C. Hofmann, F. Topputo
Rapid Low-Thrust Trajectory Optimization in Deep Space Based on Convex Programming
Journal of Guidance Control and Dynamics, In press - Published online 30/04/2021
doi:10.2514/1.G005839

Rapid Low-Thrust Trajectory Optimization in Deep Space Based On Convex Programming

Christian Hofmann* and Francesco Topputo†
Polytechnic University of Milan, 20156 Milan, Italy

I. Introduction

Lowering the costs of interplanetary space missions through the use of small satellites (for example, CubeSats) will be of key importance for solar system science and human space flight. As CubeSats can only provide very low thrust, new challenges arise. In particular, the guidance design is a complex optimization problem and usually takes several days or even months to complete. It is always performed on ground, and every deficiency of the algorithm has to be resolved by the operator. It would be desirable, however, to develop CubeSats that are able to autonomously approach minor or major bodies in the solar system without human intervention. Shifting the guidance task on board poses a great challenge as the algorithm must repeatedly recompute the reference trajectory and guarantee a (near-optimal) solution in real-time. Due to the nonlinear dynamics and low-thrust propulsion characteristics, this results in complex and computationally burdensome optimization tasks. The criteria of feasibility (convergence to a solution), optimality (cost function minimization), and sustainability (compatibility with available resources) must be traded off when the guidance is to be computed on-the-fly. Current techniques focus on optimality due to the large power capabilities of working stations that allow to compute optimal solutions offline, without the need of having real-time capabilities. Only little attention has been paid to designing a computationally simple, non-failing algorithm that can potentially be implemented on board. As a consequence, standard techniques have mainly been used to calculate low-thrust trajectories: metaheuristics, indirect, direct or feedback-driven methods.

Metaheuristics (optimization algorithms that employ heuristic rules to find an optimal solution, for example evolutionary algorithms [1]) and indirect methods (based on the calculus of variations, see [2, 3]) are generally capable of finding locally optimal solutions. Yet, they are not suitable for on-board applications as they suffer from poor robustness and computational difficulties. Although improvements have been proposed during the past years (for example through smoothing techniques [4]), the convergence issues remain unsolved.

Feedback-driven methods are computationally simple and hence a popular choice for preliminary trajectory design [5]. Still, they are in general not optimal and do not guarantee convergence. More recent approaches make use of artificial intelligence techniques to design robust, but less optimal and less flexible on-board guidance schemes [6].

Direct methods transcribe the infinite-dimensional optimal control problem into a (often large-scale) finite-dimensional

*Ph.D. Candidate, Department of Aerospace Science and Technology, Via La Masa 34; christian.hofmann@polimi.it.
†Associate Professor, Department of Aerospace Science and Technology, Via La Masa 34; francesco.topputo@polimi.it. AIAA Senior Member.

constrained optimization problem [7]. On-board computers, however, lack in general of the computational capability to solve the full nonlinear program. Still, because of the higher robustness compared to indirect methods, evolutionary algorithms and similar techniques [8], some studies refined the existing methods to further lower the computational effort and increase accuracy and reliability [9]. Nevertheless, those solvers cannot guarantee convergence to a local minimum and still do not operate in real-time. As a consequence, novel sequential convex programming (SCP) methods [10] have been developed to overcome these issues. The original nonlinear problem is transformed into an equivalent convex program that is iteratively solved using sophisticated interior point methods [11]. Due to their rapid calculation speed and the fact that convex programs are shown to converge to the global minimum under certain conditions [12], such techniques are a popular choice for real-time applications, especially within the path planning of robots and quad-rotors [13, 14]. Because of the high demand for more and more autonomy in aerospace vehicles, tremendous effort was made to exploit the advantages of convex optimization in aerospace applications. Therefore, power descent landing guidance [15, 16] and entry trajectory optimization problems [17, 18] have recently been solved using SCP. In this context, an improved Radau pseudospectral discretization scheme has been applied in [19] to increase the sparsity of the powered descent and landing problem, and thus lower the computational effort. In contrast to the majority of researchers that use a modeling language for convex programming [20] to facilitate the SCP implementation, the work in [21] aims to improve the computational performance by tailoring the algorithm to the actual flight code requirements.

With regard to low-thrust trajectory design, [22, 23] applied sequential convex programming to solve time- and fuel-optimal transfers for the first time. Their simple numerical examples show that the computational time can be reduced considerably compared to standard nonlinear programming (NLP) solvers while still getting near-optimal solutions with rather poor initial guesses. Yet, no conclusion can be drawn on how SCP performs when more complex interplanetary transfers are addressed. Moreover, extensive testing would be necessary to assess the robustness against poor initial guesses. Even simple examples show that bang-off-bang control structures cannot be captured accurately.

This Note presents an improved method based on convex programming to generate complex interplanetary trajectories for low-thrust spacecraft in deep space. The goal is to have a computationally simple and robust algorithm that produces near-optimal solutions in little time. Building on the work of [19] and [24], we employ an adaptive flipped Radau pseudospectral method to lower the computational time and add a mesh refinement strategy for bang-off-bang control structures. A shape-based method generates initial guesses of various quality and several numerical simulations assess the overall robustness of the algorithm.

The Note is structured as follows. Section II states the optimal control problem for space flight and its transcription into a convex program. In Section III, the adaptive flipped Radau pseudospectral method is explained and Section IV addresses the mesh refinement method. The results of numerical simulations are presented and discussed in Section V to assess the performance of the proposed method when compared to state-of-the-art solvers. Final remarks are given in Section VI.

II. Problem Formulation

We consider the motion of a spacecraft in the two-body environment with the Sun as the primary and no other perturbations. The equations of motion are given in Cartesian coordinates by

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) = \begin{bmatrix} \dot{\mathbf{r}} \\ \dot{\mathbf{v}} \\ \dot{m} \end{bmatrix} = \begin{bmatrix} \mathbf{v} \\ -\mu/r^3\mathbf{r} + \mathbf{T}/m \\ -T/v_e \end{bmatrix} \tag{1}$$

where $\mathbf{r} = [r_x, r_y, r_z]^\top$, $\mathbf{v} = [v_x, v_y, v_z]^\top$, and m denote the position, velocity, and mass of the spacecraft, respectively. μ is the gravitational constant, T the thrust magnitude, $\mathbf{T} = [T_x, T_y, T_z]^\top$ the thrust components and $v_e = I_{sp} g_0$ the exhaust velocity of the engine, assumed constant throughout this work (I_{sp} is the specific impulse and g_0 the gravitational acceleration at sea level). Two different convex optimal control problem (OCP) formulations are considered in this work:

OCP1: Partial Linearization

Using $\mathbf{x} = [\mathbf{r}, \mathbf{v}, m]^\top$ as state and $\mathbf{u} = [\mathbf{T}, T]^\top$ as control variables, we rewrite Eq. (1) to obtain

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) = \underbrace{\begin{bmatrix} \mathbf{v} \\ -\mu/r^3\mathbf{r} \\ 0 \end{bmatrix}}_{\mathbf{p}(\mathbf{x})} + \underbrace{\begin{bmatrix} & \mathbf{0}_{3x4} & \\ 1/m & 0 & 0 & 0 \\ 0 & 1/m & 0 & 0 \\ 0 & 0 & 1/m & 0 \\ 0 & 0 & 0 & -1/v_e \end{bmatrix}}_{\mathbf{B}(\mathbf{x})} \begin{bmatrix} \mathbf{T} \\ T \end{bmatrix} = \mathbf{p}(\mathbf{x}) + \mathbf{B}(\mathbf{x})\mathbf{u} \tag{2}$$

Linearizing Eq. (2) only partially at $\bar{\mathbf{x}}$ by setting $\mathbf{f}(\mathbf{x}, \mathbf{u}) \approx \mathbf{p}(\mathbf{x}) + \mathbf{B}(\bar{\mathbf{x}})\mathbf{u}$ yields

$$\mathbf{f}(\mathbf{x}, \mathbf{u}) \approx \mathbf{A}(\bar{\mathbf{x}})\mathbf{x} + \mathbf{B}(\bar{\mathbf{x}})\mathbf{u} + \mathbf{q}(\bar{\mathbf{x}}) \tag{3}$$

where $\mathbf{A}(\bar{\mathbf{x}}) = \left.\frac{\partial \mathbf{p}}{\partial \mathbf{x}}\right|_{\mathbf{x}=\bar{\mathbf{x}}}$ denotes the Jacobian matrix evaluated at the reference point $\bar{\mathbf{x}}$ and $\mathbf{q}(\bar{\mathbf{x}}) = \mathbf{p}(\bar{\mathbf{x}}) - \mathbf{A}(\bar{\mathbf{x}})\bar{\mathbf{x}}$ is the constant part of the linearization. This way the current solution is independent of the previous control history $\bar{\mathbf{u}}$, which significantly enhances the convergence properties [25]. The convex fuel-optimal low-thrust trajectory optimization problem can now be stated as follows:

