C. Hofmann, A.C. Morelli, F. Topputo
Performance Assessment of Convex Low-Thrust Trajectory Optimization Methods
Journal of Spacecraft and Rockets, Published online 27/10/2022
doi:10.2514/1.A35461

Performance Assessment of Convex Low-Thrust Trajectory Optimization Methods*

Christian Hofmann†, Andrea C. Morelli‡ and Francesco Topputo§
Politecnico di Milano, Milan, Italy, 20156

Different discretization and trust-region methods are compared for the low-thrust fuel-optimal trajectory optimization problem using successive convex programming. In particular, the differential and integral formulations of the adaptive pseudospectral Legendre–Gauss–Radau method, an arbitrary-order Legendre–Gauss–Lobatto technique based on Hermite interpolation, and a first-order-hold discretization are considered. The number of discretization points and segments is varied. Moreover, two hard trust-region methods and a soft trust-region strategy are compared. It is briefly discussed whether these methods, if implemented on relevant hardware, would fulfill the general requirements for onboard guidance. A perturbed cubic interpolation and the propagation of the nonlinear dynamics are used to generate initial guesses of varying quality. Interplanetary transfers to a near-Earth asteroid, Venus, and asteroid Dionysus are chosen to assess the overall performance.

Nomenclature

t	=	time
$\mathbf{r}$, $\mathbf{v}$	=	position and velocity vectors, respectively
m	=	mass
$\mathbf{T}$	=	thrust vector
$T_{\max}$	=	maximum thrust magnitude
$\mathbf{x}$, $\mathbf{u}$	=	state and control vectors, respectively
$\boldsymbol{\tau}$	=	acceleration vector due to thrust
Γ	=	magnitude of $\boldsymbol{\tau}$
$\mathbf{f}$	=	dynamics of the problem
$\bar{(\cdot)}$	=	reference value of quantity $(\cdot)$
$\mathbf{A}$, $\mathbf{B}$	=	Jacobian matrices related to states and controls, respectively

*Part of this work was presented as paper AIAA 2022-1892 at the 2022 AIAA SciTech Forum.
†Ph.D. Candidate, Department of Aerospace Science and Technology, Via La Masa 34, 20156 Milan, Italy. Email: christian.hofmann@polimi.it.
‡Ph.D. Candidate, Department of Aerospace Science and Technology, Via La Masa 34, 20156 Milan, Italy. Email: andreacarlo.morelli@polimi.it.
§Professor, Department of Aerospace Science and Technology, Via La Masa 34, 20156 Milan, Italy. Email: francesco.topputo@polimi.it.

$\boldsymbol{\nu}, \eta$ = slack variables

R = trust-region radius

K = number of trajectory segments

N_k = number of nodes of trajectory segment k in pseudospectral methods

$\mathbf{D}, \mathbf{I}$ = differentiation and integration matrices in pseudospectral methods, respectively

$\mathbf{1}$ = identity matrix

n_x, n_u = number of states and controls, respectively

$\mathbf{a}, \mathbf{a_u}$ = coefficient vectors for states and controls in Legendre–Gauss–Lobatto method, respectively

n_p, n_c = number of nodes and collocation points per segment in Legendre–Gauss–Lobatto method, respectively

$\boldsymbol{\Phi}$ = state transition matrix

ρ = ratio of actual and predicted cost decrease

α, β = trust-region shrinking and growing rates, respectively

δ = adjustment parameter in hard trust-region methods

λ_{TR} = penalty parameter in soft trust-region method

N = total number of discretization points

Subscripts

0 = initial value

f = final value

l = lower bound

u = upper bound

Acronyms

PDG = powered descent guidance

LTO = low-thrust trajectory optimization

SCP = sequential convex programming

ZOH = zero-order hold

FOH = first-order hold

LGL = Legendre–Gauss–Lobatto

LG = Legendre–Gauss

RPM = Radau pseudospectral method

LGR = Legendre–Gauss–Radau

FRPM = flipped Radau pseudospectral method

I. Introduction

The number of new space missions has grown rapidly, and especially interplanetary CubeSats are becoming increasingly important as the success of NASA's MarCO mission has shown [1]. Their low development costs make them an appealing option for many missions. Although the computational capability of onboard computers has increased continuously in the past years, only minor advances in the guidance and control systems have taken place. It is therefore not surprising that a paradigm shift is currently happening [2]. Rather than calculating the guidance and control actions on ground, these tasks shall be performed on board to reduce the operational costs even further [3].

Although feasibility is often sufficient for many applications, fuel consumption is of utmost importance for space flight. The costs to launch a satellite still amount to several thousand dollars per kilogram, and decreasing the launch mass can contribute to reducing the overall mission cost [4]. In addition, spacecraft (especially CubeSats) have severe limitations regarding the propellant mass. Therefore, it is desirable to not only compute feasible, but also (near-)optimal trajectories. This, however, requires solving a nonlinear optimal control problem. Direct and indirect methods are most commonly used to find a solution [5]. Given the requirements for onboard applications, such as high reliability and low computational effort, indirect methods play a rather secondary role, often due to the small convergence domain [6]. On the contrary, the recent developments of convex programming techniques have made direct methods a promising approach for real-time guidance [7–9]. The successive optimization method allows solving nonconvex optimal control problems with nonlinear dynamics and constraints, therefore being a viable alternative for many aerospace applications such as powered descent guidance (PDG) [10] and low-thrust trajectory optimization (LTO) [11, 12]. Solving a series of simpler, convex subproblems makes this method numerically tractable compared to solving a nonlinear program directly. Two key characteristics of this so-called sequential convex programming technique (SCP) are the discretization and trust-region methods; they strongly affect the results, especially the convergence properties. Yet, previous research activities lack a thorough assessment and comparison of relevant techniques for low-thrust trajectory optimization.

The most popular discretization techniques are collocation and control interpolation methods. The former parameterize both the states and controls using some basis functions, whereas the latter parameterize only the control history [5]. Due to its simplicity, the trapezoidal rule is one of the most important collocation methods [13–15]. Yet, its poor accuracy often prevents the solver to find solutions that satisfy the nonlinear dynamics for long-duration interplanetary transfers. Higher-order methods such as Hermite–Simpson collocation offer instead a good compromise between computational effort and accuracy [16]. The arbitrary-order Hermite–Legendre–Gauss–Lobatto discretization is a generalization of this method and a popular choice in nonlinear optimization as it allows approximating the states with higher-order Hermite interpolating polynomials [17]. In global pseudospectral methods, in contrast, single Lagrange interpolating polynomials are used to approximate the states and controls, respectively. As this results in dense matrices, adaptive methods were developed where piecewise polynomials are used to approximate the trajectories [18]. There are several categories of pseudospectral methods; the most important ones are based on Legendre–Gauss–Lobatto

(LGL) [17], Legendre–Gauss (LG) [19], or Legendre–Gauss–Radau (LGR) points [20]. Several works adapted collocation methods to convex programming and solved powered descent [21, 22] and low-thrust guidance problems [11, 12, 23]. With regard to control interpolation methods, the control is approximated using a zero-order-hold (ZOH) or first-order-hold (FOH) discretization. For ZOH, the control history is assumed piecewise constant, whereas for FOH, it is approximated as a piecewise affine function [24]. Both methods performed well for powered descent guidance problems [10, 24]. The ZOH discretization was also applied to low-thrust trajectory optimization problems in the circular restricted three-body problem [25]. However, it is yet to be investigated how FOH performs in LTO.

The work in [26] compares different discretization methods for the PDG problem. However, the results cannot be extended directly to the low-thrust trajectory optimization problem due to the different dynamics, constraints, and number of switching times for fuel-optimal problems. One major difference is that the time horizon is considerably shorter in PDG compared to the long-duration interplanetary transfers in LTO which can last several years. Therefore, many more nodes are required to capture the state and control profiles accurately, especially if the number of revolutions increases. Moreover, a global pseudospectral method using only one high-order polynomial as in [26] would result in dense matrices and a considerable higher solving time (if a solution is found at all). For this reason, the differential and integral formulations of an adaptive pseudospectral method developed by the authors are considered in this work. Standard and flipped Legendre–Gauss–Radau points are used because they are a natural choice in an adaptive framework due to the definition of the collocation points. Finally, the work in [26] only compared control interpolation methods (ZOH, FOH, and Runge-Kutta) and pseudospectral methods. By considering the Hermite interpolation-based collocation, pseudospectral methods, and the control interpolation method FOH, we believe that our selection covers the most important and relevant discretization methods for solving the LTO problem.

The second part of this paper assesses different trust-region methods. Trust regions are imposed to keep the linearization close to a reference solution. Most of the SCP methods found in literature use some type of trust-region approach. For powered descent guidance problems, soft trust regions (where the constraint is penalized in the cost function) are often used [24, 27]. In low-thrust trajectory optimization, simple hard trust regions (where the trust-region constraint is imposed directly) are more common [11, 13]. The choice of the trust-region approach is often crucial as this can decide whether a feasible solution is found or not. Furthermore, a poor choice of the parameters can deteriorate the convergence. Such a behavior is undesirable and unacceptable for onboard applications. Soft trust regions often work well in powered descent guidance problems, but they have not been used to determine low-thrust trajectories so far.

The contribution of this paper is threefold: first, none of the existing works has investigated how different trust-region methods and their parameters affect the performance of the SCP algorithm for complex interplanetary low-thrust fuel-optimal transfers. Moreover, no conclusion can be drawn on how the choice might affect the real-time capability of the algorithm. For this reason, we compare the performance of two different hard trust-region methods (standard and adaptive [11, 12]), and a modified soft trust-region method. Secondly, a convex formulation of the integral form of the

adaptive Radau pseudospectral method (RPM) is presented for the first time and compared with existing methods such as FOH, LGL, and the differential form of RPM. Lastly, to the best of the authors' knowledge, this is the first time that such an extensive assessment is carried out where the influence of discretization and trust-region methods, different orders of the interpolating polynomial, number of nodes, and initial guesses on the performance is analyzed at the same time. Additionally, general requirements for onboard guidance applications are discussed.

The paper is structured as follows. Section II states the optimal control problem and the convexification. Section III describes the discretization methods, and Section IV presents the trust-region methods. The results are presented and discussed in Section V. Section VI concludes this paper.

II. Problem Formulation

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

$$\dot{\mathbf{r}}(t) = \mathbf{v}(t) \tag{1}$$

$$\dot{\mathbf{v}}(t) = -\frac{\mu\,\mathbf{r}(t)}{\|\mathbf{r}(t)\|_2^3} + \frac{\mathbf{T}(t)}{m(t)} \tag{2}$$

$$\dot{m}(t) = -\frac{\|\mathbf{T}(t)\|_2}{g_0\, I_{\mathrm{sp}}} \tag{3}$$

where $\mathbf{r}(t) \in \mathbb{R}^{3\times1}$, $\mathbf{v}(t) \in \mathbb{R}^{3\times1}$, and $m(t) \in \mathbb{R}$ denote the position, velocity, and mass of the spacecraft, respectively. μ is the gravitational parameter of the primary body and g_0 denotes the gravitational acceleration at sea level. The control actions are governed by the thrust components $\mathbf{T}(t) \in \mathbb{R}^{3\times1}$, $I_{\mathrm{sp}} \in \mathbb{R}$ being the specific impulse.

We seek to minimize fuel usage, which is equivalent to maximizing the mass at the final time t_f:

$$\underset{\mathbf{T}(t)}{\text{minimize}} \quad -m(t_f) \tag{4}$$

In this work, we intend to target a specific point $\left[\mathbf{r}_f^\top, \mathbf{v}_f^\top\right]^\top$ in space. Therefore, the boundary conditions at the initial t_0 and final t_f times are

$$\mathbf{r}(t_0) = \mathbf{r}_0, \quad \mathbf{v}(t_0) = \mathbf{v}_0, \quad m(t_0) = m_0 \tag{5}$$

$$\mathbf{r}(t_f) = \mathbf{r}_f, \quad \mathbf{v}(t_f) = \mathbf{v}_f \tag{6}$$

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

$$0 \leq T_{\min} \leq \|\mathbf{T}(t)\|_2 \leq T_{\max} \tag{7}$$

with the minimum $T_{\min}$ and maximum $T_{\max}$ available thrust magnitudes. Note that $T_{\min} = 0$ is used throughout this paper.

