CLOSED-LOOP GUIDANCE FOR LOW-THRUST INTERPLANETARY TRAJECTORIES USING CONVEX PROGRAMMING

Christian Hofmann$^{(1)}$ **and Francesco Topputo**$^{(2)}$

$^{(1)}$ *Politecnico di Milano, Via La Masa 34, 20156 Milan, Italy, christian.hofmann@polimi.it*
$^{(2)}$ *Politecnico di Milano, Via La Masa 34, 20156 Milan, Italy, francesco.topputo@polimi.it*

ABSTRACT

A closed-loop guidance method for low-thrust fuel-optimal interplanetary transfers is developed. The algorithm is based on convex programming and a flipped Radau pseudospectral discretization scheme. New reference trajectories are repeatedly recomputed in certain time intervals. The state of the spacecraft is propagated during these periods using the obtained controls until a new trajectory is to be calculated. A mesh refinement procedure adjusts the number of nodes based on the linearization error. The effectiveness of the approach is demonstrated in a transfer from Earth to an asteroid. The proposed method is a promising step towards autonomous guidance in real space missions due to its rapid speed and excellent robustness.

1 INTRODUCTION

The interest in and market for new space missions have been growing tremendously in the past few years; especially asteroids and planets are popular targets [1]. Due to the ongoing miniaturization, CubeSats are now a viable low-cost alternative to conventional satellites [2]. The state of the art is to operate all spacecraft from ground. However, this limits the mission design and will soon be unsustainable considering the increasing number of launches [3]. As technology advances, there is a need and desire to gradually increase the autonomy of spacecraft [4]. Even though performing regular checks on ground will still be required, human intervention can be limited to a minimum. Flight-related tasks such as Guidance, Navigation, and Control (GNC) are particularly important as they are essential during the whole lifetime of a spacecraft, often several years for interplanetary transfers. As the operational costs do not scale with the system mass [5], more and more researchers address on-board GNC. Similar to autonomous driving, autonomous GNC seems to slowly become reality as technology advances [4, 6, 7, 8].
In contrast to autonomous navigation where some promising results have been obtained recently [9], literature on real-time guidance and control is scarce. Model predictive control is a popular choice for tracking a reference trajectory [10]. The guidance design, however, requires solving a nonlinear optimal control problem (OCP). This is already a demanding task. Shifting it on-board is therefore a great challenge and poses risks as the algorithm must repeatedly recompute the reference trajectory in real-time. The main criteria reliability (a feasible solution must be obtained at any instant), optimality (a cost function is to be minimized) and compatibility (the algorithm must be compatible with available on-board hardware) are essential for autonomous guidance. As the guidance design has always been performed on ground, none of the current techniques fulfills all criteria. Common methods are often divided into direct and indirect approaches. Even though the latter tend to be advantageous for

transfers with hundreds of spirals, their relatively high sensitivity to the initial guess is critical for real-time applications [11]. Direct methods, in contrast, are often more robust. They transcribe the infinite-dimensional OCP into a finite-dimensional mathematical programming problem. Solving the resulting nonlinear programming problem is often computationally expensive and may also require a decent initial guess [12]. As spacecraft computers have only limited computational power, this approach is not an option on board. Recently, sequential convex programming (SCP) techniques have been regarded as a viable alternative [6, 13]. Instead of solving the full nonlinear program, an easier convex problem is solved. One key advantage is that a rather poor initial guess (that does not need to satisfy the constraints) is often sufficient. Moreover, solving a series of simpler convex problems allows to use sophisticated interior-point methods [14]. As only first-order information is required, the algorithm becomes tractable if sparse linear algebra is exploited. SCP is thus a popular choice for powered descent and landing problems [15, 16]. Moreover, the constrained attitude control problem will be solved with convex programming techniques in real-time in space soon [17]. Applications to the low-thrust trajectory optimization problem are rather recent [18]. In [19], SCP is used to generate an initial guess for an indirect method. Other researchers address the trajectory optimization problem for solar-sail missions [20] or solve the circular restricted three-body problem in cislunar space [21]. The work in [22] focuses on reliability and low computational effort; it demonstrates that SCP is able to solve complex interplanetary transfers in little time. In [23], a SCP algorithm was implemented on a single-board computer to assess the performance on power-limited hardware. However, it was not investigated how SCP performs when integrated into a closed-loop guidance simulation. This task is particularly demanding as the algorithm must reoptimze the trajectory several times and at most guarantee a feasible solution. In this work, we create a closed-loop algorithm that is fed with the current spacecraft state and repeatedly determines new trajectories in certain intervals. Instead of using the more common differential form of the flipped Radau pseudospectral discretization, we formulate the dynamics using the integral formulation. A mesh refinement is added where the linearization error is large to obtain feasible solutions that satisfy the nonlinear dynamics. A transfer to an asteroid is considered to assess the overall performance.
The paper is structured as follows. In Section 2, we state the optimal control problem and convert it into a convex optimization problem. Section 3 addresses the mesh refinement procedure in detail. The closed-loop guidance method is described in Section 4 and the results of the numerical simulations are presented in Section 5. Finally, Section 6 concludes this paper.