$$\text{Minimize } J_0 \coloneqq -m_f \tag{4}$$

subject to

$$\dot{\mathbf{x}} = \mathbf{A}(\bar{\mathbf{x}})\mathbf{x} + \mathbf{B}(\bar{\mathbf{x}})\mathbf{u} + \mathbf{q}(\bar{\mathbf{x}}) \tag{5a}$$

$$0 \leq T \leq T_{\max} \tag{5b}$$

$$T_x^2 + T_y^2 + T_z^2 \leq T^2 \tag{5c}$$

$$\|\mathbf{x} - \bar{\mathbf{x}}\|_1 \leq r_{tr} \tag{5d}$$

$$\mathbf{x}(t_0) = \mathbf{x}_0, \quad \mathbf{x}(t_f) = \mathbf{x}_f \tag{5e}$$

$$\mathbf{x}_l \leq \mathbf{x} \leq \mathbf{x}_u, \quad \mathbf{u}_l \leq \mathbf{u} \leq \mathbf{u}_u \tag{5f}$$

where the nonlinear constraint on the thrust $T_x^2 + T_y^2 + T_z^2 = T^2$ has been convexified into the second-order cone constraint in Eq. (5c). The trust region constraint in Eq. (5d) is enforced to keep the approximation of the real dynamics accurate enough (that is, solution trajectory close to the reference). The trust region radius r_{tr} depends on the problem and must be chosen accordingly. $\mathbf{x}_0$ and $\mathbf{x}_f$ are the initial and final position, respectively. Eq. (5f) defines the lower (subscript l) and upper bounds (subscript u) for states and controls, respectively.

OCP2: Decoupling States and Controls

As demonstrated in [23], we can fully decouple states and controls by defining $\tau = T/m$, $\boldsymbol{\tau} = \mathbf{T}/m$ and $z = \ln m$ to obtain

$$\dot{\check{x}} = \mathbf{f}(\check{\mathbf{x}}, \check{\mathbf{u}}) = \underbrace{\begin{bmatrix} \mathbf{v} \\ -\mu/r^3\mathbf{r} \\ 0 \end{bmatrix}}_{\mathbf{p}(\check{\mathbf{x}})} + \underbrace{\begin{bmatrix} & \mathbf{0}_{3x4} & \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1/v_e \end{bmatrix}}_{\check{\mathbf{B}}} \begin{bmatrix} \boldsymbol{\tau} \\ \tau \end{bmatrix} = \mathbf{p}(\check{\mathbf{x}}) + \check{\mathbf{B}}\check{\mathbf{u}} \tag{6}$$

The new states and controls are $\check{\mathbf{x}} = [\mathbf{r}, \mathbf{v}, z]^\top$ and $\check{\mathbf{u}} = [\boldsymbol{\tau}, \tau]^\top$, respectively. In contrast to OCP1, the dynamics can be fully linearized due to this decoupling, and OCP2 reads [23]

$$\text{Minimize } \check{J}_0 \coloneqq \int_{t_0}^{t_f} \tau(t)\, \mathrm{d}t \tag{7}$$

subject to

$$\dot{\breve{x}} = \breve{\mathbf{A}}(\bar{\breve{\mathbf{x}}})\mathbf{x} + \breve{\mathbf{B}}\mathbf{u} + \breve{\mathbf{q}}(\bar{\breve{\mathbf{x}}}) \tag{8a}$$

$$0 \leq \tau \leq T_{\max} e^{-\bar{z}}[1 - (z - \bar{z})] \tag{8b}$$

$$\tau_x^2 + \tau_y^2 + \tau_z^2 \leq \tau^2 \tag{8c}$$

$$\text{Eqs. (5d)–(5f)} \tag{8d}$$

with equivalent notation as in OCP1. Note that the constraint on the maximum thrust $\tau e^z \leq T_{\max}$ has been linearized in Eq. (8b).

Remark 1: Since we want to target a specific point in space and not an orbit, we use Cartesian coordinates to eliminate the two nonlinear boundary constraints $\sin(\theta(t_f)) = \sin(\theta_f)$, $\cos(\theta(t_f)) = \cos(\theta_f)$ on the final position that would be necessary to keep the number of revolutions free for spherical coordinates. $\theta(t_f)$ and θ_f denote the azimuthal angle at the final time t_f and its target value, respectively.

Remark 2: Although OCP2 results in a fully decoupled system that is expected to show better convergence, Section IV will show that OCP1 reduces the complexity when the bang-off-bang mesh refinement is applied.

Remark 3: When nonlinear constraints are linearized about a reference solution, one may encounter infeasible convex subproblems even though the original problem is feasible. This phenomenon is called *artificial infeasibility* [26]. An unconstrained virtual control $\boldsymbol{\nu} \in \mathbb{R}^{n_x}$ (where $n_x = 7$ is the number of states) is added to the linearized dynamical constraints in Eq. (5a) and (8a), respectively, to prevent this [12]:

$$\dot{\mathbf{x}} = \mathbf{A}(\bar{\mathbf{x}})\mathbf{x} + \mathbf{B}(\bar{\mathbf{x}})\mathbf{u} + \mathbf{q}(\bar{\mathbf{x}}) + \boldsymbol{\nu} \tag{9}$$

As $\boldsymbol{\nu}$ is not constrained, the system can always reach a feasible point. The same problem can arise for the linearized control constraint in Eq. (8b). Therefore, this constraint is relaxed by adding a slack variable $\eta \geq 0$:

$$0 \leq \tau \leq T_{\max} e^{-\bar{z}}[1 - (z - \bar{z})] + \eta \tag{10}$$

Although these terms maintain feasibility, they also result in constraint violations when active. To ensure that $\boldsymbol{\nu}$ and η are only used when infeasibility is detected, we incorporate them in our objective function with large penalty parameters

μ_m and λ_m:

$$\text{Minimize} \quad J \coloneqq J_0 + \sum_{m \in I_{eq}} \mu_m \left\| \boldsymbol{\nu}_m \right\|_1 + \underbrace{\sum_{m \in I_{ineq}} \lambda_m \max(0, \eta_m)}_{\text{only for OCP2}} \tag{11}$$

where I_{eq} and I_{ineq} denote the set of equality and inequality constraints, respectively. Even though the virtual control was only active during the first few iterations in our examples, a large and non-decreasing artificial control might result in a solution that does not satisfy the nonlinear dynamics. In that case, it is often sufficient to modify the initial guess, the number of nodes or the trust region mechanism.

III. Adaptive Pseudospectral Convex Optimization

In order to compute an optimal trajectory, the OCP is to be transformed into a parameter optimization problem. Pseudospectral methods are a popular choice for solving NLPs because they show spectral convergence, do not suffer from Runge's phenomenon, and allow to retrieve the costates from the Lagrange multipliers [27]. In [15, 19] a flipped Radau pseudospectral method (FRPM) was developed to solve convexified powered descent and landing problems without adding measures for artificial infeasibility. Instead of keeping a constant number of nodes per segment, we extend this approach and allow this number to vary, hence resulting in the adaptive FRPM. In contrast to other optimization tools that use pseudospectral methods to solve NLPs (for example, the General Purpose Optimal Control Software (GPOPS-II) [28], Shefex-3 Pseudospectral Algorithm for Reentry Trajectory Analysis (SPARTAN) [29] and Direct and Indirect Dynamic Optimization (DIDO) [30]), we adapt the FRPM to convex programs.

A. Adaptive Flipped Radau Pseudospectral Method

The time horizon $[t_0, t_f]$ is divided into K segments and the equations of motion are collocated at N_k nodes in each segment. Throughout this Note, the notation $X_i^{(k)}$, $U_i^{(k)}$ is used to address the ith node of the kth segment of states and controls at time $t_i^{(k)}$, where $i = 0, 1, ..., N_k$ and $k = 1, ..., K$. Fig. 1 illustrates the discretization and nomenclature. Note that the initial node is not a collocation point. The dynamics are approximated at the roots of the flipped Legendre–Radau polynomial, which are defined in the pseudospectral time domain $\hat{t}_i^{(k)} \in (-1, 1]$. The transformation between the physical t and pseudospectral time domain $\hat{t}$ is given by [19]

$$t_i^{(k)} = t_{N_k}^{(k-1)} + \frac{t_{N_k}^{(k)} - t_0^{(k)}}{2} \hat{t}_i^{(k)} + \frac{t_{N_k}^{(k)} + t_0^{(k)}}{2} \qquad \text{for } i = 0, 1, ..., N_k, \quad k = 1, ..., K \tag{12}$$