The nonlinear part $\mathbf{T}(t)/m(t)$ in Eq. (2) is eliminated by a change of variables [10]:

$$\Gamma(t) \coloneqq \frac{\|\mathbf{T}(t)\|_2}{m(t)}, \quad \boldsymbol{\tau}(t) \coloneqq \frac{\mathbf{T}(t)}{m(t)}, \quad z(t) \coloneqq \ln m(t) \tag{8}$$

As the thrust constraint in Eq. (7) becomes nonconvex now, it is linearized about the reference $\bar{z}$:

$$0 \leq \Gamma(t) \leq T_{\max} \mathrm{e}^{-\bar{z}} \left(1 - z(t) + \bar{z}(t)\right) \tag{9}$$

Moreover, the original nonconvex constraint $\|\boldsymbol{\tau}(t)\|_2 = \Gamma(t)$ is relaxed to $\|\boldsymbol{\tau}(t)\|_2 \leq \Gamma(t)$. It can be shown that the solution of the relaxed problem is also an optimal solution of the original problem [10]. Defining the states and controls as $\mathbf{x} = [\mathbf{r}^\top, \mathbf{v}^\top, z]^\top$ and $\mathbf{u} = [\boldsymbol{\tau}^\top, \Gamma]^\top$, respectively, the new dynamics are

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) = \begin{bmatrix} \mathbf{v}(t) \\ -\mu\, \mathbf{r}(t)/\|\mathbf{r}(t)\|_2^3 + \boldsymbol{\tau}(t) \\ -\Gamma(t)/(g_0\, I_{\mathrm{sp}}) \end{bmatrix} \tag{10}$$

Linearizing Eq. (10) about a reference solution $\bar{\mathbf{x}}$ and adding a trust-region constraint to keep the linearization valid, the convexified optimization problem is stated as follows:

$$\underset{\mathbf{u}(t)}{\text{minimize}} \quad -z(t_f) + \lambda\ \|\boldsymbol{\nu}(t)\|_1 + \lambda\ \max(0, \eta(t)) \tag{11a}$$

$$\text{subject to:} \quad \dot{\mathbf{x}}(t) = \mathbf{A}(\bar{\mathbf{x}}(t))\,\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) + \mathbf{q}(\bar{\mathbf{x}}(t), \bar{\mathbf{u}}(t)) + \boldsymbol{\nu}(t) \tag{11b}$$

$$\Gamma(t) \leq T_{\max} \mathrm{e}^{-\bar{z}(t)} \left(1 - z(t) + \bar{z}(t)\right) + \eta(t) \tag{11c}$$

$$\|\boldsymbol{\tau}(t)\|_2 \leq \Gamma(t) \tag{11d}$$

$$\|\mathbf{x}(t) - \bar{\mathbf{x}}(t)\|_1 \leq R \tag{11e}$$

$$\mathbf{r}(t_0) = \mathbf{r}_0,\ \mathbf{v}(t_0) = \mathbf{v}_0,\ z(t_0) = z_0 \tag{11f}$$

$$\mathbf{r}(t_f) = \mathbf{r}_f,\ \mathbf{v}(t_f) = \mathbf{v}_f \tag{11g}$$

$$\mathbf{x}_l \leq \mathbf{x}(t) \leq \mathbf{x}_u,\ \mathbf{u}_l \leq \mathbf{u}(t) \leq \mathbf{u}_u \tag{11h}$$

where

$$\mathbf{A}(\bar{\mathbf{x}}(t)) \coloneqq \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{\bar{\mathbf{x}}(t)},\ \mathbf{B} \coloneqq \left.\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\right|_{\bar{\mathbf{u}}(t)},\ \mathbf{q}(\bar{\mathbf{x}}(t), \bar{\mathbf{u}}(t)) \coloneqq \mathbf{f}(\bar{\mathbf{x}}(t), \bar{\mathbf{u}}(t)) - \mathbf{A}(\bar{\mathbf{x}}(t))\,\bar{\mathbf{x}}(t) - \mathbf{B}\bar{\mathbf{u}}(t) \tag{12}$$

Note that Eqs. (11b) and (11c) are augmented with slack variables $\nu(t)$ and $\eta(t) \geq 0$ to avoid artificial infeasibility. They are penalized in the cost function in Eq. (11a) with a sufficiently large parameter $\lambda > 0$. The trust-region constraint in Eq. (11e) with radius R is imposed to keep the linearization close to the reference. Equation (11h) represents the upper and lower bounds on the state and control variables, respectively.

III. Discretization Methods

The following discretization methods are considered in this work [28]:

1) An adaptive Legendre–Gauss–Radau pseudospectral method.
2) An arbitrary-order Legendre–Gauss–Lobatto method based on Hermite interpolation.
3) A first-order-hold discretization.

The RPM has demonstrated to perform well for nonlinear programs and also low-thrust trajectory design within convex programming [11, 18]. Moreover, as either the initial or final point is not collocated, it is an appropriate choice for an adaptive framework as there is no redundancy or even lack of nodes between consecutive segments compared to other pseudospectral methods that are based on Legendre–Gauss or Legendre–Gauss–Lobatto points [29]. We also consider the arbitrary-order Legendre–Gauss–Lobatto method based on Hermite interpolation as it is a generalization of the well-known Hermite-Simpson collocation. It therefore covers a wide range of methods that have proven effective to solve nonlinear programs [30]. FOH is chosen as it belongs to the class of control interpolation techniques. Note that the zero-order-hold discretization is not considered due to the poor approximation of the thrust profile.

A. Adaptive Legendre–Gauss–Radau Pseudospectral Method

In an adaptive pseudospectral method, the trajectory is divided into K segments and the states and controls are approximated using Lagrange interpolating polynomials of arbitrary degrees. The collocation points are defined as the roots of the polynomial $P_{N-1}(\xi) + P_N(\xi)$ where P_N is the Nth degree Legendre polynomial. These points are defined in the pseudospectral time domain $\xi \in [-1, 1]$. The transformation between the physical t and pseudospectral time is given by [31]

$$t_i^{(k)} = \frac{t_{N_k}^{(k)} - t_0^{(k)}}{2} \xi_i^{(k)} + \frac{t_{N_k}^{(k)} + t_0^{(k)}}{2} \qquad i = 0, 1, ..., N_k \tag{13}$$

Throughout this section, the number of collocation points per segment (and hence, the degree of the interpolating polynomial) is denoted as N_k, and $\mathbf{x}_i^{(k)}$, $\mathbf{u}_i^{(k)}$ refer to the ith point of the kth segment of states and controls at time $t_i^{(k)}$, with $i = 0, 1, ..., N_k$ and $k = 1, ..., K$. The states and controls are approximated in the interval $[-1, 1]$ as follows:

$$\mathbf{x}^{(k)}(\xi) = \sum_{i=0}^{N_k} \mathbf{x}_i^{(k)} L_i^{(k)}(\xi), \qquad \mathbf{u}^{(k)}(\xi) = \sum_{i=0}^{N_k} \mathbf{u}_i^{(k)} L_i^{(k)}(\xi) \tag{14}$$

with

$$L_i^{(k)}(\xi) = \prod_{\substack{j=0 \\ j\neq i}}^{N_k} \frac{\xi - \xi_j}{\xi_i - \xi_j} \tag{15}$$

Depending on the chosen method, the initial or final point may or may not be included in the interpolation. The basic idea of pseudospectral methods is to approximate the differential operator as $\dot{\mathbf{x}}^{(k)} \approx \mathbf{D}^{(k)}\mathbf{x}^{(k)}$ where $\mathbf{D}^{(k)} \in \mathbb{R}^{N_k \times (N_k+1)}$ is a non-square differentiation matrix whose elements are defined as $D_{ij} \coloneqq L'_j(\xi_i)$ [32].

This work presents two formulations of the adaptive Radau pseudospectral method:

1) A Radau pseudospectral method using the standard Legendre–Gauss–Radau points.

2) A flipped Radau pseudospectral method (FRPM) using the flipped LGR points.

We dedicate individual sections to each method as they differ in several aspects. The differential and integral formulations of each method are considered. RPM and the integral form of FRPM are presented for the first time.

1. Adaptive Radau Pseudospectral Method

The dynamics are approximated at the LGR points defined on $[-1, 1)$. Therefore, the last node of each segment is not a collocation point, but it is used to approximate the state. Note, however, that the final control $\mathbf{u}_{N_K}^{(K)}$ is not obtained in the solution process and must therefore be determined by extrapolation. Using the differentiation matrix $\mathbf{D}^{(k)}$, the dynamics can be written in differential form as follows [33]:

$$\begin{aligned} \sum_{j=0}^{N_k} D_{ij}^{(k)} \mathbf{x}_j^{(k)} &= \frac{t_{N_k}^{(k)} - t_0^{(k)}}{2} \mathbf{f}(\mathbf{x}_i^{(k)}, \mathbf{u}_i^{(k)}) \\ &= \frac{t_{N_k}^{(k)} - t_0^{(k)}}{2} \left[\mathbf{A}(\bar{\mathbf{x}}_{\mathbf{i}}^{(\mathbf{k})}) \mathbf{x}_i^{(k)} + \mathbf{B}\mathbf{u}_i^{(k)} + \mathbf{q}(\bar{\mathbf{x}}_i^{(k)}) + \boldsymbol{\nu}_i^{(k)} \right], \qquad i = 0, 1, ..., N_k - 1 \end{aligned} \tag{16}$$

The factor $\Delta \coloneqq \frac{t_{N_k}^{(k)} - t_0^{(k)}}{2}$ is needed due to the time transformation from the physical time to $[-1, 1]$. As the final point of each segment is not collocated, the linking condition $\mathbf{x}_{N_k}^{(k)} = \mathbf{x}_0^{(k+1)}$ must hold for segments $k < K$. The dynamics can then be written in standard form as a linear equality constraint:

$$\left[\begin{array}{ccc:ccc:ccc} \hat{\mathbf{A}}^{(1)} & & & \hat{\mathbf{B}}^{(1)} & & & \mathbf{1}^{(1)} & & \\ & \ddots & \mathbf{0} & & \ddots & \mathbf{0} & & \ddots & \mathbf{0} \\ \mathbf{0} & & & \mathbf{0} & & & \mathbf{0} & & \\ & & \hat{\mathbf{A}}^{(K)} & & & \hat{\mathbf{B}}^{(K)} & & & \mathbf{1}^{(K)} \end{array}\right] \cdot \begin{bmatrix} \mathbf{X} \\ \mathbf{U} \\ \boldsymbol{\nu} \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{q}}^{(1)} \\ \vdots \\ \hat{\mathbf{q}}^{(K)} \end{bmatrix} \tag{17}$$

where the matrices $\hat{\mathbf{A}}^{(k)}$ and $\hat{\mathbf{B}}^{(k)}$ are calculated using $\mathbf{D}^{(k)}$ and the Jacobian matrices defined in Eq. (12). $\mathbf{1}^{(k)}$ are identity matrices and $\hat{\mathbf{q}}^{(k)}$ are comprised of the constant parts of the linearization. $\mathbf{X}$, $\mathbf{U}$, and $\boldsymbol{\nu}$ denote the concatenated

states, controls, and virtual controls, respectively. The $\hat{\mathbf{A}}^{(k)}$ take the following form

$$\hat{\mathbf{A}}^{(k)} = \begin{bmatrix} D_{00}\,\mathbf{1} - \Delta\,\mathbf{A}_0 & D_{01}\,\mathbf{1} & \dots & D_{0,N_k-1}\,\mathbf{1} & D_{0,N_k}\,\mathbf{1} \\ D_{10}\,\mathbf{1} & D_{11}\,\mathbf{1} - \Delta\,\mathbf{A}_1 & \dots & D_{1,N_k-1}\,\mathbf{1} & D_{1,N_k}\,\mathbf{1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ D_{N_k-1,0}\,\mathbf{1} & \dots & & D_{N_k-1,N_k-1}\,\mathbf{1} - \Delta\,\mathbf{A}_{N_k-1} & D_{N_k-1,N_k}\,\mathbf{1} \end{bmatrix} \tag{18}$$

where we omitted $(\cdot)^{(k)}$ for the sake of conciseness. $\hat{\mathbf{B}}^{(k)}$ are diagonal matrices with entries $-\Delta\,\mathbf{B}_i$, and $\hat{\mathbf{q}}^{(k)}$ is a concatenated vector where the elements take the form $\Delta\,\mathbf{q}_i$.

It was observed that an equivalent integral formulation of Eq. (16) may yield more consistent results for solving nonlinear programs [34]. We extend this approach to our convex optimization framework: defining an integration matrix $\mathbf{I} \in \mathbb{R}^{N_k \times N_k}$ as $\mathbf{I} \coloneqq \mathbf{D}_{1:N_k}^{-1}$ where $\mathbf{D}_{1:N_k}$ is obtained by removing the first column of $\mathbf{D}$, the dynamics are

$$\begin{aligned} \mathbf{x}_{i+1}^{(k)} &= \mathbf{x}_0^{(k)} + \frac{t_{N_k}^{(k)} - t_0^{(k)}}{2} \sum_{j=0}^{N_k-1} I_{ij}^{(k)} \mathbf{f}(\mathbf{x}_j^{(k)}, \mathbf{u}_j^{(k)}) \\ &= \mathbf{x}_0^{(k)} + \frac{t_{N_k}^{(k)} - t_0^{(k)}}{2} \sum_{j=0}^{N_k-1} I_{ij}^{(k)} \left[\mathbf{A}(\bar{\mathbf{x}}_j^{(k)}) \mathbf{x}_j^{(k)} + \mathbf{B}\mathbf{u}_j^{(k)} + \mathbf{q}(\bar{\mathbf{x}}_j^{(k)}) + \boldsymbol{\nu}_j^{(k)} \right] \qquad i = 0, ..., N_k - 1 \end{aligned} \tag{19}$$

The $\hat{\mathbf{A}}_{\text{int}}^{(k)}$ are now given as

$$\hat{\mathbf{A}}_{\text{int}}^{(k)} = \begin{bmatrix} -\mathbf{1} - \Delta\, I_{00}\mathbf{A}_0 & \mathbf{1} - \Delta\, I_{01}\mathbf{A}_1 & -\Delta\, I_{02}\mathbf{A}_2 & \dots & -\Delta\, I_{0,N_k-1}\mathbf{A}_{N_k-1} & \mathbf{0} \\ -\mathbf{1} - \Delta\, I_{10}\mathbf{A}_0 & -\Delta\, I_{11}\mathbf{A}_1 & -\mathbf{1} - \Delta\, I_{12}\mathbf{A}_2 & \dots & -\Delta\, I_{1,N_k-1}\mathbf{A}_{N_k-1} & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ -\mathbf{1} - \Delta\, I_{N_k-2,0}\mathbf{A}_0 & -\Delta\, I_{N_k-2,1}\mathbf{A}_1 & -\Delta\, I_{N_k-2,2}\mathbf{A}_2 & \dots & -\mathbf{1} - \Delta\, I_{N_k-2,N_k-1}\mathbf{A}_{N_k-1} & \mathbf{0} \\ -\mathbf{1} - \Delta\, I_{N_k-1,0}\mathbf{A}_0 & -\Delta\, I_{N_k-1,1}\mathbf{A}_1 & -\Delta\, I_{N_k-1,2}\mathbf{A}_2 & \dots & -\Delta\, I_{N_k-1,N_k-1}\mathbf{A}_{N_k-1} & \mathbf{1} \end{bmatrix} \tag{20}$$

$\hat{\mathbf{B}}_{\text{int}}^{(k)}$ and $\hat{\mathbf{q}}_{\text{int}}^{(k)}$ can be calculated in a similar way using Eq. (19).