2 PROBLEM FORMULATION

2.1 Optimal Control Problem and Convexification

Using a simple two-body model with the Sun as the primary, the equations of motion in Cartesian coordinates are given by

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \Rightarrow \begin{bmatrix} \dot{\mathbf{r}} \\ \dot{\mathbf{v}} \\ \dot{m} \end{bmatrix} = \begin{bmatrix} \mathbf{v} \\ -\mu\mathbf{r}/r^3 + \mathbf{T}/m \\ -\left\|\mathbf{T}\right\|/(g_0 I_{\text{sp}}) \end{bmatrix} \tag{1}$$

where $\mathbf{r}$, $\mathbf{v}$, and m are the position, velocity and mass, respectively. μ is the gravitational constant, $\mathbf{T}$ is the thrust vector, g_0 is the gravitational acceleration at sea level and I_{sp} the specific impulse which is assumed constant throughout this paper. If not stated otherwise, $\left\|\cdot\right\|$ refers to the 2-norm. The fuel-optimal control problem in space flight then is to minimize

$$J_0 = \int_{t_0}^{t_f} \|\mathbf{T}\| \, \mathrm{d}t \tag{2}$$

subject to

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \tag{3a}$$

$$0 \le \|\mathbf{T}\| \le T_{\text{max}} \tag{3b}$$

$$\mathbf{x}(t_0) = \mathbf{x}_0, \quad \mathbf{x}(t_f) = \mathbf{x}_f \tag{3c}$$

$$\mathbf{x}_l \le \mathbf{x} \le \mathbf{x}_u, \quad \mathbf{u}_l \le \mathbf{u} \le \mathbf{u}_u \tag{3d}$$

where the subscripts 0 and f refer to initial and final values, respectively. Eq. 3d defines the lower (subscript l) and upper bounds (subscript u) for states and controls, respectively.
We change the variables to $\tau = \|\mathbf{T}\|/m$, $\boldsymbol{\tau} = \mathbf{T}/m$ and $z = \ln m$, and define the new states and controls as $\mathbf{x} = [\mathbf{r}, \mathbf{v}, z]^\top$ and $\mathbf{u} = [\boldsymbol{\tau}, \tau]^\top$. This allows us to decouple states and controls in Eq. 1 [24]:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\mathbf{r}} \\ \dot{\mathbf{v}} \\ \dot{z} \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{v} \\ -\mu\mathbf{r}/r^3 \\ 0 \end{bmatrix}}_{\mathbf{p}(\mathbf{x})} + \underbrace{\begin{bmatrix} \multicolumn{2}{c}{\mathbf{0}_{3\times4}} \\ \mathbf{1}_{3\times3} & \mathbf{0}_{3\times1} \\ \mathbf{0}_{1\times3} & -1/(g_0 I_{\text{sp}}) \end{bmatrix}}_{\mathbf{B}} \begin{bmatrix} \boldsymbol{\tau} \\ \tau \end{bmatrix} = \mathbf{p}(\mathbf{x}) + \mathbf{B}\mathbf{u} \tag{4}$$

where $\mathbf{1}$ is the identity matrix. Linearizing the dynamics in Eq. 4 at a reference point $\bar{\mathbf{x}}$ yields