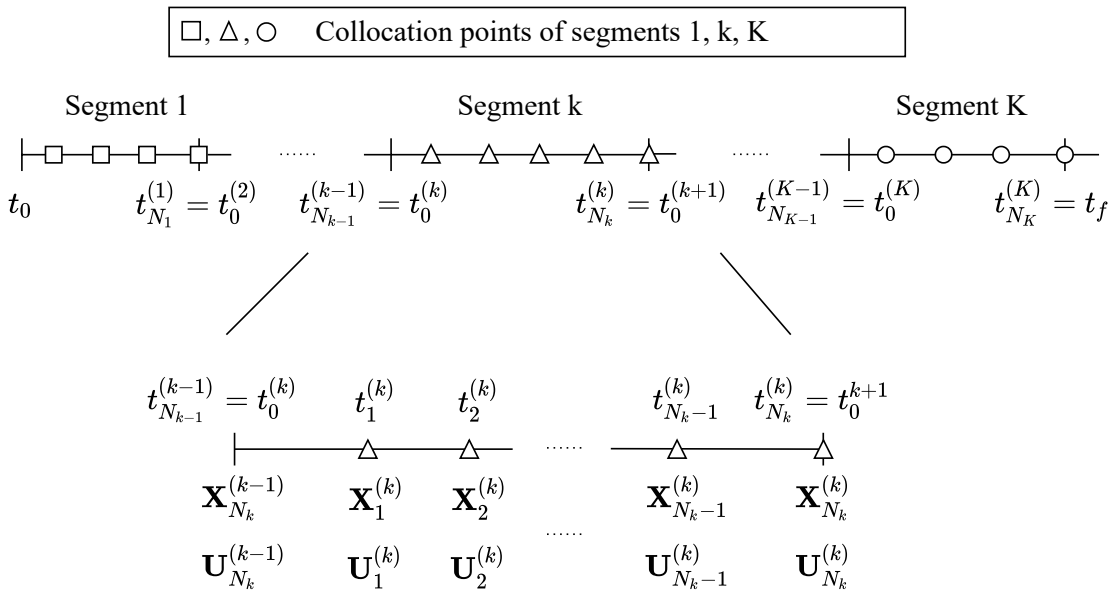


Fig. 1 Overview of the discretization and nomenclature.

with $t_{N_k}^{(0)} = 0$. An integral constraint is then discretized as

$$\int_{t_0}^{t_{N_k}} L(\mathbf{x}(t), \mathbf{u}(t))\, \mathrm{d}t \quad \longrightarrow \quad \sum_{k=1}^{K} \frac{t_{N_k}^{(k)} - t_0^{(k)}}{2} \sum_{i=1}^{N_k} w_i^{(k)} L(\mathbf{X}_i^{(k)}, \mathbf{U}_i^{(k)}) \tag{13}$$

where $w_i^{(k)}$ are the Radau quadrature weights [31]. Similarly, collocating the dynamics yields for each segment k

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \quad \longrightarrow \quad \sum_{j=0}^{N_k} D_{ij}^{(k)} \mathbf{X}_j^{(k)} = \frac{t_{N_k}^{(k)} - t_0^{(k)}}{2} \mathbf{f}(\mathbf{X}_i^{(k)}, \mathbf{U}_i^{(k)}) \qquad i = 1, ..., N_k \tag{14}$$

$D_{ij}^{(k)}$ denotes the entry in the ith row and jth column of the $N_k \times N_k + 1$ differentiation matrix of segment k [32].

B. Discretization of Dynamics

Our goal is to write the dynamics in the linear form $\mathbf{M}_{\text{dyn}}\, \mathbf{Y} = \mathbf{b}_{\text{dyn}}$ so that standard solvers can handle it. We define the discrete vectors as

$$\begin{aligned}
\mathbf{Y} &= [\mathbf{X}, \mathbf{U}, \boldsymbol{\nu}]^\top \\
\mathbf{X} &= [\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, ..., \mathbf{X}^{(K)}]^\top = [\mathbf{X}_1^{(1)}, \mathbf{X}_2^{(1)}, ..., \mathbf{X}_{N_1}^{(1)}, \mathbf{X}_1^{(2)} ..., \mathbf{X}_{N_K}^{(K)}]^\top \\
\mathbf{U} &= [\mathbf{U}^{(1)}, \mathbf{U}^{(2)}, ..., \mathbf{U}^{(K)}]^\top = [\mathbf{U}_1^{(1)}, \mathbf{U}_2^{(1)}, ..., \mathbf{U}_{N_1}^{(1)}, \mathbf{U}_1^{(2)} ..., \mathbf{U}_{N_K}^{(K)}]^\top \\
\boldsymbol{\nu} &= [\boldsymbol{\nu}^{(1)}, \boldsymbol{\nu}^{(2)}, ..., \boldsymbol{\nu}^{(K)}]^\top = [\boldsymbol{\nu}_1^{(1)}, \boldsymbol{\nu}_2^{(1)}, ..., \boldsymbol{\nu}_{N_1}^{(1)}, \boldsymbol{\nu}_1^{(2)} ..., \boldsymbol{\nu}_{N_K}^{(K)}]^\top
\end{aligned} \tag{15}$$

where $(\cdot)^{(k)}$ denotes the column vector of concatenated states, controls or virtual controls of segment k. Note that $\mathbf{X}_0$ and $\mathbf{U}_0$ are not collocation points in the FRPM. Adding the virtual control and substituting the linearized dynamics of Eq. (5a) and (8a) into Eq. (14), we get for segments $k = 1, ..., K$

$$D_{i,0}^{(k)}\mathbf{X}_0^{(k)} + \sum_{n=1}^{N_k} D_{i,n}^{(k)}\mathbf{X}_n^{(k)} = \frac{t_{N_k}^{(k)} - t_0^{(k)}}{2}\left[\mathbf{A}(\bar{\mathbf{X}}_\mathbf{i}^{(\mathbf{k})})\mathbf{X}_i + \mathbf{B}(\bar{\mathbf{X}}_\mathbf{i}^{(\mathbf{k})})\mathbf{U}_i^{(k)} + \mathbf{q}(\bar{\mathbf{X}}_i^{(k)}) + \boldsymbol{\nu}_i^{(k)}\right] \qquad i = 1, ..., N_k \tag{16}$$

$\mathbf{X}_0^{(k)}$ is the initial state of the kth segment. For $k = 1$ this is the initial boundary condition $\mathbf{x}_0$. Keeping in mind that the initial states of each segment are not collocated but the final states are collocation points, the following condition holds true for subsequent segments $k > 1$:

$$\mathbf{X}_{N_k}^{(k-1)} = \mathbf{X}_0^{(k)} \tag{17}$$

Setting $\Delta^{(k)} := (t_{N_k}^{(k)} - t_0^{(k)})/2$, Eq. (16) can be rewritten in matrix form to obtain

$$\underbrace{\left[\begin{array}{c:ccc:c:ccc:c}
\hat{\mathbf{D}}^{(1)} & \mathbf{0} & \mathbf{0} & & \hat{\mathbf{B}}^{(1)} & & & \\
\hdashline
\mathbf{0}\ \hat{\mathbf{I}}^{(2)} & \hat{\mathbf{D}}^{(2)} & \mathbf{0} & \cdots & & \hat{\mathbf{B}}^{(2)} & & \mathbf{0} \\
\hdashline
\mathbf{0} & \mathbf{0}\ \hat{\mathbf{I}}^{(3)} & \hat{\mathbf{D}}^{(3)} & & & & \hat{\mathbf{B}}^{(3)} & & \mathbf{I}_\nu \\
& \vdots & & \ddots & & & & \ddots & \\
& & & \hat{\mathbf{D}}^{(K)} & & \mathbf{0} & & \hat{\mathbf{B}}^{(K)} &
\end{array}\right]}_{\mathbf{M}_{\text{dyn}}} \cdot \begin{bmatrix}\mathbf{X}\\ \mathbf{U}\\ \boldsymbol{\nu}\end{bmatrix} = \underbrace{\begin{bmatrix}
\Delta^{(1)}\mathbf{q}_1^{(1)} - D_{10}^{(1)}\mathbf{X}_0 \\
\Delta^{(1)}\mathbf{q}_2^{(1)} - D_{20}^{(1)}\mathbf{X}_0 \\
\vdots \\
\Delta^{(1)}\mathbf{q}_{N_1}^{(1)} - D_{N_1 0}^{(1)}\mathbf{X}_0 \\
\Delta^{(2)}\mathbf{q}_1^{(2)} \\
\vdots \\
\Delta^{(K)}\mathbf{q}_{N_K}^{(K)}
\end{bmatrix}}_{\mathbf{b}_{\text{dyn}}} \tag{18}$$

with

$$\hat{\mathbf{D}}^{(k)} = \begin{bmatrix}
D_{11}^{(k)}\mathbf{I}_{n_x} - \Delta^{(k)}\mathbf{A}_1^{(k)} & D_{12}^{(k)}\mathbf{I}_{n_x} & \cdots & D_{1N_k}^{(k)}\mathbf{I}_{n_x} \\
D_{21}^{(k)}\mathbf{I}_{n_x} & D_{22}^{(k)}\mathbf{I}_{n_x} - \Delta^{(k)}\mathbf{A}_2^{(k)} & \cdots & D_{2N_k}^{(k)}\mathbf{I}_{n_x} \\
\vdots & \vdots & & \vdots \\
D_{N_k 1}^{(k)}\mathbf{I}_{n_x} & \cdots & & D_{N_k N_k}^{(k)}\mathbf{I}_{n_x} - \Delta^{(k)}\mathbf{A}_{N_k}^{(k)}
\end{bmatrix} \tag{19}$$

and

$$\hat{\mathbf{B}}^{(k)} = \begin{bmatrix} -\Delta^{(k)}\mathbf{B}_1^{(k)} & & & \mathbf{0} \\ & -\Delta^{(k)}\mathbf{B}_2^{(k)} & & \\ & & \ddots & \\ \mathbf{0} & & & -\Delta^{(k)}\mathbf{B}_{N_k}^{(k)} \end{bmatrix}, \quad \hat{\mathbf{I}}^{(k)} = [D_{10}^{(k)}\mathbf{I}_{n_x}, D_{20}^{(k)}\mathbf{I}_{n_x}, \ldots, D_{N_k0}^{(k)}\mathbf{I}_{n_x}]^\top \tag{20}$$

