^* = (x_1^*, x_2^*, x_3^*, \ldots, x_d^*)^\top$ be a reference point and $\delta x = x - x^*$.

### 3.1.1. Inner-Convex Approximation

In this subsection, a definition of an Inner-Convex Approximation (ICA) is given. An ICA of $f$ around $x^*$ is a function $f_{ica}^{(x^*)}$ verifying [55]:

$$\begin{cases} f_{ica}^{(x^*)} \text{ is convex} & (3.1a) \\ \forall x \in \mathcal{E},\ f_{ica}^{(x^*)}(x) \geq f(x) & (3.1b) \\ f_{ica}^{(x^*)}(x^*) = f(x^*) & (3.1c) \\ \nabla f_{ica}^{(x^*)}(x^*) = \nabla f(x^*) & (3.1d) \end{cases}$$

### 3.1.2. Convexification

The TSE of $f$ up to order 2 (see eq. (B.4) in appendix B) yields:

$$f(x) \simeq f(x^*) + \delta x^\top \nabla f(x^*) + \delta x^\top \mathbf{H}(x^*) \delta x \tag{3.2}$$

For eq. (3.2) to be convex, one has to convexify the Hessian term $\delta x^\top \mathbf{H}(x^*)\delta x$. Indeed, the linear part is convex and a sum of two convex functions is also convex. Hence, the second-order TSE's convexity depends only on $\delta x^\top \mathbf{H}\delta x$ (which is a quadratic form that is convex if and only if $\mathbf{H}$ is positive semi-definite [9, 55]). In our case, we must modify this term to convexify it. Hence, we write [58]:

$$\mathbf{H} = \mathbf{H}^+ + \mathbf{H}^- \tag{3.3}$$

where $\mathbf{H}^+$ and $\mathbf{H}^-$ are respectively positive semi-definite and negative semi-definite. With $\mathbf{V}$ the eigenvectors matrix of $\mathbf{H}$ and $\mathbf{\Lambda}^+$ the eigenvalues matrix containing only the positive ones, $\mathbf{H}^+$ yields [55]:

$$\mathbf{H}^+ = \mathbf{V}\mathbf{\Lambda}^+\mathbf{V}^\top \tag{3.4}$$

The convex version of eq. (3.2) (and an ICA of $f$) is:

$$f(x) \simeq f(x^*) + \delta x^\top \nabla f(x^*) + \delta x^\top \mathbf{H}^+(x^*)\delta x = f_{ica}^{(x^*)}(x) \tag{3.5}$$

To convexify higher-order terms, we can follow the same procedure with the tensors $\mathbf{T}_{f,m}$ (see eq. (B.2) for a definition of $\mathbf{T}_{f,m}$). But according to [55], the computational cost is high and an other method, towards which the reader is referred, is proposed to avoid this burden.

## 3.2. Formulation

Usually, as previously said, the dynamics is convexified by linearizing the terms with respect to a reference solution (see eq. (2.8)). However, this is an inaccurate way of approximating a function, making the algorithm's reliability drop fast when highly deviating from the reference. The goal of this section is to introduce new methods of approximating and including the dynamics in a CO algorithm, using ICAs.

### 3.2.1. First Approach

Ideally, our purpose is to solve the OCP 2.16, imposing eq. (3.5) instead of eq. (2.17a) for the nonlinear terms. In eq. (3.6), the EoM are recalled.

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x},\mathbf{u}) = \begin{bmatrix} \mathbf{v} \\ \frac{-\mathbf{r}}{r^3} \\ 0 \end{bmatrix} + \mathbf{Bu} = \begin{bmatrix} \mathbf{v} \\ \mathbf{p_r}(\mathbf{x}) \\ 0 \end{bmatrix} + \mathbf{Bu} \tag{3.6}$$

And $\mathbf{p_r}(\mathbf{x})$ is defined as:

$$\mathbf{p_r}(\mathbf{x}) = \begin{bmatrix} -r_x/r^3 \\ -r_y/r^3 \\ -r_z/r^3 \end{bmatrix} = \begin{bmatrix} p_{rx} \\ p_{ry} \\ p_{rz} \end{bmatrix} \tag{3.7}$$

Only the terms involving $\mathbf{p_r}(\mathbf{x})$ are nonlinear. They are approximated by their convexified TSE up to order 2 (for sake of clarity). We would like to impose for the dynamics:

$$\begin{cases} \dot{\mathbf{r}} = \mathbf{v} & (3.8a) \\ \dot{v}_x = p_{rx}(\mathbf{x}^*) + \delta\mathbf{x}^\top \nabla p_{rx}(\mathbf{x}^*) + \delta\mathbf{x}^\top \mathbf{H}^+_{rx}(\mathbf{x}^*)\delta\mathbf{x} + c\tau_x & (3.8b) \\ \dot{v}_y = p_{ry}(\mathbf{x}^*) + \delta\mathbf{x}^\top \nabla p_{ry}(\mathbf{x}^*) + \delta\mathbf{x}^\top \mathbf{H}^+_{ry}(\mathbf{x}^*)\delta\mathbf{x} + c\tau_y & (3.8c) \\ \dot{v}_z = p_{rz}(\mathbf{x}^*) + \delta\mathbf{x}^\top \nabla p_{rz}(\mathbf{x}^*) + \delta\mathbf{x}^\top \mathbf{H}^+_{rz}(\mathbf{x}^*)\delta\mathbf{x} + c\tau_z & (3.8d) \\ \dot{z} = -c\tau/v_e & (3.8e) \end{cases}$$

where $\nabla p_{rx}(\mathbf{x}^*)$ and $\mathbf{H}^+_{rx}(\mathbf{x}^*)$ are the gradient and the convexified Hessian of $p_{rx}$. Similar notations are used for $p_{ry}$ and $p_{rz}$. However, according to the *DCP* rules [49], in CO, each side of an equality constraint shall be affine. It is not the case for eq. (3.8b), eq. (3.8c) and eq. (3.8d) because the right-hand side is convex. Thus, this approach is not valid, we have to adapt it.

### 3.2.2. Second Approach

If we can not impose directly the equality constraints, we can try to include them into the cost function by strongly penalizing when the latter isn't respected [59]. The quantity we would like to add in the cost (with $\lambda_d$ a sufficiently large penalty weight) is:

$$\lambda_d|| -\dot{v}_k + p_{rk}(\mathbf{x}^*) + \delta\mathbf{x}^\top \nabla p_{rk}(\mathbf{x}^*) + \delta\mathbf{x}^\top \mathbf{H}^+_{rk}(\mathbf{x}^*)\delta\mathbf{x} + c\tau_k|| \quad \text{with} \quad k \in \{x,y,z\}$$

Again, this is not feasible because *DCP* rules set prevents the use of a convex expression inside a norm. Hence we can transform the cost into:

$$J = J_0 + \lambda_d \left(\Delta_x + \Delta_y + \Delta_z\right) \tag{3.9}$$

where:

$$\begin{aligned}
\Delta_x &:= -\dot{v}_x + p_{rx}(\mathbf{x}^*) + \delta\mathbf{x}^\top \nabla p_{rx}(\mathbf{x}^*) + \delta\mathbf{x}^\top \mathbf{H}^+_{rx}(\mathbf{x}^*)\delta\mathbf{x} + c\tau_x \\
\Delta_y &:= -\dot{v}_y + p_{ry}(\mathbf{x}^*) + \delta\mathbf{x}^\top \nabla p_{ry}(\mathbf{x}^*) + \delta\mathbf{x}^\top \mathbf{H}^+_{ry}(\mathbf{x}^*)\delta\mathbf{x} + c\tau_y \\
\Delta_z &:= -\dot{v}_z + p_{rz}(\mathbf{x}^*) + \delta\mathbf{x}^\top \nabla p_{rz}(\mathbf{x}^*) + \delta\mathbf{x}^\top \mathbf{H}^+_{rz}(\mathbf{x}^*)\delta\mathbf{x} + c\tau_z
\end{aligned}$$

are the dynamical defects relative to each coordinate $x, y, z$. Only the linear dynamics is remaining as a constraint:

$$\begin{cases} \dot{\mathbf{r}} = \mathbf{v} & (3.10\text{a}) \\ \dot{z} = -c\tau/v_e & (3.10\text{b}) \end{cases}$$

This formulation satisfies the *DCP* rules but, during simulation, the term $\lambda_d \left(\Delta_x + \Delta_y + \Delta_z\right)$ takes negative values (to minimize $J$) which doesn't match our aim because we want the latter to be zero.

## 3.2.3. Final Approach

From the last paragraph, we want to find a way to prevent the dynamical defects to take highly negative values. First, we know that the convexified Hessian $\mathbf{H}^+(\mathbf{x}^*)$ is positive semi-definite, hence:

$$\forall \mathbf{v} \in \mathbb{R}^7,\ \mathbf{v}^\top \mathbf{H}^+(\mathbf{x}^*)\mathbf{v} \geq 0 \tag{3.11}$$

Thus, by definition of the defects $\Delta$:

$$\forall k \in \{x, y, z\},\quad \Delta_k \leq 0 \implies -\dot{v}_k + p_{rk}(\mathbf{x}^*) + \delta\mathbf{x}^\top \nabla p_{rk}(\mathbf{x}^*) + c\tau_k \leq 0 \tag{3.12}$$

In other words, if the defects are negative, the linear part is necessarily negative too. Hence, if we prevent the linear part to be negative, our problem is solved. However, imposing:

$$\forall k \in \{x, y, z\},\quad -\dot{v}_k + p_{rk}(\mathbf{x}^*) + \delta\mathbf{x}^\top \nabla p_{rk}(\mathbf{x}^*) + c\tau_k \geq 0$$

is not a good solution because in that case, due to eq. (3.11), we would miss the chance to obtain:

$$-\dot{v}_k + p_{rk}(\mathbf{x}^*) + \delta\mathbf{x}^\top \nabla p_{rk}(\mathbf{x}^*) + c\tau_k + \delta\mathbf{x}^\top \mathbf{H}^+_{rk}(\mathbf{x}^*)\delta\mathbf{x} = 0$$

We could impose :

$$-\dot{v}_k + p_{rk}(\mathbf{x}^*) + \delta\mathbf{x}^\top \nabla p_{rk}(\mathbf{x}^*) + c\tau_k + \delta\mathbf{x}^\top \mathbf{H}^+_{rk}(\mathbf{x}^*)\delta\mathbf{x} \geq 0 \tag{3.13}$$

But a constraint involving a convex expression to be greater or equal to zero is forbidden by *DCP* rules. Let $X_{i-1}$ be the quantity $X$ at the iteration $i-1$. We impose finally, with $\mathbf{S}$ an unconstrained slack variable:

$$-\dot{v}_k + p_{rk}(\mathbf{x}^*) + \delta\mathbf{x}^\top \nabla p_{rk}(\mathbf{x}^*) + c\tau_k + \delta\mathbf{x_{i-1}}^\top \mathbf{H}^+_{rk}(\mathbf{x_{i-1}}^*)\delta\mathbf{x_{i-1}} + \mathbf{S_k} \geq 0 \tag{3.14}$$

Where $\delta\mathbf{x_{i-1}}^\top \mathbf{H}^+_{rk}(\mathbf{x_{i-1}}^*)\delta\mathbf{x_{i-1}}$ is a constant computed with the previous iteration to prevent the situation explained before. This formulation allows the linear part to be slightly negative and compensated with the constant Hessian term. We expect that, along the iterations, this constant term gets closer and closer to the actual one. The slack variable prevents the linear part to take highly negative values by including a penalty into the cost function ($\lambda_s$ is the slack variable penalty weight):

$$J = J_0 + \lambda_d\left(\Delta_x + \Delta_y + \Delta_z\right) + \lambda_s\left(\max\left(\mathbf{S_x}, 0\right) + \max\left(\mathbf{S_y}, 0\right) + \max\left(\mathbf{S_z}, 0\right)\right) \tag{3.15}$$

*Remark: there is no need to tackle artificial infeasibility directly here because the non-linear dynamics' approximation is now included in the cost and isn't a constraint anymore. The feasibility of an iteration will thus never be altered by a poor dynamics' approximation imposed as a constraint. However, the aforementioned method is directly inspired of the one used to tackle artificial infeasibility, especially the use of a slack variable (see eq.* (2.12)*).*

### 3.2.4. Summary

The final formulation, which we will refer to as Convexified Taylor Series Expansion (CTSE) method is summed up here:

$$\min_{\mathbf{u}} J = J_0 + \lambda_d\left(\Delta_x + \Delta_y + \Delta_z\right) + \lambda_s\left(\max\left(\mathbf{S_x}, 0\right) + \max\left(\mathbf{S_y}, 0\right) + \max\left(\mathbf{S_z}, 0\right)\right) \tag{3.16}$$

Such that (with $k \in \{x, y, z\}$):

$$
\begin{cases}
\dot{\mathbf{r}} = \mathbf{v} & (3.17a) \\
\dot{z} = -c\tau/v_e & (3.17b) \\
-\dot{v}_k + p_{rk}(\mathbf{x}^*) + \delta\mathbf{x}^\top \nabla p_{rk}(\mathbf{x}^*) + c\tau_k + \delta\mathbf{x_{i-1}}^\top \mathbf{H}_{rk}^+(\mathbf{x_{i-1}}^*)\delta\mathbf{x_{i-1}} + \mathbf{S_k} \geq 0 & (3.17c) \\
\tau_x^2 + \tau_y^2 + \tau_z^2 \leq \tau^2 & (3.17d) \\
0 \leq \tau \leq e^{-z^*}\left(1 - (z - z^*)\right) & (3.17e) \\
||\mathbf{x} - \mathbf{x}^*|| \leq \mathrm{R}_{tr} & (3.17f) \\
\mathbf{x}(t_0) = \mathbf{x_0}, \quad \mathbf{x}(t_f) = \mathbf{x_f} & (3.17g) \\
\mathbf{x_l} \leq \mathbf{x} \leq \mathbf{x_u}, \quad \mathbf{u_l} \leq \mathbf{u} \leq \mathbf{u_u} & (3.17h)
\end{cases}
$$

## 3.3. Alternative Method

In the CTSE method, the formulation is quite complex and causes some numerical issues (cf next chapter). Hence, another method has been developed, whose purpose is to be simpler and to improve some features of the CTSE. The new method, that we call Alternative CTSE (ACTSE), still relies on ICA to model the spacecraft's dynamics.

### 3.3.1. Main Idea

We define squared defects for the nonlinear dynamical constraints with $\mathbf{s} = [\mathbf{x}, \mathbf{u}]^\top$:

$$
\begin{cases}
D_x(\mathbf{s}) := \left(\dot{v}_x - p_{rx}(\mathbf{x}) - c\tau_x\right)^2 \\
D_y(\mathbf{s}) := \left(\dot{v}_y - p_{ry}(\mathbf{x}) - c\tau_y\right)^2 \\
D_z(\mathbf{s}) := \left(\dot{v}_z - p_{rz}(\mathbf{x}) - c\tau_z\right)^2
\end{cases}
\tag{3.18}
$$

By definition:

$$
\forall k \in \{x, y, z\}, \ \forall \mathbf{s} \in \mathbb{R}^{11} \setminus \mathbf{0}_{11\times 1}, \ D_k(\mathbf{s}) \geq 0 \tag{3.19}
$$

Their convexified TSE up to order 2 (for conciseness) are:

$$
\begin{cases}
D_x^{ica}(\mathbf{s}, \mathbf{s}^*) = D_x(\mathbf{s}^*) + \delta\mathbf{s}^\top \nabla D_x(\mathbf{s}^*) + \delta\mathbf{s}^\top \mathbf{H}_x^+(\mathbf{s}^*)\delta\mathbf{s} \\
D_y^{ica}(\mathbf{s}, \mathbf{s}^*) = D_y(\mathbf{s}^*) + \delta\mathbf{s}^\top \nabla D_y(\mathbf{s}^*) + \delta\mathbf{s}^\top \mathbf{H}_y^+(\mathbf{s}^*)\delta\mathbf{s} \\
D_z^{ica}(\mathbf{s}, \mathbf{s}^*) = D_z(\mathbf{s}^*) + \delta\mathbf{s}^\top \nabla D_z(\mathbf{s}^*) + \delta\mathbf{s}^\top \mathbf{H}_z^+(\mathbf{s}^*)\delta\mathbf{s}
\end{cases}
\tag{3.20}
$$

The previous approximation is an ICA of the defects $D_k$. By definition of an ICA (eq. (3.1b)) and eq. (3.19), we have:

$$\forall k \in \{x, y, z\},\ \forall \mathbf{s} \in \mathbb{R}^{11} \setminus \mathbf{0}_{11\times 1},\ \forall \mathbf{s}^* \in \mathbb{R}^{11} \setminus \mathbf{0}_{11\times 1},\ D_k^{ica}(\mathbf{s}, \mathbf{s}^*) \geq D_k(\mathbf{s}) \geq 0 \tag{3.21}$$

From now on, we will refer to $D_k^{ica}(\mathbf{s}, \mathbf{s}^*)$ by $D_k(\mathbf{s})$ for clarity.

With the assertion eq. (3.21), we know that the convexified defects are always positive, meaning that we can include them directly in the objective function without having to tackle negative values during minimization.

### 3.3.2. Formulation

The ACTSE method is formulated as:

$$\min_{\mathbf{s}} J = J_0 + \lambda_d \left(D_x + D_y + D_z\right) \tag{3.22}$$

Such that:

$$\begin{cases} \dot{\mathbf{r}} = \mathbf{v} & (3.23\text{a}) \\ \dot{z} = -c\tau/v_e & (3.23\text{b}) \\ \tau_x^2 + \tau_y^2 + \tau_z^2 \leq \tau^2 & (3.23\text{c}) \\ 0 \leq \tau \leq e^{-z^*}\left(1 - (z - z^*)\right) & (3.23\text{d}) \\ ||\mathbf{x} - \mathbf{x}^*|| \leq \mathrm{R}_{tr} & (3.23\text{e}) \\ \mathbf{x}(t_0) = \mathbf{x_0}, \quad \mathbf{x}(t_f) = \mathbf{x_f} & (3.23\text{f}) \\ \mathbf{x_l} \leq \mathbf{x} \leq \mathbf{x_u}, \quad \mathbf{u_l} \leq \mathbf{u} \leq \mathbf{u_u} & (3.23\text{g}) \end{cases}$$

This problem is now simpler with respect to CTSE because we removed 3 constraints (eq. (3.17c)) and we don't use the slack variable. Furthermore, there is no need to use the previous iterations' information. The problem is also less constrained than the classical SCvx formulation eq. (2.16) because only the linear dynamics is imposed. Again there is no need to tackle artificial infeasibility for the same reason explained in subsection 3.2.4.

## 3.4. A Few Remarks

The theory presented here has been developed by trial and errors using numerical tools. Once an idea to improve the ongoing formulation was found, it was implemented, tested, and its performances were assessed with respect to previous versions of the methods.

Several ideas that are presented in the previous pages are combinations and adaptations of pieces of work found in the literature (mainly [11, 55, 59]). Effort was put in this thesis to stay close to the current science that is developed nowadays in the field of Convex Optimization.

We acknowledge that theoretical developments sustained by rigorous mathematical proofs are lacking in this work but the choice was made, in agreement with the supervising team, to seek for numerical evidences first.

To the best of the author's knowledge, those methods are original and weren't mentioned in the literature.

## 3.5. Algorithms

The algorithms' principles are explained in this section. As in algorithm 2.1, they start by generating an initial guess and the initial values of the required parameters (cost functions, number of iteration, trust-region radius, etc). While the convergence condition isn't met, the convex sub-problem is solved and the reference solution is updated as well as the trust-region radius. Note that in CTSE, one SCP iteration has to be done before because information about two trajectories (here the initial guess and the first SCvx solution) are required to run a CTSE iteration (because of eq. (3.14)). It is not necessary with ACTSE. The initial guesses are generated like explained in section 2.5.