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

with the Jacobian matrix $\mathbf{A}(\bar{\mathbf{x}}) = \left.\frac{\partial \mathbf{p}}{\partial \mathbf{x}}\right|_{\mathbf{x}=\bar{\mathbf{x}}}$ and the constant part $\mathbf{q}(\bar{\mathbf{x}}) = \mathbf{p}(\bar{\mathbf{x}}) - \mathbf{A}(\bar{\mathbf{x}})\bar{\mathbf{x}}$. As the constraint on the thrust magnitude in Eq. 3b now becomes $0 \le \tau \le T_{\text{max}}\mathrm{e}^{-z}$, we also linearize it at some reference $\bar{z}$ to obtain

$$0 \le \tau \le T_{\text{max}}\mathrm{e}^{-\bar{z}}\left(1 - z + \bar{z}\right) \tag{6}$$

The original nonlinear optimal control problem in Eqs. 2-3 can now be formulated as a convex problem where we minimize

$$J_0 = \int_{t_0}^{t_f} \tau \, \mathrm{d}t \tag{7}$$

subject to

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

$$0 \le \|\boldsymbol{\tau}\| \le \tau \tag{8b}$$

$$0 \le \tau \le T_{\text{max}}\mathrm{e}^{-\bar{z}}\left(1 - z + \bar{z}\right) \tag{8c}$$

$$\|\mathbf{x} - \bar{\mathbf{x}}\|_1 \le r_{\text{tr}} \tag{8d}$$

$$\mathbf{x}(t_0) = \mathbf{x}_0, \quad \mathbf{x}(t_f) = \mathbf{x}_f \tag{8e}$$

$$\mathbf{x}_l \le \mathbf{x} \le \mathbf{x}_u, \quad \mathbf{u}_l \le \mathbf{u} \le \mathbf{u}_u \tag{8f}$$

The constraint in Eq. 8b is a relaxed version of $\|\boldsymbol{\tau}\| = \tau$. It can be shown that the optimal solution of the relaxed problem is also an optimal solution of the original problem [24]. The trust region constraint in Eq. 8d with the trust region radius r_{tr} is enforced to keep the linearization accurate enough.
As various simulations suggest [22], we add an unconstrained virtual control $\boldsymbol{\nu} \in \mathbb{R}^{n_x}$ ($n_x = 7$ is the number of states) to Eq. 8a and relax the linearized constraint in Eq. 8c with $\eta \geq 0$:

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

$$0 \leq \tau \leq T_{\text{max}} \mathrm{e}^{-\bar{z}} \left(1 - z + \bar{z}\right) + \eta \tag{10}$$

This prevents the problem to become artificially infeasible. As $\boldsymbol{\nu}$ and η would result in additional constraint violations, we need to penalize them in the objective function with sufficiently large penalty parameters μ_m (for the set of equality constraints I_{eq}) and λ_m (for the set of inequality constraints I_{ineq}):

$$J = J_0 + \sum_{m \in I_{\text{eq}}} \mu_m \left\|\boldsymbol{\nu}_m\right\|_1 + \sum_{m \in I_{\text{ineq}}} \lambda_m \max(0, \eta_m) \tag{11}$$

It directly follows that an optimal solution requires $\boldsymbol{\nu}$ and η to be zero.