As the initial $\mathbf{x}_0^{(1)}$ and final $\mathbf{x}_{N_K}^{(K)}$ states are included in the optimization, initial and final boundary conditions can be imposed as simple equality constraints.

2. Adaptive Flipped Radau Pseudospectral Method

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

$$\tilde{\theta} = \text{sort}(-\theta) \tag{21}$$

where *sort* sorts the values in ascending order. Consequently, the initial control is not part of the optimization process. In contrast to RPM, we do not include the initial node in the state approximation this time. Rather, we make use of the fact that $\mathbf{x}_0^{(1)}$ is equal to the initial boundary condition $\mathbf{x}_0$. The dynamics in differential form then read [11]

$$D_{i0}^{(k)}\mathbf{x}_0^{(k)} + \sum_{j=1}^{N_k} D_{ij}^{(k)}\mathbf{x}_j^{(k)} = \frac{t_{N_k}^{(k)} - t_0^{(k)}}{2}\left[\mathbf{A}(\bar{\mathbf{x}}_\mathbf{i}^{(\mathbf{k})})\mathbf{x}_i^{(k)} + \mathbf{B}(\bar{\mathbf{x}}_\mathbf{i}^{(\mathbf{k})})\mathbf{u}_i^{(k)} + \mathbf{q}(\bar{\mathbf{x}}_i^{(k)}) + \boldsymbol{\nu}_i^{(k)}\right] \qquad i = 1, ..., N_k \tag{22}$$

where $\mathbf{x}_0^{(k)}$ is the initial state of each segment, and $\mathbf{x}_0^{(1)} = \mathbf{x}_0$. The integral form is

$$\mathbf{x}_i^{(k)} = \mathbf{x}_0^{(k)} + \frac{t_{N_k}^{(k)} - t_0^{(k)}}{2}\sum_{j=1}^{N_k} I_{ij}^{(k)}\left[\mathbf{A}(\bar{\mathbf{x}}_j^{(k)})\mathbf{x}_j^{(k)} + \mathbf{B}\mathbf{u}_j^{(k)} + \mathbf{q}(\bar{\mathbf{x}}_j^{(k)}) + \boldsymbol{\nu}_j^{(k)}\right] \qquad i = 1, ..., N_k \tag{23}$$

The dynamics can again be formulated as a single constraint. The interested reader is referred to [11] for details on the differential form. We present only the integral formulation as it has not been reported in the literature:

$$\hat{\mathbf{A}}_{\text{int,FRPM}}^{(k)} = \begin{bmatrix} \mathbf{1} - \Delta\, I_{11}\mathbf{A}_1 & -\Delta\, I_{12}\mathbf{A}_1 & \dots & -\Delta\, I_{1,N_k}\mathbf{A}_{N_k} \\ -\Delta\, I_{21}\mathbf{A}_1 & \mathbf{1} - \Delta\, I_{22}\mathbf{A}_2 & \dots & -\Delta\, I_{2,N_k}\mathbf{A}_{N_k} \\ \vdots & \vdots & \ddots & \vdots \\ -\Delta\, I_{N_k,1}\mathbf{A}_1 & -\Delta\, I_{N_k,2}\mathbf{A}_2 & \dots & \mathbf{1} - \Delta\, I_{N_k,N_k}\mathbf{A}_{N_k} \\ \mathbf{0} & \mathbf{0} & \dots & -\tilde{\mathbf{1}} \end{bmatrix}, \quad \hat{\mathbf{q}}_{\text{int,FRPM}}^{(k)} = \begin{bmatrix} \Delta \sum_{j=1}^{N_k} I_{1j}\mathbf{q}(\bar{\mathbf{x}}_j) \\ \vdots \\ \Delta \sum_{j=1}^{N_k} I_{N_k,j}\mathbf{q}(\bar{\mathbf{x}}_j) \end{bmatrix} \tag{24}$$

where $(\cdot)^{(k)}$ was again omitted. $\tilde{\mathbf{1}} \in \mathbb{R}^{N_k n_x \times n_x}$ consists of vertically concatenated identity matrices, $n_x = 7$ being the number of states. For $k = 1$, a concatenated vector of the initial states, i.e. $[\mathbf{x}_0^\top, \mathbf{x}_0^\top, \dots]^\top$, is to be added to $\hat{\mathbf{q}}_{\text{int,FRPM}}^{(k)}$ in Eq. (24) to account for the non-collocated initial node. The structure of $\hat{\mathbf{B}}_{\text{int,FRPM}}^{(k)}$ is similar to $\hat{\mathbf{A}}_{\text{int,FRPM}}^{(k)}$.

B. Arbitrary-Order Legendre–Gauss–Lobatto Method

The arbitrary-order Legendre–Gauss–Lobatto discretization method relies on Hermite interpolation [35]. The idea is to use the information of the states and the dynamics at the nodal points and express the constraints at the collocation points by approximating the state variables with arbitrary-order polynomials in each segment [35, 36]. The total time of

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

$$t \;\rightarrow\; \frac{h}{2}\xi + \frac{t_{k+1} + t_k}{2}, \quad k = 1, \ldots, K-1 \tag{25}$$

where $\xi \in [-1, 1]$ and $h = t_{k+1} - t_k$ is the time step. In this work, nodes and collocation points are defined inside the interval $[-1, 1]$ as the roots of the derivative of the $(n-1)$th order Legendre polynomial [36], where n is the order of the method. Given n, the state $\mathbf{x}^{(k)}(\xi) \in \mathbb{R}^{n_x \times 1}$ ($n_x = 7$) is approximated inside the kth segment as

$$\mathbf{x}^{(k)}(\xi) \approx \mathbf{a}_0^{(k)} + \mathbf{a}_1^{(k)}\xi + \cdots + \mathbf{a}_n^{(k)}\xi^n, \quad k = 1, \ldots, K \tag{26}$$

where the column vectors of coefficients $\mathbf{a}_m^{(k)} \in \mathbb{R}^{n_x \times 1}$, $m = 0, \ldots, n$ are unknowns that are found by solving the following linear system:

$$\underbrace{\begin{bmatrix} \mathbf{1}_{n_x} & \theta_1 \mathbf{1}_{n_x} & \theta_1^2 \mathbf{1}_{n_x} & \ldots & \theta_1^n \mathbf{1}_{n_x} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{1}_{n_x} & \theta_{n_p} \mathbf{1}_{n_x} & \theta_{n_p}^2 \mathbf{1}_{n_x} & \ldots & \theta_{n_p}^n \mathbf{1}_{n_x} \\ \mathbf{0}_{n_x} & \mathbf{1}_{n_x} & 2\theta_1 \mathbf{1}_{n_x} & \ldots & n\theta_1^{n-1} \mathbf{1}_{n_x} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{0}_{n_x} & \mathbf{1}_{n_x} & 2\theta_{n_p} \mathbf{1}_{n_x} & \ldots & n\theta_{n_p}^{n-1} \mathbf{1}_{n_x} \end{bmatrix}}_{\boldsymbol{\theta}} \underbrace{\begin{bmatrix} \mathbf{a}_0^{(k)} \\ \vdots \\ \mathbf{a}_{n_p}^{(k)} \\ \vdots \\ \mathbf{a}_{n-1}^{(k)} \\ \mathbf{a}_n^{(k)} \end{bmatrix}}_{\mathbf{a}^{(k)}} = \underbrace{\begin{bmatrix} \mathbf{x}^{(k)}(\theta_1) \\ \vdots \\ \mathbf{x}^{(k)}(\theta_{n_p}) \\ \frac{h}{2}\mathbf{f}_l^{(k)}(\theta_1) \\ \vdots \\ \frac{h}{2}\mathbf{f}_l^{(k)}(\theta_{n_p}) \end{bmatrix}}_{\mathbf{b}^{(k)}} \tag{27}$$

In Eq. (27), θ_j are the positions of the nodal points, $n_p = (n+1)/2$ is the number of nodes in each segment, $\mathbf{1}_{n_x}$ is the $n_x \times n_x$ identity matrix, $\mathbf{0}_{n_x}$ the $n_x \times n_x$ null matrix, and $\mathbf{f}_l(\theta_j)$ the linearized dynamics as in Eq (11b). Once the coefficients $\mathbf{a}_m^{(k)}$ have been determined as $\mathbf{a}^{(k)} = \boldsymbol{\theta}^{-1}\mathbf{b}^{(k)}$, Eq. (26) can be used to define the state at the collocation points:

$$\mathbf{x}^{(k)}(\zeta) = \underbrace{\begin{bmatrix} \mathbf{1}_{n_x} & \zeta_1 \mathbf{1}_{n_x} & \ldots & \zeta_1^n \mathbf{1}_{n_x} \\ \mathbf{1}_{n_x} & \zeta_2 \mathbf{1}_{n_x} & \ldots & \zeta_2^n \mathbf{1}_{n_x} \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{1}_{n_x} & \zeta_{n_c} \mathbf{1}_{n_x} & \ldots & \zeta_{n_c}^n \mathbf{1}_{n_x} \end{bmatrix}}_{\boldsymbol{\zeta}} \underbrace{\begin{bmatrix} \mathbf{a}_0^{(k)} \\ \mathbf{a}_1^{(k)} \\ \vdots \\ \mathbf{a}_n^{(k)} \end{bmatrix}}_{\mathbf{a}^{(k)}} = \boldsymbol{\zeta}\boldsymbol{\theta}^{-1}\mathbf{b}^{(k)} = \boldsymbol{\phi}\mathbf{b}^{(k)} \tag{28}$$

where ζ_j are the positions of the collocation points, and $n_c = (n-1)/2$ is the number of collocation points within each segment. The derivative of the state at the collocation points can be found in a similar fashion by deriving the matrix $\boldsymbol{\zeta}$.

Similarly, the control $\mathbf{u}^{(k)}(\xi) \in \mathbb{R}^{n_u \times 1}$ ($n_u = 4$) is approximated in each segment as

$$\mathbf{u}^{(k)}(\xi) \approx \mathbf{a}_{u,0}^{(k)} + \mathbf{a}_{u,1}^{(k)}\xi + \cdots + \mathbf{a}_{u,n}^{(k)}\xi^{n_p-1}, \quad k = 1, \ldots, K \tag{29}$$

where the column vectors of coefficients $\mathbf{a}_{u,m}^{(k)} \in \mathbb{R}^{n_u \times 1}$, $m = 0, \ldots, n$ are unknowns, obtained in a similar fashion as for the coefficients $\mathbf{a}_m^{(k)}$ inside Eq. (27). Note, however, that for the control no information about its dynamics is available and thus only the first n_p rows of the system can be considered. For this reason, the control is approximated by means of a polynomial of order $n_p - 1$. The quantities $\boldsymbol{\phi}_u$ and $\mathbf{b}_u^{(k)}$ are defined accordingly. Once the matrices and vectors of all trajectory segments are computed, the dynamical constraints can be written as

$$\boldsymbol{\Delta} = \boldsymbol{\Phi}'\hat{\mathbf{b}} - \frac{h}{2}[\hat{\mathbf{f}}_f + \hat{\mathbf{A}}(\boldsymbol{\Phi}\hat{\mathbf{b}} - \boldsymbol{\Phi}\hat{\mathbf{b}}^*) + \hat{\mathbf{B}}\boldsymbol{\Phi}_u\hat{\mathbf{b}}_u] = \mathbf{0} \tag{30}$$

where the capital letters and $\hat{(\cdot)}$ indicate the concatenated quantities, and $\hat{\mathbf{f}}_f$ denotes the assembled free dynamics of the spacecraft. For a detailed explanation of the method, the interested reader is referred to [12].

