AAS 21-350

TOWARD ON-BOARD GUIDANCE OF LOW-THRUST SPACECRAFT IN DEEP SPACE USING SEQUENTIAL CONVEX PROGRAMMING

Christian Hofmann* **and Francesco Topputo†**

A robust algorithm to solve the low-thrust fuel-optimal trajectory optimization problem for interplanetary spacecraft is developed in this work. The original nonlinear optimal control problem is convexified and transformed into a parameter optimization problem using the adaptive flipped Radau pseudospectral discretization scheme. The switching times are determined and added as optimization variables in a subsequent optimization process to obtain an accurate on-off control structure. The efficiency of the proposed algorithm is shown in two numerical examples: minimum-fuel transfers from Earth to Venus and Earth to asteroid Dionysus. Simulations are carried out on a desktop computer and a single-board computer. The overall robustness is assessed by varying the quality of the initial guess through a shape-based method and compared to results in the literature and state-of-the-art solvers. The results show a superior performance in terms of computational effort compared to standard nonlinear programming solvers while yielding similar accuracy.

INTRODUCTION

The number of satellites launched into near-Earth orbits is rapidly increasing, and satellite mega-constellations are becoming reality. There is evidence that numerous new deep-space missions will be carried out in the next decade, aiming at increasing our knowledge of the solar system.[1] As miniaturization advanced, small satellites (for example, CubeSats) are now a viable low-cost alternative for many space missions. Their faster and cheaper design compared to conventional spacecraft will be of key importance when it comes to lowering the cost of interplanetary missions.[2] Yet, new challenges arise due to their severe limitations regarding power, orbit control, propulsion and communication. Especially the guidance design is a complex optimization problem that can take several days or even months to complete. Even though it has always been performed on ground, it would be desirable to shift this task on board so that the reference trajectory can be computed on-the-fly. This, however, poses a risk as every deficiency of the algorithm must then be handled by the spacecraft itself. The three criteria feasibility (convergence to a solution at any instant), optimality (minimization of propellant or time of flight) and sustainability (algorithm must be compatible with the limited resources of a nanosatellite) are crucial for the success of a mission. Still, only little attention has been paid to designing an algorithm that respects all of the criteria. Standard techniques have mainly been used because they can simply be rerun in case of non-convergence.

*PhD Candidate, Department of Aerospace Science and Technology, Politecnico di Milano, Via La Masa 34, 20156 Milan, Italy. Email: christian.hofmann@polimi.it

†Associate Professor, Department of Aerospace Science and Technology, Politecnico di Milano, Via La Masa 34, 20156 Milan, Italy. Email: francesco.topputo@polimi.it

As of now, standard indirect,[3,4] direct[5–7] and feedback-driven methods[8] cannot guarantee convergence to a local minimum and still cannot operate in real-time on hardware with limited computational power. More recent approaches make use of artificial intelligence techniques to design computational simple, but less robust and less flexible on-board guidance schemes.[9–11] In recent years, sequential convex programming (SCP) methods have been developed to overcome these issues. Instead of solving the original nonlinear problem, an equivalent convex program is solved iteratively. Due to their rapid calculation speed and high robustness, such techniques are becoming more and more important for real-time aerospace applications.[12,13] Recently, SCP was applied to entry trajectory optimization and power descent landing guidance problems.[14,15] Moreover, simple time- and fuel-optimal low-thrust trajectories were computed with a convex approach[16,17] and it was demonstrated that the computational effort can be lowered significantly compared to nonlinear programming (NLP) solvers. Yet, it is still to be investigated how SCP algorithms perform in case of more complex orbital transfers. In addition, even those simple examples indicate that bang-off-bang control structures cannot be determined accurately. Therefore, we develop an improved, robust method to generate interplanetary low-thrust trajectories in little time. We combine an adaptive flipped Radau pseudospectral method with a bang-off-bang mesh refinement strategy to lower the computational burden while ensuring accurate control histories. Initial guesses of different quality are generated with a shape-based method to assess the robustness of the proposed algorithm.
The paper 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. Section IV addresses the mesh refinement method and the overall SCP algorithm is described in Section V. The results of numerical simulations are presented and discussed in Section VI to assess the performance of the proposed method when compared to state-of-the-art solvers. Finally, Section VII concludes this paper.

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). We use $\mathbf{x} = [\mathbf{r}, \mathbf{v}, m]^\top$ as state and $\mathbf{u} = [\mathbf{T}, T]^\top$ as control variables and rewrite Equation (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 Equation (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 $\mathbf{A}(\bar{\mathbf{x}}) = \frac{\partial \mathbf{p}}{\partial \mathbf{x}}\Big|_{\mathbf{x}=\bar{\mathbf{x}}}$ 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 enhances the convergence properties.[18] The convex fuel-optimal low-thrust trajectory optimization problem can now be stated as follows:

$$\text{Minimize } J_0 \coloneqq -m(t_f) \tag{4}$$

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 a second-order cone constraint in Equation (5c). The trust region constraint in Equation (5d) with the trust region radius r_{tr} is enforced to keep the approximation of the real dynamics accurate enough. x_0 and x_f are the initial and final position, respectively (with t_0 and t_f being the initial and final time, respectively). Equation (5f) defines the lower (subscript l) and upper bounds (subscript u) for states and controls.

We add an unconstrained virtual control $\boldsymbol{\nu} \in \mathbb{R}^{n_x}$ (where $n_x = 7$ is the number of states) to the linearized dynamical constraints in Equation (5a) to prevent *artificial infeasibility*[19] during the linearization process:

$$\dot{\mathbf{x}} = \mathbf{A}(\bar{\mathbf{x}})\mathbf{x} + \mathbf{B}(\bar{\mathbf{x}})\mathbf{u} + \mathbf{q}(\bar{\mathbf{x}}) + \boldsymbol{\nu} \tag{6}$$

We leave $\boldsymbol{\nu}$ unconstrained so that the system can always reach a feasible point. Adding a sufficiently large penalty parameter μ_m in the objective function ensures that $\boldsymbol{\nu}$ is only active when necessary:

$$\text{Minimize} \quad J \coloneqq J_0 + \sum_{m \in I_{eq}} \mu_m \|\boldsymbol{\nu}_m\|_1 \tag{7}$$

I_{eq} denotes the set of equality constraints.

ADAPTIVE PSEUDOSPECTRAL CONVEX OPTIMIZATION

We transform the optimal control problem (OCP) into a parameter optimization problem using the flipped Radau pseudospectral method (FRPM) as it allows us to retrieve the costates from the Lagrange multipliers.[20] We extend the approach of Sagliano,[14,21] include a virtual control and allow a variable number of nodes per segment.