We introduced the notation $\mathbf{A}_i^{(k)} \coloneqq \mathbf{A}(\bar{\mathbf{X}}_\mathbf{i}^{(\mathbf{k})})$ ($\mathbf{B}$ and $\mathbf{q}$ accordingly) for the sake of brevity. $\mathbf{I}_{n_x}$ is a $n_x \times n_x$ and $\mathbf{I}_\nu$ a $(N\ n_x) \times (N\ n_x)$ identity matrix with N being the total number of collocation points and n_x the number of states. As the initial condition is not a collocation point, $\mathbf{X}_0$ is incorporated on the right-hand side of Eq. (18). Final boundary conditions, in contrast, can easily be considered by adding another linear constraint such that $\mathbf{X}_{N_K}^{(K)} = \mathbf{x}_f$, where $\mathbf{x}_f$ denotes the desired final state.

IV. Bang-Off-Bang Mesh Refinement

Determining the switching times, that is, the times when the control changes from on to off or vice versa, requires the costates of the OCP. The estimated switching times are then included in a subsequent optimization process to obtain a more accurate representation of the discontinuities. [24] applied this approach to simple NLP problems. To the best of the authors knowledge, this Note adapts the bang-off-bang mesh refinement to solve complex low-thrust optimization problems in a convex programming environment for the first time. The complete procedure is depicted in Fig. 2.

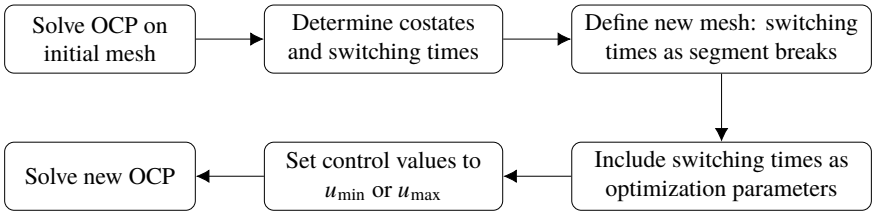


Fig. 2 Flow chart of bang-off-bang mesh refinement process.

A. Computation of Costates and Switching Times

After solving the discretized OCP, the resulting Lagrange multipliers $\mathbf{\Lambda}^{(k)} \in \mathbb{R}^{N_k \times n_x}$ are used to compute the values of the costates $\lambda^{(k)} \in \mathbb{R}^{N_k \times n_x}$ at the collocation points for each segment [33]:

$$\boldsymbol{\lambda}_0 = -[\mathbf{D}_{:,1}^{(1)}]^\top \cdot \mathbf{\Lambda}^{(1)} \tag{21}$$

$$\boldsymbol{\lambda}_i^{(k)} = \frac{\mathbf{\Lambda}_{i,:}^{(k)}}{w_i^{(k)}} \qquad i = 1, ..., N_k \tag{22}$$

The notation $(\cdot)_{i,:}$ and $(\cdot)_{:,i}$ indicates the ith row and column, respectively. After rewriting the equations of motion as

$$[\dot{\mathbf{r}}, \dot{\mathbf{v}}, \dot{m}]^\top = [\mathbf{v},\ \mathbf{g}(\mathbf{r}) + uT_{\max}\boldsymbol{\alpha}/m, -uT_{\max}/v_e]^\top \tag{23}$$

with $\mathbf{g}(\mathbf{r}) = -\mu\mathbf{r}/r^3$, the throttle factor $u = T/T_{\max}$ and the thrust direction vector $\boldsymbol{\alpha} = \mathbf{T}/T$, the Hamiltonian of the problem can be stated as [34]

$$H = \frac{uT_{\max}}{v_e} + \boldsymbol{\lambda}_r^\top \mathbf{v} + \boldsymbol{\lambda}_v^\top \left[\mathbf{g}(\mathbf{r}) + \frac{uT_{\max}}{m}\boldsymbol{\alpha}\right] - \lambda_m \frac{uT_{\max}}{v_e} \tag{24}$$

where minimizing $\frac{T_{\max}}{v_e}\int_{t_0}^{t_f} u(t)\mathrm{d}t$ is equivalent to Eq. (4). Substituting the optimal thrust direction $\boldsymbol{\alpha}_{opt} = -\boldsymbol{\lambda}_v/\|\boldsymbol{\lambda}_v\| = -\boldsymbol{\lambda}_v/\lambda_v$ into Eq. (24) and rearranging terms yields

$$H = \boldsymbol{\lambda}_r^\top \mathbf{v} + \boldsymbol{\lambda}_v^\top \mathbf{g}(\mathbf{r}) + \frac{uT_{\max}}{v_e} \underbrace{\left(1 - \frac{v_e}{m}\lambda_v - \lambda_m\right)}_{=:S} \tag{25}$$

According to Pontryagin's minimum principle [35], an optimal trajectory minimizes the Hamiltonian and hence, the characteristic bang-off-bang control profile in fuel-optimal problems solely depends on the sign of the switching function S. As a consequence, the switching times ($S = 0$) and control structure can be computed once the costates are known.

Remark 4: Using the linear formulation of the Hamiltonian $H = L + \boldsymbol{\lambda}^\top \mathbf{f} = L + \boldsymbol{\lambda}^\top(\mathbf{A}(\bar{\mathbf{x}})\mathbf{x} + \mathbf{B}(\bar{\mathbf{x}})\mathbf{u} + \mathbf{q}(\bar{\mathbf{x}}))$ to calculate the costates and switching times will yield similar results because a converged SCP solution satisfies the nonlinear dynamics. We compared both versions in our numerical simulations and did not notice significant differences.

B. Optimization of Switching Times

The solution to OCP1 and OCP2 is only an approximation as states and controls are known only at the nodes. Therefore, any direct method can intrinsically not determine a discontinuous control structure accurately. The costates and switching times (that is, the zeros of S) are also only estimations. The values of the switching times can be refined by incorporating them into the optimization process to eventually obtain an accurate bang-off-bang control.

We define the vector of all switching times as $\mathbf{T_s} = [t_{s,1}, t_{s,2}, ..., t_{s,j}]^\top$ and divide the trajectory into $K = j + 1$ segments where j is the number of switching times. Thus, each t_s lies on the corner of a segment as shown in Fig. 3. The factors $\Delta^{(k)}$ for the time transformation then become

$$\Delta^{(1)} = \frac{t_{s,1} - t_0}{2}, \qquad \Delta^{(2)} = \frac{t_{s,2} - t_{s,1}}{2}, \qquad \ldots, \qquad \Delta^{(K)} = \frac{t_f - t_{s,j}}{2} \tag{26}$$

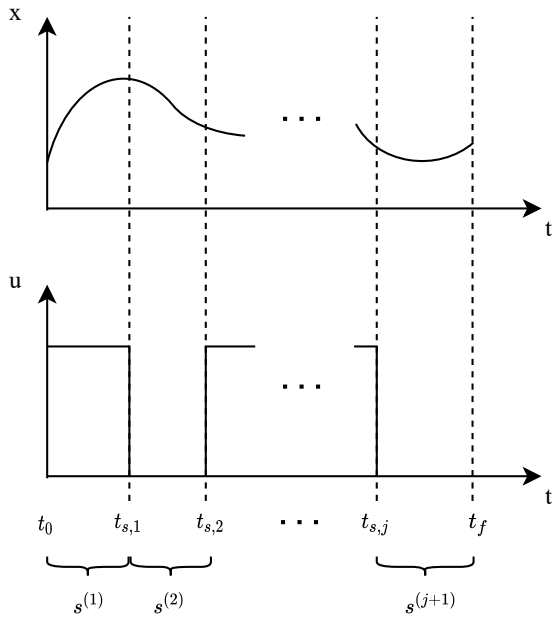


Fig. 3 2D illustration of switching times t_s and segments $s^{(k)}$ for some state x and control u curves.