C. First-Order-Hold Method

Given N nodes, the time horizon is divided into $N - 1$ equidistant segments with

$$t_0 = t_1 < t_2 < \cdots < t_N = t_f \tag{31}$$

The control history $\mathbf{u}(t)$ is approximated as a piecewise affine function using

$$\mathbf{u}(t) = \underbrace{\frac{t_{i+1} - t}{t_{i+1} - t_i}}_{=:\lambda_-(t)}\mathbf{u}_i + \underbrace{\frac{t - t_i}{t_{i+1} - t_i}}_{=:\lambda_+(t)}\mathbf{u}_{i+1} = \lambda_-(t)\,\mathbf{u}_i + \lambda_+(t)\,\mathbf{u}_{i+1}, \qquad t \in [t_i, t_{i+1}] \tag{32}$$

where $\mathbf{u}_i$ denotes the discretized control at node i, $i = 1, \ldots, N - 1$. The linearized dynamics are then

$$\dot{\mathbf{x}}(t) = \mathbf{A}(t)\,\mathbf{x}(t) + \mathbf{B}\,\lambda_-(t)\,\mathbf{u}_i + \mathbf{B}\,\lambda_+(t)\,\mathbf{u}_{i+1} + \mathbf{q}(t) \tag{33}$$

Given the state transition matrix $\boldsymbol{\Phi}$ that satisfies

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\Phi}(t, t_0) = \mathbf{A}(t)\,\boldsymbol{\Phi}(t, t_0), \qquad \boldsymbol{\Phi}(t_0, t_0) = \mathbf{1} \tag{34}$$

Eq. (33) can be rewritten in discretized form to obtain [37]

$$\mathbf{x}_{i+1} = \mathbf{A}_i\,\mathbf{x}_i + \mathbf{B}_i^-\,\mathbf{u}_i + \mathbf{B}_i^+\,\mathbf{u}_{i+1} + \mathbf{q}_i + \boldsymbol{\nu}_i \tag{35}$$

with

$$\mathbf{A}_i = \boldsymbol{\Phi}(t_{i+1}, t_i) \tag{36a}$$

$$\mathbf{B}_i^- = \mathbf{A}_i \int_{t_i}^{t_{i+1}} \boldsymbol{\Phi}^{-1}(t, t_i)\,\mathbf{B}(t)\,\lambda_-(t)\,\mathrm{d}t \tag{36b}$$

$$\mathbf{B}_i^+ = \mathbf{A}_i \int_{t_i}^{t_{i+1}} \boldsymbol{\Phi}^{-1}(t, t_i)\,\mathbf{B}(t)\,\lambda_+(t)\,\mathrm{d}t \tag{36c}$$

$$\mathbf{q}_i = \mathbf{A}_i \int_{t_i}^{t_{i+1}} \boldsymbol{\Phi}^{-1}(t, t_i)\,\mathbf{q}(t)\,\mathrm{d}t \tag{36d}$$

The state transition matrix in Eq. (34), the nonlinear dynamics in Eq. (10), and the integrands of Eqs. (36b)–(36d) are integrated simultaneously to compute $\mathbf{A}_i, \mathbf{B}_i^-, \mathbf{B}_i^+$, and $\mathbf{q}_i$ at each node. These matrices are then used to create a single equality constraint for the discretized dynamics.

Remark: The nonlinear dynamics are integrated using

$$\bar{\mathbf{x}}(t) = \bar{\mathbf{x}}_i + \int_{t_i}^{t} \mathbf{f}(\mathbf{x}(\xi), \mathbf{u}(\xi))\,\mathrm{d}\xi \tag{37}$$

to obtain the reference state at $t \in [t_i, t_{i+1}]$. In this work, an explicit fixed-step 8th-order Runge-Kutta method is used for numerically integrating Eq. (37).

IV. Trust-Region Methods

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

$$\rho^{(k)} = \frac{\text{actual cost decrease}}{\text{predicted cost decrease}} \tag{38}$$

where the actual cost decrease is calculated using the nonlinear constraints, and the predicted cost decrease is based on the linear constraint violations [27]. Depending on the value of $\rho^{(k)}$, the solution is accepted or rejected. In this work, three trust-region methods are considered:

1) Hard trust region with constant trust-region shrinking and growing rates.
2) Hard trust region with varying trust-region shrinking and growing rates.
3) Soft trust region with constant shrinking and growing rates.

The constraint in Eq. (11e) is imposed explicitly in hard trust-region methods, whereas in soft trust-region methods it is

penalized in the objective function.

A. Hard Trust Region With Constant Rates

Hard trust regions with constant parameters are most often used in space trajectory optimization problems due to their simplicity [11, 13, 14]. Defining three parameters $0 < \rho_0 < \rho_1 < \rho_2 < 1$, a step at iteration k is rejected if $\rho^{(k)} < \rho_0$ because this indicates that there is no (sufficiently large) progress. When a solution is accepted, the trust-region radius R is updated as follows:

$$R^{(k+1)} = \begin{cases} R^{(k)}/\alpha & \text{if} \quad \rho_0 \leq \rho^{(k)} < \rho_1 \\ R^{(k)} & \text{if} \quad \rho_1 \leq \rho^{(k)} < \rho_2 \\ \beta\, R^{(k)} & \text{if} \quad \rho^{(k)} \geq \rho_2 \end{cases} \tag{39}$$

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

B. Hard Trust Region With Varying Rates

We allow α and β to vary based on the values of ρ in the current k and previous iteration $k-1$. Defining the constant parameter $\delta > 1$, α and β are updated as follows [12]:

1) If $\rho^{(k)} \geq \rho_0$ and $\rho^{(k-1)} \geq \rho_0$, then $\beta^{(k)} = \delta\beta^{(k-1)}$ and $\alpha^{(k)} = \alpha^{(k-1)}/\delta$. The growing rate is increased and the shrinking rate decreased if the previous and current iterations are accepted.
2) If $\rho^{(k)} \geq \rho_0$ and $\rho^{(k-1)} < \rho_0$, then $\beta^{(k)} = \beta^{(k-1)}/\delta$ and $\alpha^{(k)} = \delta\alpha^{(k-1)}$. The growing rate is decreased and the shrinking rate increased if only the current step is accepted.
3) If $\rho^{(k)} < \rho_0$ and $\rho^{(k-1)} \geq \rho_0$, then $\alpha^{(k)} = \alpha^{(k-1)}$ and $\beta^{(k)} = \beta^{(k-1)}$. The rates remain constant if the previous step was accepted and the current one is rejected.
4) If $\rho^{(k)} < \rho_0$ and $\rho^{(k-1)} < \rho_0$, then $\alpha^{(k)} = \delta\alpha^{(k-1)}$. The shrinking rate is increased if both steps are rejected.

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

C. Soft Trust Region

Instead of imposing the trust-region constraint in Eq. (11e) directly, the performance index in Eq. (11a) is augmented with a penalty function $p(\mathbf{x})$:

$$\underset{\mathbf{u}(t)}{\text{minimize}} \qquad -z(t_f) + \lambda\ \|\boldsymbol{\nu}(t)\|_1 + \lambda\ \max(0, \eta(t)) + p(\mathbf{x}) \tag{40}$$

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

$$p(\mathbf{x}) = \lambda_{\mathrm{TR}} \ [\max(0, \|\mathbf{x}(t) - \bar{\mathbf{x}}(t)\|_1 - R)]^2 \tag{41}$$

with the penalty parameter $\lambda_{\mathrm{TR}} > 0$. The update mechanism is as follows [38]:

Case 1: $g_{\mathrm{TR}} > 0$: reject step and set $\lambda_{\mathrm{TR}}^{(k+1)} = \zeta \, \lambda_{\mathrm{TR}}^{(k)}$ for $\zeta > 0$.

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

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

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

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

V. Numerical Simulations

The performance of the discretization and trust-region methods is assessed in several thousand simulations. The number of converged simulations, iterations, final mass, computational time, propagation error, and the sparsity of the matrices of the discretized problem are compared. All simulations are carried out in MATLAB. The computational times are measured on an Intel Core i5-6300 2.30 GHz Laptop with four cores and 8 GB of RAM. The Embedded Conic Solver (ECOS) is used to solve the second-order cone program in Eqs. (11) [39]. Details about the trust-region-based SCP algorithm can be found in [40, 41]. The algorithm converges if the maximum constraint violation and the relative change of the modified final mass $z(t_f)$ are lower than thresholds ε_c and ε_ϕ, respectively. The algorithm also terminates without successful convergence to an optimal solution if there is no sufficient progress, that is, if the relative difference of the solution vector in two consecutive iterations i and $i+1$ is smaller than ε_x:

$$\frac{\|\mathbf{x}_i - \mathbf{x}_{i+1}\|}{\|\mathbf{x}_i\|} < \varepsilon_x \tag{42}$$

Relevant SCP parameters are given in Table 1. Different combinations of ρ_i $(i = 0, 1, 2)$, α, β, and δ for the hard, and λ_{TR}, $\lambda_{\mathrm{TR},0}$, and ζ for the soft trust-region method were assessed in preliminary simulations. We found that the values in Table 1 perform well and are often a good compromise in terms of convergence, optimality, and number of iterations. The physical constants and scaling parameters are given in Table 2. Throughout this section, we refer to (F)RPM-D and (F)RPM-I for the differential and integral formulations of the (flipped) Radau pseudospectral method, LGL for the Legendre–Gauss–Lobatto method based on Hermite interpolation, and FOH for the first-order-hold method.

Table 1 Parameters of the SCP algorithms.

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

Table 2 Physical constants in all simulations.

Parameter	Value
Gravitational constant μ	1.3271244×10^{11} km^3/s^2
Gravitational acceleration g_0	9.80665×10^{-3} km/s^2
Length unit LU = AU	$1.495978707 \times 10^{8}$ km
Velocity unit VU	$\sqrt{\mu/\text{LU}}$
Time unit TU	LU/VU
Acceleration unit ACU	VU/TU
Mass unit MU	m_0

A. Overview of Simulations

Three different targets (near-Earth asteroid 2000 SG344, Venus, and asteroid Dionysus) where the complexity of the transfers increases, and two methods to generate (infeasible) initial guesses of different quality (perturbed shape-based cubic interpolation approach and propagation of dynamics) are taken into account. The simulation values of each target are given in Table 3, and Table 4 summarizes the number of initial guesses. The comparison of an optimal trajectory and the ones generated with cubic interpolation and propagation are illustrated in Fig. 1. Clearly, the optimal one deviates significantly, and the final positions of the initial guesses are far from the target position. Note that the initial controls are set to zero in all simulations. Figures 3, 4, and 5 show typical optimized trajectories and the corresponding thrust profiles (linearly interpolated between the nodes) for the SEL_2-2000 SG344, Earth-Venus, and Earth-Dionysus transfers, respectively.

Moreover, trust-region and discretization methods as per Sections III and IV, different numbers of discretization points (ranging from 100 to 300) and orders of the interpolating polynomial (ranging from 3 to 23) are compared (see also Table 5). For each target, all combinations of different initial guesses, numbers of nodes, degrees of the interpolating polynomials, and trust-region methods are considered. An overview of the performed simulations is shown in Fig. 2.

The total number of simulations n_{sim} for each discretization method is given in Table 6. It is determined using

$$n_{\text{sim}} = n_{\text{guess}} \cdot n_{\text{nodes}} \cdot n_{\text{TR}} \cdot n_{\text{orders}} \tag{43}$$

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

Table 3 Simulation values for SEL_2 to 2000 SG344, Earth-Venus and Earth-Dionysus transfers [42–44].

Parameter	SEL_2 - 2000 SG344	Earth - Venus	Earth - Dionysus
$\mathbf{r}_0$, LU	$[-0.70186065, 0.70623244, -3.51115 \times 10^{-5}]^\top$	$[0.97083220, 0.23758440, -1.67106 \times 10^{-6}]^\top$	$[-0.02431767, 0.98330142, -1.51168 \times 10^{-5}]^\top$
$\mathbf{v}_0$, VU	$[-0.73296949, -0.71590485, 4.40245 \times 10^{-5}]^\top$	$[-0.25453902, 0.96865497, 1.50402 \times 10^{-5}]^\top$	$[-1.01612926, -0.02849401, 1.69550 \times 10^{-6}]^\top$
m_0, kg	22.6	1500	4000
$\mathbf{r}_f$, LU	$[0.41806795, 0.82897114, -0.00143382]^\top$	$[-0.32771780, 0.63891720, 0.02765929]^\top$	$[-2.04061782, 2.05179130, 0.55428895]^\top$
$\mathbf{v}_f$, VU	$[-0.96990332, 0.43630220, -0.00123381]^\top$	$[-1.05087702, -0.54356747, 0.05320953]^\top$	$[-0.14231932, -0.45108800, 0.01894690]^\top$
m_f, kg	free	free	free
$T_{\max}$, N	2.2519×10^{-3}	0.33	0.32
I_{sp}, s	3067	3800	3000
t_f, days	700	1000	3534

Table 4 Types of initial guesses and their number for each transfer.

Parameter	SEL_2 - 2000 SG344	Earth - Venus	Earth - Dionysus
Cubic Interpolation	101	251	301
Propagation	101	101	101
Total	202	352	402

B. Results

The performance of the algorithms is assessed by means of four comparisons for each of the transfers. For each of the aforementioned parameters (discretization and trust-region method, polynomial order, and number of nodes), we take the median of the obtained results by varying all the other parameters. For example, the comparison of the discretization methods is performed by taking the median of all results obtained with a given method for all polynomial orders, all considered nodes, and all trust-region strategies. Throughout our analysis, we comment on three key aspects: general performance (i.e. convergence, iterations, and final spacecraft mass), accuracy, and computational time and memory required by the discretization and trust-region methods. We also compare our results with the ones for the PDG problem in [26]. Finally, we assess the performance of our algorithms when compared with the state-of-the-art optimal control software GPOPS-II [45] in combination with the Sparse Nonlinear Optimizer (SNOPT) [46].

Table 5 Number of nodes and orders of the interpolating polynomials for each transfer.

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

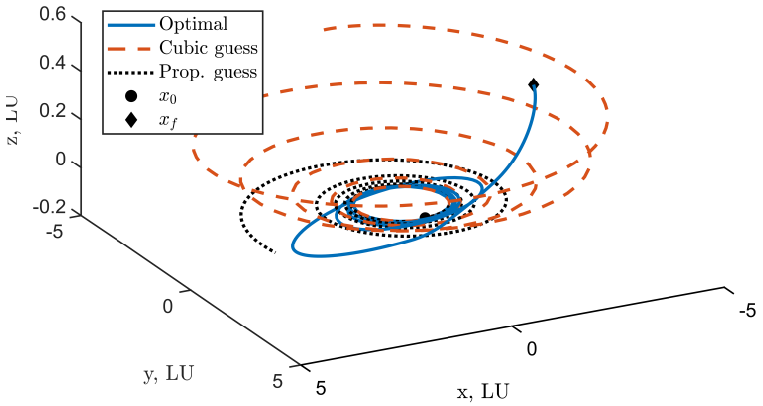


Fig. 1 Optimal trajectory and initial guesses generated using cubic interpolation with 5.4 revolutions and propagation with $T = 0.5\,T_{\max}$ for a Dionysus transfer ($N = 250$).