Algorithm 3.1 CTSE algorithm

```
1: Generate initial guess $(\mathbf{x^{(0)}}, \mathbf{u^{(0)}})$ and initial trust-region radius $\mathrm{R}_{tr,1}$
2: $(\mathbf{x}^*, \mathbf{u}^*) = (\mathbf{x^{(0)}}, \mathbf{u^{(0)}})$
3: $J^{(0)} = 100,\ J^{(1)} = 10$
4: Solve problem 2.16 for $(\mathbf{x^{(1)}}, \mathbf{u^{(1)}})$
5: $(\mathbf{x}^*, \mathbf{u}^*) = (\mathbf{x^{(1)}}, \mathbf{u^{(1)}})$
6: $i = 1$
7: Update trust-region radius $\mathrm{R}_{tr,2}$ with the chosen strategy
8: while $|J^{(i)} - J^{(i-1)}| \geq \delta$ do
9:     Evaluate the required natural dynamics, gradients and Hessians with $(\mathbf{x}^*, \mathbf{u}^*)$
10:    Solve problem 3.16 for $(\mathbf{x^{(i)}}, \mathbf{u^{(i)}})$
11:    $(\mathbf{x^{(i-1)}}, \mathbf{u^{(i-1)}}) = (\mathbf{x}^*, \mathbf{u}^*)$
12:    $(\mathbf{x}^*, \mathbf{u}^*) = (\mathbf{x^{(i)}}, \mathbf{u^{(i)}})$
13:    $J^{(i-1)} = J^{(i)}$
14:    $J^{(i)} = J(\mathbf{x^{(i)}}, \mathbf{u^{(i)}})$
15:    $i = i + 1$
16:    Update trust-region radius $\mathrm{R}_{tr,i}$
17: end while
18: The solution is $(\mathbf{x}^*, \mathbf{u}^*)$
```

Algorithm 3.2 ACTSE algorithm

```
1: Generate initial guess $(\mathbf{x^{(0)}}, \mathbf{u^{(0)}})$ and initial trust-region radius $\mathrm{R}_{tr,1}$
2: $(\mathbf{x}^*, \mathbf{u}^*) = (\mathbf{x^{(0)}}, \mathbf{u^{(0)}})$
3: $i = 1,\ J^{(0)} = 100,\ J^{(1)} = 10$
4: while $|J^{(i)} - J^{(i-1)}| \geq \delta$ do
5:     Evaluate the required natural dynamics, gradients and Hessians with $(\mathbf{x}^*, \mathbf{u}^*)$
6:     Solve problem 3.22 for $(\mathbf{x^{(i)}}, \mathbf{u^{(i)}})$
7:     $(\mathbf{x}^*, \mathbf{u}^*) = (\mathbf{x^{(i)}}, \mathbf{u^{(i)}})$
8:     $J^{(i-1)} = J^{(i)}$
9:     $J^{(i)} = J(\mathbf{x^{(i)}}, \mathbf{u^{(i)}})$
10:    $i = i + 1$
11:    Update trust-region radius $\mathrm{R}_{tr,i}$ with the chosen strategy
12: end while
13: The solution is $(\mathbf{x}^*, \mathbf{u}^*)$
```

# 4 | Simulations and Results

In this chapter, numerical simulations will be performed to assess the performances of the previous part's theoretical development.

## 4.1. Discretization Techniques

First, the two discretization methods used in this chapter will be described. The goal is to discretize the integral equation eq. (4.1) and to transform differential constraints into algebraic ones. By doing so, the OCP becomes a finite-parameter optimization problem.

$$\int_{t_k}^{t_{k+1}} \dot{\mathbf{x}}(t)dt = \int_{t_k}^{t_{k+1}} \mathbf{f}(\mathbf{x}, \mathbf{u})dt \tag{4.1}$$

In our framework, the time segment $[0, t_f]$ (with $t_f$ the time of flight) on which the trajectory must be performed is divided into $N-1$ intervals $[t_k, t_{k+1}]$, where $k \in \{1, \cdots, N-1\}$. $N$ is the number of discretization points, called nodes. In other words, nodes are the boundaries of the $N-1$ time intervals. A different type of points will be mentioned in this work: the collocation points. They represent the time instants where the dynamical constraints are enforced, meaning where the dynamics (approximated or not) will be respected in the obtained solution. Nodes and collocation points can coincide in some discretization techniques but it is not a generality.

Choosing the right discretization method for our purpose is important because it has a direct impact on the accuracy and the speed of the algorithm. In chapter 1, some of them were described but two will be particularly under focus: the trapezoidal discretization and the Hermite-Simpson scheme.

For the following subsections, we define $X_k = X(t_k)$ the quantity $X$ evaluated at the instant $t_k$, and $h = t_{k+1} - t_k$ the length of the time sub-intervals. The left-hand side of eq. (4.1) yields:

$$\int_{t_k}^{t_{k+1}} \dot{\mathbf{x}}(t)dt = \mathbf{x}_{k+1} - \mathbf{x}_k \tag{4.2}$$

Hence, the discretization techniques are applied to the right-hand side.

### 4.1.1. Trapezoidal Discretization

The trapezoidal rule approximates the integral by the area of a trapezoid. It yields for the right-hand side of eq. (4.1):

$$\int_{t_k}^{t_{k+1}} \mathbf{f}(\mathbf{x}, \mathbf{u}) dt \simeq \frac{h}{2} \left( \mathbf{f}(\mathbf{x}_{k+1}, \mathbf{u}_{k+1}) + \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) \right) \tag{4.3}$$

The corresponding constraint as used by Wang in [42] is:

$$\mathbf{x}_{k+1} - \mathbf{x}_k = \frac{h}{2} \left( \mathbf{f}(\mathbf{x}_{k+1}, \mathbf{u}_{k+1}) + \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) \right) \tag{4.4}$$

In this method, the nodes and the collocation points coincide.

### 4.1.2. Hermite-Simpson Scheme

The second one is the Hermite-Simpson scheme (see [24] for details). Let $t_c = (t_k + t_{k+1})/2$ be the center of the time interval. The integral is approximated using Simpson's rule [30], yielding for the right-hand side of eq. (4.1):

$$\int_{t_k}^{t_{k+1}} \mathbf{f}(\mathbf{x}, \mathbf{u}) dt \simeq \frac{h}{6} \left( \mathbf{f}(\mathbf{x}_{k+1}, \mathbf{u}_{k+1}) + 4\mathbf{f}(\mathbf{x}_c, \mathbf{u}_c) + \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) \right) \tag{4.5}$$

where $(\mathbf{x}_c, \mathbf{u}_c) = (\mathbf{x}(t_c), \mathbf{u}(t_c))$. A cubic and a linear interpolations are respectively used to evaluate $\mathbf{x}_c$ and $\mathbf{u}_c$. The details of the derivation are available in [24]. We have:

$$\mathbf{x}_c = \frac{1}{2} \left( \mathbf{x}_k + \mathbf{x}_{k+1} \right) + \frac{h}{8} \left( \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) - \mathbf{f}(\mathbf{x}_{k+1}, \mathbf{u}_{k+1}) \right) \tag{4.6a}$$

$$\mathbf{u}_c = \frac{1}{2} \left( \mathbf{u}_{k+1} + \mathbf{u}_k \right) \tag{4.6b}$$

The dynamical constraint to impose is finally:

$$\mathbf{x}_{k+1} - \mathbf{x}_k = \frac{h}{6} \left( \mathbf{f}(\mathbf{x}_{k+1}, \mathbf{u}_{k+1}) + 4\mathbf{f}(\mathbf{x}_c, \mathbf{u}_c) + \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) \right) \tag{4.7}$$

In that case, the collocation points are located on the nodes and at the center of the time intervals. Hence, they don't coincide.

## 4.2. Earth-Mars Transfer

In this section, a simple Earth-Mars Transfer (EMT) will be solved using the previous methods to check if they are valid. The reference data are found in [16].

### 4.2.1. Parameters

The trust-region constraint is, with $i > 1$ the current iteration ($\mathbf{x}_{i-1} = \mathbf{x}^*$):

$$||\mathbf{x}_i - \mathbf{x}_{i-1}||_2 \leq \gamma ||\mathbf{x}_{i-1} - \mathbf{x}_{i-2}||_2 \tag{4.8}$$

The initial guess is an integrated constant control $\mathbf{u}_{guess} = [0,\ 0.6,\ 0.4,\ \sqrt{0.52}]^\top$ (dynamically accurate) and the algorithms' numerical parameters are available in table 4.1.

Table 4.1: Numerical parameters for the Earth-Mars transfer

| **Parameter** | **Value** |
|---|---|
| Time of flight $t_f$ | 253 $days$ |
| Initial position $\mathbf{r}_0$ | $[1,\ 0,\ 0]^\top$ |
| Initial velocity $\mathbf{v}_0$ | $[0,\ 1,\ 0]^\top$ |
| Initial mass $m_0$ | $659.3kg$ |
| Final position $\mathbf{r}_f$ | $[-1.5229,\ 0,\ 0.0492]^\top$ |
| Final velocity $\mathbf{v}_f$ | $[0,\ -0.8101,\ 0]^\top$ |
| Final mass $m_f$ | free |
| Maximum thrust $\mathrm{T}_{max}$ | $0.55N$ |
| Specific impulse $I_{sp}$ | $3300s$ |
| Number of nodes $N$ | 100 |
| w (artificial infeasibility) | $10^2$ |
| $\lambda_d$, $\lambda_s$ (see eq. (3.15)) | $10^2$, $10^4$ |
| Convergence criterion $\delta$ | $10^{-4}$ |
| Initial trust region radius $\mathrm{R}_{tr,0}$ | 1 |

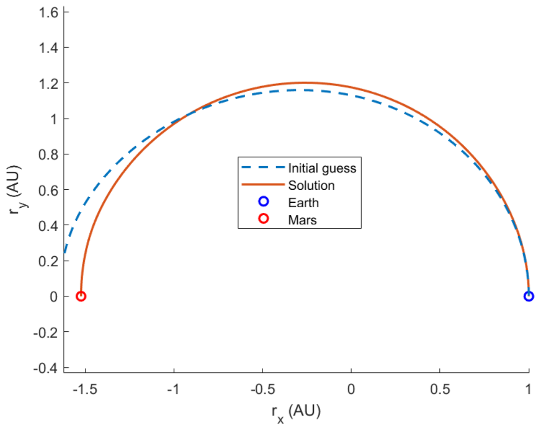
### 4.2.2. Reference Solution

In this subsection, we compute a reference solution with the SCP method described in algorithm 2.1, trapezoidal discretization and $\gamma = 0.9$ (see eq. (4.8)) to validate the new methods. The outcomes are:

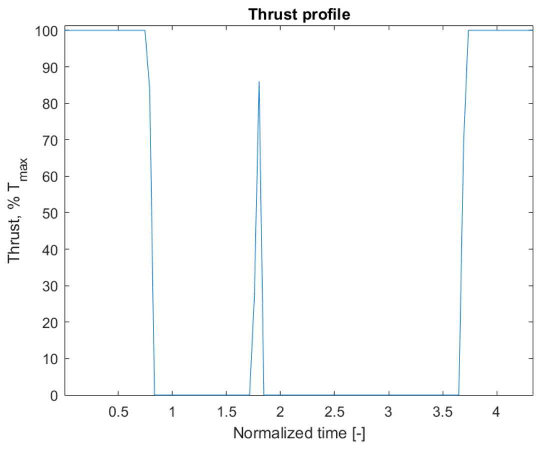
Table 4.2: Results for the EMT (classic SCP, trapezoidal discretization)

| Final mass [kg] | Iterations | Running time [s] |
|---|---|---|
| 530.59 | 4 | 4.84 |

The algorithm needs 4 iterations to converge and 4.84$s$ to operate. The solution, represented on fig. 4.1, is similar to the one of [16], validating our simple algorithm 2.1. Note that the thrust magnitude is expressed as a percentage of T$_{max}$ and that the time is normalized according to table 2.1 (every figure will use those units).



(a) Trajectory seen from above



(b) Thrust profile

Figure 4.1: Earth-Mars transfer, Classic SCP, trapezoidal discretization

## 4.2.3. New Methods Testing

Everything is now ready to test the performances of the CTSE and ACTSE methods for this specific problem. Here $\gamma = 0.8$ (eq. (4.8)). The outcomes of the algorithms 3.1 and 3.2 are gathered in appendix A. Only the necessary ones for our discussion will appear in the next pages. Please note that in those algorithms, only the convexified quadratic term has been included in the dynamics enforcement (see eq. (3.5), eq. (3.8) and eq. (3.20)). A dedicated section on high-order terms inclusion will follow. The goal of this subsection is to obtain numerical evidences that the two methods developed earlier are valid, the study of their accuracy and speed are kept for later.

## CTSE, Trapezoidal Discretization

First, we test the CTSE method with trapezoidal discretization, same initial guess and trust-region strategy as the reference solution (that we will call 'Classic SCP' in the following). The results are:

Table 4.3: Results for the EMT (CTSE, trapezoidal discretization)

| Final mass [kg] | Iterations | Running time [s] |
|---|---|---|
| 530.54 | 8 | 138.76 |

And the thrust profile is:

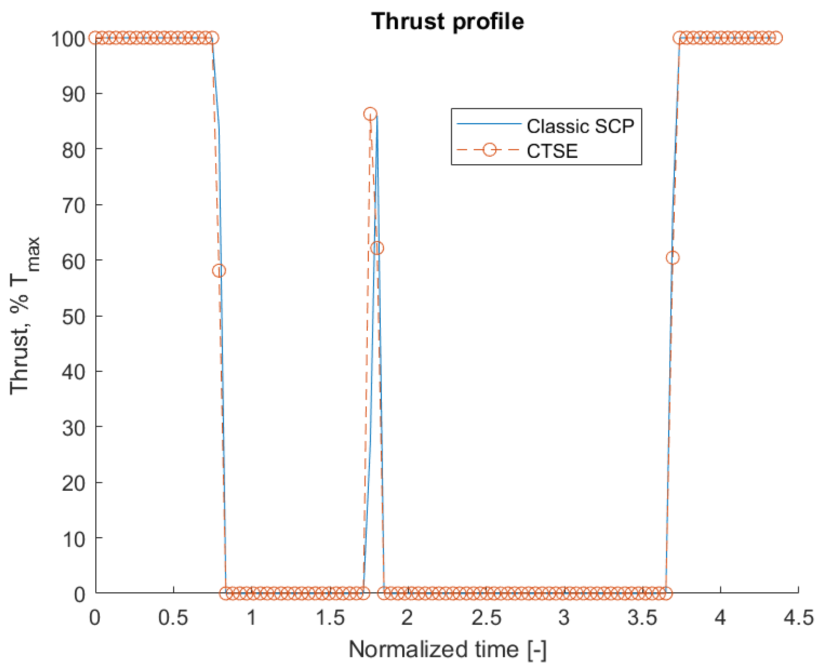


Figure 4.2: Thrust profile, EMT, CTSE, trapezoidal discretization

On the figure above, we notice that the thrust profiles of the Classic SCP (fig. 4.1b) and of the CTSE are matching, giving a first proof of validity for the latter. The final masses are also similar. However, the number of iterations is doubled with respect to the previous case and the running time is multiplied by a factor 29. Those issues will be addressed later.

The trajectory's plot isn't given here but is available on fig. A.2. In all the following, the trajectories will be provided only if needed for the current discussion. If not, they are available in appendix A for any simulation.

## CTSE, Hermite-Simpson Scheme

In this paragraph, we follow the same procedure but changing the type of discretization. We use now a Hermite-Simpson scheme (eq. (4.7)). The results are:

Table 4.4: Results for the EMT (CTSE, Hermite-Simpson scheme)

| **Final mass [kg]** | **Iterations** | **Running time [s]** |
|---|---|---|
| 530.49 | 8 | 221.91 |

The running time is longer with respect to the previous simulation. It is understandable by considering the amount of supplementary information (Gradient, Hessian, convexification, etc) needed for this scheme due to the additional collocation point with respect to the trapezoidal discretization. The thrust profile is:

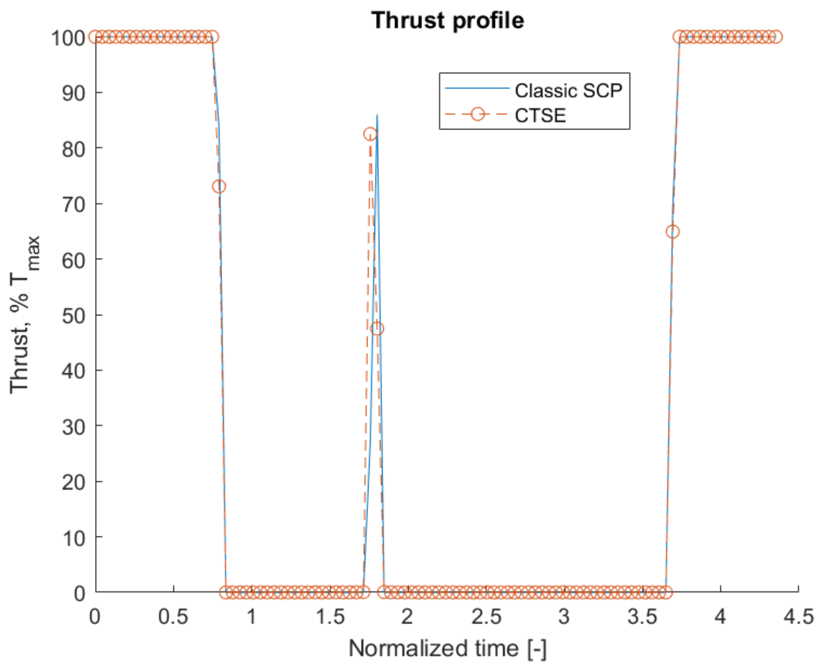


Figure 4.3: Thrust profile, EMT, CTSE, Hermite-Simpson scheme

Again, the thrust profiles and the final masses are matching with the classic SCP, validating our algorithm with this other discretization type.

## ACTSE, Trapezoidal Discretization

We run now the algorithm employing ACTSE to observe its behavior. In that case only, $\lambda_d = 10000$. The other parameters are the same as table 4.1. The algorithm yields :

Table 4.5: Results for the EMT (ACTSE, trapezoidal discretization)

| Final mass [kg] | Iterations | Running time [s] |
|---|---|---|
| 547.96 | 13 | 108.29 |

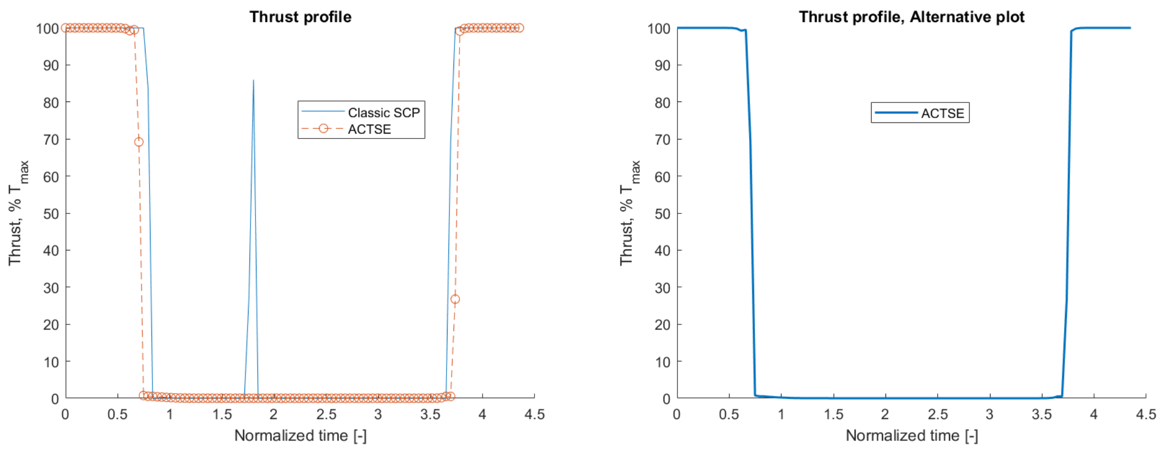


Figure 4.4: Thrust profiles, EMT, ACTSE

The outcomes are surprising (and thus interesting). First, the final mass is greater than the previous ones, granting a better solution in theory (even if the accuracy of the method is poor, as discussed later) because our final aim is to minimize the fuel spent and thus, equivalently, maximize the final mass. Then, the ACTSE needs more iterations to converge but is time-wise faster than CTSE for this problem, which is consistent because it contains less constraints in its formulation (see eq. (3.16) and eq. (3.22)). Finally, in theory, solving a minimum fuel problem such as eq. (2.16) should yields for the thrust magnitude a "bang-off-bang" profile as proved in [60]. It means that the thrust magnitude should be either 100% of T$_{max}$ or 0 along the trajectory. This effect is illustrated on fig. 4.1b for instance (the central peak isn't worth completely 100% for numerical reasons, but this value is reached when more nodes are used). It is not perfectly the case on fig. 4.4 but it is close: we observe on the alternative plot that the line representing the ACTSE's thrust magnitude is slightly curved at the normalized switching times 0.7 and 3.6 while they are expected to be perfectly straight. Hence, ACTSE raises several interrogations concerning its behavior that will be partially addressed later.