and cause the formerly convex dynamical constraints in Eq. (16) to become nonconvex. Introducing and linearizing

$$\dot{\mathbf{x}} = \frac{t_{N_k}^{(k)} - t_0^{(k)}}{2}\mathbf{f}(\mathbf{x},\mathbf{u}) =: \tilde{\mathbf{f}}(\mathbf{x},\mathbf{u},t_0^{(k)},t_{N_k}^{(k)}), \tag{27}$$

at $\bar{\mathbf{x}}, \bar{\mathbf{u}}, \bar{t}_0^{(k)}, \bar{t}_{N_k}^{(k)}$ yields

$$\begin{aligned}\dot{\mathbf{x}} \approx &\tilde{\mathbf{A}}(\bar{\mathbf{x}},\bar{\mathbf{u}},\bar{t}_0^{(k)},\bar{t}_{N_k}^{(k)})\mathbf{x} + \tilde{\mathbf{B}}(\bar{\mathbf{x}},\bar{\mathbf{u}},\bar{t}_0^{(k)},\bar{t}_{N_k}^{(k)})\mathbf{u} + \mathbf{T}_0(\bar{\mathbf{x}},\bar{\mathbf{u}},\bar{t}_0^{(k)},\bar{t}_{N_k}^{(k)})t_0^{(k)} + \mathbf{T}_f(\bar{\mathbf{x}},\bar{\mathbf{u}},\bar{t}_0^{(k)},\bar{t}_{N_k}^{(k)})t_{N_k}^{(k)} \\ &+ \tilde{\mathbf{q}}(\bar{\mathbf{x}},\bar{\mathbf{u}},\bar{t}_0^{(k)},\bar{t}_{N_k}^{(k)})\end{aligned} \tag{28}$$

The Jacobian matrices $\tilde{\mathbf{A}}(\bar{\mathbf{x}},\bar{\mathbf{u}},\bar{t}_0^{(k)},\bar{t}_{N_k}^{(k)}) = \frac{\partial\tilde{\mathbf{f}}}{\partial\mathbf{x}}$, $\tilde{\mathbf{B}}(\bar{\mathbf{x}},\bar{\mathbf{u}},\bar{t}_0^{(k)},\bar{t}_{N_k}^{(k)}) = \frac{\partial\tilde{\mathbf{f}}}{\partial\mathbf{u}}$, $\mathbf{T}_0(\bar{\mathbf{x}},\bar{\mathbf{u}},\bar{t}_0^{(k)},\bar{t}_{N_k}^{(k)}) = \frac{\partial\tilde{\mathbf{f}}}{\partial t_0}$ and $\mathbf{T}_f(\bar{\mathbf{x}},\bar{\mathbf{u}},\bar{t}_0^{(k)},\bar{t}_{N_k}^{(k)}) = \frac{\partial\tilde{\mathbf{f}}}{\partial t_{N_k}}$ are evaluated at the reference point $\bar{\mathbf{x}},\bar{\mathbf{u}},\bar{t}_0^{(k)},\bar{t}_{N_k}^{(k)}$. The constant part $\tilde{\mathbf{q}}(\bar{\mathbf{x}},\bar{\mathbf{u}},\bar{t}_0^{(k)},\bar{t}_{N_k}^{(k)})$ is given by $\tilde{\mathbf{f}} - \tilde{\mathbf{A}}\bar{\mathbf{x}} - \tilde{\mathbf{B}}\bar{\mathbf{u}} -$

$\tilde{\mathbf{T}}_0\bar{t}_0 - \tilde{\mathbf{T}}_f\,\bar{t}_{N_k}$ where the dependencies are omitted for the sake of brevity. The new matrix representation is now

$$\underbrace{\left[\begin{array}{c|c|c|c|c|c} \tilde{\mathbf{D}}^{(1)} & \mathbf{0} & \cdots & \tilde{\mathbf{B}}^{(1)} & & \\ \hline \mathbf{0}\quad \tilde{\mathbf{I}}^{(2)} & \tilde{\mathbf{D}}^{(2)} & & & \mathbf{0} & \\ & \ddots & & \tilde{\mathbf{B}}^{(2)} & & \mathbf{I}_\nu \quad \tilde{\mathbf{T}} \\ \vdots & & \ddots & & \ddots & \\ & & \tilde{\mathbf{D}}^{(K)} & \mathbf{0} & \tilde{\mathbf{B}}^{(K)} & \end{array}\right]}_{\tilde{\mathbf{M}}_{\text{dyn}}} \cdot \underbrace{\begin{bmatrix} \mathbf{X} \\ \mathbf{U} \\ \boldsymbol{\nu} \\ \mathbf{T}_s \end{bmatrix}}_{\tilde{\mathbf{Y}}} = \underbrace{\begin{bmatrix} \tilde{\mathbf{q}}_1^{(1)} - D_{10}^{(1)} \cdot \mathbf{X}_0 \\ \tilde{\mathbf{q}}_2^{(1)} - D_{20}^{(1)} \cdot \mathbf{X}_0 \\ \vdots \\ \tilde{\mathbf{q}}_{N_k}^{(1)} - D_{N_k 0}^{(1)} \cdot \mathbf{X}_0 \\ \tilde{\mathbf{q}}_1^{(2)} \\ \vdots \\ \tilde{\mathbf{q}}_{N_k}^{(K)} \end{bmatrix}}_{\tilde{\mathbf{b}}_{\text{dyn}}} \tag{29}$$

The matrices $\tilde{\mathbf{D}}^{(k)}$, $\tilde{\mathbf{I}}^{(k)}$ and $\tilde{\mathbf{B}}^{(K)}$ are similar to those in Eq. (19) with the only difference that the factor $\Delta^{(k)}$ is already included here. Furthermore, $\tilde{\mathbf{T}} = [\mathbf{T}^{(1)}, \mathbf{T}^{(2)}, \ldots, \mathbf{T}^{(K)}]^\top$ with

$$\mathbf{T}^{(1)} = \begin{bmatrix} -\mathbf{T}_{f,1}^{(1)} & \\ \vdots & \mathbf{0} \\ -\mathbf{T}_{f,N_1}^{(1)} & \end{bmatrix}, \quad \mathbf{T}^{(2)} = \begin{bmatrix} -\mathbf{T}_{0,1}^{(2)} & -\mathbf{T}_{f,1}^{(2)} & \\ \vdots & \vdots & \mathbf{0} \\ -\mathbf{T}_{0,N_2}^{(2)} & -\mathbf{T}_{f,N_2}^{(2)} & \end{bmatrix}, \ \ldots, \quad \mathbf{T}^{(K)} = \begin{bmatrix} & -\mathbf{T}_{0,1}^{(K)} \\ \mathbf{0} & \vdots \\ & -\mathbf{T}_{0,N_K}^{(K)} \end{bmatrix} \tag{30}$$

and $\mathbf{T}_{f,i}^{(k)}$ refers to $\mathbf{T}_f\,(\bar{\mathbf{X}}_{\mathbf{i}}^{(\mathbf{k})}, \bar{\mathbf{U}}_{\mathbf{i}}^{(\mathbf{k})}, \bar{t}_0^{(k)}, \bar{t}_f^{(k)})$ ($\mathbf{T}_{0,i}^{(k)}$ equivalently). As the (estimated) switching times are known, the thrust magnitude can be predefined based on the sign of the switching function:

$$\begin{aligned} T &= 0 \qquad \text{if } S > 0 \\ T &= T_{\max} \qquad \text{if } S < 0 \end{aligned} \tag{31}$$

This is done by setting the upper and lower bounds of T accordingly. The linear constraint $t_0 \leq t_{s,1} \leq \ldots \leq t_{s,j} \leq t_f$ is added to ensure the correct order of the switching times.

Remark 5: Although the formulation of OCP2 for optimizing $\mathbf{T_s}$ could have been used in the bang-off-bang mesh refinement procedure, approximating the performance index in Eq. (7) with Eq. (13) results in an additional nonconvex constraint. This can be avoided by simply employing OCP1.

Remark 6: We fully linearize Eq. (27) because we predefine the control $\mathbf{u}$ based on the sign of the switching function. Therefore, oscillations due to large changes of $\mathbf{u}$ are avoided and the convergence does not deteriorate.

Remark 7: Note that a similar mesh refinement strategy as in [36] can be used to increase or decrease the number of

nodes in a continuous segment or subdivide such a segment in several intervals. This may be beneficial in terms of computational effort when there are only few switching times.

V. Numerical Simulations

We simulate two fuel-optimal transfers to compare our sequential convex programming algorithm with results in the literature and the optimization framework GPOPS-II in combination with the Sparse Nonlinear OPTimizer (SNOPT) [28, 37]. Note that GPOPS-II solves the full nonlinear program whereas our SCP algorithm solves the convexified version. To ensure a fair comparison, no mesh refinement procedures are applied in the calculations with SCP and GPOPS-II.

The SCP algorithm is based on [12, 38]. Since feasibility is more important than optimality, a step is accepted as long as the actual cost decrease is greater than zero, even if the predicted decrease is negative. This may result in a higher fuel consumption during intermediate iterations, but drives the state and control vectors towards a feasible trajectory with respect to the nonlinear dynamics. When the feasibility threshold ϵ_c is reached, the algorithm switches back to the original formulation and accepts the step only when both the actual and predicted decreases are positive. It converges when ϵ_c and ϵ_ϕ (cost decrease) are lower than predefined values. The resulting second-order cone program is internally solved by the open source Embedded Conic Solver (ECOS) [11]. The number of (major) iterations and computational time are compared. All simulations are performed in MATLAB version 2018b on an Intel Core i5-6300 2.30 GHz Laptop with four cores and 8 GB of RAM.