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 (see Figure 1). $X_i^{(k)}$, $U_i^{(k)}$ denote the ith node of the kth segment of states and controls at time $t_i^{(k)}$, where $i = 0, 1, ..., N_k$ and $k = 1, ..., K$. Note that the first node of each segment is not a collocation point. The dynamics are approximated at the roots of the flipped

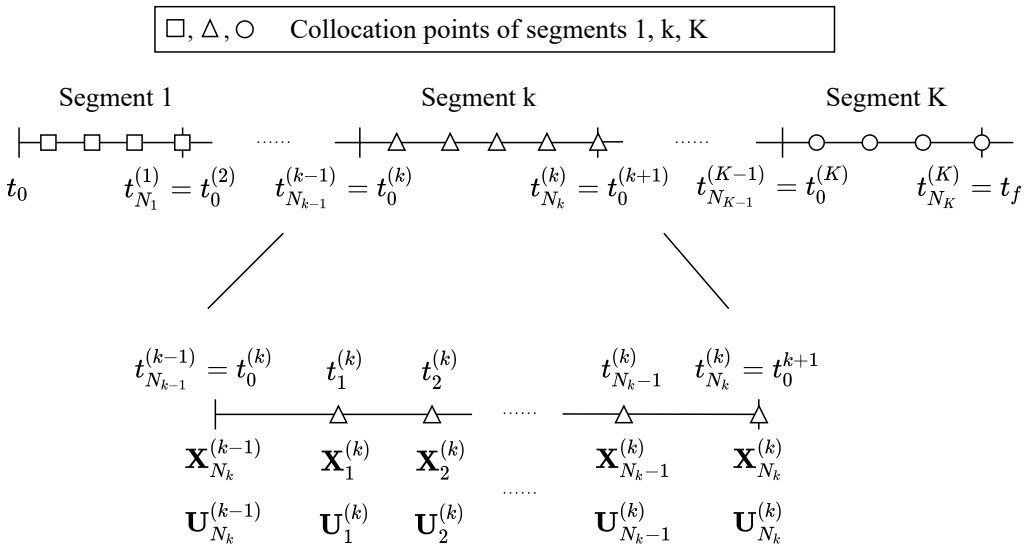


Figure 1: Overview of the discretization and nomenclature.

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[21]

$$t_i^{(k)} = t_{N_k}^{(k-1)} + \frac{t_{N_k}^{(k)} - t_0^{(k)}}{2}\hat{t}_i^{(k)} + \frac{t_{N_k}^{(k)} + t_0^{(k)}}{2} \qquad \text{for } i = 0, 1, ..., N_k, \quad k = 1, ..., K \tag{8}$$

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

$$\int_{t_0}^{t_{N_k}} L(\mathbf{x}(t), \mathbf{u}(t))\mathrm{d}t \quad \longrightarrow \quad \sum_{k=1}^{K} \frac{t_{N_k}^{(k)} - t_0^{(k)}}{2} \sum_{i=1}^{N_k} w_i^{(k)} L(\mathbf{X}_i^{(k)}, \mathbf{U}_i^{(k)}) \tag{9}$$

where $w_i^{(k)}$ are the Radau quadrature weights.[22] Similarly, collocating the dynamics yields for each segment k

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \quad \longrightarrow \quad \sum_{j=0}^{N_k} D_{ij}^{(k)} \mathbf{X}_j^{(k)} = \frac{t_{N_k}^{(k)} - t_0^{(k)}}{2} \mathbf{f}(\mathbf{X}_i^{(k)}, \mathbf{U}_i^{(k)}) \qquad i = 1, ..., N_k \tag{10}$$

$D_{ij}^{(k)}$ denotes the entry in the ith row and jth column of the $N_k \times N_k + 1$ differentiation matrix of segment k.[23]

Discretization of Dynamics

Standard convex programming solvers require the dynamics to be written in the linear form $\mathbf{M}_{dyn} \cdot \mathbf{Y} = \mathbf{b}_{dyn}$. We define the discrete vectors as

$$\begin{aligned}
\mathbf{Y} &= [\mathbf{X}, \mathbf{U}, \boldsymbol{\nu}]^\top \\
\mathbf{X} &= [\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, ..., \mathbf{X}^{(K)}]^\top = [\mathbf{X}_1^{(1)}, \mathbf{X}_2^{(1)}, ..., \mathbf{X}_{N_1}^{(1)}, \mathbf{X}_1^{(2)} ..., \mathbf{X}_{N_K}^{(K)}]^\top \\
\mathbf{U} &= [\mathbf{U}^{(1)}, \mathbf{U}^{(2)}, ..., \mathbf{U}^{(K)}]^\top = [\mathbf{U}_1^{(1)}, \mathbf{U}_2^{(1)}, ..., \mathbf{U}_{N_1}^{(1)}, \mathbf{U}_1^{(2)} ..., \mathbf{U}_{N_K}^{(K)}]^\top \\
\boldsymbol{\nu} &= [\boldsymbol{\nu}^{(1)}, \boldsymbol{\nu}^{(2)}, ..., \boldsymbol{\nu}^{(K)}]^\top = [\boldsymbol{\nu}_1^{(1)}, \boldsymbol{\nu}_2^{(1)}, ..., \boldsymbol{\nu}_{N_1}^{(1)}, \boldsymbol{\nu}_1^{(2)} ..., \boldsymbol{\nu}_{N_K}^{(K)}]^\top
\end{aligned} \tag{11}$$

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 included because they are not collocation points in the FRPM. Adding the virtual control and substituting the linearized dynamics of Equation (5a) into Equation (10), we get for segments $k = 1, ..., K$

$$D_{i,0}^{(k)}\mathbf{X}_0^{(k)} + \sum_{n=1}^{N_k} D_{i,n}^{(k)}\mathbf{X}_n^{(k)} = \frac{t_{N_k}^{(k)} - t_0^{(k)}}{2}\left[\mathbf{A}(\bar{\mathbf{X}}_\mathbf{i}^{(\mathbf{k})})\mathbf{X}_i + \mathbf{B}(\bar{\mathbf{X}}_\mathbf{i}^{(\mathbf{k})})\mathbf{U}_i^{(k)} + \mathbf{q}(\bar{\mathbf{X}}_i^{(k)}) + \boldsymbol{\nu}_i^{(k)}\right] \quad (12)$$

with $i = 1, ..., N_k$. $\mathbf{X}_0^{(k)}$ is the initial state of the kth segment. For $k = 1$ this is the initial boundary condition $\mathbf{x}_0$. 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 $k > 1$:

$$\mathbf{X}_{N_k}^{(k-1)} = \mathbf{X}_0^{(k)} \quad (13)$$

Setting $\Delta^{(k)} := \frac{t_{N_k}^{(k)} - t_0^{(k)}}{2}$, Equation (12) can be rewritten in matrix form to obtain