## Summary

To sum up, we got numerical evidences of the CTSE method's validity: the final masses are similar and the thrust profiles matches well for both discretization techniques. However, a higher computation time and more iterations to reach convergence with respect to classic SCP lower our expectations.

Regarding the ACTSE, interesting results have been found. A perfect bang-off-bang control isn't obtained but the thrust profile shows a close behavior to this phenomenon. Furthermore, the final mass differs by circa $17.5kg$ with respect to the Classic SCP and the CTSE. Hence, doubts are raising about the validity of the method. Consequently, ACTSE with Hermite-Simpson haven't been implemented yet because more proofs of the method's validity are desirable.

The relative simplicity of the Earth-Mars transfer was of good help to try those new approaches but isn't sufficient to assess their performances. Solving a multi-revolution problem with longer propagation time will thus be the next step to test our methods' capacities.

A classic and abundantly studied problem is thus chosen to take further our work: the Earth-Venus transfer with 1000 days of time-of-flight [14]. 2,3,4-revolutions solutions exist in the literature. We will restrain our study to the 2 and 3-revolutions problem. The challenge here is to test our approaches against dynamically inaccurate initial guesses. In all the following, we will use multi-revolution cubic approximations for the trajectory (see subsection 2.5).

## 4.3. Earth-Venus Transfer

The reference data for the Earth-Venus Transfer (EVT) are found in [14]. In that section, we will solve 2 and 3-revolutions EVT and we will refer to those as respectively EVT2r and EVT3r.

### 4.3.1. Parameters

In table 4.6 are gathered all the main numerical parameters used for this section's simulations. Again, the trust-region constraint is, with $i > 1$ the current iteration ($\mathbf{x}_{i-1} = \mathbf{x}^*$):

$$||\mathbf{x}_i - \mathbf{x}_{i-1}||_2 \leq \gamma||\mathbf{x}_{i-1} - \mathbf{x}_{i-2}||_2 \tag{4.9}$$

Table 4.6: Numerical parameters for the Earth-Venus transfer

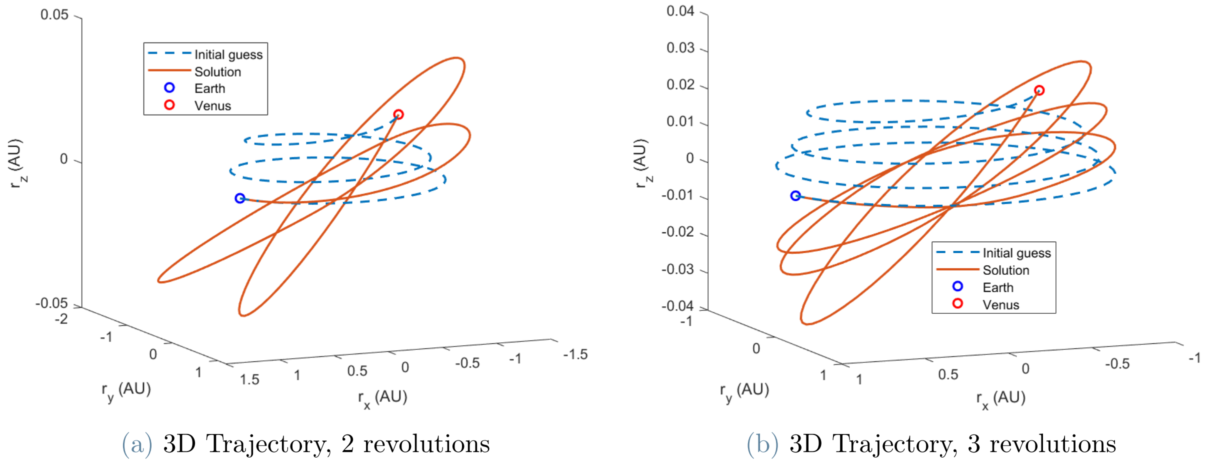
| **Parameter** | **Value** |
|---|---|
| Time of flight $t_f$ | 1000 $days$ |
| Initial position $\mathbf{r}_0$ | $[0.9708,\ 0.2376,\ -1.6711 \times 10^{-6}]^\top$ |
| Initial velocity $\mathbf{v}_0$ | $[-0.2545,\ 0.9687,\ 1.5040 \times 10^{-5}]^\top$ |
| Initial mass $m_0$ | $1500kg$ |
| Final position $\mathbf{r}_f$ | $[-0.3277,\ 0.6389,\ 0.0277]^\top$ |
| Final velocity $\mathbf{v}_f$ | $[-1.0509,\ -0.5436,\ 0.0532]^\top$ |
| Final mass $m_f$ | free |
| Maximum thrust $\mathrm{T}_{max}$ | $0.33N$ |
| Specific impulse $I_{sp}$ | $3800s$ |
| Number of nodes $N$ | 200 |
| w (artificial infeasibility) | $10^2$ |
| $\lambda_d$ (see eq. (3.15)) | $10^2$ |
| $\lambda_s$ (see eq. (3.15)) | $10^4$ |
| Convergence criterion $\delta$ | $5 \times 10^{-3}$ |

## 4.3.2. Reference Solution with Trapezoidal Discretization

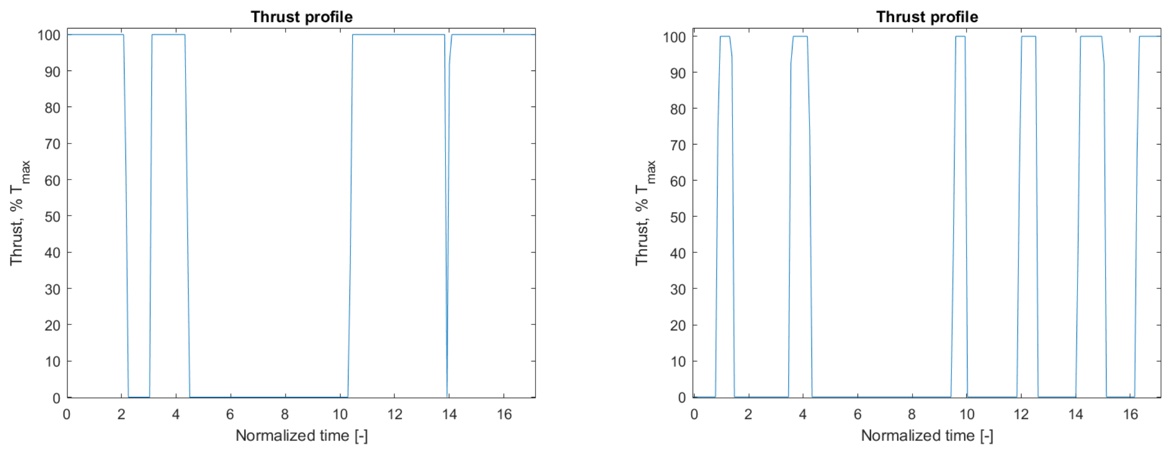
In that subsection, we compute a reference solution with SCP, trapezoidal discretization, $\gamma = 0.98$ and $\mathrm{R}_{tr,0} = 3$. The outcomes are not close to those obtained in [14], but it is not surprising when comparing the simplicity of algorithm 2.1 to the performance-oriented features of the cited work.

Table 4.7: Results for the EVT (classic SCP, trapezoidal discretization)

| $N_{rev}$ (see section 2.5) | **Final mass [kg]** | **Iterations** | **Running time [s]** |
|---|---|---|---|
| 2 | 1047.09 | 21 | 70.85 |
| 3 | 1305.31 | 23 | 76.72 |



(a) 3D Trajectory, 2 revolutions (b) 3D Trajectory, 3 revolutions



(c) Thrust profile, 2 revolutions (d) Thrust profile, 3 revolutions

Figure 4.5: Earth-Venus transfer, Classic SCP, trapezoidal discretization

### 4.3.3. Assessment of CTSE Performances with Trapezoidal Discretization

#### Convergence Issue

During the process of tuning the CTSE algorithm's parameters to find a set that solves efficiently the problem using the multi-revolution cubic approximation initial guess, an issue was met: the algorithm never converged. Several sets of parameters were tried, and an other trust region strategy (the Hard Trust Region with Constant Rates, see [25]) was implemented to reach convergence: none of these attempts was successful. Considering the high running time and number of iterations performed until algorithm failure (more than 50min and 80 iterations), we concluded that the computational effort was not worth it, especially when no satisfying results were obtained.

Keeping in mind our goal, we decided to use the solution of the classic SCP as the initial guess, sticking with the same trust-region strategy eq. (4.9) with $\gamma = 0.8$ and $\mathrm{R}_{tr,0} = 1$. The trajectories and the thrust profiles for 2 and 3-revolutions are available on fig. 4.6, the numerical results are given in table 4.8.

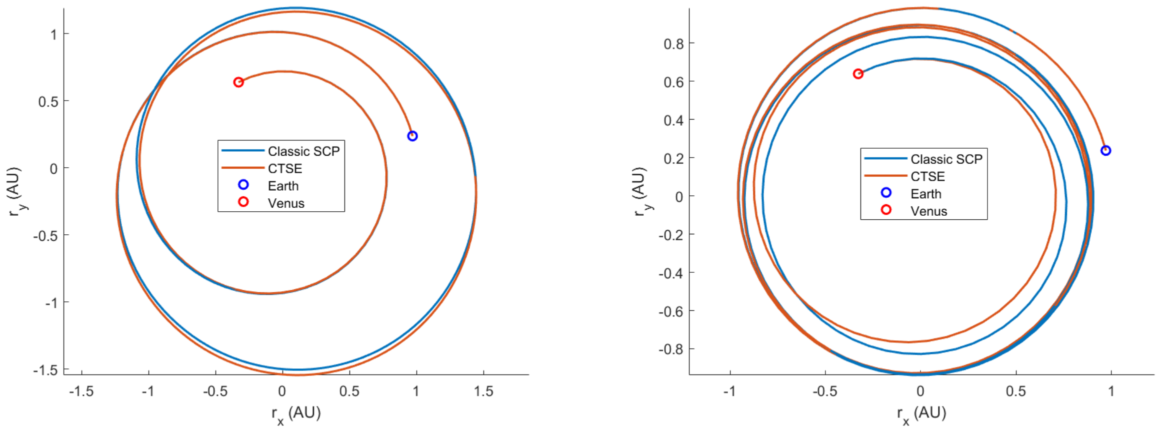
#### First Discussion

To start, it can be seen on fig. 4.6 and table 4.8 that the trajectory, the final mass and the thrust profile of the CTSE stay close to the SCP solution for 2 revolutions (even if some slight differences are noticeable), whereas it is not the case for 3. The bang-off-bang behavior of the thrust magnitude is still present for both. 6 iterations are necessary to reach convergence, with circa 210$s$ of running time. Those numbers show the computational burden of including convexified Hessians in our algorithm (we recall that only the quadratic term is included in the dynamics' TSE).

Finally, to check the accuracy of the method, the EoM are integrated from node to node using the obtained control and the final state is compared to the 6 final boundary conditions. The absolute value of the differences are represented on fig. 4.7. We observe a slight improvement of the solution's accuracy for both cases, that seems to be more effective on the 3-revolution problem. However, the improvement is worth less than an order of magnitude, which doesn't allow us to conclude that CTSE completely outperforms the classic SCP in terms of accuracy. To push further our analysis, we can upgrade the algorithms with the Hermite-Simpson scheme and perform the same comparison to see if CTSE is still interesting against a more accurate classic SCP.

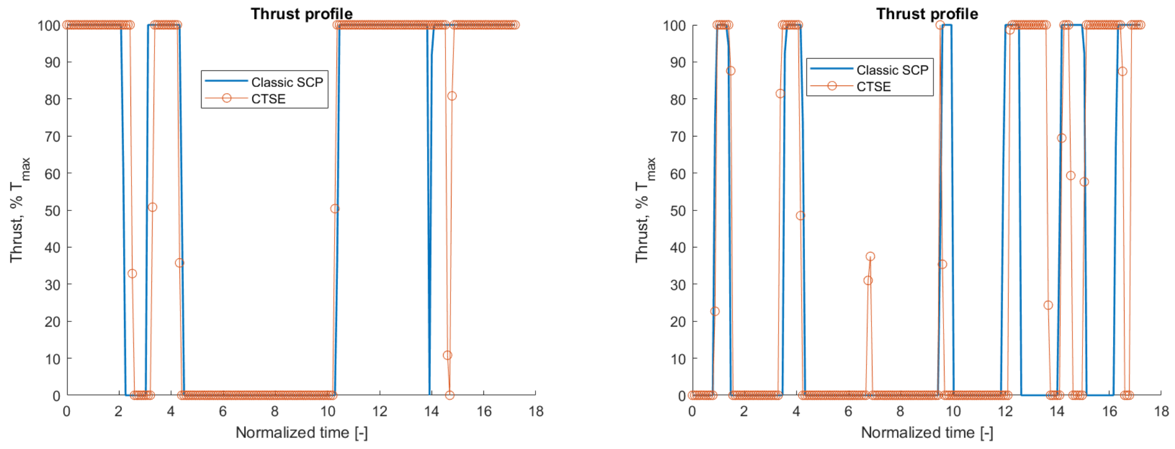
Table 4.8: Results for the EVT (CTSE, trapezoidal discretization)

| $N_{rev}$ (see section 2.5) | **Final mass [kg]** | **Iterations** | **Running time [s]** |
|---|---|---|---|
| 2 | 1046.23 | 6 | 211.63 |
| 3 | 1263.16 | 6 | 210.42 |



(a) 2D Trajectory, 2 revolutions

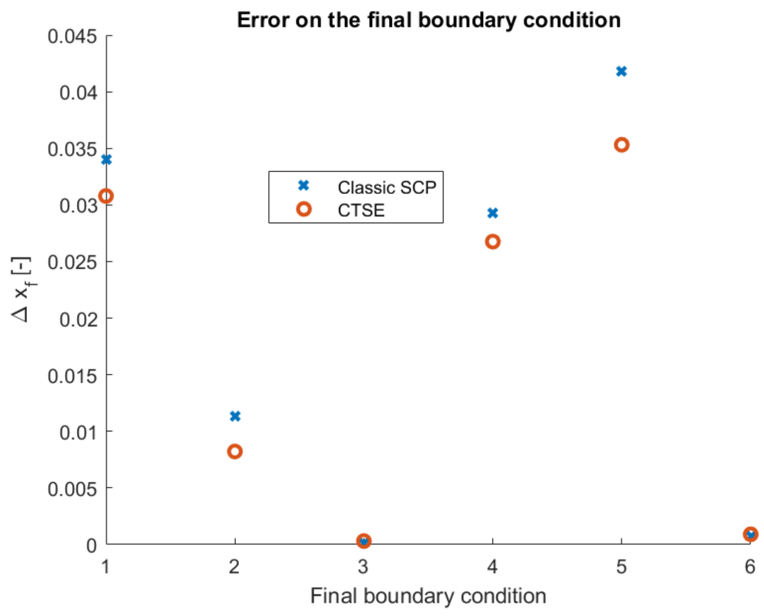
(b) 2D Trajectory, 3 revolutions



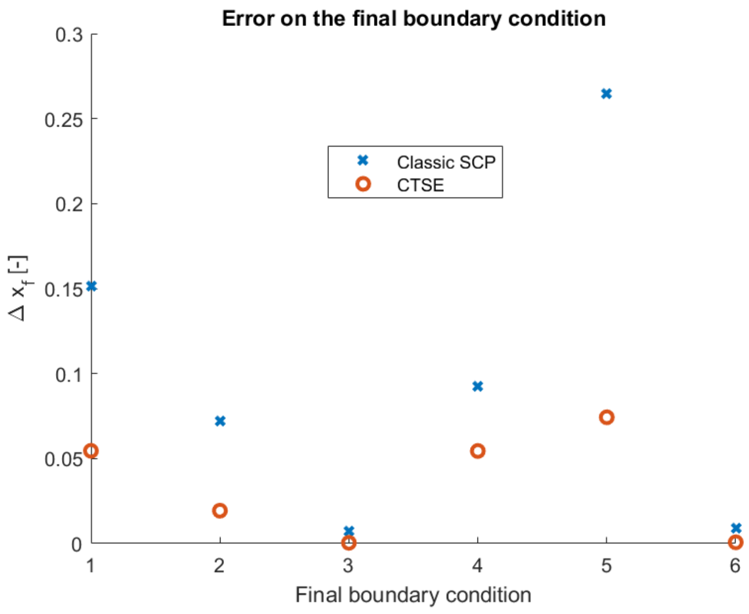
(c) Thrust profile, 2 revolutions

(d) Thrust profile, 3 revolutions

Figure 4.6: Earth-Venus transfer, CTSE, trapezoidal discretization



(a) 2 revolutions



(b) 3 revolutions

Figure 4.7: Error on the final boundary conditions with respect to the integrated solution, trapezoidal discretization

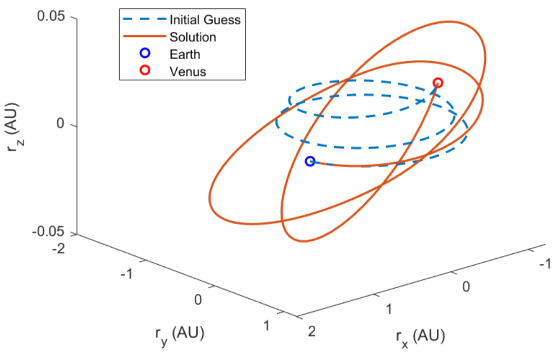
*Remark: it can be observed that the error's order of magnitude for the 2 and 3-revolutions problems are varying a lot. It is due to the fact that the same number of nodes are used for the two problems, but the traveled distance within one time interval is higher in the 3-revolutions case, decreasing the accuracy.*

### 4.3.4. Reference Solution with Hermite-Simpson Scheme

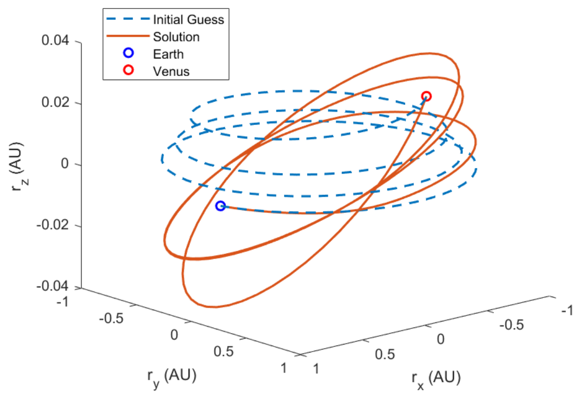
A new reference solution is computed to ensure a fair comparison with the updated CTSE.

Table 4.9: Results for the EVT (classic SCP, Hermite-Simpson scheme)

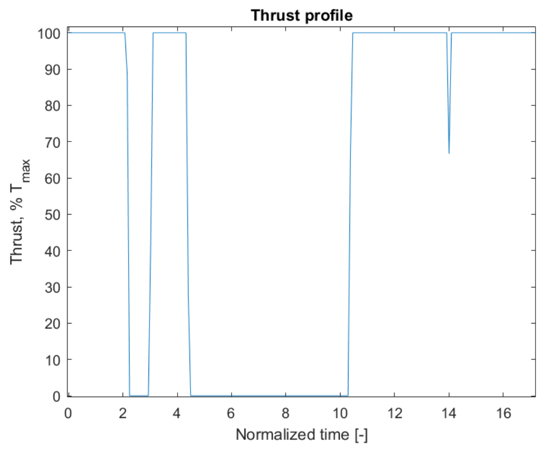
| $N_{rev}$ (see section 2.5) | **Final mass [kg]** | **Iterations** | **Running time [s]** |
|---|---|---|---|
| 2 | 1041.27 | 26 | 118.19 |
| 3 | 1290.35 | 25 | 116.59 |



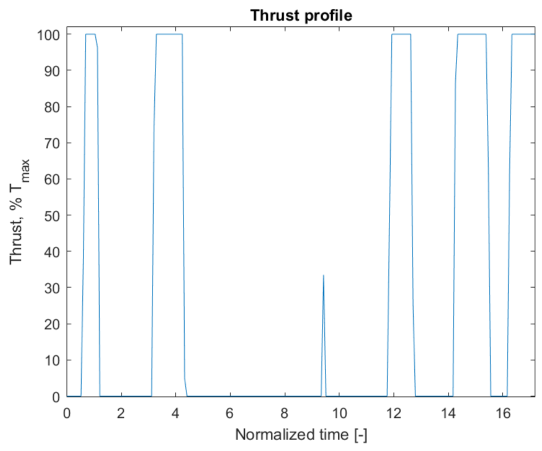
(a) 3D Trajectory, 2 revolutions



(b) 3D Trajectory, 3 revolutions



(c) Thrust profile, 2 revolutions



(d) Thrust profile, 3 revolutions

Figure 4.8: Earth-Venus transfer, Classic SCP, Hermite-Simpson scheme

### 4.3.5. Assessment of CTSE Performances with Hermite-Simpson Scheme

#### Convergence Issue

The same amount of time as in subsection 4.3.3 has been spent trying to make this algorithm converge (which is basically the same algorithm as previously but with Hermite-Simpson scheme instead of trapezoidal discretization) without giving much more positive results. Hence, we used the same method as before: the initial guess is the solution of the classic SCP, and the trust-region is the simplest one (eq. (4.9)) with $\gamma = 0.8$ and $\mathrm{R}_{tr,0} = 1$. The trajectories and the thrust profiles for 2 and 3-revolutions are available on fig. 4.9, the numerical results are given in table 4.10.