Table 6 Total number of simulations for each target and each discretization method.

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

1. Convergence, Iterations, and Final Mass

One key objective is to understand the influence of the discretization method, the order of the interpolating polynomial, the number of nodes, and the trust-region method on the success rate, the iterations required to reach convergence, and the optimality of the solutions. The outcome of the analyses is reported in Figs. 6 - 8, where the results related to asteroid 2000 SG344, Venus, and asteroid Dionysus are presented, respectively. The error bars represent the 70th percentile of the considered quantity.

With regard to asteroid 2000 SG344, the largest number of converged cases is obtained with FOH (success rate of approximately 80 %); LGL yields only slightly fewer, and pseudospectral methods approximately 10 % fewer converged cases. The number of iterations is similar, even though FOH requires the fewest. The obtained final masses are almost the same for all methods. This is in accordance with the results of the PDG problem [26] where all methods achieve a similar fuel consumption. Notably, lower polynomial orders require fewer iterations than higher orders. The convergence increases for higher polynomial degrees up to 15, and then remains almost constant. The convergence is slightly higher when N is increased, whereas iterations decrease for larger number of nodes. Similar findings are reported in the PDG study. The hard trust-region with $\delta = 1.0$ and the soft trust-region method yield equivalent results. Remarkably, the hard trust-region approach with $\delta = 1.2$ requires only half as many iterations as with $\delta = 1.0$, at the cost of fewer converged simulations.

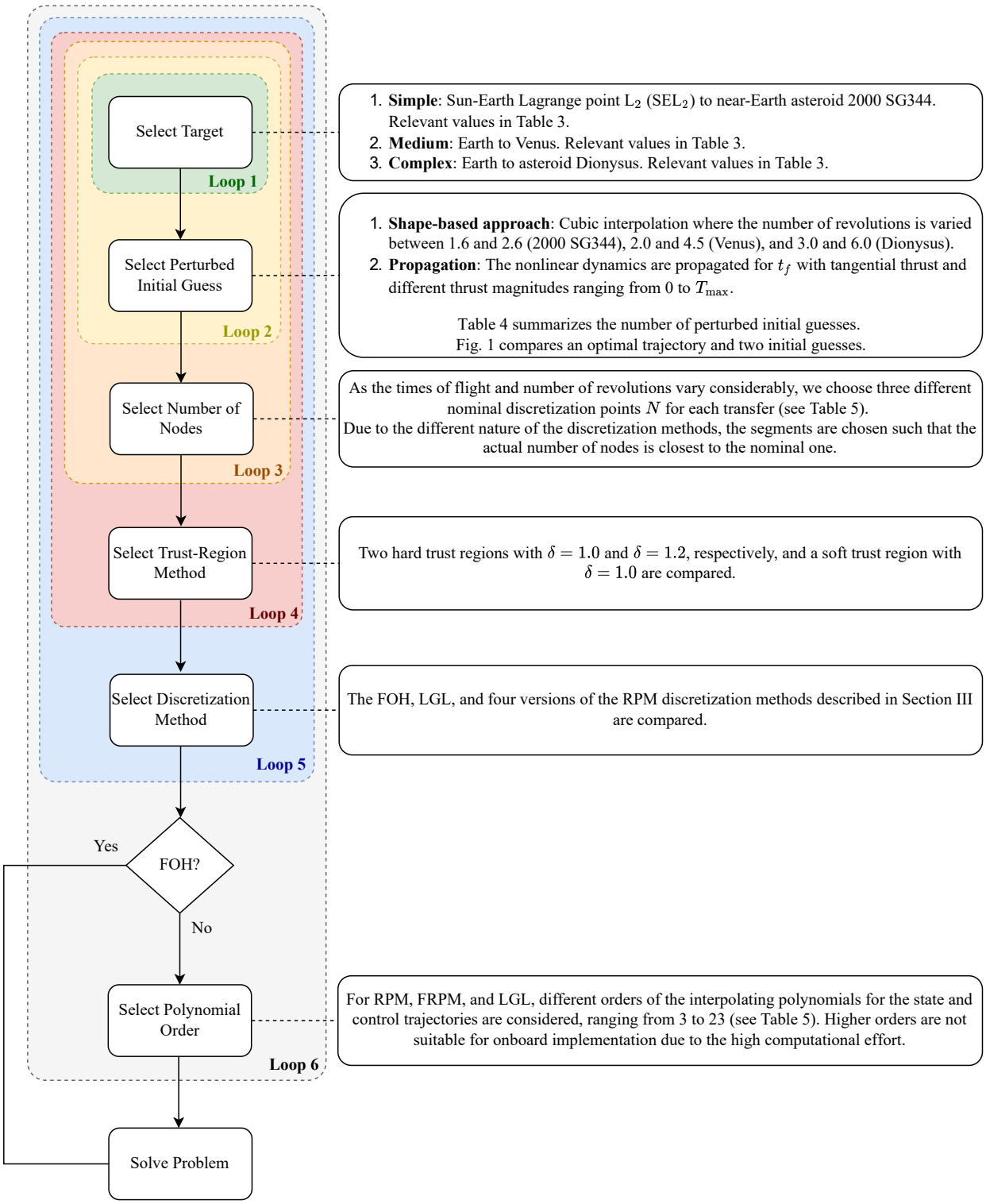
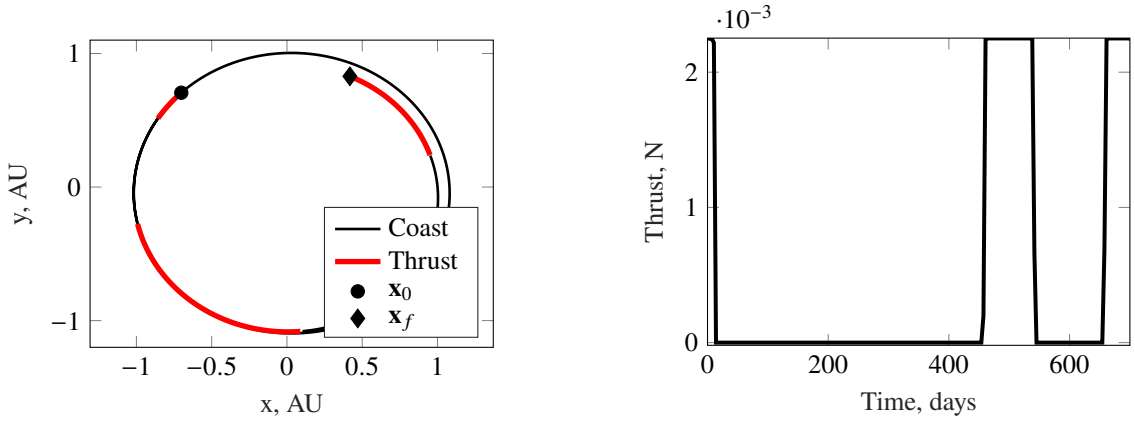


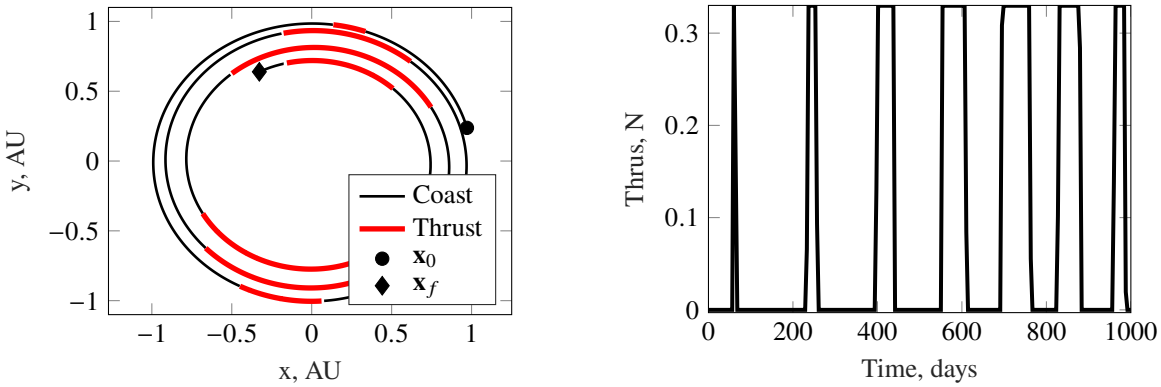
Fig. 2 Overview of performed simulations.

Regarding Venus, the tendency of the methods is opposite: the pseudospectral methods are able to find solutions in almost 60 % of the cases, therefore having a 10 % higher success rate compared to FOH and LGL (see Fig. 7).



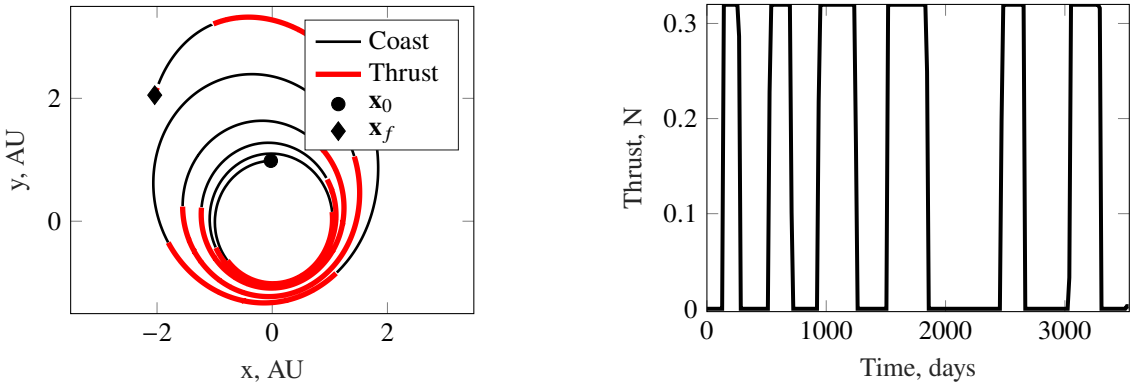
(a) Trajectory **(b) Thrust profile**

Fig. 3 Typical SEL_2 to 2000 SG344 transfer trajectory and corresponding linearly interpolated thrust profile.



(a) Trajectory **(b) Thrust profile**

Fig. 4 Typical Earth-Venus transfer trajectory and corresponding linearly interpolated thrust profile.



(a) Trajectory **(b) Thrust profile**

Fig. 5 Typical Earth-Dionysus transfer trajectory and corresponding linearly interpolated thrust profile.

Furthermore, the overall convergence is worse compared to 2000 SG344. The number of iterations and final masses are similar. Even though the convergence also improves for higher orders, the difference is less significant. The number of nodes and the trust-region method seem to have a small impact on the results (except for the fewer number of iterations when choosing $\delta = 1.2$).

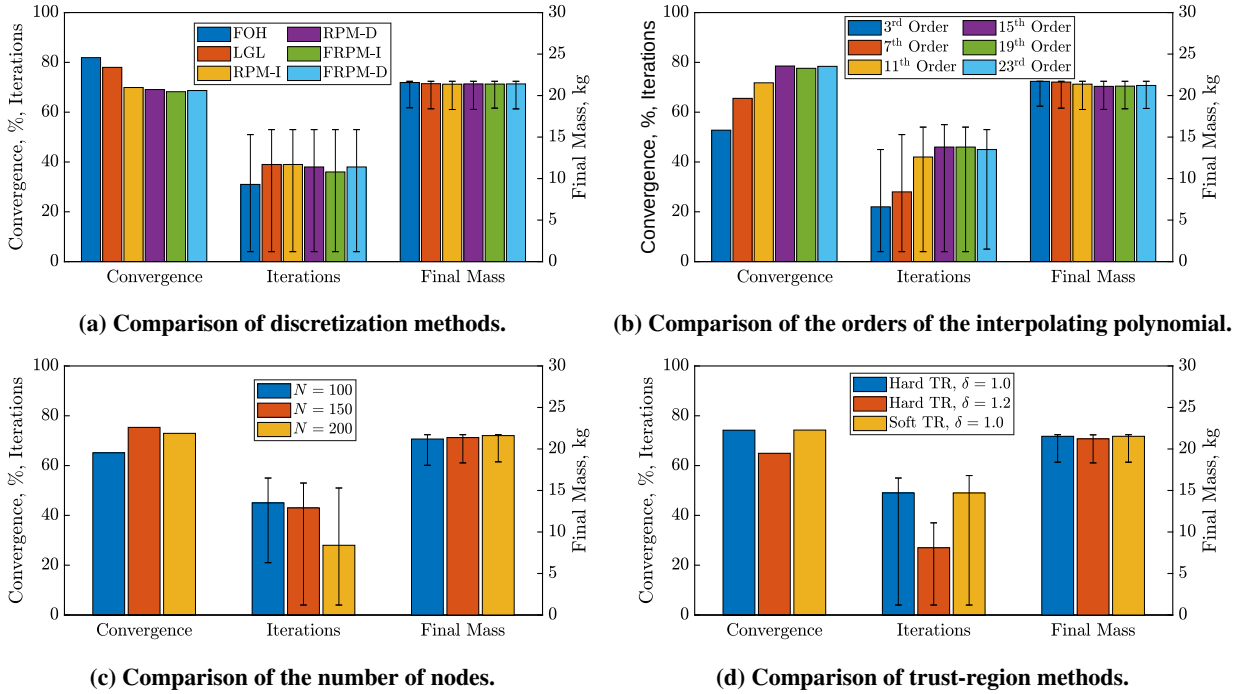
In the Dionysus case, FOH and LGL yield 15 % more converged simulations than the pseudospectral methods (see Fig. 8). Remarkably, the behavior of the polynomial orders is opposite in this case: apart from the third order that yields the lowest success rate, the convergence is best for the 7th order and decreases as the order increases. The success rate slightly changes again depending on the number of nodes, whereas iterations and final mass are not particularly affected by N. The trust-region methods show the same behavior for the iterations and final mass as in the previous cases. Note, however, that the hard trust region with $\delta = 1.2$ achieves slightly higher success rates than the other methods in this example.