2.2 Integral Flipped Radau Pseudospectral Discretization

We only address the dynamical constraints here; the reader is referred to [22, 25] for more details on the flipped Radau pseudospectral method (FRPM). The trajectory is divided into segments and the dynamics are approximated in each segment at the roots of the flipped Legendre–Radau polynomial. $\mathbf{X}_i^{(k)}$, $\mathbf{U}_i^{(k)}$ denote the i-th node of the k-th segment of states and controls at time $t_i^{(k)}$, where $i = 0, 1, ..., N_k$ and $k = 1, ..., K$, N_k being the number of collcaotion points in segment k. The differential operator is approximated as $\dot{\mathbf{X}} \approx \mathbf{D}\mathbf{X}$ with the differentiation matrix $\mathbf{D}$. Thus, the discrete dynamics read in the FRPM for each segment k:

$$\sum_{j=0}^{N_k} D_{ij}^{(k)} \mathbf{X}_j^{(k)} = \frac{t_{N_k}^{(k)} - t_0^{(k)}}{2} \mathbf{f}(\mathbf{X}_i^{(k)}, \mathbf{U}_i^{(k)}) \qquad i = 1, ..., N_k \tag{12}$$

with $D_{ij}^{(k)}$ being the entries of $\mathbf{D}$. The factor $(t_{N_k}^{(k)} - t_0^{(k)})/2$ stems from the transformation between the physical time t and the pseudospectral time that is defined in (-1, 1]. As demonstrated in [25], there is an equivalent integral form of Eq. 12 that uses the integration matrix $\mathbf{I} = \mathbf{D}_{:,1:N}^{-1}$ (the subscript $:, 1 : N$ refers to columns 1 to N):

$$\mathbf{X}_i^{(k)} = \mathbf{X}_0^{(k)} + \frac{t_{N_k}^{(k)} - t_0^{(k)}}{2} \sum_{j=1}^{N_k} I_{ij}^{(k)} \mathbf{f}(\mathbf{X}_j^{(k)}, \mathbf{U}_j^{(k)}) \qquad i = 1, ..., N_k \tag{13}$$

Substituting the linearized dynamics of Eq. 9 into Eq. 13 yields

$$\mathbf{X}_i^{(k)} = \mathbf{X}_0^{(k)} + \frac{t_{N_k}^{(k)} - t_0^{(k)}}{2} \sum_{j=1}^{N_k} I_{ij}^{(k)} \left[\mathbf{A}(\bar{\mathbf{X}}_j^{(k)})\mathbf{X}_j^{(k)} + \mathbf{B}\mathbf{U}_j^{(k)} + \mathbf{q}(\bar{\mathbf{X}}_j^{(k)}) + \boldsymbol{\nu}_j^{(k)}\right] \qquad i = 1, ..., N_k \tag{14}$$

Considering that the initial node is not a collocation point and that $\mathbf{X}_{N_k}^{(k-1)} = \mathbf{X}_0^{(k)}$ for $k > 1$, we can write the dynamics in the standard linear form $\mathbf{MY} = \mathbf{b}$:

$$
\begin{aligned}
&\underbrace{\left[\begin{array}{c:c:c:c:cccc:c}
\hat{\mathbf{I}}^{(1)} & \mathbf{0} & \mathbf{0} & & \hat{\mathbf{B}}^{(1)} & & & \mathbf{0} & \\
\hdashline
\mathbf{0} \quad -\hat{\mathbf{1}} & \hat{\mathbf{I}}^{(2)} & \mathbf{0} & \cdots & & \hat{\mathbf{B}}^{(2)} & & & \\
\hdashline
\mathbf{0} & \mathbf{0} \quad -\hat{\mathbf{1}} & \hat{\mathbf{I}}^{(3)} & & & & \hat{\mathbf{B}}^{(3)} & & \mathbf{I}_\nu \\
 & & & \ddots & & \mathbf{0} & & \ddots & \\
 & \vdots & & \hat{\mathbf{I}}^{(K)} & & & & \hat{\mathbf{B}}^{(K)} &
\end{array}\right]}_{\mathbf{M}} \cdot \underbrace{\begin{bmatrix} \mathbf{X} \\ \mathbf{U} \\ \boldsymbol{\nu} \end{bmatrix}}_{\mathbf{Y}} \\
&= \underbrace{\begin{bmatrix}
\mathbf{X}_0 + \Delta^{(1)} \sum_{j=1}^{N_1} I_{1j}^{(1)} \mathbf{q}(\bar{\mathbf{X}}_j^{(1)}) \\
\mathbf{X}_0 + \Delta^{(1)} \sum_{j=1}^{N_1} I_{2j}^{(1)} \mathbf{q}(\bar{\mathbf{X}}_j^{(1)}) \\
\vdots \\
\mathbf{X}_0 + \Delta^{(1)} \sum_{j=1}^{N_1} I_{N_1 j}^{(1)} \mathbf{q}(\bar{\mathbf{X}}_j^{(1)}) \\
\Delta^{(2)} \sum_{j=1}^{N_2} I_{1j}^{(2)} \mathbf{q}(\bar{\mathbf{X}}_j^{(2)}) \\
\vdots \\
\Delta^{(K)} \sum_{j=1}^{N_K} I_{N_K j}^{(K)} \mathbf{q}(\bar{\mathbf{X}}_j^{(K)})
\end{bmatrix}}_{\mathbf{b}}
\end{aligned}
\tag{15}
$$