#### Second Discussion

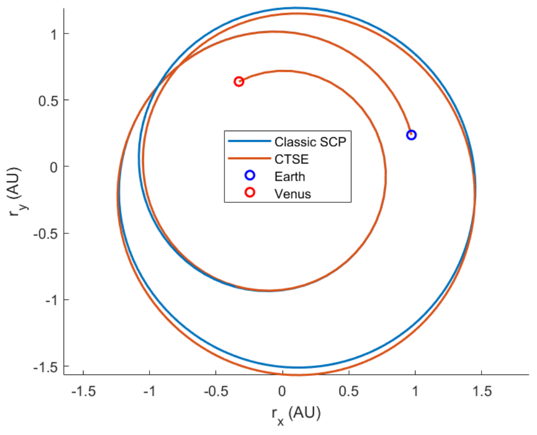
Again, the 2-revolution CTSE solution is comparable to the SCP's. It takes 2 iterations less than the CTSE with trapezoidal discretization to converge and thus is $20s$ faster. On the contrary, the 3-revolution solution differs from the reference and needs the same number of iterations as CTSE with trapezoidal discretization. It is $110s$ slower, which is consistent due to the additional data (collocation point) to compute for Hermite-Simpson.

Concerning the accuracy, the same method is employed as earlier: the results are available on fig. 4.10. With respect to fig. 4.7, we can read on the y-axis that the error's order of magnitude is lower with Hermite-Simpson than with the trapezoidal rule. This fact argues in favor of our results' reliability (indeed, we switched the discretization method to improve the accuracy). From these figures, it can be seen that CTSE improve the 2-revolutions case's accuracy but hinder the latter for the 3-revolutions'. Hence, the CTSE isn't, for the moment, reliable for our purpose: it is able to improve the accuracy in some cases but could also decrease it. Our evolution perspectives are discussed in the next lines.

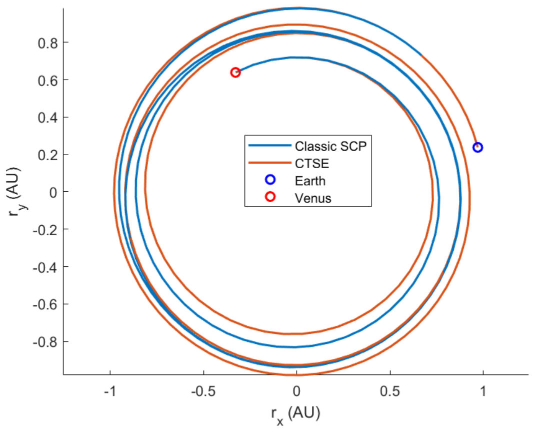
First, an alternative is available to solve the convergence issue, that will be discussed in the next subsection. Then, a hypothetical solution to improve the accuracy could be to specifically design a trust-region strategy adapted to our formulation: those available in the literature [25] were thought to fit the state-of-the-art SCvx or more generally the SCP methods, but ours are a bit borderline with those framework (cf section 3.2). Unsuccessful attempts have been done, that do not appear in this report. This part is thus left to a future work. Finally, the inclusion of convexified higher-order terms will be treated in the next section.

Table 4.10: Results for the EVT (CTSE, Hermite-Simpson scheme)

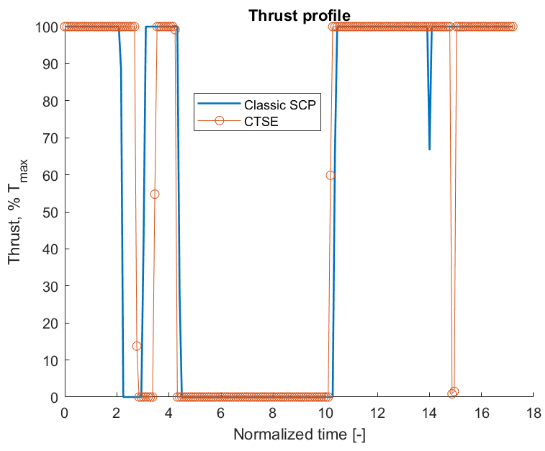
| $N_{rev}$ (see section 2.5) | **Final mass [kg]** | **Iterations** | **Running time [s]** |
|---|---|---|---|
| 2 | 1039.79 | 4 | 192.06 |
| 3 | 1288.57 | 6 | 319.42 |



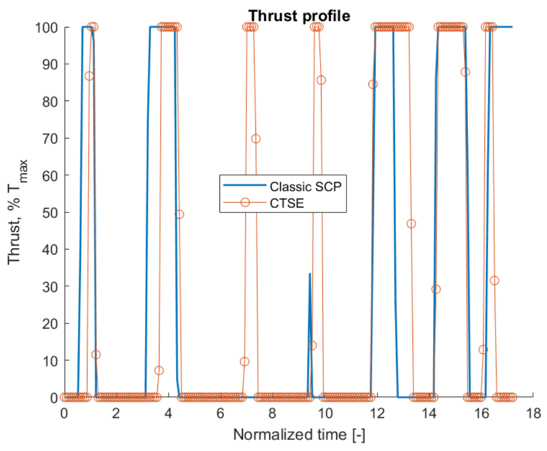
(a) 2D Trajectory, 2 revolutions



(b) 2D Trajectory, 3 revolutions

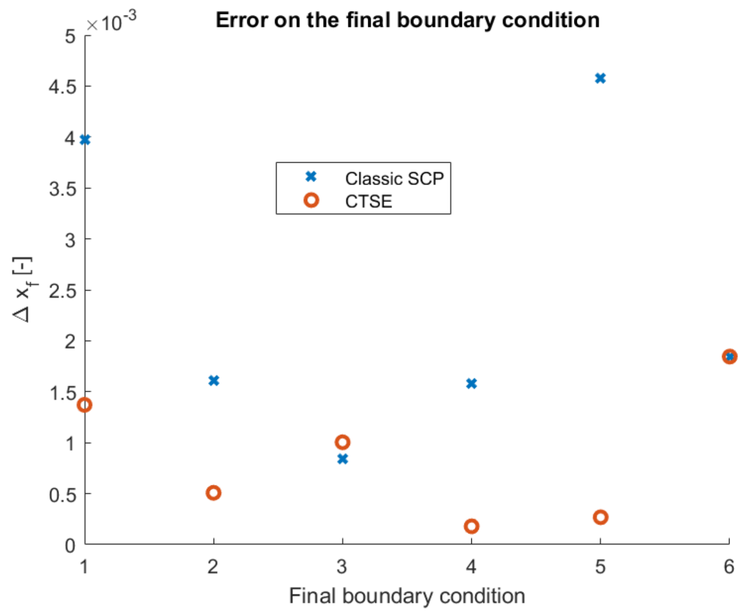


(c) Thrust profile, 2 revolutions

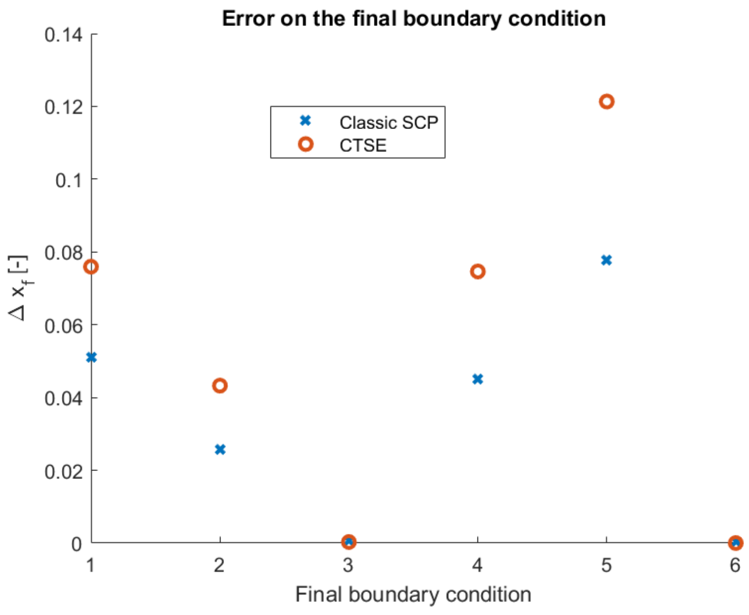


(d) Thrust profile, 3 revolutions

Figure 4.9: Earth-Venus transfer, CTSE, Hermite-Simpson scheme



(a) 2 revolutions



(b) 3 revolutions

Figure 4.10: Error on the final boundary conditions with respect to the integrated solution, CTSE, Hermite-Simpson scheme

*Remark: it can be observed that the error's order of magnitude for the 2 and 3-revolutions problems are varying a lot. It is due to the fact that the same number of nodes are used for the two problems, but the traveled distance within one time interval is higher in the 3-revolutions case, decreasing the accuracy.*

### 4.3.6. Numerical Simulations with ACTSE Method

As mentioned before, the CTSE method doesn't converge with the cubic interpolation initial guess of section 2.5, which brings us to the reason why the ACTSE method has been developed originally. We thought that, by simplifying our formulation and still using inner-convex approximations (staying in the frame of our work), we could get a chance of obtaining a more robust algorithm with respect to CTSE. To run the ACTSE, we changed some parameters with respect to table 4.6:

Table 4.11: New parameters with respect to table 4.6

| $N_{rev}$ | $N$ | $\gamma$ | $\lambda_d$ | $\mathrm{R}_{tr,0}$ |
|---|---|---|---|---|
| 2 | 200 | 0.95 | 50000 | 5 |
| 3 | 250 | 0.97 | 50000 | 5 |

All the others are kept the same. It is important to note that, now, we use the initial guess described in section 2.5. Trapezoidal discretization is employed. When running the algorithm 3.2 with those parameters, the former converges, which is a great achievement with respect to CTSE. The results are gathered in table 4.12. The corresponding trajectories and thrust profiles are available on fig. A.13 and fig. A.14.

Table 4.12: Results for the EVT (ACTSE, trapezoidal discretization)

| $N_{rev}$ (see section 2.5) | **Final mass [kg]** | **Iterations** | **Running time [s]** |
|---|---|---|---|
| 2 | 1104.16 | 37 | 720.21 |
| 3 | 1029.38 | 93 | 2145.01 |

However, looking at the aforementioned table and comparing it to table 4.7, the very high running times of the ACTSE have to be evoked. As a matter of fact, those are 10 to 28 times higher depending on the number of revolutions (see table 4.7). The algorithm needs 16 and 70 iterations more to converge than the classic SCP for 2 and 3 revolutions respectively. Even if ACTSE converges, it is heavier computationally, and thus not interesting in terms of speed with respect to state-of-the-art method. From fig. A.13c and fig. A.14c, the bang-off-bang behavior [60] of the thrust magnitude is still not present, but the curve is close.

Concerning the accuracy of the method, fig. 4.11 shows that the error of the integrated control's final state with respect to the final boundary condition is at least one order of magnitude higher than the classic SCP (with trapezoidal discretization and for the 2-revolutions problem). The approach is thus not interesting in that sense because those errors are excessively high (the first two are of the order of half an astronomical unit).

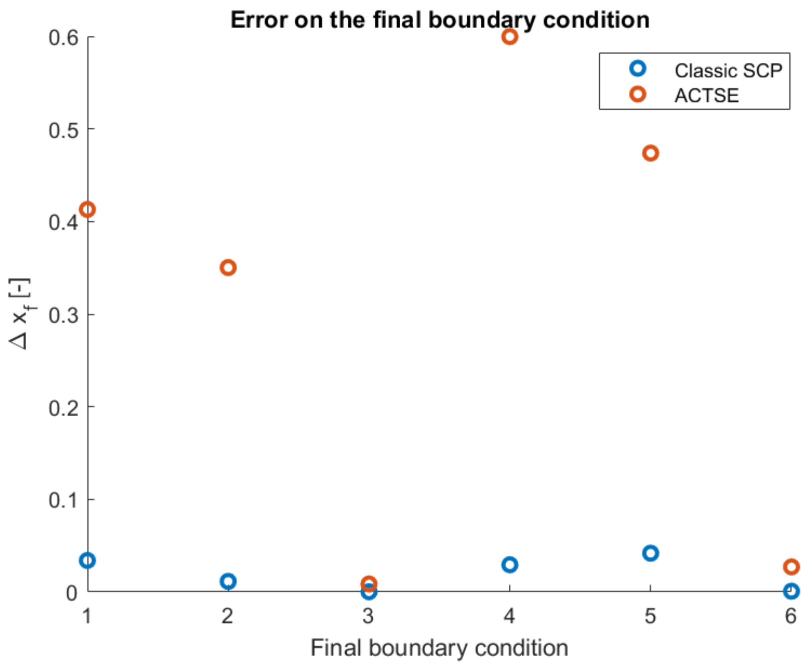


Figure 4.11: Error on the final boundary conditions with respect to the integrated solution, EVT2r, ACTSE, trapezoidal discretization

Finally, the main advantage of the ACTSE method compared to the CTSE is its robustness. Combining the two methods would allow us to propose an approach fully employing the convexification of TSE to model the dynamics, without having to use the classic SCP to generate an initial guess for the CTSE technique. The principle would be to use ACTSE to generate the latter, and then plug the solution in the CTSE algorithm 3.1. However, the computational cost of this idea quickly dissuaded us of implementing such combination.

## 4.4. High-Order Terms Inclusion

We recall that only the convexified second-order term of the TSE (Hessian) has been included in all our simulations since the beginning. Therefore, a natural development of our work would be to use higher-order terms in our algorithms. This section proposes some insights to understand if it is indeed an interesting perspective or not. To begin with, table 4.13 gathers all the running times for every simulation performed so far. Those are sorted by increasing values for each problem (Earth-Mars, 2-revolutions and 3-revolutions Earth-Venus) and the methods and discretization types are given.

Table 4.13: Running times for different cases

| **Problem** | **Method** | **Discretization** | **Running time [s]** |
|---|---|---|---|
| Earth-Mars | Classical SCP | Trapezoidal | 4.84 |
| | ACTSE | Trapezoidal | 108.29 |
| | CTSE | Trapezoidal | 138.76 |
| | CTSE | Hermite-Simpson | 221.91 |
| Earth-Venus, 2 rev.(1) | Classic SCP | Trapezoidal | 70.85 |
| | Classic SCP | Hermite-Simpson | 118.19 |
| | CTSE(2) | Hermite-Simpson | 192.06 |
| | CTSE(2) | Trapezoidal | 211.63 |
| | ACTSE | Trapezoidal | 720.21 |
| Earth-Venus, 3 rev. | Classic SCP | Trapezoidal | 76.72 |
| | Classic SCP | Hermite-Simpson | 116.59 |
| | CTSE(2) | Trapezoidal | 210.42 |
| | CTSE(2) | Hermite-Simpson | 319.42 |
| | ACTSE | Trapezoidal | 2145.01 |

(1): *rev. : revolutions*

(2): *with very good initial guess (solution of Classical SCP)*

Several conclusions can be drawn from these data. First, the classic SCP is always faster by a factor 2 minimum for equivalent problems and discretization types. It is worth mentioning that the former is 20 to 44 times faster than any other method for the Earth-Mars transfer. Thus, our 2 new methods are computationally heavier with respect to the state-of-the-art algorithm. Indeed, CTSE and ACTSE require Hessians to be computed and convexified for each node, for all iterations.

Then, ACTSE is faster for the EMT than CTSE, but it is the contrary for the EVT. It is due to the fact that CTSE is provided with a good initial guess (the solution of the classic SCP), thus the convergence is reached faster. It is worth recalling that without it, CTSE doesn't converge at all.

Finally, in most cases, the algorithms based on trapezoidal discretization are faster than those using the Hermite-Simpson scheme. As already explained, the additional collocation point needed for Hermite-Simpson might be a reason for this extra time. It is not the case for the EVT2r with CTSE because Hermite-Simpson's algorithm required 4 iterations to converge while 6 were needed with the trapezoidal technique.

The inclusion of higher-order terms is now discussed. As said in the previous lines, computing the convexified quadratic term requires a lot of extra time with respect to a simple linearization. For orders greater than 2, the corresponding tensor is of an higher dimension ($m$-dimensional for an order $m$, see appendix B), meaning that the computational effort to compute it (without convexification) is higher than for the $2^{nd}$ order. We are pushing open doors here but it is necessary to mention it for our following discussion.

Furthermore, in [55], a "computationally efficient route" is used to convexify higher-order tensors. The said technique overestimates the higher-order terms to make them convex. But in subsection 3.1.2 and specifically in eq. (3.5), we obtained an Inner-Convex Approximation (ICA) of the dynamics that already overestimates the function by definition of an ICA (eq. (3.1b)). A legitimate question (that will remain unanswered here) is to understand if we can improve the accuracy of an approximation by adding over-estimations to an already overestimated quantity. It is obvious that those terms are smaller and smaller along the growing orders, hence their impacts on the estimation decrease fast. But still, a deeper analysis is needed.

To conclude this part, it is necessary to get further guarantees that adding the higher-order terms will actually improve the algorithm's accuracy because, surely, it will add a computational burden (as witnessed when adding the quadratic term). However, the way of computing the higher-order tensors has been developed in appendix B and implemented in numerical functions.

## 4.5. Final Discussion and Improvement Directions

To sum up, CTSE converges for dynamically accurate initial guesses (integrated control or solution of a classic SCP). When it does, the error on the final boundary condition usually decreases but it is not a certainty with the simple trust-region strategy we used. The method is computationally heavier than the classic SCP for all cases because the high-order terms' computation and convexification are expensive with respect to a simple linearization.

The ACTSE method converges for any initial guess which is a great achievement with respect to CTSE. However, the price to pay is a high computational cost and a poor accuracy. Combining it with the CTSE would result in an algorithm employing fully the convexification of high-order TSE to model the dynamics, but some improvements regarding the speed are needed before implementing this.

The improvement directions for these methods are, in that order:

1. Working on a trust-region strategy purposely designed for those, to make sure that the accuracy is improved with respect to classic SCP (it could also result in a decreased number of iteration to reach convergence).
2. Understanding why CTSE does not converge with the multi-revolution cubic approximation initial guess.
3. Answering the following question: is it worth it to include higher-order terms in the CTSE (greater than 2) ?
4. Getting more knowledge about the behavior of ACTSE: why the bang-off-bang profile isn't found although the solution is so close ? What are the performances if other discretization techniques are used ? Is the inclusion of higher-order terms interesting ?
5. Starting to do some theoretical work to understand deeper those methods in a mathematical point of view, if all the previous directions have been successfully explored.

# 5 | Conclusion

## 5.1. Review of the Work Performed in this Thesis

To conclude, two interesting methods employing convexified Taylor Series Expansion to model the dynamics of a spacecraft have been developed and studied in this thesis.

First, the CTSE converges with dynamically accurate initial guesses or for simple problems, but meets some troubles for harder ones. Better quality guesses can be employed in that case. In terms of accuracy, the CTSE often improves the solution of the classic SCP. It is an achievement and a very interesting result to see that CTSE have the potential to reach the initial goal (improving reliability with respect to state-of-the-art techniques), although its computational weight and convergence issues are burdens to be taken into account if future developments are considered.

The ACTSE is also a novelty worth of interests. The method is very poor in terms of accuracy but has better convergence properties and computational efficiency than CTSE. It is still heavy with respect to classic SCP but that is the price to pay when computing and convexifying high-order terms. ACTSE generates solutions that are unexpected (related to the bang-off-bang behavior) and thus worth studying.