A constant maximum thrust and specific impulse are assumed. Furthermore, two-body dynamics without any additional perturbations are considered. The values of all physical constants are given in Table 1, additional SCP parameters in Table 2 where $\rho_i, \alpha_i, \beta_i$ $(i = 0, 1, 2)$ are parameters to adjust the trust region size [38]. All variables are scaled with LU (1 LU = 1 astronomical unit), TU, VU, ACU and MU, respectively. In order to test the robustness against poor initial guesses, a simple cubic interpolation and a shape-based Fourier series method [39] are used to generate guesses of different quality. In all simulations, SCP1 and SCP2 refer to solving OCP1 and OCP2, respectively.

For SCP, the same definition of feasibility as in SNOPT is used:

$$c_{\max} = \max_i \mathrm{viol}_i / \|\mathbf{x}\| \leq \epsilon_c \tag{32}$$

with i being the ith nonlinear constraint violation and $\mathbf{x}$ is the solution vector. Moreover, we define the final position $\Delta\mathbf{r}$ and velocity error $\Delta\mathbf{v}$ as

$$\Delta\mathbf{r} = \left\| \mathbf{r}(t_f)_{\text{propagated}} - \mathbf{r}(t_f)_{\text{solver}} \right\|_2 \tag{33}$$

$$\Delta\mathbf{v} = \left\| \mathbf{v}(t_f)_{\text{propagated}} - \mathbf{v}(t_f)_{\text{solver}} \right\|_2 \tag{34}$$

where the subscript *propagated* defines the final position (velocity) when the full nonlinear equations of motion are propagated with the obtained controls. The subscript *solver* refers to the final position (velocity) that is obtained with SCP and GPOPS-II, respectively.

Table 1 Physical constants in all simulations.

Parameter	Value
Gravitational constant μ	1.32712×10^{11} km^3/s^2
Gravitational acceleration g_0	9.80665×10^{-3} km/s^2
Length unit LU	1.49597×10^{8} km
Velocity unit VU	$\sqrt{\mu/LU}$ km/s
Time unit TU	LU/VU s
Acceleration unit ACU	VU/TU km/s^2
Mass unit MU	m_0

Table 2 Parameters of our SCP algorithm [38].

Parameter	Value
Penalty weight λ	1.0
Penalty weight μ	1.0
Trust region r_0	1.0
$[\rho_0, \rho_1, \rho_2]$	$[0.01, 0.25, 0.9]$
$\alpha_0, \alpha_1, \alpha_2$	$[1.3, 1.5, 1.5]$
$\beta_0, \beta_1, \beta_2$	$[1.3, 1.5, 1.5]$
ϵ_c	10^{-6}
ϵ_ϕ	10^{-5}
Max. iterations	500

A. Earth to Venus Transfer

The Earth to Venus rendezvous transfer is investigated in [40] with an indirect method. Boundary conditions along with other relevant parameters are summarized in Table 3.

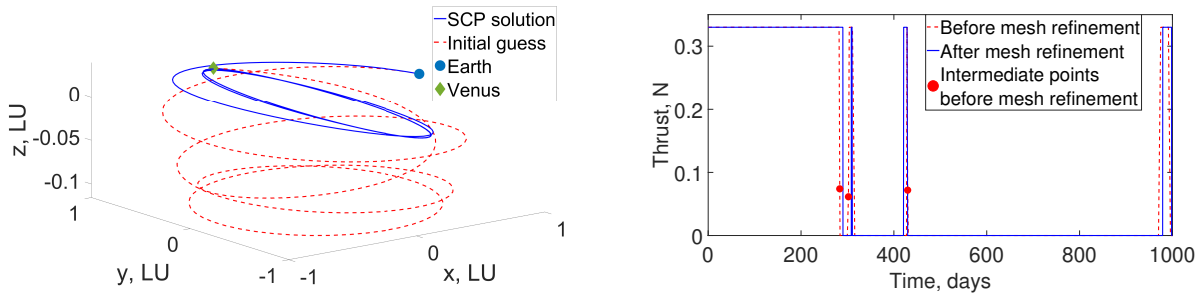
Table 3 Simulation values for Earth-Venus and Earth-Dionysus transfers [4, 40].

Parameter	Earth - Venus	Earth - Dionysus
Initial position $[r_x, r_y, r_z]_0^\top$, LU	$[0.9708, 0.2376, -1.6711 \times 10^{-6}]^\top$	$[-0.0243, 0.9833, -1.5117 \times 10^{-5}]^\top$
Initial velocity $[v_x, v_y, v_z]_0^\top$, VU	$[-0.2545, 0.9687, 1.5040 \times 10^{-5}]^\top$	$[-1.0161, -0.0285, 1.6955 \times 10^{-6}]^\top$
Initial mass $m(t_0)$, kg	1500	4000
Final position $[r_x, r_y, r_z]_f^\top$, LU	$[-0.3277, 0.6389, 0.0277]^\top$	$[-2.0406, 2.0518, 0.5543]^\top$
Final velocity $[v_x, v_y, v_z]_f^\top$, VU	$[-1.0509, -0.5436, 0.0532]^\top$	$[-0.1423, -0.4511, 0.0189]^\top$
Final mass $m(t_f)$, kg	free	free
Maximum thrust $T_{\max}$, N	0.33	0.32
Specific impulse I_{sp}, s	3800	3000
Time of flight t_f, days	1000	3534

1. Comparison with GPOPS-II

The trajectory is discretized in 15 segments with 10 nodes each. Each run is repeated ten times and mean values for computational times are reported. *CubicX* (simple cubic interpolation with *X* revolutions) and *FFSX* (fully optimized Fourier-series approach with *X* revolutions, considered more accurate) denote initial guesses of different quality. A representative trajectory is illustrated in Fig. 4a and shows the large discrepancy between the initial guess and final

solution. Note again that no mesh refinement is considered. The results in Table 4 show that both SCP algorithms



(a) Earth to Venus transfer trajectory **(b) Thrust magnitude before and after mesh refinement**

Fig. 4 Example of Earth to Venus trajectory and thrust profile before/after mesh refinement.

required considerably fewer iterations compared to GPOPS-II. As a consequence, the computational times are also lower, but less significant (only by a factor of two to five). The final masses $m(t_f)$ of GPOPS-II are always lower than SCP masses. When compared to final masses computed with an indirect method (last row in Table 4), it is evident that the values obtained with SCP agree well regardless of the initial guess. Also note that in almost all cases the number of iterations in SCP1 is lower than that in SCP2. As expected, the behavior of computational times is similar.

In many cases, GPOPS-II obtained different trajectories and solutions with oscillations in the control structure. The latter results in larger $\Delta\mathbf{r}$ and $\Delta\mathbf{v}$ values (order of 10^{-2} to 10^{-3} LU and VU, respectively) compared to SCP (order of 10^{-3} to 10^{-4} LU and VU, respectively). Apparently, GPOPS-II could not determine fuel-optimal solutions accurately in the performed simulations with poor initial guesses. This might be an example of the Lavrentiev phenomenon that occurs in adaptive pseudospectral methods where the control is discontinuous; the interested reader is referred to [41] for details.

2. Bang-Off-Bang Mesh Refinement

As the virtual control and slack variables were zero in all converged SCP simulations, the solutions are locally optimal [12]. This is also evident from the characteristic bang-off-bang control structure. Although it is captured quite accurately, there are a few intermediate control values that are neither 0 nor $T_{\max}$. After refinement, however, the control represents the on-off structure more precisely (see Fig. 4b) and agrees very well when compared to an indirect method (see, for example, Fig. 3 in [40]).

3. Perturbed Initial Condition

Each component of the initial condition is randomly perturbed by values between -100.000 km and +100.000 km (position) and -1 km/s and +1 km/s (velocity). The robustness of our algorithm is then tested by calculating 1000 trajectories and counting the number of converged cases, iterations and computational time. A simple cubic interpolation is used to generate (poor) initial guesses. The perturbed initial conditions are illustrated in Fig. 5a.

Table 4 Results of Earth to Venus transfer on the initial mesh.

Element \ Initial Guess		Cubic2	Cubic3	Cubic4	FFS2	FFS3	FFS4
Iterations	GPOPS-II	500*	137	291	135	492	136
	SCP1	28	24	32	37	20	26
	SCP2	43	31	46	35	30	34
Comp. time, s	GPOPS-II	44.9	13.1	24.3	13.4	45.1	14.9
	SCP1	5.4	7.1	8.4	9.4	6.1	6.8
	SCP2	9.1	7.3	10.5	7.8	7.6	7.6
$m(t_f)$, kg	GPOPS-II	943	1278	1066	978	1275	1184
	SCP1	1038	1287	1245	1038	1290	1258
	SCP2	1038	1289	1254	1038	1290	1258
$m(t_f)$, kg, from indirect method [40]		1007, 1036, 1260, 1291					

* Iteration limit reached without convergence.