$$\underbrace{\left[\begin{array}{c|c|c|c|c} \hat{\mathbf{D}}^{(1)} & \mathbf{0} & \cdots & \hat{\mathbf{B}}^{(1)} & \mathbf{0} \\ \hline \mathbf{0}\ \hat{\mathbf{I}}^{(2)} & \hat{\mathbf{D}}^{(2)} & & & \\ \hline \mathbf{0} & \mathbf{0}\ \hat{\mathbf{I}}^{(3)} & & & \mathbf{I}_\nu \\ & & \ddots & \mathbf{0} & \\ \vdots & & \hat{\mathbf{D}}^{(K)} & & \hat{\mathbf{B}}^{(K)} \end{array}\right]}_{\mathbf{M}_{\text{dyn}}} \begin{bmatrix}\mathbf{X}\\ \mathbf{U}\\ \boldsymbol{\nu}\end{bmatrix} = \underbrace{\begin{bmatrix} \Delta^{(1)}\mathbf{q}_1^{(1)} - D_{10}\mathbf{X}_0 \\ \Delta^{(1)}\mathbf{q}_2^{(1)} - D_{20}\mathbf{X}_0 \\ \vdots \\ \Delta^{(1)}\mathbf{q}_{N_1}^{(1)} - D_{N_1 0}\mathbf{X}_0 \\ \Delta^{(2)}\mathbf{q}_1^{(2)} \\ \vdots \\ \Delta^{(K)}\mathbf{q}_{N_K}^{(K)} \end{bmatrix}}_{\mathbf{b}_{\text{dyn}}} \quad (14)$$

with

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

$$\hat{\mathbf{B}}^{(k)} = \begin{bmatrix} -\Delta^{(k)}\mathbf{B}_1^{(k)} & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & -\Delta^{(k)}\mathbf{B}_{N_k}^{(k)} \end{bmatrix}, \quad \hat{\mathbf{I}}^{(k)} = [D_{10}^{(k)}\mathbf{I}_{n_x}, D_{20}^{(k)}\mathbf{I}_{n_x}, \ldots, D_{N_k 0}^{(k)}\mathbf{I}_{n_x}]^\top \quad (15)$$

We introduced the notation $\mathbf{A}_i^{(k)} := \mathbf{A}(\bar{\mathbf{X}}_\mathbf{i}^{(\mathbf{k})})$ ($\mathbf{B}$ 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 $N \cdot n_x \times N \cdot n_x$ identity matrix with N being the total number of collocation points. $\mathbf{X}_0$ is incorporated on the right-hand side of Equation (14) because the initial condition is not a collocation point. In contrast, we can easily include final boundary conditions such as the desired final state $\mathbf{x}_f$ by adding the linear constraint $\mathbf{X}_{N_K}^{(K)} = \mathbf{x}_f$.

BANG-OFF-BANG MESH REFINEMENT

Once the costates are known, we can compute the times when the control changes from on to off or vice versa (switching times). As these values are not exact, we refine the switching times by including them as new optimization parameters in another optimization procedure (see Figure 2). Agamawi et al.[24] applied this approach to simple NLP problems; to the best of our knowledge, we adapt the bang-off-bang mesh refinement to solve problems in a convex programming environment for the first time.

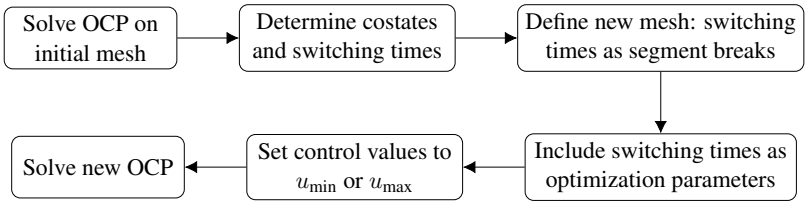


Figure 2: Flow chart of bang-off-bang mesh refinement process.

Computation of Costates and Switching Times

After solving the second-order cone program in Equations (4) and (5), 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:[25]

$$\boldsymbol{\lambda}_0 = -[\mathbf{D}_{:,1}^{(1)}]^\top \cdot \mathbf{\Lambda}^{(1)} \tag{16}$$

$$\boldsymbol{\lambda}_i^{(k)} = \frac{\mathbf{\Lambda}_{i,:}^{(k)}}{w_i^{(k)}} \qquad i = 1, ..., N_k \tag{17}$$

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

$$[\dot{\mathbf{r}}, \dot{\mathbf{v}}, \dot{m}]^\top = [\mathbf{v}, \mathbf{g}(\mathbf{r}) + uT_{\max}\boldsymbol{\alpha}/m, -uT_{\max}/v_e]^\top \tag{18}$$

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[26]

$$H = \frac{uT_{\max}}{v_e} + \boldsymbol{\lambda}_r^\top \mathbf{v} + \boldsymbol{\lambda}_v^\top \left[\mathbf{g}(\mathbf{r}) + \frac{uT_{\max}}{m}\boldsymbol{\alpha}\right] - \lambda_m \frac{uT_{\max}}{v_e} \tag{19}$$

where minimizing $\frac{T_{\max}}{v_e}\int_{t_0}^{t_f} u(t)\mathrm{d}t$ is equivalent to Equation (4). Substituting the optimal thrust direction $\boldsymbol{\alpha}_{opt} = -\boldsymbol{\lambda}_v / \|\boldsymbol{\lambda}_v\| = -\boldsymbol{\lambda}_v/\lambda_v$ into Equation (19) and rearranging terms yields

$$H = \boldsymbol{\lambda}_r^\top \mathbf{v} + \boldsymbol{\lambda}_v^\top \mathbf{g}(\mathbf{r}) + \frac{uT_{\max}}{v_e} \underbrace{\left(1 - \frac{v_e}{m}\lambda_v - \lambda_m\right)}_{=:S} \tag{20}$$

According to Pontryagin's minimum principle,[27] 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.
Note that 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.

Optimization of Switching Times

As states and controls are known only at the nodes, the solution to our OCP is only an approximation. Therefore, any direct method can intrinsically not determine a discontinuous control structure accurately. The switching times (that is, the zeros of S) and costates are also only estimations. Our goal is to refine the values of the switching times by incorporating them into another optimization process to 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 Figure 3. The factors $\Delta^{(k)}$ for the time transformation become

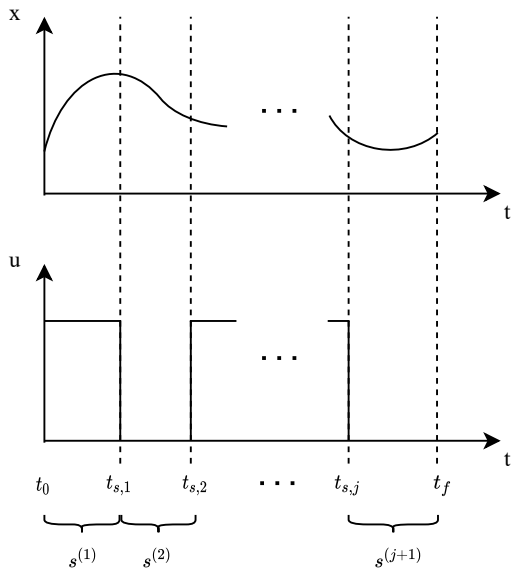


Figure 3: 2D illustration of switching times t_s and segments $s^{(k)}$ for some state x and control u curves.