To the research questions stated in the introduction, it can be answered that the CTSE improves the reliability of the optimization solution but only in some cases, while the ACTSE does not. With respect to state-of-the-art methods, both techniques are slower and less efficient, and the CTSE has convergence issues.

Finally, several interesting development directions are available for both methods, and thus expectations could be raised concerning their futures. They are able to solve the low-thrust trajectory problem for deep-space guidance, and they can sometimes increase the reliability of the classical methods solution. Hence, this thesis is successful by means of the short-term goals defined in the introduction.

## 5.2. Differences of Convex Functions to Obtain Inner-Convex Approximations

To extend our work, a novel idea that could fit our purposes is presented in that section. It comes from the same formulation as the ACTSE method, and uses also principles that are explained in [55]. The method that will be mentioned here hasn't been implemented because it doesn't employ TSE and is thus out of this work's frame. It is a different theoretical concept that has the potential to help reaching this thesis' goal.

We start by defining the defects for the non-linear dynamical constraints as we did for the ACTSE method. We recall $\mathbf{s} = [\mathbf{x}, \mathbf{u}]^\top$.

$$\begin{cases} D_x(\mathbf{s}) := (\dot{v}_x - p_{rx}(\mathbf{x}) - c\tau_x)^2 \\ D_y(\mathbf{s}) := (\dot{v}_y - p_{ry}(\mathbf{x}) - c\tau_y)^2 \\ D_z(\mathbf{s}) := (\dot{v}_z - p_{rz}(\mathbf{x}) - c\tau_z)^2 \end{cases} \tag{5.1}$$

We could also define, with $||.||$ any norm:

$$\begin{cases} D_x(\mathbf{s}) := ||\dot{v}_x - p_{rx}(\mathbf{x}) - c\tau_x|| \\ D_y(\mathbf{s}) := ||\dot{v}_y - p_{ry}(\mathbf{x}) - c\tau_y|| \\ D_z(\mathbf{s}) := ||\dot{v}_z - p_{rz}(\mathbf{x}) - c\tau_z|| \end{cases} \tag{5.2}$$

The definition of a so-called d.c. function is given in the following lines (d.c. stands for difference of convex functions). Let $f : \mathcal{E} \longrightarrow \mathbb{R}$ be a function of a variable $x$ with $\mathcal{E} \subset \mathbb{R}^d$, $d \in \mathbb{N}$ and $\mathcal{E}$ convex. $f$ represents in practice $D_x$, $D_y$ or $D_z$. From [55, 61], $f$ is d.c. if it exists convex functions $c_1, c_2 : \mathcal{E} \longrightarrow \mathbb{R}$ such that:

$$f = c_1 - c_2 \tag{5.3}$$

From [55], an inner-convex approximation (ICA) of $f$ is obtained if the concave component $-c_2$ of the above equality is linearized, and the said approximation can then be included in the cost function as done in the theoretical development of this work. It is an other way of obtaining an ICA for $f$, different from the one used in this thesis (see eq. (3.20) for instance), but equally interesting. The main issue (a mathematical one) is to find $c_1$ and $c_2$ because they are not unique and can thus grant various performances. Several ways of solving the former are explained in [55] depending on the features of $f$.

The aforementioned idea is interesting in many ways. First, the d.c. property is the topic of several articles in the literature [61, 62], thus their theoretical features are well known. Then the *convex-concave procedure*, meaning the linearization of the component $-c_2$ to obtain a convex expression, has already been studied and employed [63, 64]. Also, the use of d.c. functions in optimization is not a complete novelty and is itself a widely developed programming method called *DC Programming* [65] that has already been applied to real-world problems [66]. It shows that the method explained before could be relevant, even if the application seems new. The main problem is hence finding $c_1$ and $c_2$, but it has already been done in particular cases like non-convex polynomial functions [67].

# Bibliography

[1] E. Kulu, "In-space economy in 2021 - statistical overview and classification of commercial entities," October 2021.

[2] G. Di Domenico, E. Andreis, A. Morelli, G. Merisio, V. Franzese, C. Giordano, A. Morselli, P. Panicucci, F. Ferrari, and F. Topputo, "Toward self-driving interplanetary cubesats: the erc-funded project extrema," October 2021.

[3] J. M. Longuski, J. J. Guzmán, and J. E. Prussing, *Optimal Control with Aerospace Applications*. Springer New York, NY, July 2014.

[4] D. Malyuta, Y. Yu, P. Elango, and B. Açıkmeşe, "Advances in trajectory optimization for space vehicle control," *Annual Reviews in Control*, vol. 52, pp. 282–315, 2021. `https://doi.org/10.1016/j.arcontrol.2021.04.013`.

[5] R. F. Hartl, S. P. Sethi, and R. G. Vickson, "A survey of the maximum principles for optimal control problems with state constraints," *SIAM Review*, vol. 37, no. 2, pp. 181–218, 1995. `https://doi.org/10.1137/1037043`.

[6] S. D. S. Fernandes, "Optimum low-thrust limited power transfers between neighbouring elliptic non-equatorial orbits in a non-central gravity field," *Acta Astronautica*, vol. 35, no. 12, pp. 763–770, 1995. `https://doi.org/10.1016/0094-5765(95)00002-H`.

[7] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear Programming*. John Wiley Sons, Ltd, 2006. `https://doi.org/10.1002/0471787779`.

[8] F. Topputo and C. Zhang, "Survey of direct transcription for low-thrust space trajectory optimization with applications," *Abstract and Applied Analysis*, vol. 2014, pp. 1–15, June 2014. `https://doi.org/10.1155/2014/851720`.

[9] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[10] A. Ben-Tal and A. Nemirovski, "Convex optimization in engineering: Modeling, analysis, algorithms," 1998.

[11] Y. Mao, M. Szmuk, and B. Açıkmeşe, "A tutorial on real-time convex optimization based guidance and control for aerospace applications," pp. 2410–2416, June 2018. `https://doi.org/10.23919/ACC.2018.8430984`.

[12] B. Açıkmeşe and L. Blackmore, "Lossless convexification of a class of optimal control problems with non-convex control constraints," *Automatica*, vol. 47, no. 2, pp. 341–347, 2011. `https://doi.org/10.1016/j.automatica.2010.10.037`.

[13] B. Açıkmeşe and S. R. Ploen, "Convex programming approach to powered descent guidance for mars landing," *Journal of Guidance, Control, and Dynamics*, vol. 30, no. 5, pp. 1353–1366, 2007. `https://doi.org/10.2514/1.27553`.

[14] C. Hofmann and F. Topputo, "Rapid low-thrust trajectory optimization in deep space based on convex programming," *Journal of Guidance, Control, and Dynamics*, vol. 44, pp. 1–10, 04 2021. `https://doi.org/10.2514/1.G005839`.

[15] A. C. Morelli, C. Hofmann, and F. Topputo, "Robust low-thrust trajectory optimization using convex programming and a homotopic approach," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 58, no. 3, pp. 2103–2116, 2022. `https://doi.org/10.1109/TAES.2021.3128869`.

[16] Z. Wang and M. J. Grant, "Minimum-fuel low-thrust transfers for spacecraft: A convex approach," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 54, no. 5, pp. 2274–2290, 2018. `https://doi.org/10.1109/TAES.2018.2812558`.

[17] X. Liu, Z. Shen, and P. Lu, "Exact convex relaxation for optimal flight of aerodynamically controlled missiles," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 52, no. 4, pp. 1881–1892, 2016. `https://doi.org/10.1109/TAES.2016.150741`.

[18] Y. Mao, M. Szmuk, and B. Açıkmeşe, "Successive convexification of non-convex optimal control problems and its convergence properties," August 2016.

[19] D. Malyuta, T. Reynolds, M. Szmuk, T. Lew, R. Bonalli, M. Pavone, and B. Açıkmeşe, "Convex optimization for trajectory generation," June 2021.

[20] R. Bonalli, A. Cauligi, A. Bylard, and M. Pavone, "Gusto: Guaranteed sequential trajectory optimization via sequential convex programming," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 6741–6747, 2019. `https://doi.org/10.1109/ICRA.2019.8794205`.

[21] J. F. Bonnans, J. C. Gilbert, C. Lemaréchal, and C. A. Sagastizábal, *Numerical Optimization, Theoretical and Practical Aspects*. Springer Berlin, Heidelberg, 2 ed., 2006. `https://doi.org/10.1007/978-3-540-35447-5`.

[22] A. Domahidi, E. Chu, and S. Boyd, "Ecos: An socp solver for embedded systems," in *2013 European Control Conference (ECC)*, pp. 3071–3076, 2013. `https://doi.org/10.23919/ECC.2013.6669541`.

[23] D. Dueri, B. Açıkmeşe, D. P. Scharf, and M. W. Harris, "Customized real-time interior-point methods for onboard powered-descent guidance," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 2, pp. 197–212, 2017. `https://doi.org/10.2514/1.G001480`.

[24] C. Zhang and F. Topputo, "Space trajectory optimization via direct transcription and collocation: A note for practical implementation," tech. rep., Politecnico di Milano, Department of Aerospace Science and Technology, Scientific Report DSTA-SR 13-02, July 2013.

[25] C. Hofmann, A. Morelli, and F. Topputo, "On the performance of discretization and trust-region methods for on-board convex low-thrust trajectory optimization," January 2022. `https://doi.org/10.2514/6.2022-1892`.

[26] W. W. Hager, H. Hou, and A. Rao, "Convergence rate for a gauss collocation method applied to unconstrained optimal control," *J Optim Theory Appl*, vol. 169, p. 801–824, 2016. `https://doi.org/10.1007/s10957-016-0929-7`.

[27] W. W. Hager, J. Liu, S. Mohapatra, A. V. Rao, and X.-S. Wang, "Convergence rate for a gauss collocation method applied to constrained optimal control," *SIAM Journal on Control and Optimization*, vol. 56, no. 2, pp. 1386–1411, 2018. `https://doi.org/10.1137/16M1096761`.

[28] W. W. Hager, H. Hou, and A. V. Rao, "Convergence rate for a radau collocation method applied to unconstrained optimal control," 2015. `https://arxiv.org/abs/1508.03783`.

[29] A. Spitzbart, "A generalization of hermite's interpolation formula," *The American Mathematical Monthly*, vol. 67, no. 1, pp. 42–46, 1960. `http://www.jstor.org/stable/2308924`.

[30] E. Süli and D. Mayers, *An Introduction to Numerical Analysis*. Cambridge University Press, 2003. `https://doi.org/10.1017/CBO9780511801181`.

[31] P. Williams, "Hermite-legendre-gauss-lobatto direct transcription in trajectory optimization," *Journal of Guidance, Control, and Dynamics*, vol. 32, no. 4, pp. 1392–1395, 2009. `https://doi.org/10.2514/1.42731`.

[32] G. Elnagar, M. Kazemi, and M. Razzaghi, "The pseudospectral legendre method for

discretizing optimal control problems," *IEEE Transactions on Automatic Control*, vol. 40, no. 10, pp. 1793–1796, 1995. `https://doi.org/10.1109/9.467672`.

[33] M. Sagliano, S. Theil, M. Bergsma, V. D'Onofrio, L. J. Whittle, and G. Viavattene, "On the radau pseudospectral method: theoretical and implementation advances," *CEAS Space Journal*, vol. 9, pp. 313–331, 2017.

[34] M. Sagliano, "Pseudospectral convex optimization for powered descent and landing," *Journal of Guidance, Control, and Dynamics*, vol. 41, no. 2, pp. 320–334, 2018. `https://doi.org/10.2514/1.G002818`.

[35] C. Hofmann and F. Topputo, "Toward on-board guidance of low-thrust spacecraft in deep space using sequential convex programming," February 2021.

[36] C. Hofmann and F. Topputo, "Pseudospectral convex low-thrust trajectory optimization in a high-fidelity model," August 2021.

[37] Z. Wang and M. Grant, "Optimization of minimum-time low-thrust transfers using convex programming," *Journal of Spacecraft and Rockets*, vol. 55, pp. 1–13, December 2017. `https://doi.org/10.2514/1.A33995`.

[38] X. Liu, P. Lu, and B. Pan, "Survey of convex optimization for aerospace applications," *Astrodynamics*, vol. 1, 02 2017. `https://doi.org/10.1007/s42064-017-0003-8`.

[39] Y. Mao, D. Dueri, M. Szmuk, and B. Açıkmeşe, "Successive convexification of non-convex optimal control problems with state constraints," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 4063–4069, 2017. 20th IFAC World Congress. `https://doi.org/10.1016/j.ifacol.2017.08.789`.

[40] B. Benedikter, A. Zavoli, and G. Colasurdo, "A convex approach to rocket ascent trajectory optimization," July 2019. `https://doi.org/10.13009/EUCASS2019-430`.

[41] P. Simplício, A. Marcos, and S. Bennani, "Guidance of reusable launchers: Improving descent and landing performance," *Journal of Guidance, Control, and Dynamics*, vol. 42, no. 10, pp. 2206–2219, 2019. `https://doi.org/10.2514/1.G004155`.

[42] Z. Wang and M. Grant, "Constrained trajectory optimization for planetary entry via sequential convex programming," *Journal of Guidance, Control, and Dynamics*, vol. 40, pp. 1–13, 07 2017. `https://doi.org/10.2514/1.G002150`.

[43] E. Belbruno and J. Miller, "A ballistic lunar capture trajectory for the japanese spacecraft hiten," tech. rep., JPL, California Institute of Technology, 1990. Technical Report IOM 312/90.4-1731-EAB.

[44] S. Campagnola and M. Lo, “Bepicolombo gravitational capture and the elliptic restricted three-body problem,” *PAMM*, vol. 7, pp. 1030905 – 1030906, December 2007. `https://doi.org/10.1002/pamm.200700330`.

[45] A. Morelli, G. Merisio, C. Hofmann, and F. Topputo, “A convex guidance approach to target ballistic capture corridors at mars,” February 2022.

[46] B. Williams, P. Antreasian, E. Carranza, C. Adam, J. Leonard, D. Nelson, B. Page, D. Stanbridge, D. Wibben, K. Williams, M. Moreau, K. Berry, K. Getzandanner, A. Liounis, A. Mashiku, D. Highsmith, B. Sutter, and D. Lauretta, “Osiris-rex flight dynamics and navigation design,” *Space Science Reviews*, vol. 214, April 2018. `https://doi.org/10.1007/s11214-018-0501-x`.

[47] V. Franzese, F. Topputo, F. Ankersen, and R. Walker, “Deep-space optical navigation for m-argo mission,” *The Journal of the Astronautical Sciences*, vol. 68, 10 2021. `https://doi.org/10.1007/s40295-021-00286-9`.

[48] R. M. Pinson and P. Lu, “Trajectory design employing convex optimization for landing on irregularly shaped asteroids,” *Journal of Guidance, Control, and Dynamics*, vol. 41, no. 6, pp. 1243–1256, 2018. `https://doi.org/10.2514/1.G003045`.

[49] M. Grant, S. Boyd, and Y. Ye, *Disciplined Convex Programming*, pp. 155–210. Boston, MA: Springer US, 2006. `https://doi.org/10.1007/0-387-30528-9_7`.

[50] M. Grant and S. Boyd, “CVX: Matlab software for disciplined convex programming, version 2.1.” `http://cvxr.com/cvx`, Mar. 2014.

[51] M. Grant and S. Boyd, “Graph implementations for nonsmooth convex programs,” in *Recent Advances in Learning and Control* (V. Blondel, S. Boyd, and H. Kimura, eds.), Lecture Notes in Control and Information Sciences, pp. 95–110, Springer-Verlag Limited, 2008. `http://stanford.edu/~boyd/graph_dcp.html`.

[52] Y. M. Mao, M. Szmuk, and B. Açıkmeşe, “Successive convexification: A superlinearly convergent algorithm for non-convex optimal control problems,” April 2018. https://arxiv.org/abs/1804.06539.

[53] X. Lei, R. he, H.-B. Zhang, and G. Tang, “Oscillation phenomenon in trust-region-based sequential convex programming for the nonlinear trajectory planning problem,” *IEEE Transactions on Aerospace and Electronic Systems*, February 2022. `https://doi.org/10.1109/TAES.2022.3153761`.

[54] R. Foust, S.-J. Chung, and F. Y. Hadaegh, “Optimal guidance and control with nonlinear dynamics using sequential convex programming,” *Journal of Guidance,*

*Control, and Dynamics*, vol. 43, no. 4, pp. 633–644, 2020. `https://doi.org/10.2514/1.G004590`.

[55] J. Virgili-Llop and M. Romano, "A recursively feasible and convergent sequential convex programming procedure to solve non-convex problems with linear equality constraints," September 2018. `https://doi.org/10.13140/RG.2.2.28622.10564`.

[56] B. R. Marks and G. P. Wright, "A general inner approximation algorithm for non-convex mathematical programs," *Operations Research*, vol. 26, no. 4, pp. 681–683, 1978. `http://www.jstor.org/stable/169728`.

[57] E. Taheri and O. Abdelkhalik, "Initial three-dimensional low-thrust trajectory design," *Advances in Space Research*, vol. 57, no. 3, pp. 889–903, 2016. `https://www.sciencedirect.com/science/article/pii/S0273117715008315`.

[58] C. Meyer, *Matrix Analysis and Applied Linear Algebra.* Other Titles in Applied Mathematics, Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104), 2000. `https://books.google.fr/books?id=-7JeAwAAQBAJ`.

[59] R. M. Freund, "Penalty and barrier methods for constrained optimization," 2004. `https://ocw.mit.edu/courses/15-084j-nonlinear-programming-spring-2004/pages/lecture-notes/`.

[60] C. Zhang, F. Topputo, F. Bernelli-Zazzera, and Y.-S. Zhao, "Low-thrust minimum-fuel optimization in the circular restricted three-body problem," *Journal of Guidance, Control, and Dynamics*, vol. 38, pp. 1–9, March 2015. `https://doi.org/10.2514/1.G001080`.

[61] J.-B. Hiriart-Urruty, "Generalized differentiability / duality and optimization for problems dealing with differences of convex functions," in *Convexity and Duality in Optimization* (J. Ponstein, ed.), (Berlin, Heidelberg), pp. 37–70, Springer Berlin Heidelberg, 1985.

[62] P. Hartman, "On functions representable as a difference of convex functions," *Pacific Journal of Mathematics*, vol. 9, pp. 707–713, 1959.

[63] A. Yuille and A. Rangarajan, "The concave-convex procedure," *Neural Computation*, vol. 15, pp. 915–936, April 2003.

[64] T. Lipp and S. Boyd, "Variations and extension of the convex–concave procedure," *Optimization and Engineering*, vol. 17, pp. 263–287, June 2016. `https://doi.org/10.1007/s11081-015-9294-x`.

[65] R. Horst and N. V. Thoai, “Dc programming: Overview,” *Journal of Optimization Theory and Applications*, vol. 103, pp. 1–43, October 1999. `https://doi.org/10.1023/A:1021765131316`.

[66] L. T. H. An and P. D. Tao, “The dc (difference of convex functions) programming and dca revisited with dc models of real world nonconvex optimization problems,” *Annals of Operations Research*, vol. 133, pp. 23–46, January 2005. `https://doi.org/10.1007/s10479-004-5022-1`.

[67] A. Ahmadi and G. Hall, “Dc decomposition of nonconvex polynomials with algebraic techniques,” *Mathematical Programming*, vol. 169, 10 2015.

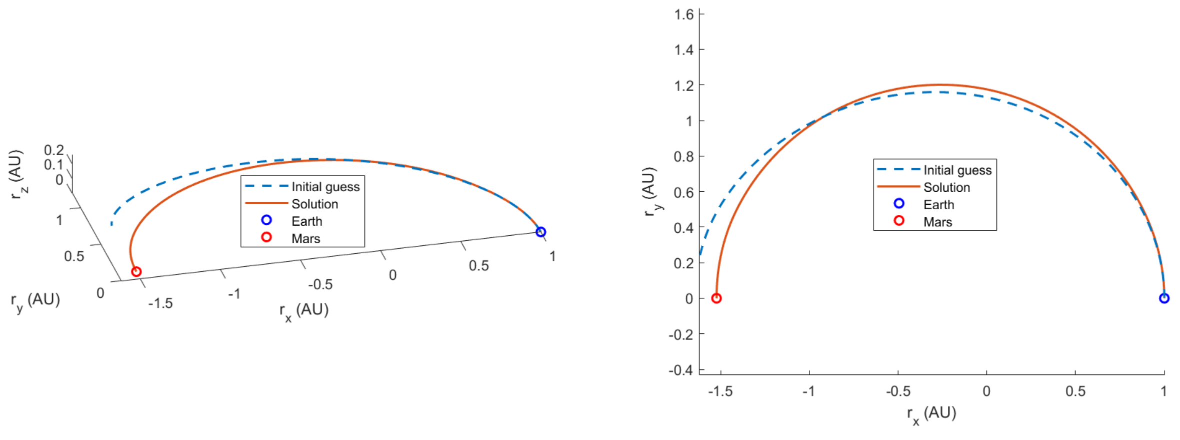
[68] L. Hörmander, *The Analysis of Linear Partial Differential Operators I: Distribution Theory and Fourier Analysis*, pp. 5–32. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003. `https://doi.org/10.1007/978-3-642-61497-2_2`.