The results are reported in Table 5 (where averaged values ± 1σ are shown) and Fig. 5 (where the error bars indicate maximum and minimum values). Despite the very large displacements, both SCP algorithms converged in all cases and could determine final masses that are very close to a locally optimal one of 1290 kg as reported in [40]. Still, more iterations were required on average than in the unperturbed case. Computational times, in contrast, are similar. Overall, SCP1 and SCP2 yielded very similar results.

Table 5 Overview of results after 1000 simulations for Earth to Venus transfer.

Method	Total simulations	Converged cases	Avg. # iterations	Avg. comp. time, s	Avg. $m(t_f)$, kg
SCP1	1000	1000	38.4 ± 9.3	8.3 ± 2.0	1284 ± 19
SCP2	1000	1000	41.6 ± 12.9	8.3 ± 2.9	1285 ± 19

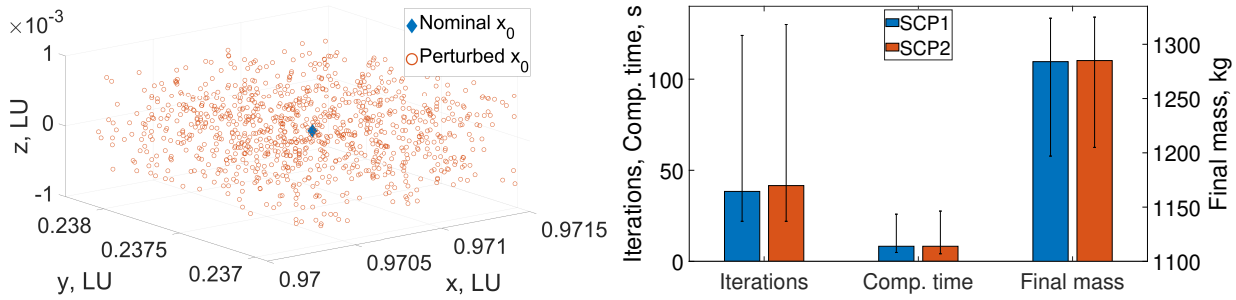
B. Earth to Asteroid Dionysus Transfer

A more complex example represents the low-thrust transfer from Earth to asteroid Dionysus in [4] with a flight time of 3534 days (see Table 3). Several revolutions with significant changes in semi-major axis (Δa = 1.2 LU), eccentricity (Δe = 0.52) and inclination (Δi = 13.5°) are needed to reach the target.

1. Comparison with GPOPS-II

As t_f is considerably higher in this example and to test the performance in case of more nodes, the number of segments is increased to 25 with 10 nodes each.

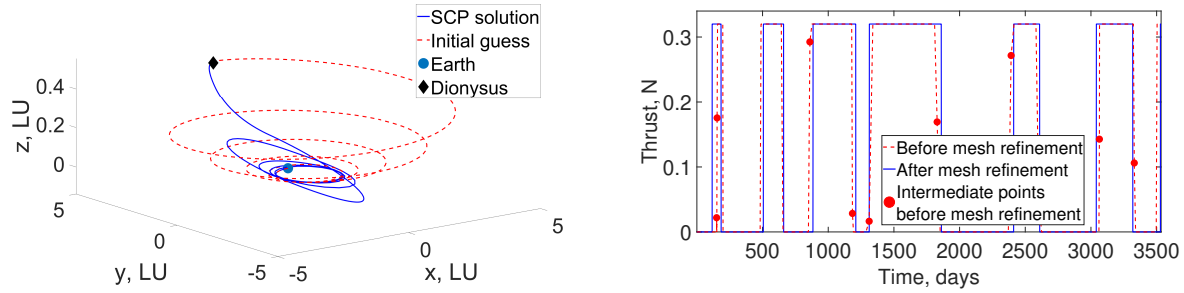
Fig. 6a demonstrates the complexity of this example where several revolutions are needed. The discrepancy between



(a) Distribution of perturbed x_0 **(b) SCP1 vs. SCP2 after 1000 simulations**

Fig. 5 Perturbed initial condition and comparison between SCP1 and SCP2 after 1000 simulations for Earth to Venus transfer.

GPOPS-II and SCP1 and SCP2 becomes clearer because GPOPS-II often reached the iteration limit without finding an optimal solution (see Table 6). Again, SCP required up to one order of magnitude fewer iterations and less computing time than GPOPS-II. Moreover, final masses of SCP are close to values obtained with an indirect method (last row in Table 6) and also notably higher than those of GPOPS-II. GPOPS-II would require many more iterations to reach a similar optimal value (if at all). The trajectories and control histories obtained with both methods are again different and the oscillations in the controls obtained with GPOPS-II increased. In all cases, the values of $\Delta\mathbf{r}$ and $\Delta\mathbf{v}$ are (slightly) bigger compared to the Venus transfer. In contrast to the previous example, the number of iterations for SCP2 is lower compared to SCP1, whereas the computational times are again similar.



(a) Earth to Dionysus transfer trajectory **(b) Thrust magnitude before and after mesh refinement**

Fig. 6 Example of Earth to Dionysus trajectory and thrust profile before/after mesh refinement.

2. Bang-Off-Bang Mesh Refinement

Even in this more complex transfer, the mesh refinement can accurately determine a control structure without intermediate points (see Fig. 6b). This clearly shows that low-thrust fuel-optimal problems can be solved with convex optimization to high accuracy. Note that we increased the number of nodes to show that the procedure also works if the accuracy is to be enhanced.

Table 6 Results of Earth to Dionysus transfer on the initial mesh.

Element \ Initial Guess		Cubic4	Cubic5	Cubic6	FFS4	FFS5	FFS6
Iterations	GPOPS-II	500*	500*	500*	500*	370	311
	SCP1	40	52	45	31	25	26
	SCP2	46	61	116	42	43	35
Comp. time, s	GPOPS-II	97.3	131.8	109.9	330.8	117.2	76.2
	SCP1	20.9	31.5	25.2	16.4	14.1	14.4
	SCP2	26.6	33.0	77.9	25.8	29.5	22.5
$m(t_f)$, kg	GPOPS-II	1984	1706	1843	2083	1964	1547
	SCP1	2352	2614	2586	2404	2445	2312
	SCP2	2247	2617	2528	2445	2491	2249
$m(t_f)$, kg, from indirect method [4]		1980, 2227, 2465, 2672, 2718					

* Iteration limit reached without convergence.

3. Perturbed Initial Condition

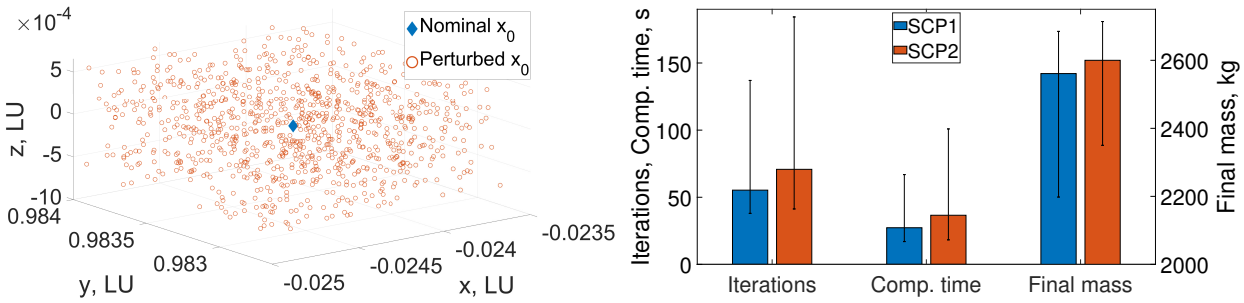
We proceed as in the previous section and perturb the initial condition (cf. Fig. 7a); the results are summarized in Table 7. Compared to SCP1 that converged in all simulations, SCP2 was not able to find a feasible solution in one case (maximum constraint violation of only 10^{-3} vs. the required tolerance of 10^{-6}). It is noticeable that this time SCP2 on average needed 15 iterations more than SCP1. The computing times behave accordingly. In summary, in all cases both algorithms perform in this perturbed environment better than GPOPS-II with the nominal $\mathbf{x}_0$ when poor initial guesses are provided (see Fig. 7b). This again demonstrates the excellent robustness of the developed method.

Table 7 Overview of results after 1000 simulations for Earth to Dionysus transfer.