$$\Delta^{(1)} = \frac{t_{s,1} - t_0}{2}, \qquad \Delta^{(2)} = \frac{t_{s,2} - t_{s,1}}{2}, \qquad \ldots, \qquad \Delta^{(K)} = \frac{t_f - t_{s,j}}{2} \tag{21}$$

and cause the formerly convex dynamical constraints in Equation (12) to become nonconvex. Introducing and linearizing

$$\dot{\mathbf{x}} = \frac{t_{N_k}^{(k)} - t_0^{(k)}}{2}\mathbf{f}(\mathbf{x}, \mathbf{u}) =: \tilde{\mathbf{f}}(\mathbf{x}, \mathbf{u}, t_0^{(k)}, t_{N_k}^{(k)}), \tag{22}$$

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

$$\begin{aligned}\dot{\mathbf{x}} \approx &\tilde{\mathbf{A}}(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \bar{t}_0^{(k)}, \bar{t}_{N_k}^{(k)})\mathbf{x} + \tilde{\mathbf{B}}(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \bar{t}_0^{(k)}, \bar{t}_{N_k}^{(k)})\mathbf{u} + \mathbf{T}_0(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \bar{t}_0^{(k)}, \bar{t}_{N_k}^{(k)})t_0^{(k)} + \mathbf{T}_f(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \bar{t}_0^{(k)}, \bar{t}_{N_k}^{(k)})t_{N_k}^{(k)} \\ &+ \tilde{\mathbf{q}}(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \bar{t}_0^{(k)}, \bar{t}_{N_k}^{(k)})\end{aligned} \tag{23}$$

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 we omit the dependencies for the sake of brevity. The new matrix representation is now

$$\begin{bmatrix} \tilde{\mathbf{M}}_{\text{dyn}} & \tilde{\mathbf{T}} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{X} \\ \mathbf{U} \\ \boldsymbol{\nu} \\ \mathbf{T}_s \end{bmatrix} = \tilde{\mathbf{b}}_{\text{dyn}} \tag{24}$$

where the matrix $\tilde{\mathbf{M}}_{\text{dyn}}$ and the vector $\tilde{\mathbf{b}}_{\text{dyn}}$ are similar to the ones in Equation (15). Moreover, we define $\tilde{\mathbf{T}} = [\mathbf{T}^{(1)}, \mathbf{T}^{(2)}, \ldots, \mathbf{T}^{(K)}]^\top$ with

$$\begin{aligned}\mathbf{T}^{(1)} &= \begin{bmatrix} -\mathbf{T}_{f,1}^{(1)} & \\ \vdots & \mathbf{0} \\ -\mathbf{T}_{f,N_1}^{(1)} & \end{bmatrix}, \quad \mathbf{T}^{(2)} = \begin{bmatrix} -\mathbf{T}_{0,1}^{(2)} & -\mathbf{T}_{f,1}^{(2)} & \\ \vdots & \vdots & \mathbf{0} \\ -\mathbf{T}_{0,N_2}^{(2)} & -\mathbf{T}_{f,N_2}^{(2)} & \end{bmatrix}, \quad \ldots \\ \mathbf{T}^{(K)} &= \begin{bmatrix} & -\mathbf{T}_{0,1}^{(K)} \\ \mathbf{0} & \vdots \\ & -\mathbf{T}_{0,N_K}^{(K)} \end{bmatrix}\end{aligned} \tag{25}$$

After calculating the (estimated) switching times, the thrust magnitude is given by the sign of the switching function:

$$\begin{aligned} T &= 0 \qquad \text{if } S > 0 \\ T &= T_{\text{max}} \qquad \text{if } S < 0 \end{aligned} \tag{26}$$

We can now set the upper and lower bounds of T accordingly. Note that the linear constraint $t_0 \leq t_{s,1} \leq \ldots \leq t_{s,j} \leq t_f$ is to be added to ensure the correct order of the switching times.

ALGORITHM

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

$$\phi \coloneqq J_0 + \sum_{m \in I_{eq}} \mu_m \cdot \|\mathbf{h}_m\|_1 + \sum_{m \in I_{ineq}} \lambda_m \cdot \max(0, g_m) \tag{27}$$

$$\hat{\phi} \coloneqq J_0 + \sum_{m \in I_{eq}} \mu_m \cdot \|\boldsymbol{\nu}_m\|_1 + \sum_{m \in I_{ineq}} \lambda_m \cdot \max(0, \omega_m) \tag{28}$$

$\mathbf{h}_m$ and g_m denote the equality and inequality constraints of the original nonlinear, nonconvex problem. As the thrust direction constraint has been convexified to an inequality constraint, $\omega_m = T_{x,m}^2 + T_{y,m}^2 + T_{z,m}^2 - T_m^2$ appears as a separate term in Equation (28).
The SCP algorithm is based on the one by Mao et al.[28] and shown in Figure 4. We solve the convex optimization problem and compute the actual and predicted cost reductions according to Equations (27) and (28). The maximum constraint violation is calculated using the original nonlinear representations to obtain $c_{max} = \left\| [\mathbf{h}_m, \max(0, g_m)]^\top \right\|_\infty$ over all m. Since feasibility is more important than optimality, a step is accepted as long as the actual 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 $\Delta\phi$ and $\Delta\hat{\phi}$ are positive. The trust region radius is updated and the next iteration starts. The parameters α and β that define the change of the trust region are adjusted based on the values of ρ of the current i and previous iteration $i-1$:

- if $\rho_i \geq (\rho_1 + \rho_2)/2$ and $\rho_{i-1} \geq \rho_0$, we set $\beta = \beta_2$ and $\alpha = \alpha_0$
- if $\rho_i \geq \rho_1$ and $\rho_{i-1} \geq \rho_0$, we set $\beta = \beta_1$ and $\alpha = \alpha_1$
- otherwise, $\beta = \beta_0$ and $\alpha = \alpha_2$

with parameters $1 < \alpha_0 < \alpha_1 < \alpha_2$ and $1 < \beta_0 < \beta_1 < \beta_2$. Our rationale is that if the previous step was accepted, the trust region radius can be increased by a larger value (or decreased by a smaller one) for the next iteration. If instead ρ_i and ρ_{i-1} are small, it is likely that we need to stay closer to the reference solution. Numerical simulations suggest that updating α and β yields faster convergence. The algorithm stops when c_{max} and $\Delta\phi$ are smaller than predefined thresholds. The bang-off-bang mesh refinement can then applied to accurately account for discontinuous control structures.
Such an algorithm that uses variable trust regions, an exact penalty method and virtual controls would eventually converge to a local minimum.[28] Yet, this solution might not be feasible with regard to the nonlinear dynamics. If the final virtual controls and slack variables are zero, however, the converged trajectory is a local optimum of the original, nonconvex problem. This means that the final solution also satisfies the nonlinear dynamics and controls at the nodes.

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).[29,30] Note that GPOPS-II solves the full nonlinear program whereas our SCP algorithm solves the convexified version. The resulting second-order cone program is internally solved by the open source Embedded Conic Solver (ECOS).[31] We compare the number of (major) iterations and computational time. To ensure a fair comparison with GPOPS-II, we do not consider our mesh refinement procedure in the calculations. Likewise, we do not apply any mesh refinement in GPOPS-II. 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. In addition, we implemented the SCP algorithm (without mesh refinement) in Python 3.5 on a Raspberry Pi Model 3 B+ to assess the performance on a single-board computer.