with

$$
\hat{\mathbf{I}}^{(k)} = \begin{bmatrix}
\mathbf{1}_7 - \Delta^{(k)} I_{11}^{(k)} \mathbf{A}_1^{(k)} & -\Delta^{(k)} I_{12}^{(k)} \mathbf{A}_2^{(k)} & \cdots & -\Delta^{(k)} I_{1N_k}^{(k)} \mathbf{A}_{N_k}^{(k)} \\
-\Delta^{(k)} I_{21}^{(k)} \mathbf{A}_1^{(k)} & \mathbf{1}_7 - \Delta^{(k)} I_{22}^{(k)} \mathbf{A}_2^{(k)} & \cdots & -\Delta^{(k)} I_{2N_k}^{(k)} \mathbf{A}_{N_k}^{(k)} \\
\vdots & \vdots & & \vdots \\
-\Delta^{(k)} I_{N_k 1}^{(k)} \mathbf{A}_1^{(k)} & -\Delta^{(k)} I_{N_k 2}^{(k)} \mathbf{A}_2^{(k)} & \cdots & \mathbf{1}_7 - \Delta^{(k)} I_{N_k N_k}^{(k)} \mathbf{A}_{N_k}^{(k)}
\end{bmatrix} \tag{16}
$$

$$
\hat{\mathbf{B}}^{(k)} = \begin{bmatrix}
-\Delta^{(k)} I_{11}^{(k)} \mathbf{B}_1^{(k)} & -\Delta^{(k)} I_{12}^{(k)} \mathbf{B}_2^{(k)} & \cdots & -\Delta^{(k)} I_{1N_k}^{(k)} \mathbf{B}_{N_k}^{(k)} \\
-\Delta^{(k)} I_{21}^{(k)} \mathbf{B}_1^{(k)} & -\Delta^{(k)} I_{22}^{(k)} \mathbf{B}_2^{(k)} & \cdots & -\Delta^{(k)} I_{2N_k}^{(k)} \mathbf{B}_{N_k}^{(k)} \\
\vdots & \vdots & & \vdots \\
-\Delta^{(k)} I_{N_k 1}^{(k)} \mathbf{B}_1^{(k)} & -\Delta^{(k)} I_{N_k 2}^{(k)} \mathbf{B}_2^{(k)} & \cdots & -\Delta^{(k)} I_{N_k N_k}^{(k)} \mathbf{B}_{N_k}^{(k)}
\end{bmatrix} \tag{17}
$$

We introduced the notation $\Delta^{(k)} = (t_{N_k}^{(k)} - t_0^{(k)})/2$ and $\mathbf{A}_j^{(k)} = \mathbf{A}(\bar{\mathbf{X}}_j^{(k)})$ ($\mathbf{B}$ and $\mathbf{q}$ are defined accordingly) for conciseness. $\hat{\mathbf{1}} \in \mathbb{R}^{N_k n_x \times n_x}$ consists of vertically concatenated identity matrices. $\mathbf{I}_\nu$ has the same structure as the $\mathbf{B}$ matrix with the only difference that all $\mathbf{B}_j^{(k)}$ are replaced by the identity matrix $\mathbf{1}_7$. $\mathbf{X}$, $\mathbf{U}$ and $\boldsymbol{\nu}$ are the concatenated states, controls, and virtual controls, respectively. The representation in Eq. 15 allows us to use standard solvers to solve the discrete convex optimization problem.