As the previous plots considered all orders, Fig. 9 shows the convergence for all targets for the most relevant polynomial degrees 7 and 11. This way the potentially poor performance of the third order does not bias the results. Apparently, the bars follow the same trend: FOH and LGL seem to outperform the pseudospectral methods for the transfers to the asteroids 2000 SG344 and Dionysus. With regard to Venus, in contrast, RPM and FRPM achieve higher success rates.

Figure 10 shows the convergence for the cubic interpolation and propagation guesses. Due to the similarity of the initial and final orbits, the success rate of approximately 70 % for the transfer to asteroid 2000 SG344 is high regardless of how the initial guess is generated. With regard to Venus and Dionysus, however, propagating with tangential thrust results in poor guesses that deviate considerably from the optimal trajectories. A success rate of almost 50 % is therefore remarkable. Still, using a cubic interpolation guess yields in general a larger number of converged cases.

2. Accuracy

It is also crucial that a discretization method is able to achieve a certain accuracy. We define the propagation error for the position as $\left\|\mathbf{r}(t_f)_{\text{prop}} - \mathbf{r}(t_f)\right\|_2$ where $\mathbf{r}(t_f)_{\text{prop}}$ is the final position that is obtained by integrating the dynamics with the optimized controls (the error for the velocity is defined accordingly). As the thrust profile is only known at the discretization points, the controls need to be interpolated for the integration using Eqs. (14), (29), and (32), respectively. Figure 11 shows the orders of magnitude of the propagation error for a Dionysus transfer. It is evident that all methods and polynomial orders achieved the desired accuracy of 10^{-6} LU except for the third order. More precisely, almost all methods found solutions with a median error of 10^{-7} LU or less; only some LGL orders failed to do so in a few cases. The propagation error for the velocity shows a similar tendency, often being one order of magnitude smaller than the position error. The same statements are true for the other targets. Even though the time horizon is significantly smaller for the PDG problem, the errors on the final boundary conditions are very similar to our results



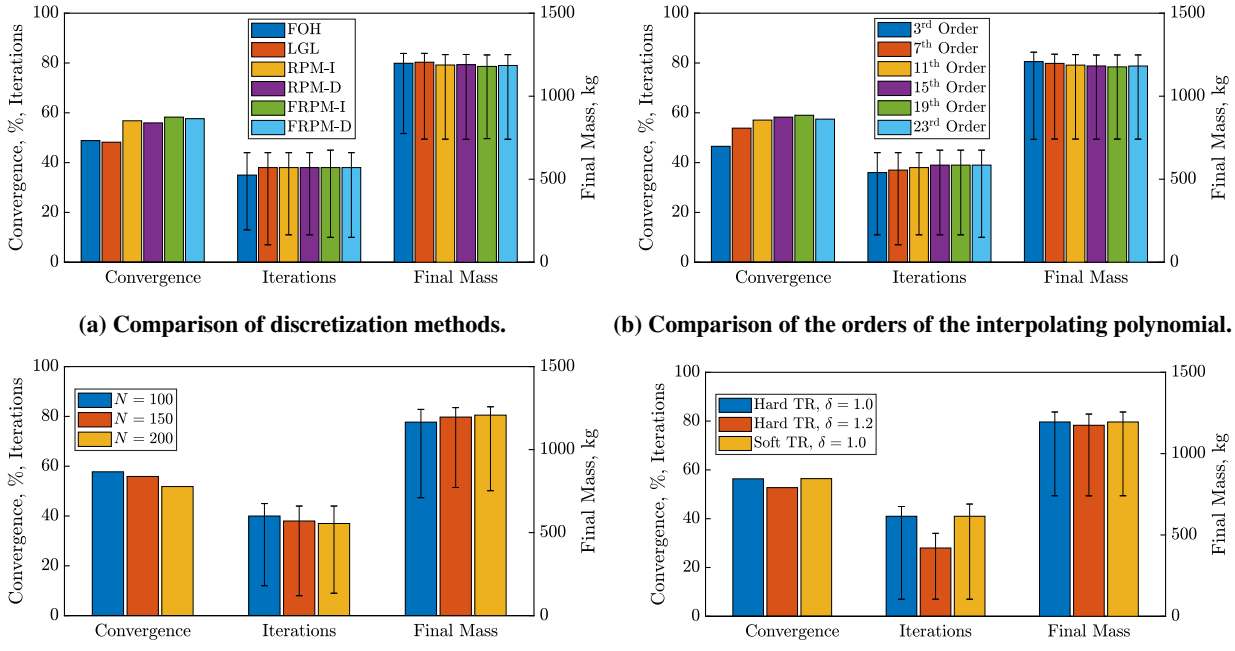
(a) Comparison of discretization methods.

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

(c) Comparison of the number of nodes.

(d) Comparison of trust-region methods.

Fig. 6 Influence of the discretization method, the order of the interpolating polynomial, the number of nodes, and the trust-region method on convergence, iterations, and final mass for the transfer to asteroid 2000 SG344.



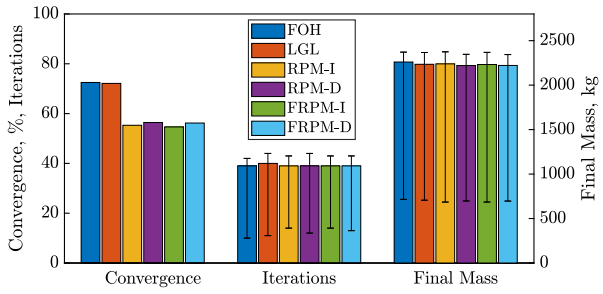
(a) Comparison of discretization methods.

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

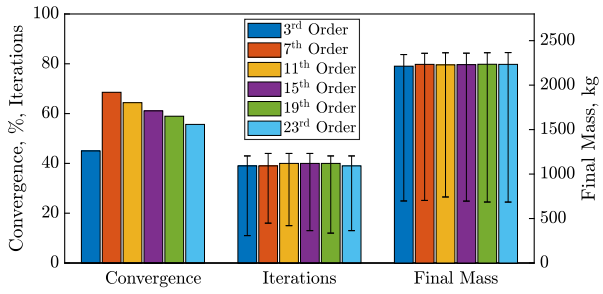
(c) Comparison of the number of nodes.

(d) Comparison of trust-region methods.

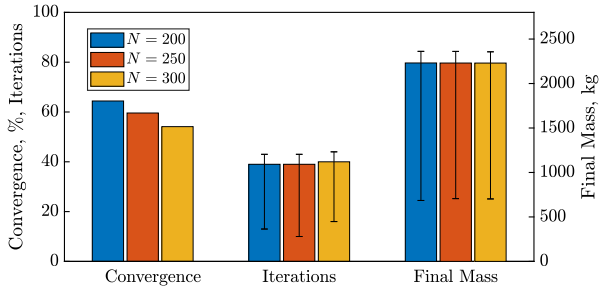
Fig. 7 Influence of the discretization method, the order of the interpolating polynomial, the number of nodes, and the trust-region method on convergence, iterations, and final mass for the transfer to Venus.



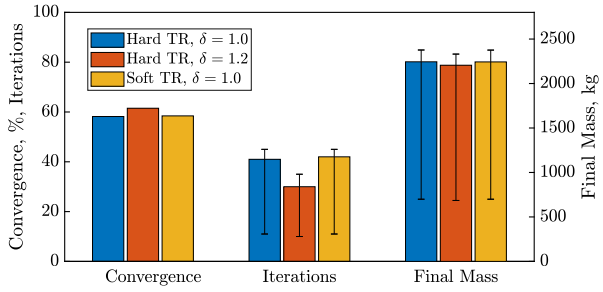
(a) Comparison of discretization methods.



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



(c) Comparison of number of nodes.



(d) Comparison of trust-region methods.

Fig. 8 Influence of the discretization method, the order of the interpolating polynomial, the number of nodes, and the trust-region method on convergence, iterations, and final mass for the transfer to Dionysus.

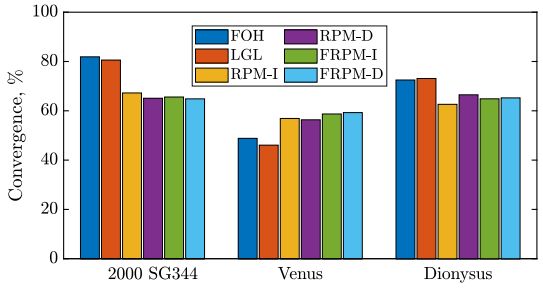


Fig. 9 Comparison of convergence for polynomial orders 7 and 11 for all targets.

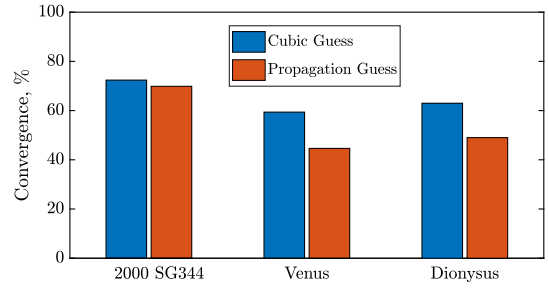


Fig. 10 Comparison of convergence for different initial guesses for all targets.

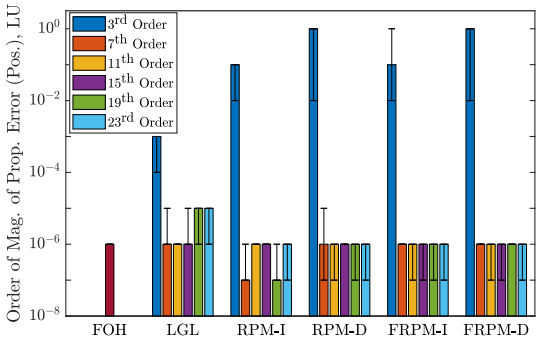


Fig. 11 Comparison of the order of magnitude of the propagation error (position) for a Dionysus transfer (N = 250). Median with minimum and maximum values is shown.

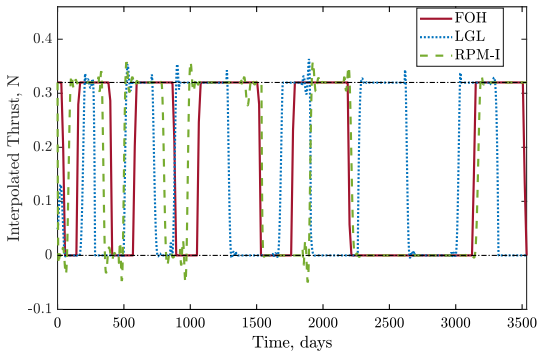


Fig. 12 Interpolated control profiles obtained with FOH and 11th order polynomials (LGL/RPM) for a Dionysus transfer (N = 250). The black dash-dotted lines represent the lower and upper bounds.

(provided that a higher-order polynomial is used) [26]. This confirms the high accuracy of the proposed discretization methods. Although the propagation error is small for all methods, the interpolated controls violate the constraints on the thrust magnitude for LGL and RPM/FRPM as shown in Fig. 12. The reason is that polynomial interpolation results in oscillations close to the edges of the segments for higher polynomial orders (Runge phenomenon). Only FOH is able to generate a bang-bang control profile that does not violate the bounds due to the linear interpolation. As expected, the same is true for the control profiles obtained when solving the PDG problem [26].

3. Computational Time and Memory

Computational time and memory are two other important aspects to consider. As we are dealing with a large amount of optimization parameters, sparse linear algebra becomes crucial. Dense matrix operations would not only take longer to compute, but might also result in memory problems for large-scale optimization problems. The typical percentage of the nonzero elements in the linear equality constraints matrix is given in Fig. 13. Even though more than 99% of the elements are zero for all methods, the integral formulations of RPM/FRPM and LGL are several times denser than (F)RPM-D and FOH. This increases the time required to solve the second-order cone program (SOCP), therefore resulting in a higher total CPU time when the number of iterations does not change. Figure 14a shows the computational times per SCP iteration for a typical Dionysus transfer. As expected, the computational effort increases for higher orders as the matrices become denser. Remarkably, for the four pseudospectral methods the solver time accounts for the largest portion of the total time, whereas for FOH and LGL the time outside the solver is larger. This is because of the integration (FOH) and the transformations due to the definition of the nodes and collocation points (LGL). Moreover, LGL requires the greatest computational effort among all methods regardless of the transfer (see Fig. 14b where the most relevant orders 7 and 11 are shown). The difference becomes more significant when the number of nodes is increased. In general, pseudospectral methods with orders 7 and 11 seem to be the fastest, directly followed by

FOH which eventually outperforms all other methods when higher orders are considered. Given the typical number of iterations of 40, the maximum total CPU time is approximately 24 seconds for FOH and (F)RPM, and 52 seconds for LGL. Although the integral formulations of (F)RPM are denser, their CPU times are sometimes lower than the ones obtained with (F)RPM-D due to the smaller number of solver iterations. Considering Figs. 6a, 7a, 8a, and 14b, it is interesting to note that the results regarding CPU time are partially in accordance with the findings for the PDG problem [26]: even though the overall CPU time is not affected by the choice of either of the two methods, F(RPM) shows a higher sensitivity to the number of discretization points than FOH. Finally, albeit this work only considers fixed final boundary conditions, it was shown that the computational effort required by the SCP algorithm does not significantly increase when a moving target is considered [47].