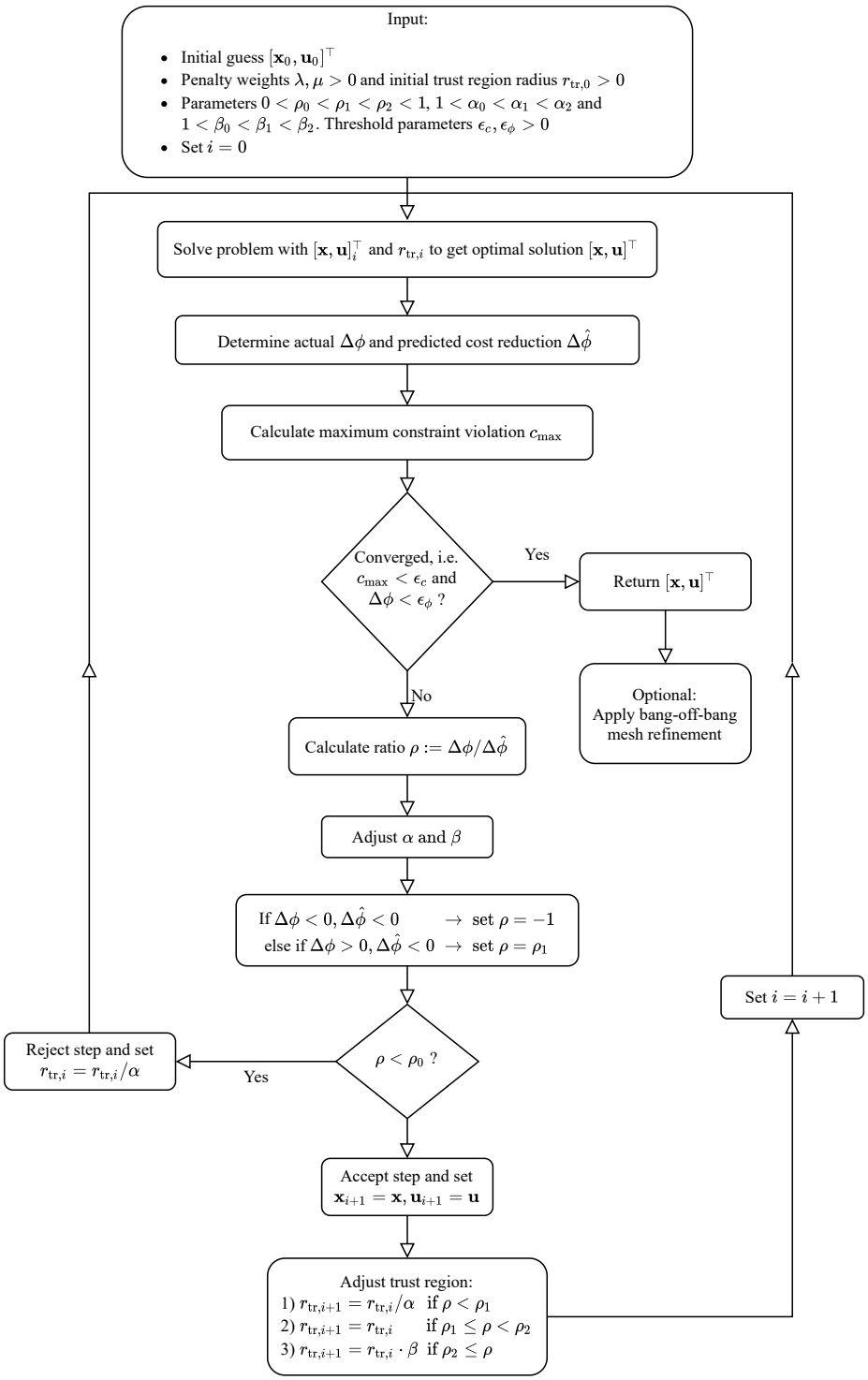


Figure 4: Flow chart of the SCP algorithm.

Furthermore, we assume a constant maximum thrust and specific impulse and two-body dynamics without any additional perturbations. The values of all physical constants are given in Table 1,

additional SCP parameters in Table 2. All variables are scaled with LU, TU, VU, AU and MU, respectively. In order to test the robustness against poor initial guesses, a simple cubic interpolation and a shape-based Fourier series method[32] are used to generate guesses of different quality.
We use in our SCP algorithm the same definition of feasibility as in SNOPT:

$$c_{max} = \max_i \text{viol}_i / \|x\| \leq \epsilon_c \tag{29}$$

where i is the ith nonlinear constraint violation.

Table 1: Physical constants in all simulations.

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

Table 2: Parameters of our SCP algorithm.

Parameter	Value
Penalty weight λ	1.0
Penalty weight μ	1.0
Trust region r_0	1.0
$[\rho_0, \rho_1, \rho_2]$	$[0.01, 0.25, 0.9]$
$[\alpha_0, \alpha_1, \alpha_2]$	$[1.3, 1.5, 1.7]$
$[\beta_0, \beta_1, \beta_2]$	$[1.3, 1.5, 1.7]$
ϵ_c	10^{-6}
ϵ_ϕ	10^{-5}
Max. iterations	500

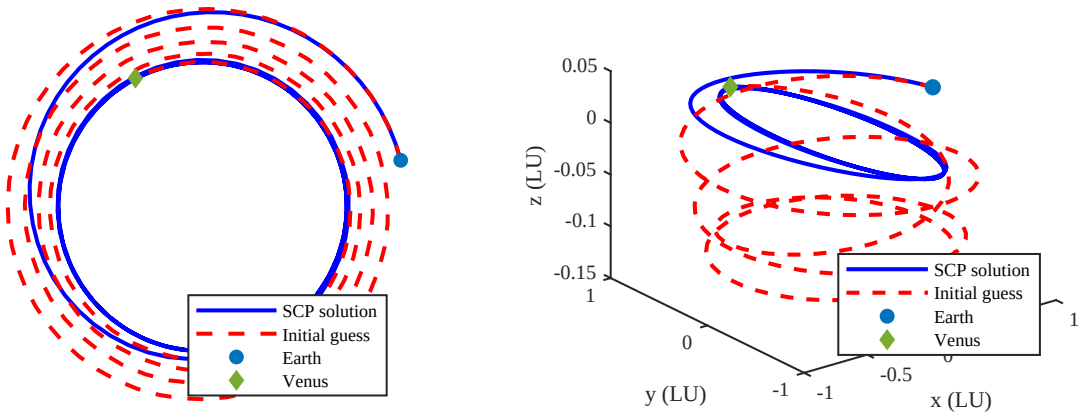
Earth to Venus Transfer

We compare the Earth to Venus rendezvous transfer with the results of an indirect method in the literature.[33] Boundary conditions along with other relevant parameters are summarized in Table 3.