[69] J. J. Duistermaat and J. A. C. Kolk, *Distributions: Theory and Applications*, pp. 59–63. Boston: Birkhäuser Boston, 2010. `https://doi.org/10.1007/978-0-8176-4675-2_6`.

# A | Appendix: Graphs & Figures

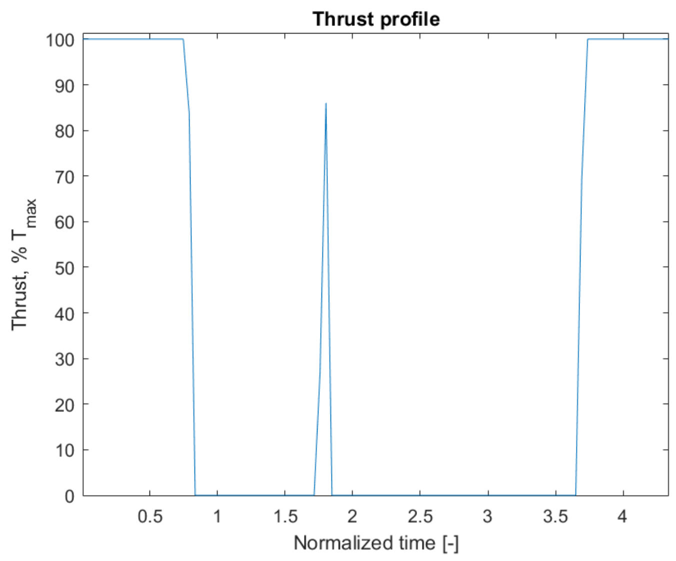
This Appendix gathers all the graphs and figures that have been generated during this thesis. Several are in the main text but all of them couldn't fit for sake of conciseness. The reader is referred to the list of acronyms for any doubts about those used in the next pages.

## A.1. EMT, Classic SCP, trapezoidal discretization



(a) 3D Trajectory

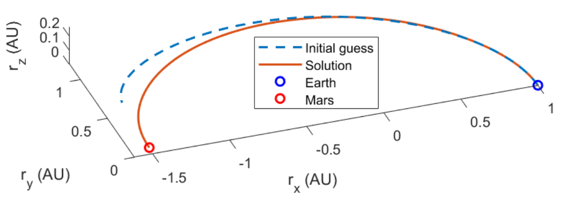
(b) 2D Trajectory



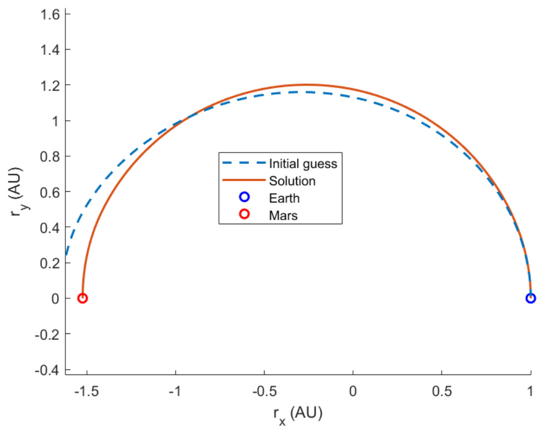
(c) Thrust profile

Figure A.1: Earth-Mars transfer, Classic SCP, trapezoidal discretization

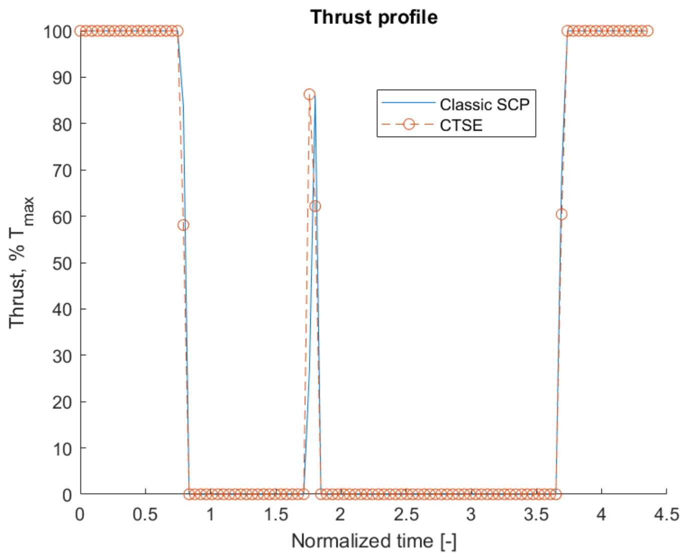
## A.2. EMT, CTSE, trapezoidal discretization



(a) 3D Trajectory



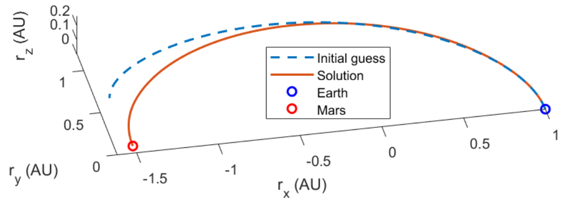
(b) 2D Trajectory



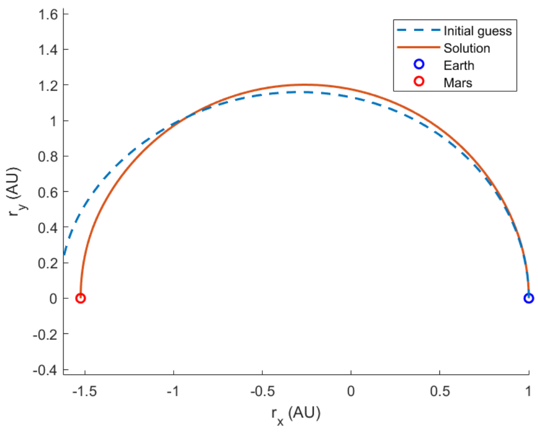
(c) Thrust profile

Figure A.2: Earth-Mars transfer, CTSE, trapezoidal discretization

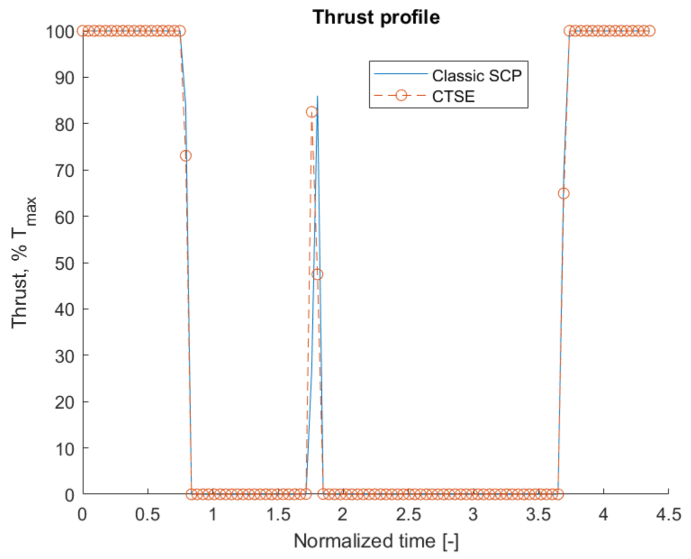
## A.3. EMT, CTSE, Hermite-Simpson scheme



(a) 3D Trajectory



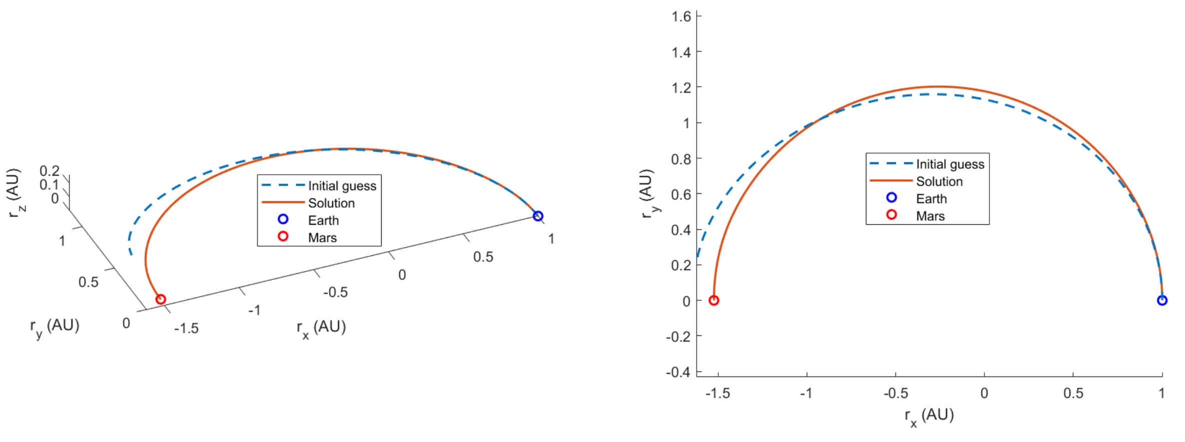
(b) 2D Trajectory



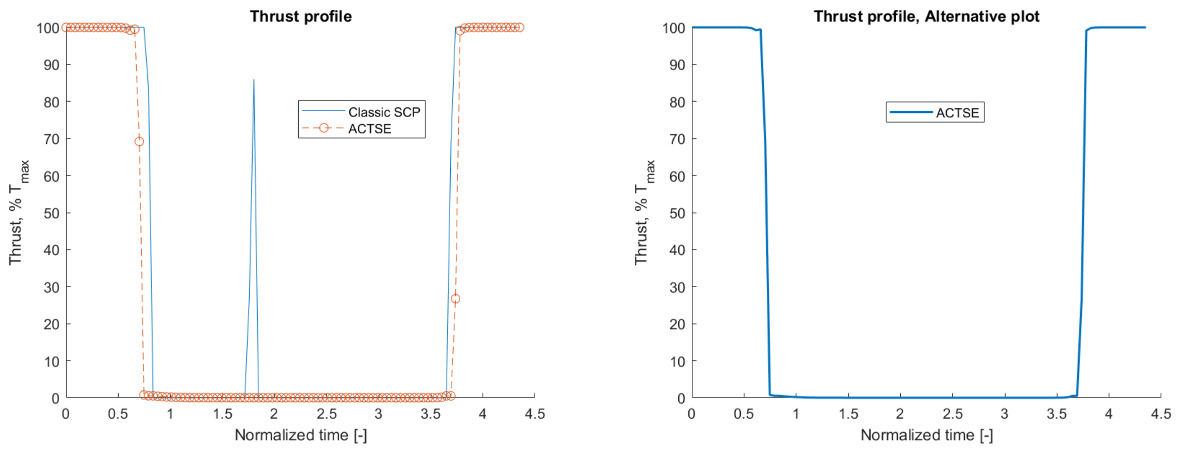
(c) Thrust profile

Figure A.3: Earth-Mars transfer, CTSE, Hermite-Simpson scheme

## A.4. EMT, ACTSE, trapezoidal discretization



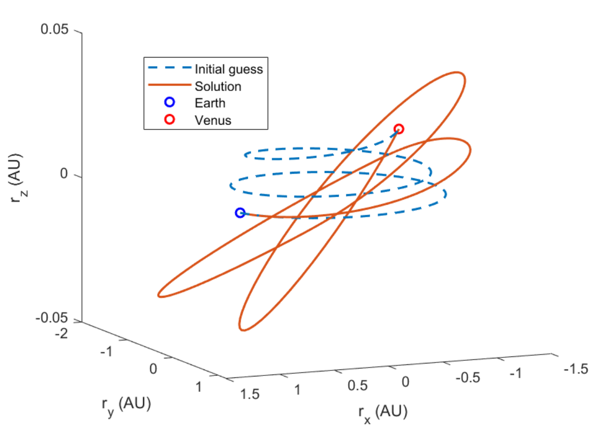
(a) 3D Trajectory (b) 2D Trajectory



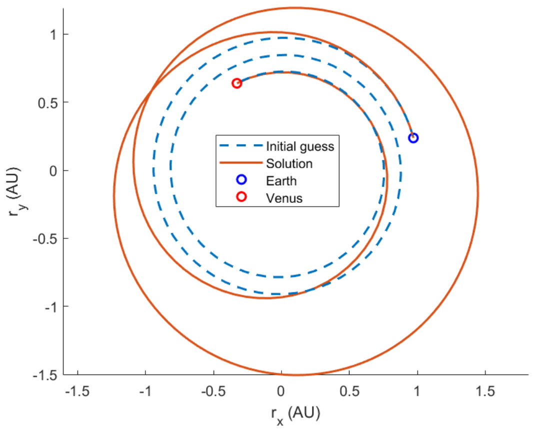
(c) Thrust profile (d) Alternative thrust profile

Figure A.4: Earth-Mars transfer, ACTSE, trapezoidal discretization

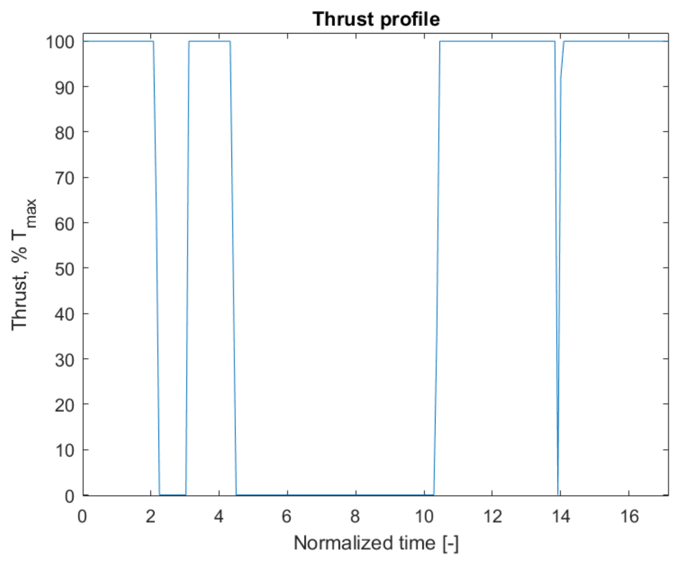
## A.5. EVT2r, Classic SCP, trapezoidal discretization



(a) 3D Trajectory



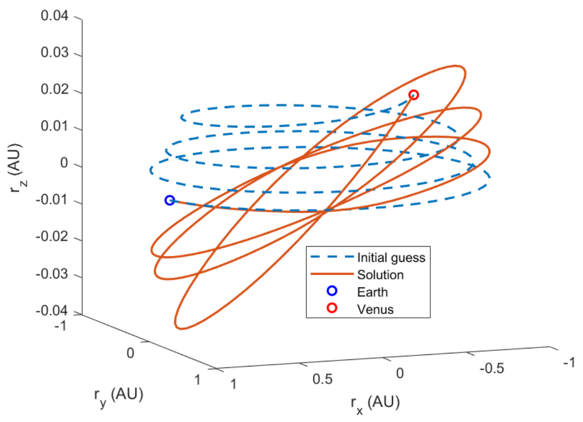
(b) 2D Trajectory



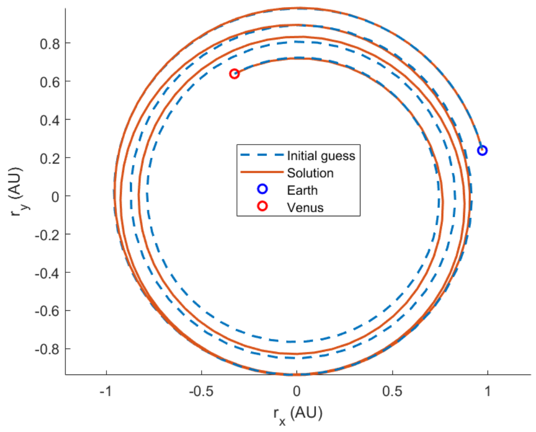
(c) Thrust profile

Figure A.5: 2-revolutions Earth-Venus transfer, Classic SCP, trapezoidal discretization

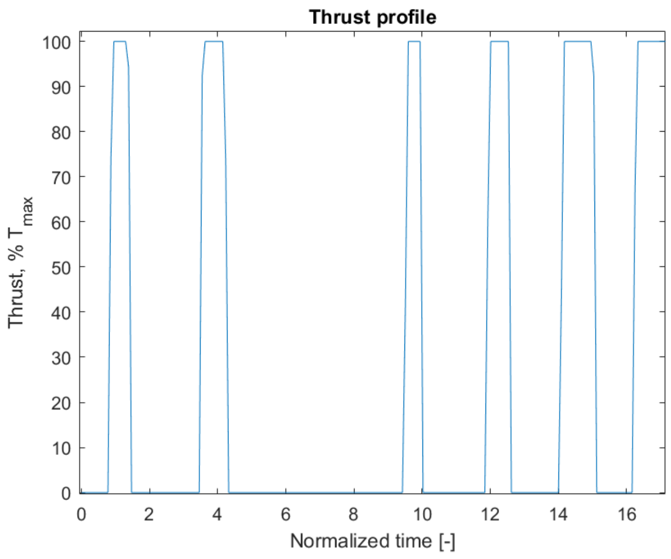
## A.6. EVT3r, Classic SCP, trapezoidal discretization



(a) 3D Trajectory



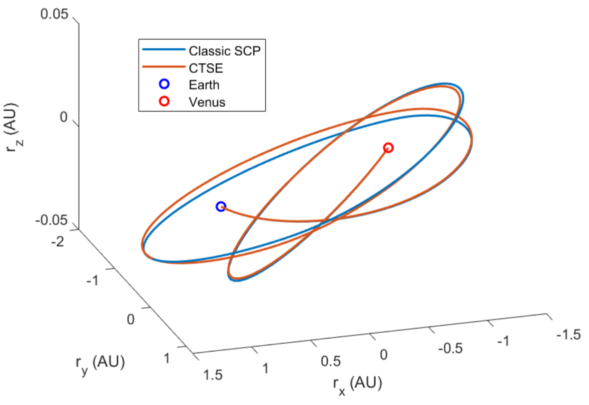
(b) 2D Trajectory



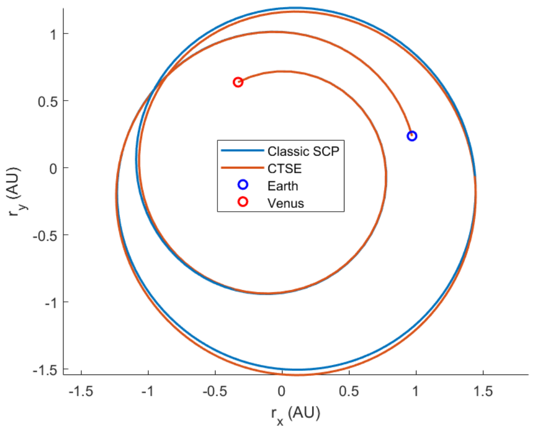
(c) Thrust profile

Figure A.6: 3-revolutions Earth-Venus transfer, Classic SCP, trapezoidal discretization

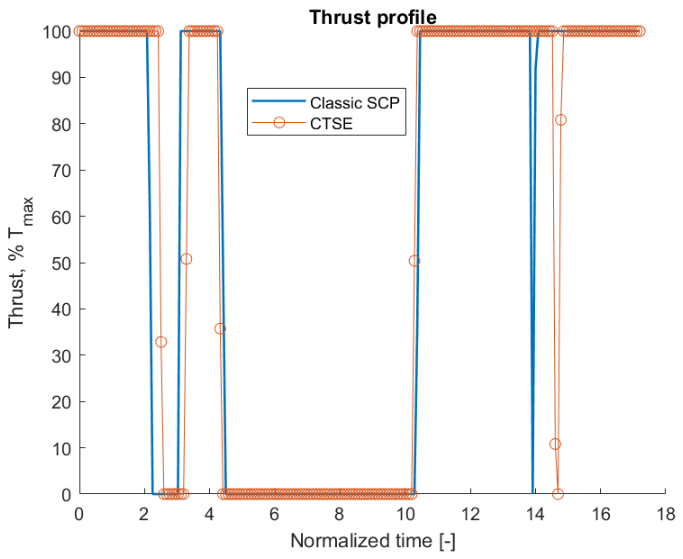
## A.7. EVT2r, CTSE, trapezoidal discretization