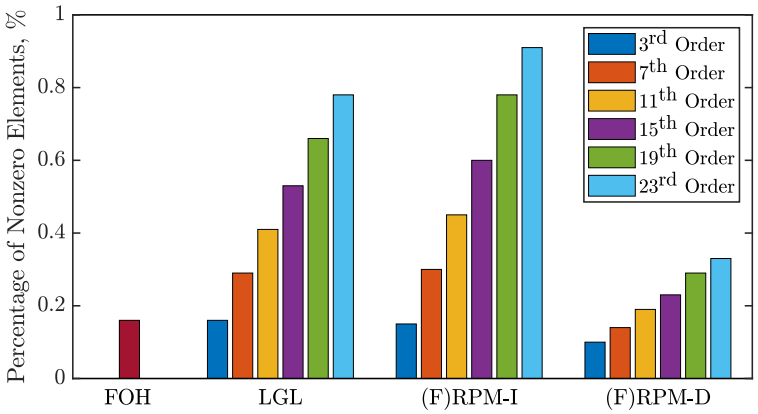
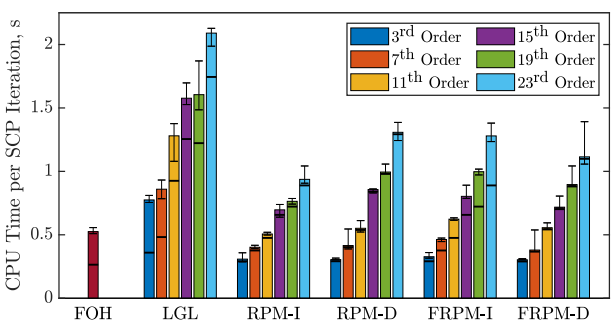


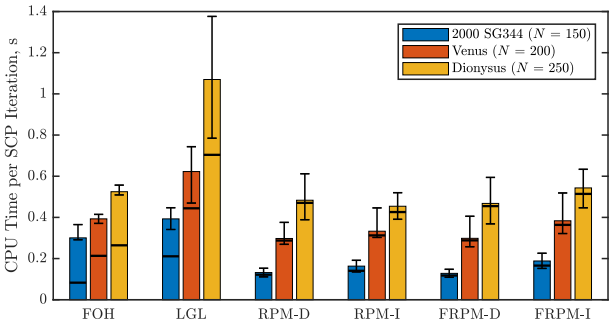
Fig. 13 Comparison of the number of nonzero elements of the linear equality constraints matrix for a Dionysus transfer (N = 250).

4. Comparison With GPOPS-II

The comparison of GPOPS-II and SCP for the transfers to 2000 SG344, Venus, and Dionysus is given in Figs. 15, 16, and 17, respectively. In the bar charts, *SCP* refers to the median of the results obtained with the SCP algorithm when all methods, orders, trust-region parameters and methods, and nodes are considered. *Best SCP* refers instead to the results when the best combination of all parameters (discretization and trust-region methods and polynomial order) is selected for each target. With regard to the 2000 SG344 transfer, the success rate of GPOPS-II is on average higher (approximately +20 %) compared to SCP. This discrepancy, however, reduces to only 5 % if the best SCP method (LGL or FOH) is considered. Yet, GPOPS-II often takes twice as long as SCP to find a solution. For this simple transfer, the final masses are nearly the same. The trend is similar with respect to the Venus transfer, albeit the success rate of SCP improves only slightly when choosing the best method (FRPM or RPM). In addition, the difference in the final mass becomes more evident now. Even though the success rate of GPOPS-II is slightly higher for the Dionysus transfer when considering the averaged SCP, *Best SCP* (11th order LGL) outperforms GPOPS-II in terms of convergence, CPU time,



(a) CPU times for different polynomial orders for a Dionysus transfer (N = 250).



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

Fig. 14 Comparison of CPU times per SCP iteration obtained with the hard trust-region method and δ = 1.0. Median with minimum and maximum values is shown. The heights of the bars show the total CPU times, the horizontal lines within each bar indicate the times required to solve one SOCP.

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

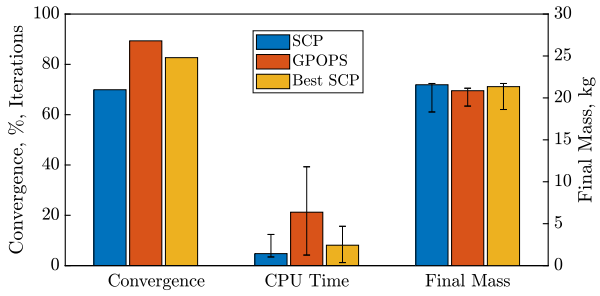
Typical thrust magnitude profiles are illustrated in Figs. 15b, 16b, and 17b. Note the jittery behavior of the control profiles obtained with GPOPS-II. Apparently, it has difficulties to find decent bang-bang trajectories if the controls are discontinuous [23, 48, 49]. Significant additional refinement would therefore be required to use such profiles on board. In contrast, SCP yields the desired bang-bang structure.

C. Overall Performance Assessment

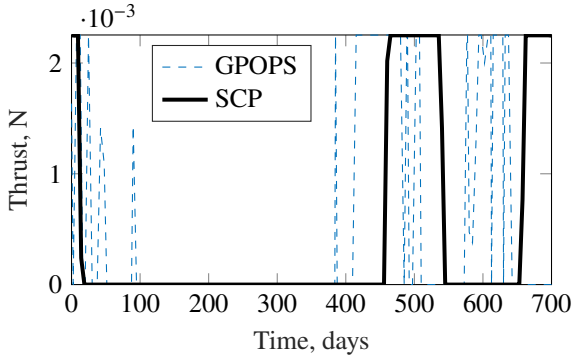
After the extensive explanation of the results in Section V, the performance of the methods is assessed from a general perspective and with respect to their use for onboard guidance applications.

1. General Performance Assessment

We summarize the general performance of the trust-region and discretization methods in Tables 7 and 8. The comparison criteria are: *convergence* (how often the method converges), *optimality* (performance index of the solution),

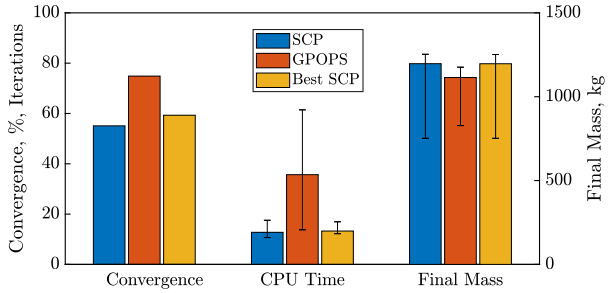


(a) Comparison of convergence, CPU time and final mass.

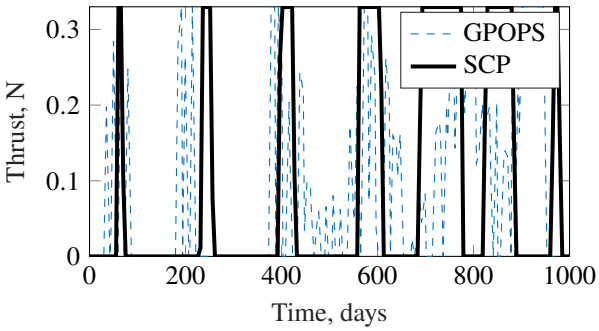


(b) Comparison of linearly interpolated thrust profiles.

Fig. 15 Comparison of GPOPS-II and SCP for the SEL_2 to 2000 SG344 transfer.

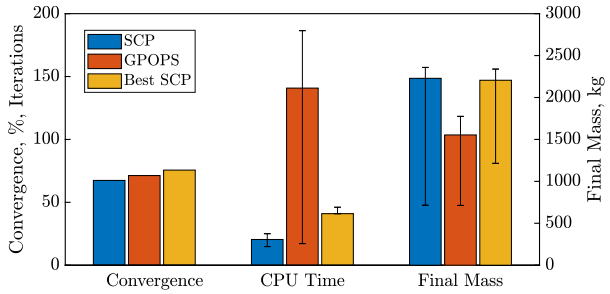


(a) Comparison of convergence, CPU time and final mass.

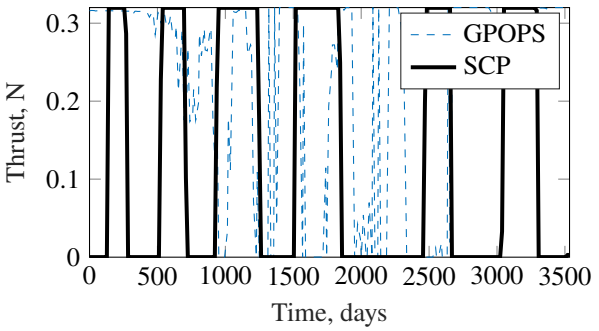


(b) Comparison of linearly interpolated thrust profiles.

Fig. 16 Comparison of GPOPS-II and SCP for the Earth-Venus transfer.



(a) Comparison of convergence, CPU time and final mass.



(b) Comparison of linearly interpolated thrust profiles.

Fig. 17 Comparison of GPOPS-II and SCP for the Earth-Dionysus transfer.

iterations and *CPU time* to reach convergence, and *thrust regularity*, i.e. to what extent a method is capable of accurately capturing the bang-off-bang structure of the optimal thrust profile. In addition, the discretization methods are also compared in terms of *sparsity* of their equality constraints matrices, and *accuracy*, i.e. the error on the final boundary conditions when propagating the dynamics with the obtained controls.

In general, hard trust-region methods seem to be preferable due to the lower computational effort as no additional second-order cone constraints are needed when a quadratic (and hence differentiable) penalty function is used. Selecting $\delta > 1$ is often beneficial, and the value can be adjusted depending on the requirements on convergence and speed of the algorithm. Lower values tend to have higher success rates, whereas larger values result in fewer iterations, but also often less accurate control histories. Even though the results indicate that the performance of a discretization method can depend on the transfer, FOH seems to yield the best overall performance given the criteria in Table 8.

Table 7 Assessment of trust-region methods.

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

Table 8 Assessment of discretization methods.

Criterion	FOH	LGL	RPM/FRPM	Comments
Convergence	Good	Good	Acceptable	Worst performance for third-order polynomials
Optimality	Good	Good	Good	No influence of the discretization method
Iterations	Good	Acceptable	Acceptable	-
CPU time	Good	Bad	Acceptable	CPU time increases with the order of the polynomial
Thrust regularity	Good	Acceptable	Acceptable	Worst performance for high-order polynomials
Sparsity	Good	Acceptable	Acceptable	Few hundreds of kilobyte of memory required
Accuracy	Good	Good	Good	Worst performance for third-order polynomials

2. Assessment of Onboard Guidance Requirements

There are several requirements for onboard guidance methods. We briefly comment on the results in terms of the onboard guidance requirements *reliability and robustness*, *onboard capability*, *accuracy*, and *optimality* in Table 9. All of the fundamental requirements seem to be satisfied by almost all methods. With regard to the collocation methods, polynomial orders between 7 and 11 are preferable due to the acceptable computational effort and high convergence rates. As the repeatedly recomputed optimal trajectories in an onboard guidance scenario will not differ considerably,

the previous solution can serve as a good initial guess for the next optimization. Therefore, the convergence is expected to increase and computational effort to decrease significantly. Using a compiled language like C or C++ will also decrease the computational time. In addition, the flexibility and also reliability of the algorithm can be increased by incorporating planetary ephemeris for dynamic endpoint targets [47]. Yet, it is still to be investigated under what conditions convergence can be guaranteed, and how the methods perform on a real spacecraft onboard computer in a real mission scenario.

Table 9 Assessment of the methods in terms of high-level onboard guidance requirements.

ID	Requirement	Comments
1	Reliability and robustness: The algorithm shall have a high success rate regardless of the quality of the initial guess	All discretization and trust-region methods achieve a high success rate. Previous optimal trajectories can be reused in an autonomous guidance scenario, so the convergence is expected to be close to 100 %.
2	Optimality: The algorithm shall minimize the fuel consumption while respecting the other mission objectives/constraints	All discretization and trust-region methods fulfil this requirement as they yield similar final masses that are close to the optimal ones found in literature.
3	Onboard capability: The algorithm shall be compatible with the limited hardware onboard	The obtained CPU times would result in a total computational time of few minutes for typical space-flight processors such as the LEON family (see e.g. [41] and [50]), therefore being acceptable for deep-space cruise. High sparsity: only few hundreds of kilobytes of memory are required.
4	Accuracy: The propagation error shall be small; the optimized thrust profile shall have as few oscillations as possible	All methods achieve a high accuracy. Interpolating the controls results in oscillations for LGL and RPM/FRPM. The linear control interpolation in FOH seems therefore more suitable.

VI. Conclusion

This paper assesses the discretization methods first-order hold, an arbitrary-order Legendre–Gauss–Lobatto method with Hermite interpolation, and different formulations of the Radau pseudospectral method for the low-thrust trajectory optimization problem. The influence of the discretization method, order of the interpolating polynomial, number of nodes, and trust-region strategy are investigated.

The results show that FOH and a hard trust-region strategy with $\delta > 1.0$ yield the best compromise in terms of convergence, onboard capability, accuracy, and optimality for the considered class of problems. Even though this work

does not present a convergence proof, the high percentage of converged cases has shown that SCP is very reliable despite the poor initial guesses. This is crucial for real-time guidance as a solution must be obtained at any time. Our results show that a convex programming approach seems to be a viable method to compute low-thrust trajectories onboard. In fact, all of the fundamental requirements seem to be satisfied by almost all methods. One of the open challenges is related to the actual use of the optimized thrust profile when fed to the onboard processor: SCP, as all direct optimization methods, only provides a solution at specific discretization points. Therefore, interpolating the controls is necessary to obtain a control trajectory for the whole time domain. Even though a linear interpolation may yield acceptable results, this must be treated carefully to avoid introducing additional errors.