Table 3: Simulation values for Earth-Venus transfer.[33]

Parameter	Value
Initial position $[r_x, r_y, r_z]_0^\top$	$[0.9708, 0.2376, -1.6711 \cdot 10^{-6}]^\top$ LU
Initial velocity $[v_x, v_y, v_z]_0^\top$	$[-0.2545, 0.9687, 1.5040 \cdot 10^{-5}]^\top$ VU
Initial mass $m(t_0)$	1500 kg
Final position $[r_x, r_y, r_z]_f^\top$	$[-0.3277, 0.6389, 0.0277]^\top$ LU
Final velocity $[v_x, v_y, v_z]_f^\top$	$[-1.0509, -0.5436, 0.0532]^\top$ VU
Final mass $m(t_f)$	free
Maximum thrust T_{max}	0.33 N
Specific impulse I_{sp}	3800 s
Time of flight t_f	1000 days

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. The trajectory that corresponds to the *Cubic4* guess is illustrated in Figure 5 and shows the large discrepancy



(a) Polar plot of trajectory

(b) Trajectory in Cartesian coordinates

Figure 5: Example of Earth to Venus trajectory in polar and Cartesian coordinates with Cubic3 initial guess.

between the initial guess and final solution. Note that we calculated the trajectories also on the Raspberry Pi. The results in Table 4 show that our SCP algorithm required considerably fewer

Table 4: Results of Earth to Venus transfer on the initial mesh (without mesh refinement).

Element \ Initial Guess		Cubic2	Cubic3	Cubic4	FFS2	FFS3	FFS4
Iterations	GPOPS-II	500*	137	291	135	492	136
	SCP	28	24	32	37	20	26
Comp. time (s)	GPOPS-II	44.9	13.1	24.3	13.4	45.1	14.9
	SCP	5.4	7.1	8.4	9.4	6.1	6.8
	SCP (Pi)	144.4	147.5	203.2	228.0	126.0	159.7
$m(t_f)$ (kg)	GPOPS-II	943	1278	1066	978	1275	1184
	SCP	1038	1287	1245	1038	1290	1258
$m(t_f)$ (kg) from indirect method[33]		1007, 1036, 1260, 1291					

* Iteration limit reached without convergence.

iterations compared to GPOPS-II. As expected, the computational times are also lower, but less significant (only by a factor of two to three). Moreover, SCP was able to determine higher final masses than GPOPS-II. When compared to results from an indirect method[33] (last row of Table 4), it is evident that the obtained values agree well with the reported ones regardless of the initial guess. Propagating the nonlinear equations of motion with the obtained controls yield final position errors $\left\|\mathbf{x}(t_f)_{\text{propagated}} - \mathbf{x}(t_f)_{\text{solver}}\right\|_2$ of the order 10^4 to 10^5 km (SCP) and 10^5 to 10^6 km (GPOPS-II). We noticed that GPOPS-II often obtained control histories with oscillations, hence resulting in non-optimal solutions with lower accuracy.

The SCP algorithm was also able to determine the same trajectories on the Raspberry Pi as on the desktop computer. As expected, the computing times increase considerably due to the lower clock

speed on a single-board computer (see row *SCP (Pi)* in Table 4). Nevertheless, the average run time of approximately two to three minutes seems acceptable for a deep-space cruise that often takes months or even years.

Bang-Off-Bang Mesh Refinement. As the virtual controls and slack variables were zero in all converged SCP simulations, the solutions are locally optimal according to Section V. This is also evident from the characteristic bang-off-bang control structure in Figure 6a. 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 and agrees very well when compared to an indirect method.[33]

Perturbed Initial Condition. We perturb each component of the initial condition randomly 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 shown in Figure 7a. As we are mainly interested in feasible trajectories, we stop the algorithm when the feasibility threshold is reached.
The results are reported in Table 5 where mean values $\pm$ 1σ are shown. Despite the very large displacements, the SCP algorithm converged in all cases and could determine final masses that are close to a locally optimal one of 1290 kg.[33] Still, more iterations were required on average than in the unperturbed case. Computational times, in contrast, are similar.

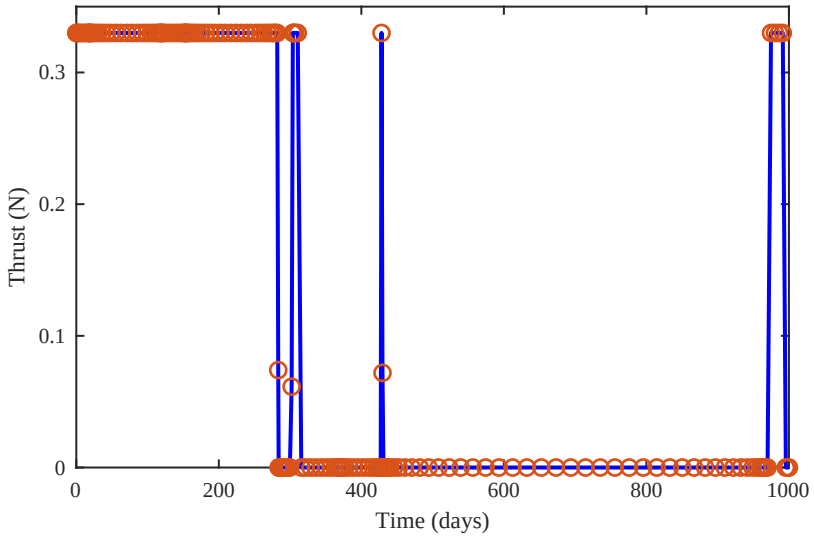
Table 5: Overview of results after 1000 simulations for Earth to Venus and Earth to Dionysus transfers using simple cubic interpolations for the initial guesses.

Transfer	Total sim.	Converged sim.	# iter.	CPU time (s)	$m(t_f)$ (kg)
Venus	1000	1000	31.9 ± 4.2	7.0 ± 1.1	1280 ± 20
Dion.	1000	999	48.4 ± 23.2	26.3 ± 13.4	2510 ± 118

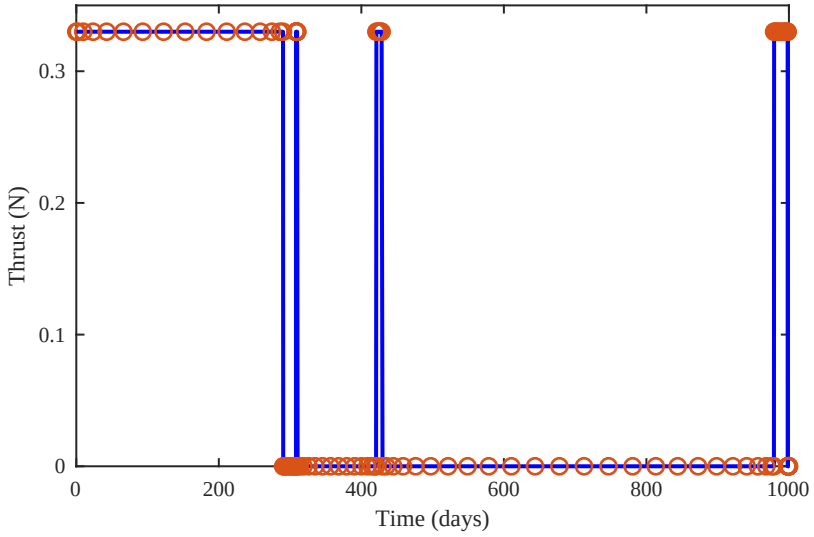
Earth to Asteroid Dionysus Transfer

We now consider a more complex example from the literature:[34] the low-thrust transfer from Earth to asteroid Dionysus with a flight time of 3534 days (see Table 6). Several revolutions with significant changes in semi-major axis ($\Delta a = 1.2$ LU), eccentricity ($\Delta e = 0.52$) and inclination ($\Delta i = 13.5°$) are required to reach the target.