(a) 3D Trajectory



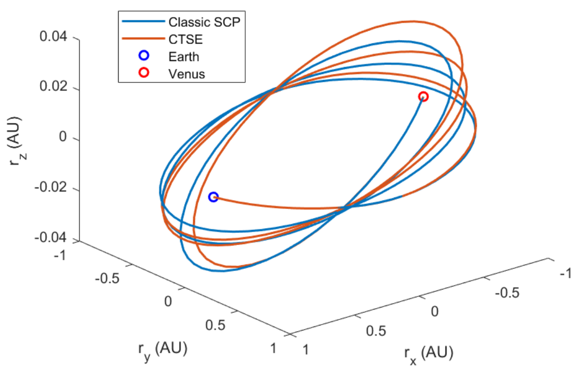
(b) 2D Trajectory



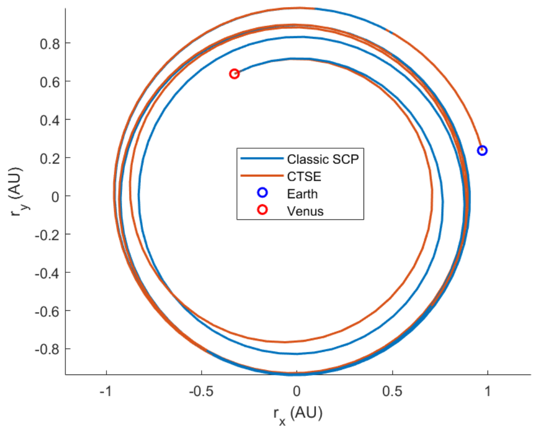
(c) Thrust profile

Figure A.7: 2-revolutions Earth-Venus transfer, CTSE, trapezoidal discretization

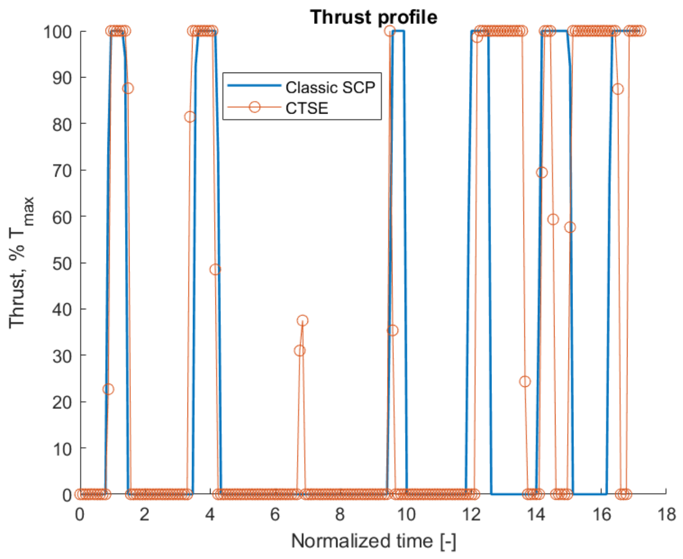
## A.8. EVT3r, CTSE, trapezoidal discretization



(a) 3D Trajectory



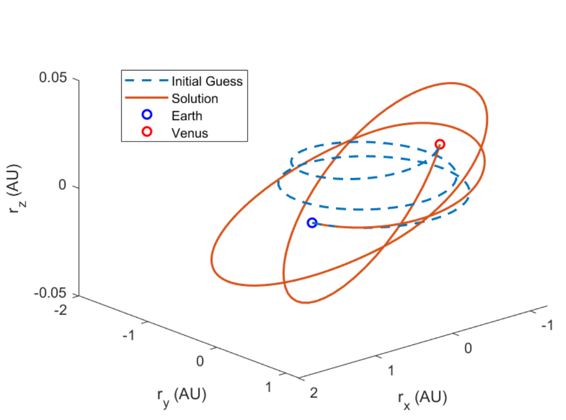
(b) 2D Trajectory



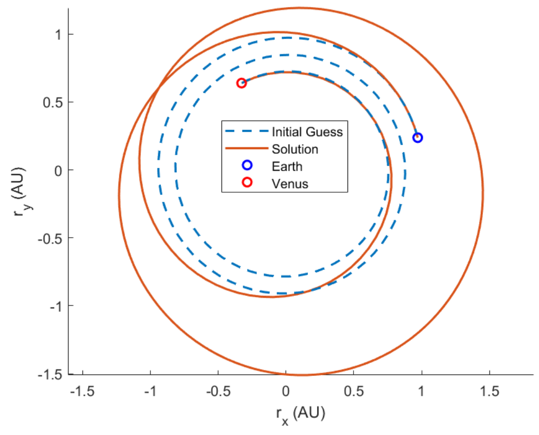
(c) Thrust profile

Figure A.8: 3-revolutions Earth-Venus transfer, CTSE, trapezoidal discretization

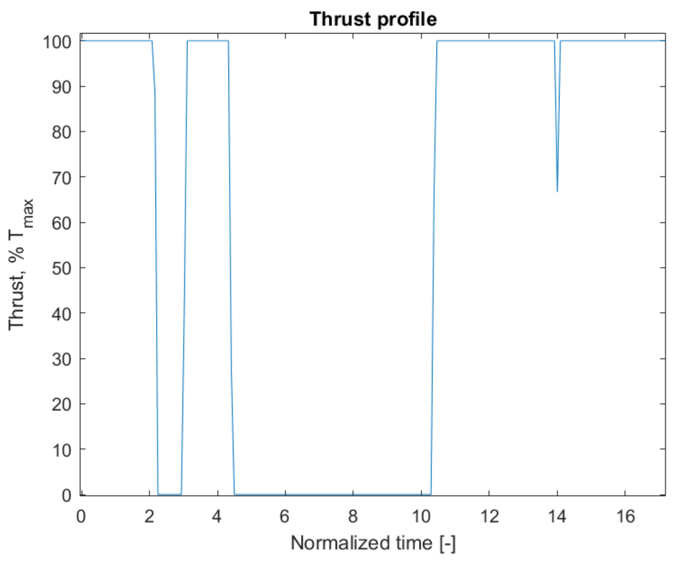
## A.9. EVT2r, Classic SCP, Hermite-Simpson scheme



(a) 3D Trajectory



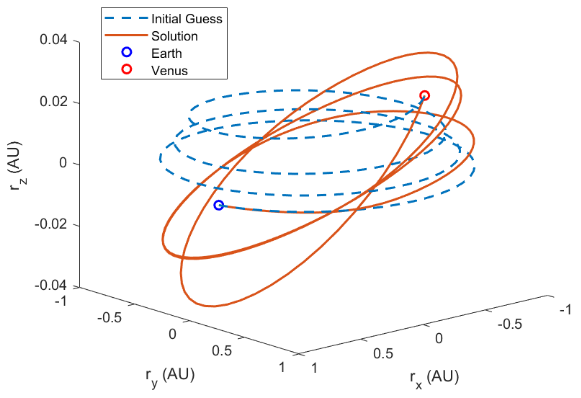
(b) 2D Trajectory



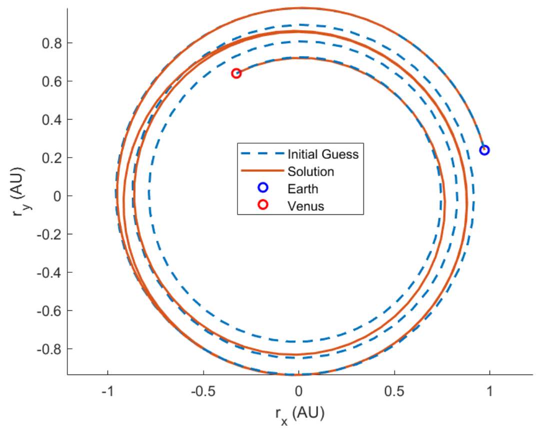
(c) Thrust profile

Figure A.9: 2-revolutions Earth-Venus transfer, Classic SCP, Hermite-Simpson scheme

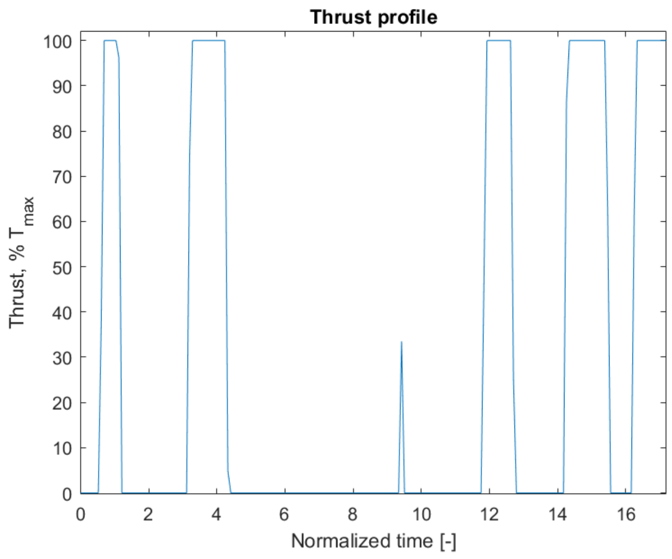
## A.10. EVT3r, Classic SCP, Hermite-Simpson scheme



(a) 3D Trajectory



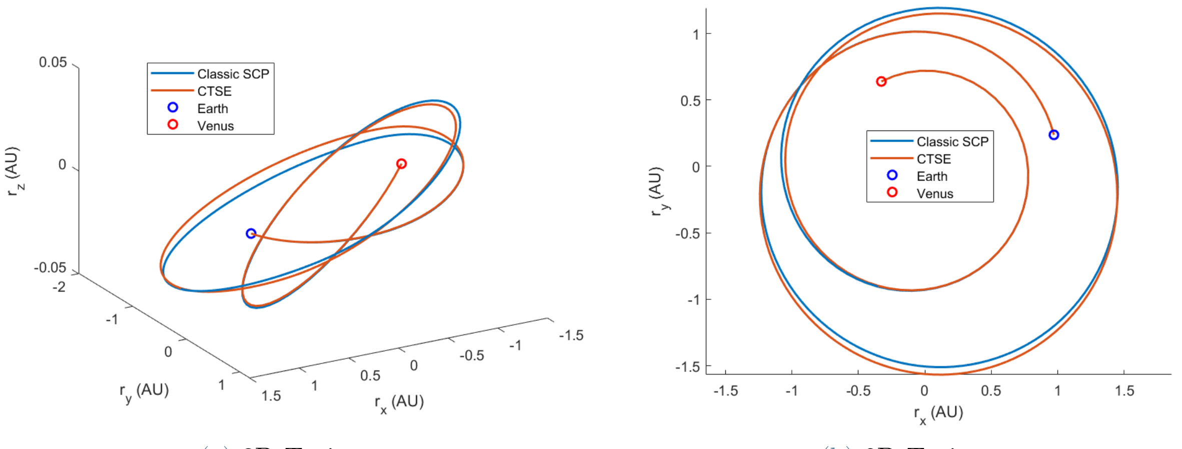
(b) 2D Trajectory



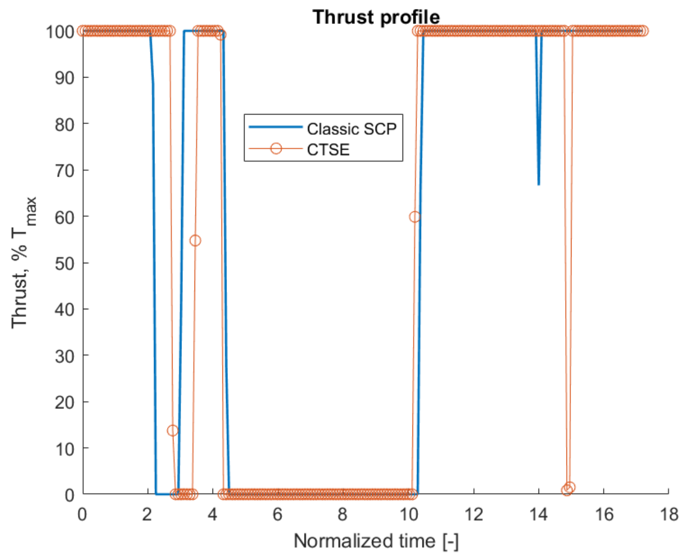
(c) Thrust profile

Figure A.10: 3-revolutions Earth-Venus transfer, Classic SCP, Hermite-Simpson scheme

## A.11. EVT2r, CTSE, Hermite-Simpson scheme



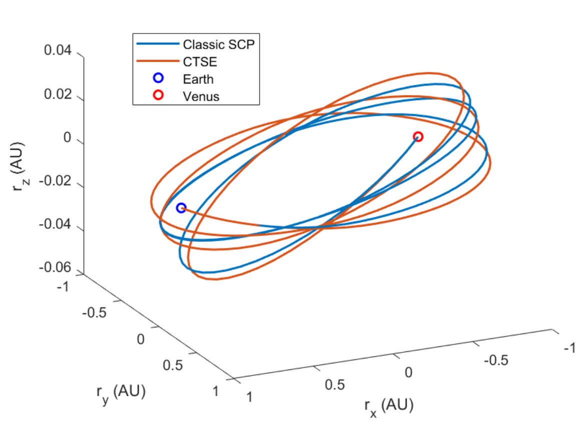
(a) 3D Trajectory (b) 2D Trajectory



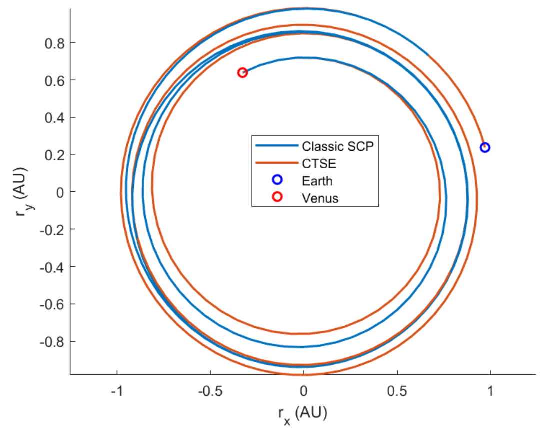
(c) Thrust profile

Figure A.11: 2-revolutions Earth-Venus transfer, CTSE, Hermite-Simpson scheme

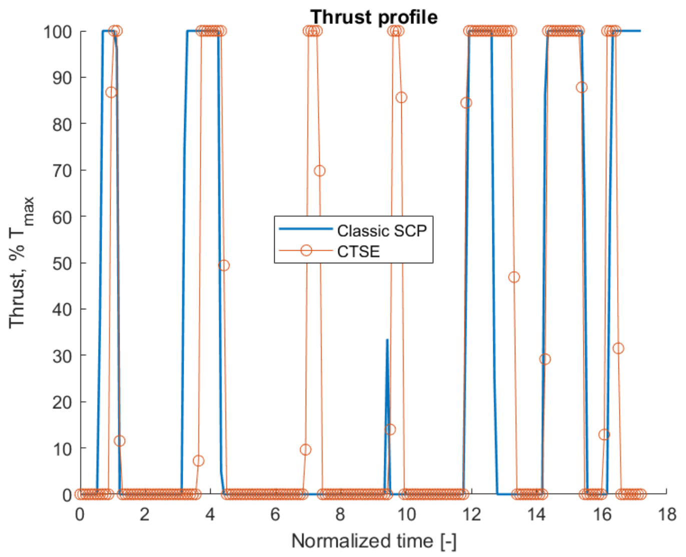
## A.12. EVT3r, CTSE, Hermite-Simpson scheme



(a) 3D Trajectory



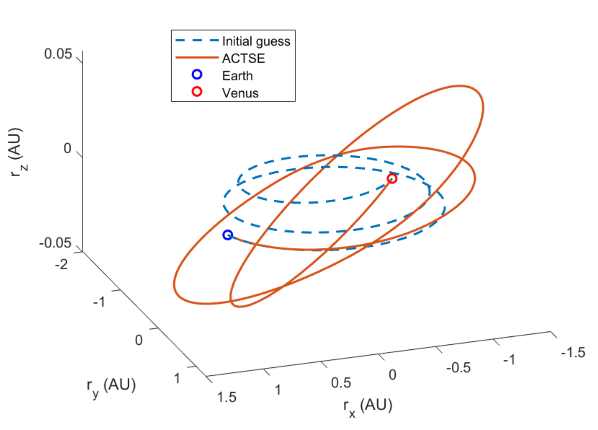
(b) 2D Trajectory



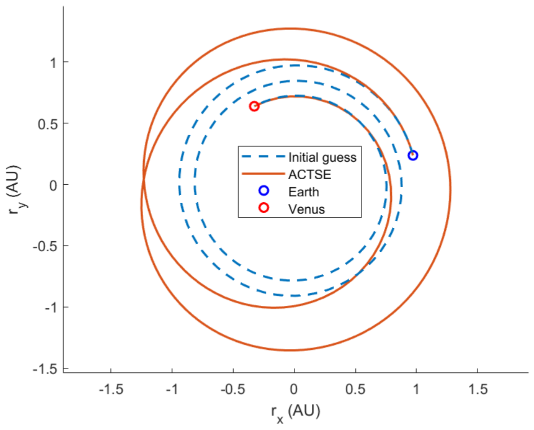
(c) Thrust profile

Figure A.12: 3-revolutions Earth-Venus transfer, CTSE, Hermite-Simpson scheme

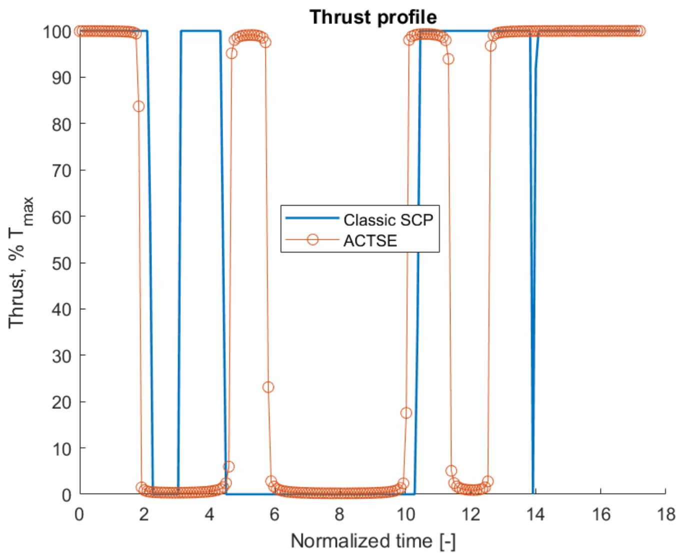
## A.13. EVT2r, ACTSE, trapezoidal discretization



(a) 3D Trajectory



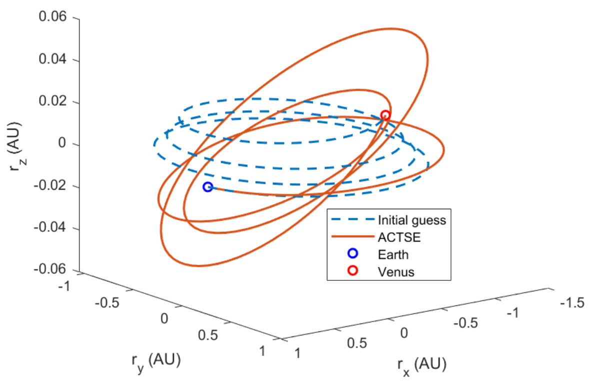
(b) 2D Trajectory



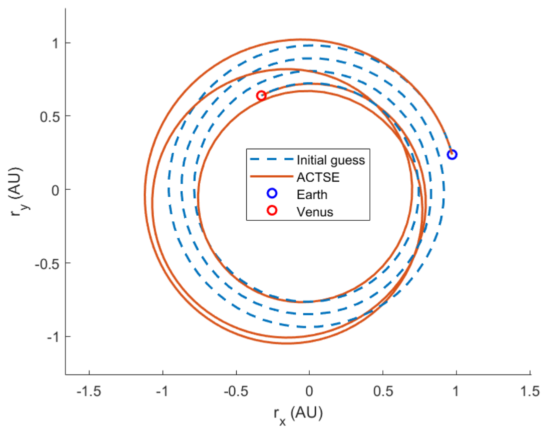
(c) Thrust profile

Figure A.13: 2-revolutions Earth-Venus transfer, ACTSE, trapezoidal discretization

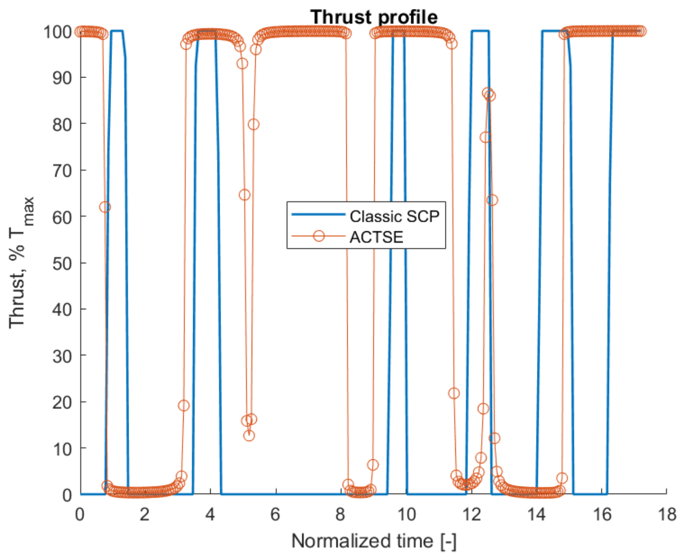
## A.14. EVT3r, ACTSE, trapezoidal discretization



(a) 3D Trajectory



(b) 2D Trajectory



(c) Thrust profile

Figure A.14: 3-revolutions Earth-Venus transfer, ACTSE, trapezoidal discretization

# B | Appendix - Analytic & Numerical Computation of Multivariate TSE

Let $f : \mathcal{E} \longrightarrow \mathbb{R}$ be a function of a variable $x = (x_1, x_2, x_3, \dots, x_d)^\top$ with $\mathcal{E} \subset \mathbb{R}^d$ and $d \in \mathbb{N}$. Let $x^* \in \mathcal{E}$, $x^* = (x_1^*, x_2^*, x_3^*, \dots, x_d^*)^\top$ be a reference point and $\delta x = x - x^*$. $f$ represents in practice any component $p_{rx}$, $p_{ry}$, $p_{rz}$ in eq. (3.7) or $D_x$, $D_y$, $D_z$ in eq. (3.18). $f$ is a multivariate scalar function.