Method	Total simulations	Converged cases	Avg. # iterations	Avg. comp. time, s	Avg. $m(t_f)$, kg
SCP1	1000	1000	55.3 ± 12.0	27.3 ± 6.7	2561 ± 72
SCP2	1000	999	70.5 ± 20.0	36.5 ± 12.2	2600 ± 58

VI. Conclusion

This work developed a refined version of the sequential convex programming algorithm that can be used for solving low-thrust trajectory optimization problems. Adding an adaptive flipped Radau pseudospectral scheme, a bang-off-bang mesh refinement strategy and measures to avoid virtual infeasibilities proved effective in terms of accuracy, computational effort and optimality when compared to the literature and state-of-the-art solvers. A simple cubic interpolation and a more accurate shape-based approach to generate initial guesses of various quality illustrated the high robustness of the approach. This was shown in orbital transfers from Earth to Venus and Earth to asteroid Dionysus.



(a) Distribution of perturbed x_0

(b) SCP1 vs. SCP2 after 1000 simulations

Fig. 7 Perturbed initial condition and comparison between SCP1 and SCP2 after 1000 simulations for Earth to Dionysus transfer.

The method is a promising first step towards autonomous guidance in real space missions. Its rapid speed together with its demonstrated excellent robustness make it suitable for preliminary trajectory design and also real-time applications where high accuracy is not crucial. Therefore, it is intended for deep-space cruise where less accurate solutions with two-body dynamics are acceptable. In contrast to related work that focused mainly on machine learning methods, the autonomous guidance problem was addressed using numerical optimization techniques. Instead of relying on the quality of training data, the proposed SCP algorithm can react to the environment because it actually calculates new trajectories. Although nonlinear programming methods may in general be superior in terms of accuracy, the simulations show that the proposed algorithm can be faster and more reliable while maintaining a sufficient level of optimality.

References

[1] Pontani, M., and Conway, B. A., "Particle Swarm Optimization Applied to Space Trajectories," *Journal of Guidance, Control, and Dynamics*, Vol. 33, No. 5, 2010, pp. 1429–1441. https://doi.org/10.2514/1.48475.

[2] Zhang, C., Topputo, F., Bernelli-Zazzera, F., and Zhao, Y. S., "Low Thrust Minimum Fuel Optimization in the Circular Restricted Three Body Model," *Advances in the Astronautical Sciences*, Vol. 153, No. 8, 2015, pp. 1597–1615. https://doi.org/10.2514/1.G001080.

[3] Haberkorn, T., Martinon, P., and Gergaud, J., "Low Thrust Minimum-Fuel Orbital Transfer: A Homotopic Approach," *Journal of Guidance, Control, and Dynamics*, Vol. 27, No. 6, 2008, pp. 1046–1060. https://doi.org/10.2514/1.4022.

[4] 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.

[5] Petropoulos, A. E., "Simple Control Laws for Low-Thrust Orbit Transfers," *AAS/AIAA Astrodynamics Specialist Conference*, Aug. 2003. AAS Paper 03-630.

[6] Izzo, D., and Öztürk, E., "Real-Time Guidance for Low-Thrust Transfers Using Deep Neural Networks," *Journal of Guidance, Control, and Dynamics*, Vol. 44, No. 2, 2021, pp. 315–327. https://doi.org/10.2514/1.G005254.

[7] 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.

[8] 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.

[9] 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.

[10] Liu, X., and Lu, P., "Solving Nonconvex Optimal Control Problems by Convex Optimization," *Journal of Guidance, Control, and Dynamics*, Vol. 37, No. 3, 2014. https://doi.org/10.2514/1.62110.

[11] 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/10.23919/ECC.2013.6669541.

[12] Mao, Y., Szmuk, M., Xu, X., and Acikmese, B., "Successive Convexification: A Superlinearly Convergent Algorithm for Non-convex Optimal Control Problems," arXiv 1804.06539, 2019. URL https://arxiv.org/abs/1804.06539.

[13] Szmuk, M., Pascucci, C. A., and Acikmese, B., "Real-Time Quad-Rotor Path Planning for Mobile Obstacle Avoidance using Convex Optimization (IROS)," *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Madrid, Spain, 2018. https://doi.org/10.1109/IROS.2018.8594351.

[14] Liu, c., Lin, C.-Y., and Tomizuka, M., "The Convex Feasible Set Algorithm for Real-Time Optimization in Motion Planning," *SIAM Journal on Control and Optimization*, Vol. 56, No. 4, 2018. https://doi.org/10.1137/16M1091460.

[15] Sagliano, M., "Pseudospectral Convex Optimization for Powered Descent and Landing," *Journal of Guidance, Control, and Dynamics*, Vol. 41, No. 2, 2018. https://doi.org/10.2514/1.G002818.

[16] Yang, H., Bai, X., and Baoyin, H., "Rapid Generation of Time-Optimal Trajectories for Asteroid Landing via Convex Optimization," *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 3, 2018. https://doi.org/10.2514/1.G002170.

[17] Wang, Z., and Grant, M. J., "Constrained Trajectory Optimization for Planetary Entry via Sequential Convex Programming," *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 10, 2017. https://doi.org/10.2514/1.G002150.

[18] Liu, X., and Shen, Z., "Entry Trajectory Optimization by Second-Order Cone Programming," *Journal of Guidance, Control, and Dynamics*, Vol. 39, No. 2, 2016. https://doi.org/10.2514/1.G001210.

[19] 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.

[20] Grant, M., and Boyd, S., "CVX: Matlab Software for Disciplined Convex Programming," http://cvxr.com/cvx, Mar. 2014.

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

[22] 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. https://doi.org/10.2514/1.A33995.

[23] 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.

[24] Agamawi, Y., Hager, W. W., and Rao, A. V., "Mesh Refinement Method for Solving Bang-Bang Optimal Control Problems using Direct Collocation," arXiv 1905.11895, 2019.

[25] Liu, X., Shen, Z., and Lu, P., "Exact Convex Relaxation for Optimal Flight of Aerodynamically Controlled Missiles," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 52, No. 4, 2016. https://doi.org/10.1109/TAES.2016.150741.

[26] Liu, X., Lu, P., and Pan, B., "Survey of Convex Optimization for Aerospace Applications," *Astrodynamics*, Vol. 1, No. 1, 2017, pp. 23–40. https://doi.org/https://doi.org/10.1007/s42064-017-0003-8.

[27] Sagliano, M., Theil, S., Bergsma, M., D'Onofrio, V., Whittle, L., and Viavattene, G., "On the Radau pseudospectral Method: Theoretical and Implementation Advances," *CEAS Space Journal*, Vol. 9, 2017. https://doi.org/10.1007/s12567-017-0165-5.

[28] Patterson, M. A., and Rao, A. V., "GPOPS-II: A MATLAB Software for Solving Multiple-Phase Optimal Control Problems using Hp-Adaptive Gaussian Quadrature Collocation Methods and Sparse Nonlinear Programming," *ACM Trans. Math. Softw.*, Vol. 41, No. 1, 2014. https://doi.org/10.1145/2558904.

[29] Sagliano, M., Theil, S., D'Onofrio, V., and Bergsma, M., "SPARTAN: A Novel Pseudospectral Algorithm for Entry, Descent, and Landing Analysis," *Advances in Aerospace Guidance, Navigation and Control*, Springer International Publishing, Cham, 2018, pp. 669–688. https://doi.org/10.1007/978-3-319-65283-2_36.

[30] Fahroo, F., and Ross, I. M., "Advances in Pseudospectral Methods for Optimal Control," *AIAA Guidance, Navigation and Control Conference and Exhibit*, 2008. https://doi.org/10.2514/6.2008-7309.

[31] Abramowitz, M., and Stegun, I. A., *Handbook of Mathematical Functions*, United States Department of Commerce, 1972, pp. 877 – 897.

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

[34] Zhang, C., Topputo, F., Bernelli-Zazzera, F., and Zhao, Y., "Low-Thrust Minimum-Fuel Optimization in the Circular Restricted Three-Body Problem," *Journal of Guidance, Control, and Dynamics*, Vol. 38, No. 8, 2015. https://doi.org/10.2514/1.G001080.

[35] Bryson, A. E., and Ho, Y.-C., *Applied Optimal Control: Optimization, Estimation, and Control*, Blaisdell Publishing Company, 1969, pp. 90–127.

[36] Patterson, M. A., Hager, W., and Rao, A. V., "A ph Mesh Refinement Method for Optimal Control," *Optimal Control Applications and Methods*, Vol. 36, No. 4, 2014. https://doi.org/10.1002/oca.2114.

[37] Gill, P. E., Murray, W., and Saunders, M. A., "SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization," *SIAM Rev.*, Vol. 47, 2005, pp. 99 – 131. https://doi.org/10.1137/S0036144504446096.

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

[39] Taheri, E., and Abdelkhalik, O., "Initial Three-Dimensional Low-Thrust Trajectory Design," *Advances in Space Research*, Vol. 57, No. 3, 2016, pp. 889 – 903. https://doi.org/10.1016/j.asr.2015.11.034.

[40] 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.

[41] Eide, J., and Rao, A. V., "Lavrentiev Phenomenon in <italic>hp </italic>Gaussian Quadrature Collocation Methods for Optimal Control," *AIAA/AAS Astrodynamics Specialist Conference*, 2016. https://doi.org/10.2514/6.2016-5575.