Comparison with GPOPS-II. We increase the number of segments to 25 with 10 nodes each as t_f is considerably higher in this example. Figure 8 shows the polar and three-dimensional plots where the higher number of revolutions can be observed. Although the algorithm converged in all calculations, the discrepancy between GPOPS-II and SCP becomes clearer. Again, the latter required up to one order of magnitude fewer iterations and less computing time than GPOPS-II (see Table 7). Due to the more complex example with more nodes, the calculation times on the Raspberry Pi also increased to few minutes. Moreover, final masses of GPOPS-II are notably lower than those of SCP because GPOPS-II could not find optimal solutions within the iteration limit in several cases. In addition, the oscillations in the controls obtained with GPOPS-II became larger and resulted in a higher error when propagating the nonlinear equations of motion.

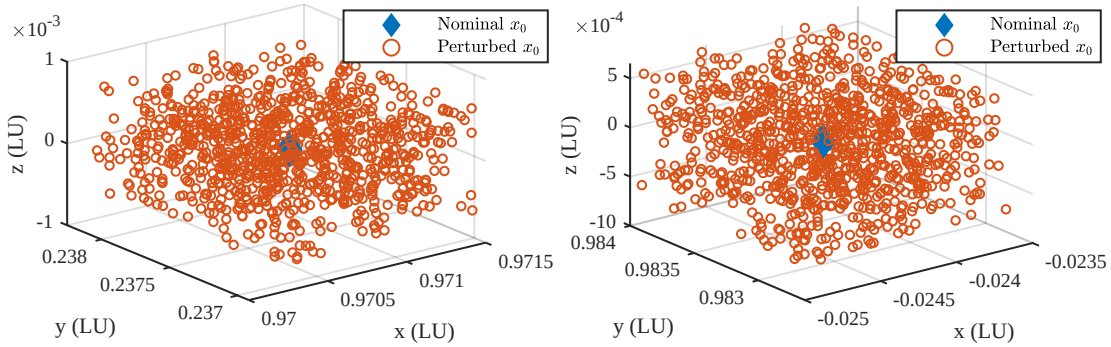


(a) Thrust magnitude before mesh refinement (N = 150).



(b) Thrust magnitude after mesh refinement (N = 100)

Figure 6: Comparison of thrust magnitude before and after bang-off-bang mesh refinement for Earth to Venus transfer calculated with SCP. Circles represent the nodes.



(a) Earth to Venus transfer

(b) Earth to Dionysus transfer

Figure 7: Perturbed initial conditions for Earth to Venus and Earth to Dionysus transfers.

Table 6: Simulation values for Earth-Dionysus transfer.[34]

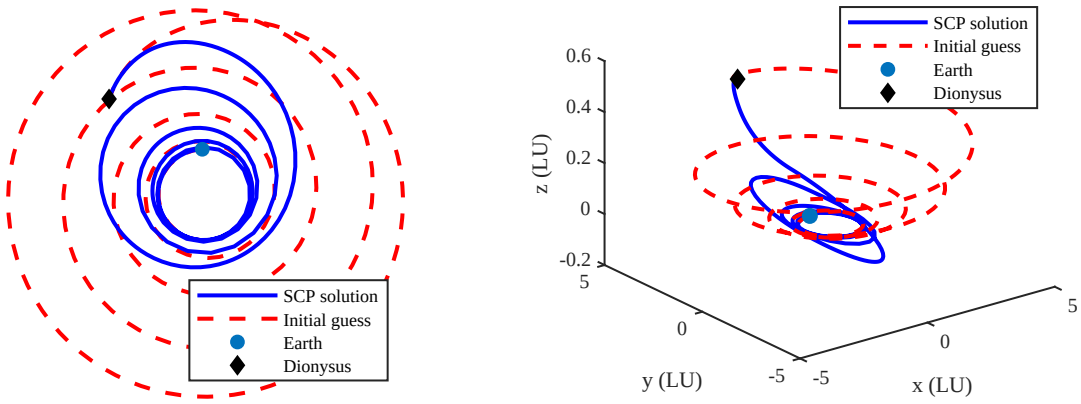
Parameter	Value
Initial position $[r_x, r_y, r_z]_0^\top$	$[-0.0243, 0.9833, -1.5117 \cdot 10^{-5}]^\top$ LU
Initial velocity $[v_x, v_y, v_z]_0^\top$	$[-1.0161, -0.0285, 1.6955 \cdot 10^{-6}]^\top$ VU
Initial mass $m(t_0)$	4000 kg
Final position $[r_x, r_y, r_z]_f^\top$	$[-2.0406, 2.0518, 0.5543]^\top$ LU
Final velocity $[v_x, v_y, v_z]_f^\top$	$[-0.1423, -0.4511, 0.0189]^\top$ VU
Final mass $m(t_f)$	free
Maximum thrust T_{max}	0.32 N
Specific impulse I_{sp}	3000 s
Time of flight t_f	3534 days

Bang-Off-Bang Mesh Refinement. Even in this more complex transfer, the mesh refinement can accurately determine a control structure without intermediate points (see Figure 9). 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.

Perturbed Initial Condition. We proceed in a similar way and perturb the initial condition (see Figure 7b). Compared to the Venus transfer, SCP 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}). Note that this issue could be resolved by adding more nodes or employing a more sophisticated updating of α and β.

CONCLUSION

We developed a refined version of the sequential convex programming algorithm that can be used for solving the low-thrust trajectory optimization problem. The combination of an adaptive flipped Radau pseudospectral scheme with a bang-off-bang mesh refinement strategy and measures to avoid virtual infeasibility proved effective in terms of accuracy, computational effort and optimality when compared to the literature and state-of-the-art solvers. We demonstrated the high robustness of our



(a) Polar plot of trajectory

(b) Trajectory in Cartesian coordinates

Figure 8: Example of Earth to Dionysus trajectory in polar and Cartesian coordinates with Cubic5 initial guess.