## B.1. Multivariate TSE

We define here some notations to discuss about the TSE of a multivariate scalar function. First we recall the TSE of $f$ at $x^*$ up to order $m$ [68, 69]:

$$\begin{aligned}
f(x) \simeq & f(x^*) \\
& + \sum_{j_1=1}^{d} \frac{\partial f(x^*)}{\partial x_{j_1}} \left(x_{j_1} - x_{j_1}^*\right) \\
& + \sum_{j_1=1}^{d} \sum_{j_2=1}^{d} \frac{1}{2!} \frac{\partial^2 f(x^*)}{\partial x_{j_1} \partial x_{j_2}} \left(x_{j_1} - x_{j_1}^*\right) \left(x_{j_2} - x_{j_2}^*\right) \\
& + \sum_{j_1=1}^{d} \sum_{j_2=1}^{d} \sum_{j_3=1}^{d} \frac{1}{3!} \frac{\partial^3 f(x^*)}{\partial x_{j_1} \partial x_{j_2} \partial x_{j_3}} \left(x_{j_1} - x_{j_1}^*\right) \left(x_{j_2} - x_{j_2}^*\right) \left(x_{j_3} - x_{j_3}^*\right) \\
& + \dots \\
& + \sum_{j_1=1}^{d} \sum_{j_2=1}^{d} \cdots \sum_{j_m=1}^{d} \frac{1}{m!} \frac{\partial^m f(x^*)}{\partial x_{j_1} \partial x_{j_2} \dots \partial x_{j_m}} \left(x_{j_1} - x_{j_1}^*\right) \left(x_{j_2} - x_{j_2}^*\right) \dots \left(x_{j_m} - x_{j_m}^*\right)
\end{aligned} \tag{B.1}$$

Let $\mathbf{T}_{f,m} \in \mathbb{R}^{d\times d\times\cdots\times d}$ be the tensor gathering the $m^{th}$ order derivatives of $f$ evaluated on the reference $x^*$:

$$\mathbf{T}_{f,m}(j_1, j_2, \ldots, j_m) = \frac{\partial^m f(x^*)}{\partial x_{j_1}\partial x_{j_2}\ldots\partial x_{j_m}} \tag{B.2}$$

$\mathbf{T}_{f,m}$ is supersymmetric, hence any permutation of two indices $j_p, j_q$ for $p, q \in \{1, 2, \ldots, m\}$ leaves the quantity $\mathbf{T}_{f,m}(j_1, j_2, \ldots, j_m)$ invariant. Then, we denote (similarly to [55]):

$$\mathbf{T}_{f,m}\delta x^m := \sum_{j_1=1}^{d}\cdots\sum_{j_m=1}^{d}\frac{1}{m!}\frac{\partial^m f(x^*)}{\partial x_{j_1}\partial x_{j_2}\ldots\partial x_{j_m}}\left(x_{j_1} - x^*_{j_1}\right)\ldots\left(x_{j_m} - x^*_{j_m}\right) \tag{B.3}$$

Compactly, eq. (B.1) is written:

$$f(x) \simeq \sum_{i=1}^{m}\mathbf{T}_{f,i}\delta x^i \tag{B.4}$$

In particular, we have:

$$\begin{cases} \mathbf{T}_{f,0} = f(x^*) & \text{the } 0^{th} \text{ order} \\ \mathbf{T}_{f,1} = \nabla f(x^*) & \text{the gradient} \\ \mathbf{T}_{f,2} = \mathbf{H}(x^*) & \text{the Hessian} \end{cases}$$

## B.2. Analytic

The goal of this section is to find a strategy to compute $\mathbf{T}_{f,m}$ for arbitrary $m \in \mathbb{N}$ in a clever way. The idea is to exploit the supersymmetric property to avoid computing $\mathbf{T}_{f,m}(j_1, j_2, \ldots, j_m)$ for all $(j_1, j_2, \ldots, j_m) \in \mathbb{N}^m$. To do so, we perform a preliminary sorting work on the tensor's indices. First we store all the possible indices in the rows of an array. For example, for a second-order tensor (Hessian) with $d = 3$, we have:

$Array^{(1)}$ :

| | |
|---|---|
| 1 | 1 |
| 2 | 1 |
| 3 | 1 |
| 1 | 2 |
| 2 | 2 |
| 3 | 2 |
| 1 | 3 |
| 2 | 3 |
| 3 | 3 |

because the Hessian can be written $\mathbf{H} = \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & H_{33} \end{bmatrix}$

Then, we sort each row:

$Array^{(2)}$ :

| | |
|---|---|
| 1 | 1 |
| 1 | 2 |
| 1 | 3 |
| 1 | 2 |
| 2 | 2 |
| 2 | 3 |
| 1 | 3 |
| 2 | 3 |
| 3 | 3 |

We convert them into numbers in a first column and add a linear index on a second one:

$Array^{(3)}$ :

| | |
|---|---|
| 11 | 1 |
| 12 | 2 |
| 13 | 3 |
| 12 | 4 |
| 22 | 5 |
| 23 | 6 |
| 13 | 7 |
| 23 | 8 |
| 33 | 9 |

And we select only the unique numbers in the $Array^{(3)}$'s first column (that is why we sorted the rows before) to avoid repetitions and to obtain a unique denomination:

$Array^{(4)}$ :

| |
|---|
| 11 |
| 12 |
| 13 |
| 22 |
| 23 |
| 33 |

By combining the last 4 arrays, we can obtain all the information we need for the next step. We can store those in a cell, which first column represents the unique indices of $Array^{(4)}$. The second one informs about the variables by which $f$ is derived (from $Array^{(1)}$). And the last one accounts for the linear indices of the similar terms in the tensor (from $Array^{(3)}$). In the second row for example, the doublet $[1, 2]$ in the second column means that we shall derive $f$ with respect to $x_1$ and $x_2$, and the doublet $[2, 4]$ gives the linear coordinates of the terms $H_{12}$ and $H_{21}$ respectively (because they are equal).

The cell for $m = 2$ and $d = 3$ is:

| Unique index | Variable indices | Linear indices |
|---|---|---|
| 11 | [1,1] | [1] |
| 12 | [1,2] | [2,4] |
| 13 | [1,3] | [3,7] |
| 22 | [2,2] | [5] |
| 23 | [2,3] | [6,8] |
| 33 | [3,3] | [9] |

The cell for $m = 3$ and $d = 3$ is:

| Unique index | Variable indices | Linear indices |
|---|---|---|
| 111 | [1,1,1] | [1] |
| 112 | [1,1,2] | [2,4,10] |
| 113 | [1,1,3] | [3,7,19] |
| 122 | [1,2,2] | [5,11,13] |
| 123 | [1,2,3] | [6,8,12,16,20,22] |
| 133 | [1,3,3] | [9,21,25] |
| 222 | [2,2,2] | [14] |
| 223 | [2,2,3] | [15,17,23] |
| 233 | [2,3,3] | [18,24,26] |
| 333 | [3,3,3] | [27] |

With this preliminary work, we can derive analytically all the different terms of the tensor $\mathbf{T}_{f,m}$, and fill an empty $d \times d \cdots \times d$ array using the linear indices for the similar ones. This preliminary work avoids unnecessary derivations. Indeed, if we compute all the terms of the tensors naively, we would have to perform $d^m$ analytical computations. With this method, it is interesting to understand how many derivations we need and to compare it to the naive way.

The number of different terms in a supersymmetric tensor is equivalent to the counting problem of possible combinations with repetition: we must choose $m$ elements in a set of $d$ possibilities (where $m$ is the order and $d$ is the dimension of the $x$ variable) allowing repetitions. In facts, it can be seen in the previous tables that the terms 123, 321 or 213 are equal and correspond to the term: $\dfrac{\partial^3 f}{\partial x_1 \partial x_2 \partial x_3}$

The number of possible combination of $m$ elements within a set of $d$ choices with repetition is:

$$\binom{d+m-1}{m}$$

For $d = 3$, the number of necessary analytical derivations are given in table B.1 for several values of $m$ (derivation order). The numbers speak for themselves: it is very interesting to consider the supersymmetric properties while computing those tensors.

Table B.1: Number of necessary analytical derivations to fill a TSE tensor for $d = 3$

| **Method** | $m = 2$ | $m = 3$ | $m = 4$ | $m = 5$ | $m = 6$ | $m = 7$ | $m = 8$ | $m = 9$ | $m = 10$ |
|---|---|---|---|---|---|---|---|---|---|
| Naive | 9 | 27 | 81 | 243 | 729 | 2187 | 6561 | 19683 | 59049 |
| Smart | 6 | 10 | 15 | 21 | 28 | 36 | 45 | 55 | 66 |

## B.3. Numerical

In practice, the tensors are analytically derived using *MatLab*'s 'Symbolic Math Toolbox' and the method explained before. Functions taking the state variable as input and returning the numerical tensors are created with the *MatLab*'s function 'matlabFunction'. The convexification of the arrays can be done using the techniques explained in [55] but haven't been implemented for the reasons explained in the main text.

# C | Computational Performances

This appendix gathers the performances of the computer used for all the computations presented in this work. The latter have been performed using *MatLab*'s version R2021b with the software *CVX* [50, 51].

Table C.1: Computational Performances

| | |
|---|---|
| **Processor** | Intel Core i7 |
| **Generation** | 10$^{th}$ |
| **Model** | 10510U |
| **Cores** | 4 |
| **Max turbo frequency** | 4.90 GHz |
| **Processor frequency** | 1.80 GHz |
| **RAM** | 16 Go |

# List of Figures

# List of Tables

# List of Symbols

*Preliminary remarks:*

- *n.a. means not applicable (usually because several different units are inside the same vector, or because normalized quantities are summed)*
- *the "SI unit" column isn't representative of the units used in practice because the dynamics equations are adimensionalized from the very beginning*

| **Variable** | **Description** | **SI unit** | **Dimension** |
|---|---|---|---|
| $\mathbf{A}$ | Jacobian matrix | $n.a.$ | $[7 \times 7]$ |
| $\mathbf{B}$ | Control matrix | $n.a.$ | $[7 \times 4]$ |
| $c$ | Normalization coefficient | $N/(kg.s^2)$ | $[1 \times 1]$ |
| $d$ | Length of the state's vector | 1 | $[1 \times 1]$ |
| $D$ | Dynamical defect | $n.a.$ | $[1 \times 1]$ |
| $\delta$ | Convergence criterion | 1 | $[1 \times 1]$ |
| $\delta\mathbf{x}$ | Difference between state and reference | $n.a.$ | $[7 \times 1]$ |
| $\Delta$ | Convex dynamical defect | $n.a.$ | $[1 \times 1]$ |
| $g_0$ | Gravitational acceleration at sea level | $m/s$ | $[1 \times 1]$ |
| $\gamma$ | Shrinking trust-region parameter | 1 | $[1 \times 1]$ |
| $\mathbf{H}$ | Hessian matrix | $n.a$ | $[d \times d]$ |
| $\mathbf{I_n}$ | Identity matrix of dimension $n$ | $n.a.$ | $[n \times n]$ |
| $I_{sp}$ | Engine's specific impulse | $s$ | $[1 \times 1]$ |
| $J$ | Modified cost function | $n.a.$ | $[1 \times 1]$ |
| $J_0$ | Standard cost function | $m/s^2$ | $[1 \times 1]$ |
| $\lambda_d$ | Dynamical penalty weight | 1 | $[1 \times 1]$ |
| $\lambda_s$ | Slack variable penalty weight | 1 | $[1 \times 1]$ |
| $m$ | Spacecraft's mass | $kg$ | $[1 \times 1]$ |
| $m_0$ | Spacecraft's initial mass | $kg$ | $[1 \times 1]$ |

| **Variable** | **Description** | **SI unit** | **Dimension** |
|---|---|---|---|
| $m_f$ | Spacecraft's final mass | $kg$ | $[1 \times 1]$ |
| $\mu$ | Gravitational constant | $km^3/s^2$ | $[1 \times 1]$ |
| $N$ | Number of nodes | 1 | $[1 \times 1]$ |
| $N_{rev}$ | Number of revolutions | 1 | $[1 \times 1]$ |
| $\boldsymbol{\nu}$ | Slack variable to tackle artificial infeasibility | 1 | $[1 \times 1]$ |
| $\mathbf{p}$ | Natural dynamics vector | $n.a.$ | $[7 \times 1]$ |
| $\mathbf{p_r}$ | nonlinear part of $\mathbf{p}$ | $n.a.$ | $[3 \times 1]$ |
| $\mathbf{r}$ | Position vector | $km$ | $[3 \times 1]$ |
| $R_0$ | Distance normalization quantity | $km$ | $[1 \times 1]$ |
| $\mathrm{R}_{tr}$ | Radius of the trust-region | $n.a.$ | $[1 \times 1]$ |
| $\mathbf{s}$ | Concatenated state and control | $n.a.$ | $[11 \times 1]$ |
| $\mathbf{S}$ | Slack variable for CTSE method | $n.a.$ | $[1 \times 1]$ |
| $t_f$ | Time of Flight | $s$ | $[1 \times 1]$ |
| $\mathrm{T}$ | Thrust magnitude | $N$ | $[1 \times 1]$ |
| $\mathbf{T}$ | Thrust vector | $N$ | $[3 \times 1]$ |
| $T_0$ | Time normalization quantity | $s$ | $[1 \times 1]$ |
| $\mathbf{T}_{f,m}$ | Tensor of $m^{th}$ order derivatives of $f$ | $n.a.$ | $[7 \times \cdots \times 7]$ |
| $\mathrm{T}_{max}$ | Maximum thrust available | $N$ | $[1 \times 1]$ |
| $\tau$ | Modified thrust magnitude | $m/s^2$ | $[1 \times 1]$ |
| $\boldsymbol{\tau}$ | Modified thrust vector | $N/kg$ | $[3 \times 1]$ |
| $\mathbf{u}$ | Control variable | $m/s^2$ | $[4 \times 1]$ |
| $\mathbf{u_l}$ | Lower boundary of $\mathbf{u}$ | $m/s^2$ | $[4 \times 1]$ |
| $\mathbf{u_u}$ | Upper boundary of $\mathbf{u}$ | $m/s^2$ | $[4 \times 1]$ |
| $\mathbf{v}$ | Velocity vector | $km/s$ | $[3 \times 1]$ |
| $V_0$ | Velocity normalization quantity | $km/s$ | $[3 \times 1]$ |
| $v_e$ | Exhaust velocity | $km/s$ | $[1 \times 1]$ |
| $\mathrm{w}$ | Penalty weight (artificial infeasibility) | 1 | $[1 \times 1]$ |
| $\mathbf{x}$ | State variable | $n.a.$ | $[7 \times 1]$ |
| $\mathbf{x}^*$ | Reference solution | $n.a.$ | $[7 \times 1]$ |
| $\mathbf{x_0}$ | Initial condition | $n.a.$ | $[7 \times 1]$ |
| $\mathbf{x_f}$ | Final condition | $n.a.$ | $[7 \times 1]$ |
| $\mathbf{x_l}$ | Lower boundary of $\mathbf{x}$ | $n.a.$ | $[7 \times 1]$ |
| $\mathbf{x_u}$ | Upper boundary of $\mathbf{x}$ | $n.a.$ | $[7 \times 1]$ |
| $z$ | Natural logarithm of the mass | 1 | $[1 \times 1]$ |

# List of Acronyms

*Remark: Some acronyms are provided with reference(s) to enable a deeper understanding.*

| **Acronym** | **Meaning** | **Reference** |
|---|---|---|
| ACTSE | Alternative CTSE | eq. (3.22) |
| AU | Astronomical Unit | |
| CO | Convex Optimization | [9, 10] |
| CTSE | Convexified Taylor Series Expansion | eq. (3.16) |
| DCP | Disciplined Convex Programming | [49–51] |
| EMT | Earth-Mars Transfer | section 4.2 |
| EoM | Equations of Motion | |
| EVT | Earth-Venus Transfer | section 4.3 |
| EVT2r | EVT with 2 revolutions | section 4.3 |
| EVT3r | EVT with 3 revolutions | section 4.3 |
| FRPD | Flipped Radau Pseudospectral Discretization | [33, 34] |
| GNC | Guidance, Navigation and Control | |
| ICA | Inner-Convex Approximation | [55, 56] |
| IPM | Interior-Point Method | [21, 23] |
| LGL | Legendre-Gauss-Lobatto | [15, 31] |
| NLP | Non-Linear Programming | [7] |
| OC | Optimal Control | [3] |
| OCP | Optimal Control Problem | [3] |
| SCP | Sequential Convex Programming | [11] |
| SCvx | Successive Convexification | [18, 52] |
| SOCP | Second-Order Cone Program | [22] |
| TSE | Taylor Series Expansion | |

# Acknowledgements

First, I would like to thank Prof. Francesco Topputo for having accepted my application to perform my Master thesis at the DART Laboratory and for his supervision. I also express my deepest gratitude to Christian Hofmann and Andrea Carlo Morelli for their valuable help during this thesis, I really appreciated working with you.

This thesis concludes the 2 wonderful years I have spent in Italy for my double degree. I thank the Ecole Centrale de Nantes and Politecnico di Milano for this life-changing opportunity. I enjoyed every second spent in this lovely country and I am grateful that I had the chance to live here. I have to address a special mention to Edo, Fede and Giulia for being the best ambassadors Italy can dream of, this experience wouldn't have been the same without you.

Je te remercie, Alizée, pour m'avoir toujours poussé à donner le meilleur de moi-même. Ta présence à mes côtés a joué un rôle important dans l'accomplissement de ce double-diplôme.

Ces lignes étant les dernières que j'écris en tant qu'étudiant, je remercie mes parents, mon frère et mes grands-parents de m'avoir fourni les ressources morales et matérielles nécessaires à la réussite de ces 6 années d'études, qui n'ont pas toujours été de tout repos. L'épanouissement personnel que j'en ai tiré n'aurait pas été atteint sans votre soutien.