Funding Sources

This research is part of EXTREMA, a project that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant Agreement No. 864697).

References

[1] Klesh, A., and Krajewski, J., "MarCO: Mars Cube One – Lessons Learned from Readying the First Interplanetary Cubesats for Flight," *49th Lunar and Planetary Science Conference*, 2018.

[2] Tsiotras, P., and Mesbahi, M., "Toward an Algorithmic Control Theory," *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 2, 2017, pp. 194–196. https://doi.org/10.2514/1.G002754.

[3] Lu, P., "Introducing Computational Guidance and Control," *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 2, 2017, pp. 193–193. https://doi.org/10.2514/1.G002745.

[4] Reynolds, T., Malyuta, D., Mesbahi, M., Açıkmeşe, B., and Carson, J. M., "A Real-Time Algorithm for Non-Convex Powered Descent Guidance," *AIAA Scitech 2020 Forum*, 2020. https://doi.org/10.2514/6.2020-0844.

[5] Betts, J. T., "Survey of Numerical Methods for Trajectory Optimization," *Journal of Guidance, Control, and Dynamics*, Vol. 21, No. 2, 1998, pp. 193 – 207. https://doi.org/10.2514/2.4231.

[6] Conway, B. A., "A Survey of Methods Available for the Numerical Optimization of Continuous Dynamic Systems," *Journal of Optimization Theory and Applications*, Vol. 152, No. 2, 2012, pp. 271 – 306. https://doi.org/10.1007/s10957-011-9918-z.

[7] Mao, Y., Szmuk, M., and Açıkmeşe, B., "A Tutorial on Real-time Convex Optimization Based Guidance and Control for Aerospace Applications," *2018 Annual American Control Conference (ACC)*, 2018, pp. 2410–2416. https://doi.org/10.23919/ACC.2018.8430984.

[8] Dueri, D., Açıkmeşe, B., Scharf, D. P., and Harris, M. W., "Customized Real-Time Interior-Point Methods for Onboard Powered-Descent Guidance," *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 2, 2017, pp. 197–212. https://doi.org/10.2514/1.G001480.

[9] Reynolds, T. P., and et al., "SOC-i: A CubeSat Demonstration of Optimization-Based Real-Time Constrained Attitude Control," *IEEE Aerospace Conference*, virtual, 2020.

[10] Açıkmeşe, B., and Ploen, S. R., "Convex Programming Approach to Powered Descent Guidance for Mars Landing," *Journal of Guidance, Control, and Dynamics*, Vol. 30, No. 5, 2007, pp. 1353–1366. https://doi.org/10.2514/1.27553.

[11] Hofmann, C., and Topputo, F., "Rapid Low-Thrust Trajectory Optimization in Deep Space Based On Convex Programming," *Journal of Guidance, Control, and Dynamics*, Vol. 44, No. 7, 2021, pp. 1379–1388. https://doi.org/10.2514/1.G005839.

[12] Morelli, A. C., Hofmann, C., and Topputo, F., "Robust Low-Thrust Trajectory Optimization Using Convex Programming and a Homotopic Approach," *IEEE Transactions on Aerospace and Electronic Systems*, 2021. https://doi.org/10.1109/TAES.2021.3128869, available online.

[13] Wang, Z., and Grant, M. J., "Minimum-Fuel Low-Thrust Transfers for Spacecraft: A Convex Approach," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 54, No. 5, 2018, pp. 2274–2290. https://doi.org/10.1109/TAES.2018.2812558.

[14] Wang, Z., and Grant, M. J., "Optimization of Minimum-Time Low-Thrust Transfers Using Convex Programming," *Journal of Spacecraft and Rockets*, Vol. 55, No. 3, 2018, pp. 586–598. https://doi.org/10.2514/1.A33995.

[15] Wang, Z., and Lu, Y., "Improved Sequential Convex Programming Algorithms for Entry Trajectory Optimization," *Journal of Spacecraft and Rockets*, Vol. 57, No. 6, 2020, pp. 1373–1386. https://doi.org/10.2514/1.A34640.

[16] Topputo, F., and Zhang, C., "Survey of Direct Transcription for Low-Thrust Space Trajectory Optimization with Applications," *Abstract and Applied Analysis*, Vol. 2014, 2014, pp. 1–15. https://doi.org/10.1155/2014/851720.

[17] Williams, P., "Hermite–Legendre–Gauss–Lobatto Direct Transcription in Trajectory Optimization," *Journal of Guidance, Control, and Dynamics*, Vol. 32, No. 4, 2009, pp. 1392–1395. https://doi.org/10.2514/1.42731.

[18] L. Darby, W. W. Hager, A. V. R., "An hp-Adaptive Pseudospectral Method for Solving Optimal Control Problems," *Optimal Control Applications and Methods*, Vol. 32, 2010, pp. 476–502. https://doi.org/10.1002/oca.

[19] Benson, D. A., Huntington, G. T., Thorvaldsen, T. P., and Rao, A. V., "Direct Trajectory Optimization and Costate Estimation via an Orthogonal Collocation Method," *Journal of Guidance, Control, and Dynamics*, Vol. 29, No. 6, 2006, pp. 1435–1440. https://doi.org/10.2514/1.20478.

[20] Garg, D., Patterson, M. . A., Francolin, C., L. Darby, C. L., Huntington, G. T., Hager, W. W., and Rao, A. V., "Direct trajectory optimization and costate estimation of finite-horizon and infinite-horizon optimal control problems using a Radau pseudospectral method," *Computational Optimization and Applications*, Vol. 49, No. 2, 2011, pp. 335–358. https://doi.org/10.2514/1.20478.

[21] Sagliano, M., "Generalized hp Pseudospectral-Convex Programming for Powered Descent and Landing," *Journal of Guidance, Control, and Dynamics*, Vol. 42, No. 7, 2019. https://doi.org/10.2514/1.G003731.

[22] Yu, C., and Zhao, Y., D. Yang, "Efficient Convex Optimization of Reentry Trajectory via the Chebyshev Pseudospectral Method," *International Journal of Aerospace Engineering*, Vol. 2019, 2019. https://doi.org/10.1155/2019/1414279, article 1414279.

[23] Tang, G., Jiang, F., and Li, J., "Fuel-Optimal Low-Thrust Trajectory Optimization Using Indirect Method and Successive Convex Programming," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 54, No. 4, 2018, pp. 2053–2066. https://doi.org/10.1109/TAES.2018.2803558.

[24] Reynolds, T. P., Szmuk, M., Malyuta, D., Mesbahi, M., Açıkmeşe, B., and Carson, J. M., "Dual Quaternion-Based Powered Descent Guidance with State-Triggered Constraints," *Journal of Guidance, Control, and Dynamics*, Vol. 43, No. 9, 2020, pp. 1584–1599. https://doi.org/10.2514/1.G004536.

[25] Kayama, Y., Howell, K. C., Bando, M., and Hokamoto, S., "Low-Thrust Trajectory Design with Successive Convex Optimization for Libration Point Orbits," *Journal of Guidance, Control, and Dynamics*, Vol. 45, No. 4, 2022, pp. 623–637. https://doi.org/10.2514/1.G005916.

[26] Malyuta, D., Reynolds, T., Szmuk, M., Mesbahi, M., Açıkmeşe, B., and Carson, J. M., "Discretization Performance and Accuracy Analysis for the Rocket Powered Descent Guidance Problem," *AIAA Scitech 2019 Forum*, 2019. https://doi.org/10.2514/6.2019-0925.

[27] Mao, Y., Szmuk, M., Xu, X., and Açıkmeşe, B., "Successive Convexification: A Superlinearly Convergent Algorithm for Non-convex Optimal Control Problems," , Preprint, submitted February 2019. https://arxiv.org/abs/1804.06539.

[28] Hofmann, C., Morelli, A. C., and Topputo, F., "On the Performance of Discretization and Trust-Region Methods for On-Board Convex Low-Thrust Trajectory Optimization," *AIAA Scitech 2020 Forum*, 2022. https://doi.org/10.2514/6.2022-1892.

[29] Garg, D., Patterson, M., Hager, W., Rao, A., Benson, D., and Huntington, G., "An overview of three pseudospectral methods for the numerical solution of optimal control problems," *Advances in the Astronautical Sciences*, Vol. 135, 2009.

[30] Williams, P., "Hermite–Legendre–Gauss–Lobatto Direct Transcription in Trajectory Optimization," *Journal of Guidance, Control, and Dynamics*, Vol. 32, No. 4, 2009, pp. 1392–1395. https://doi.org/10.2514/1.42731.

[31] Garg, D., "Advances in Global Pseudospectral Methods for Optimal Control," Ph.D. thesis, University of Florida, 2011.

[32] Berrut, J.-P., and Trefethen, L. N., "Barycentric Lagrange Interpolation," *SIAM Review*, Vol. 46, No. 3, 2004. https://doi.org/10.1137/S0036144502417715.

[33] Hofmann, C., and Topputo, F., "Pseudospectral Convex Low-Thrust Trajectory Optimization in a High-Fidelity Model," *AAS/AIAA Space Flight Mechanics Meeting*, 2021. AAS Paper 21-678.

[34] Françolin, C. C., Benson, D. A., Hager, W. W., and Rao, A. V., "Costate approximation in optimal control using integral Gaussian quadrature orthogonal collocation methods," *Optimal Control Applications and Methods*, Vol. 36, No. 4, 2015, pp. 381–397. https://doi.org/https://doi.org/10.1002/oca.2112.

[35] Topputo, F., and Zhang, C., "Survey of Direct Transcription for Low-Thrust Space Trajectory Optimization with Applications," *Abstract and Applied Analysis*, Vol. 2014, 2014. https://doi.org/10.1155/2014/851720.

[36] Williams, P., "Hermite–Legendre–Gauss–Lobatto Direct Transcription in Trajectory Optimization," *Journal of Guidance, Navigation, and Control*, Vol. 32, No. 4, 2009, pp. 1392–1395. https://doi.org/10.2514/1.42731.

[37] Rugh, W. J., *Linear System Theory*, 2nd ed., Prentice-Hall, 1996.

[38] Bonalli, R., Cauligi, A., Bylard, A., and Pavone, M., "GuSTO: Guaranteed Sequential Trajectory optimization via Sequential Convex Programming," *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 6741–6747. https://doi.org/10.1109/ICRA.2019.8794205.

[39] Domahidi, A., Chu, E., and Boyd, S., "ECOS: An SOCP Solver for Embedded Systems," *European Control Conference*, Zurich, Switzerland, 2013, pp. 3071–3076. https://doi.org/doi:10.23919/ECC.2013.6669541.

[40] Mao, Y., Szmuk, M., Xu, X., and Açıkmeşe, B., "Successive Convexification: A Superlinearly Convergent Algorithm for Non-convex Optimal Control Problems," , Preprint, submitted February 2019. https://arxiv.org/abs/1804.06539.

[41] Hofmann, C., and Topputo, F., "Toward On-Board Guidance of Low-Thrust Spacecraft in Deep Space Using Sequential Convex Programming," *Proceedings of AAS/AIAA Space Flight Mechanics Meeting AAS Paper 21-350*, 2021, pp. 1–19.

[42] Topputo, F., Wang, Y., Giordano, G., Franzese, V., Goldberg, H., Perez-Lissi, F., and Walker, R., "Envelop of Reachable Asteroids by M-ARGO CubeSat," *Advances in Space Research*, Vol. 67, No. 12, 2021, pp. 4193–4221. https://doi.org/10.1016/j.asr.2021.02.031.

[43] Jiang, F., Baoyin, H., and Li, J., "Practical Techniques for Low-Thrust Trajectory Optimization with Homotopic Approach," *Journal of Guidance, Control, and Dynamics*, Vol. 35, No. 1, 2012. https://doi.org/10.2514/1.52476.

[44] Taheri, E., Kolmanovsky, I., and Atkins, E., "Enhanced Smoothing Technique for Indirect Optimization of Minimum-Fuel Low-Thrust Trajectories," *Journal of Guidance, Control, and Dynamics*, Vol. 39, No. 11, 2016. https://doi.org/10.2514/1.G000379.

[45] Patterson, M. A., and Rao, A. V., "GPOPS-II: A MATLAB software for solving multiple-phase optimal control problems using hp-adaptive Gaussian quadrature collocation methods and sparse nonlinear programming," *ACM Transactions on Mathematical Software (TOMS)*, Vol. 41, No. 1, 2014, pp. 1–37.

[46] Gill, P. E., Murray, W., and Saunders, M. A., "SNOPT: An SQP algorithm for large-scale constrained optimization," *SIAM review*, Vol. 47, No. 1, 2005, pp. 99–131.

[47] Morelli, A. C., Merisio, G., Hofmann, C., and Topputo, F., "A Convex Guidance Approach to Target Ballistic Capture Corridors at Mars," *44th AAS Guidance, Navigation and Control Conference*, 2022, pp. 1–24.

[48] Yakimenko, O., Yakimenko, O., and Romano, M., "Real-time 6dof guidance for of spacecraft proximity maneuvering and close approach with a tumbling object," *AIAA/AAS astrodynamics specialist conference*, 2010, p. 7666.

[49] Zhou, H., Wang, X., Bai, Y., and Cui, N., "Ascent phase trajectory optimization for vehicle with multi-combined cycle engine based on improved particle swarm optimization," *Acta Astronautica*, Vol. 140, 2017, pp. 156–165.

[50] Massari, M., Lizia, P. D., Cavenago, F., and Wittig, A., "Differential Algebra software library with automatic code generation for space embedded applications," *2018 AIAA Information Systems-AIAA Infotech Aerospace*, 2018. https://doi.org/10.2514/6.2018-0398.