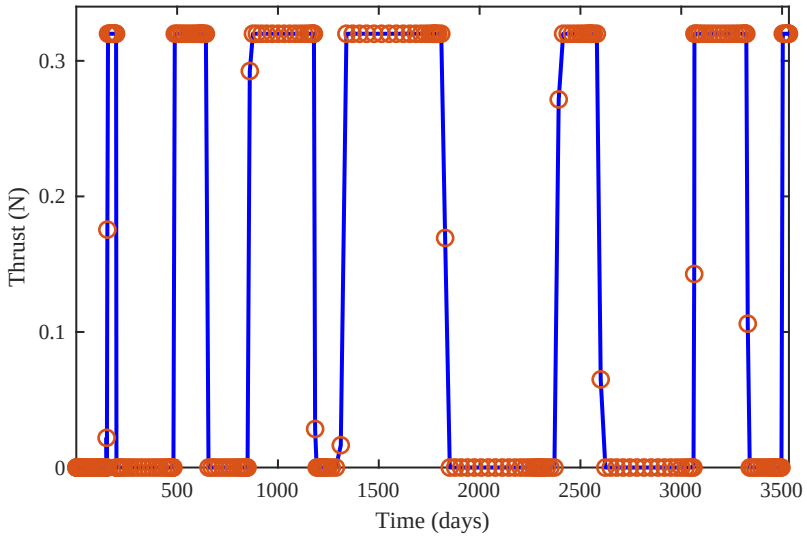
Table 7: Results of Earth to Dionysus transfer on the initial mesh (without mesh refinement).

Element \ Initial Guess		Cubic4	Cubic5	Cubic6	FFS4	FFS5	FFS6
Iterations	GPOPS-II	500*	500*	500*	500*	370	311
	SCP	40	52	45	31	25	26
Comp. time (s)	GPOPS-II	97.3	131.8	109.9	330.8	117.2	76.2
	SCP	20.9	31.5	25.2	16.4	14.1	14.4
	SCP (Pi)	263.1	478.2	321.5	243.5	180.5	184.5
$m(t_f)$ (kg)	GPOPS-II	1984	1706	1843	2093	1964	1547
	SCP	2352	2614	2586	2404	2445	2312
$m(t_f)$ (kg) from indirect method[34]		1980, 2227, 2465, 2672, 2718					

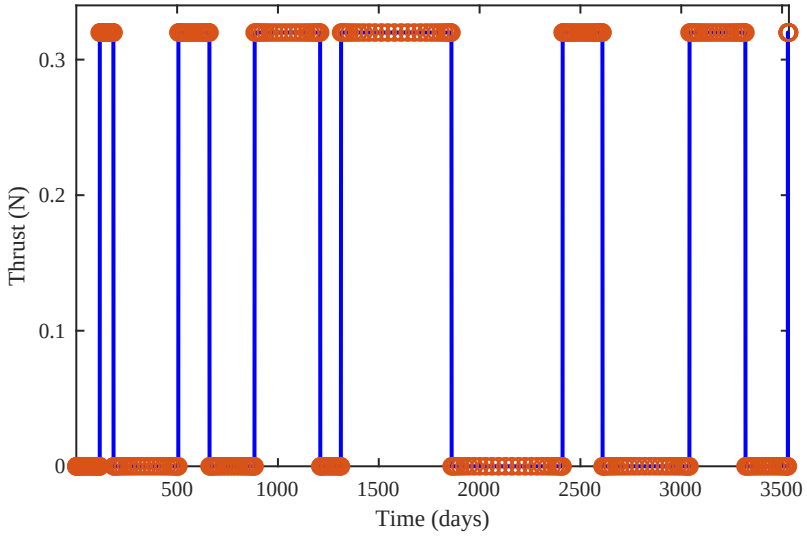
* Iteration limit reached without convergence.

approach in orbital transfers from Earth to Venus and Earth to asteroid Dionysus with poor initial guesses.

The method is a promising first step towards autonomous guidance in a real deep-space cruise where high accuracy is not crucial. The run times of only few minutes on a single-board computer seem to be acceptable considering the long duration of interplanetary transfers. In contrast to related work that focused mainly on machine learning techniques, we addressed the guidance problem with numerical optimization methods. As the proposed algorithm actually calculates and optimizes trajectories, we avoid the dependency on the quality of training data. Despite the lower accuracy compared to nonlinear programming methods in general, the simulations suggest that our method can still achieve a sufficient level of optimality and accuracy while being faster, more robust and compatible with the limited resources of a single-board computer.



(a) Thrust magnitude before mesh refinement (N = 250)



(b) Thrust magnitude after mesh refinement (N = 293)

Figure 9: Comparison of thrust magnitude before and after bang-off-bang mesh refinement for Earth to Dionysus transfer calculated with SCP. Circles represent the nodes.

REFERENCES

[1] A. Poghosyan and A. Golkar, "CubeSat evolution: Analyzing CubeSat capabilities for conducting science missions," *Progress in Aerospace Sciences*, Vol. 88, 2017, pp. 59–83, doi: 10.1016/j.paerosci.2016.11.002.

[2] R. Walker and e. al., "Deep-space CubeSats: thinking inside the box," *Progress in Aerospace Sciences*, Vol. 59, No. 5, 2018, pp. 24–30, doi: 10.1093/astrogeo/aty232.

[3] C. Zhang, F. Topputo, F. Bernelli-Zazzera, and Y. S. Zhao, "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, doi: 10.2514/1.G001080.

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

[5] G. Lantoine and R. P. Russell, "A Hybrid Differential Dynamic Programming Algorithm for Constrained Optimal Control Problems. Part 1: Theory," *Journal of Optimization Theory and Applications*, Vol. 154, No. 2, 2012, pp. 382–417, doi: 10.1007/s10957-012-0038-1.

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

[7] L. T. Biegler and V. M. Zavala, "Large-scale nonlinear programming using IPOPT: An integrating framework for enterprise-wide dynamic optimization," *Computers and Chemical Engineering*, Vol. 33, No. 3, 2009, pp. 575–582, doi: 10.1016/j.compchemeng.2008.08.006.

[8] J. M. S. P. Gabor I. Varga, "Many-Revolution Low-Thrust Orbit Transfer Computation using Equinoctial Q-Law Including J2 and Eclipse Effects," *6th International Conference on Astrodynamics Tools and Techniques*, No. 1, 2016.

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

[10] D. Izzo, C. Sprague, and D. Tailor, "Machine Learning and Evolutionary Techniques in Interplanetary Trajectory Design," arXiv 1802.00180, 9 2018.

[11] R. Furfaro, I. Bloise, M. Orlandelli, P. Di, F. Topputo, and R. Linares, "A Recurrent Deep Architecture for Quasi-Optimal Feedback Guidance in Planetary Landing," *IAA SciTech Forum on Space Flight Mechanics and Space Structures and Materials*, 2018, pp. 1–24.

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

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

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

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

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

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

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

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

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

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

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

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

[24] Y. Agamawi, W. W. Hager, and A. V. Rao, "Mesh Refinement Method for Solving Bang-Bang Optimal Control Problems using Direct Collocation," arXiv 1905.11895, 5 2019.

[25] D. Garg, *Advances in Global Pseudospectral Methods for Optimal Control*. PhD thesis, University of Florida, 2011.

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

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

[28] Y. Mao, M. Szmuk, X. Xu, and B. Acikmese, "Successive Convexification: A Superlinearly Convergent Algorithm for Non-convex Optimal Control Problems," arXiv 1804.06539, 2 2019.

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

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

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

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

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

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