3 MESH REFINEMENT

A common problem in direct methods is that the number of nodes needed to achieve a certain accuracy is not known a priori. A poor discretization can result in large constraint violations or even in non-convergence as the dynamics cannot be satisfied. This is especially problematic for convex problems where the original nonlinear constraints are convexified and linearized. Even though the incorporation of virtual controls eliminates the problem of artificial infeasibility, it does not necessarily mean that the solver is able to drive the virtual controls to zero. As a consequence, the solution may neither satisfy the linear nor the nonlinear constraints. Even if the virtual controls are zero, one might have found a solution that is not feasible with respect to the original, nonlinear problem. This is often the case in complex and highly nonlinear problems when the initial guess is poor or an insufficient number of nodes is used to discretize the problem. In this section, we propose a mesh refinement method to increase the accuracy with respect to the nonlinear dynamics that also improves convergence. The approach consists of two steps: first, we attempt to determine a solution that satisfies the nonlinear constraints using two different methods M1 and M2 (step 1). Secondly, we refine this solution by using the propagation error to measure the accuracy (step 2). The reason is that even if the nonlinear dynamics are satisfied at the nodes, the accuracy might be poor when propagating the nonlinear dynamics with the obtained controls.

Mesh Refinement Method 1

As we are mainly interested in generating feasible trajectories with respect to the nonlinear constraints, we compare the linear and nonlinear constraint violations in method 1:

$$\mathbf{h}_{i,\text{nonlin,dynamics}}^{(k)} = \mathbf{X}_i^{(k)} - \mathbf{X}_0^{(k)} - \frac{t_{N_k}^{(k)} - t_0^{(k)}}{2} \sum_{j=1}^{N_k} I_{ij}^{(k)} \mathbf{f}_{\text{nonlin}}(\mathbf{X}_j^{(k)}, \mathbf{U}_j^{(k)}) \qquad i = 1, ..., N_k \tag{18}$$

$$\mathbf{h}_{i,\text{nonlin,control}}^{(k)} = \left\| \boldsymbol{\tau}_i^{(k)} \right\| - \tau_i^{(k)} \qquad i = 1, ..., N_k \tag{19}$$

where $\mathbf{f}_{\text{nonlin}}$ refers to the nonlinear dynamics in Eq. 4. The maximum constraint violation for each segment is then calculated as

$$h_{\text{max, nonlin}}^{(k)} = \max_{1 \le i \le N_k} \left\| \begin{bmatrix} \mathbf{h}_{i,\text{nonlin,dynamics}}^{(k)} \\ \mathbf{h}_{i,\text{nonlin,control}}^{(k)} \end{bmatrix} \right\|_\infty \tag{20}$$

In the linear case, we can simply use the slack variables $\boldsymbol{\nu}$ and η to define the constraint violations:

$$\mathbf{h}_{i,\text{lin,dynamics}}^{(k)} = \boldsymbol{\nu}_i^{(k)} \qquad i = 1, ..., N_k \tag{21}$$

$$\mathbf{h}_{i,\text{lin,control1}}^{(k)} = \max\left(\eta_i^{(k)}, 0\right) \qquad i = 1, ..., N_k \tag{22}$$

$$\mathbf{h}_{i,\text{lin,control2}}^{(k)} = \max\left(\left\| \boldsymbol{\tau}_i^{(k)} \right\| - \tau_i^{(k)}, 0\right) \qquad i = 1, ..., N_k \tag{23}$$

The maximum violation for each segment is then determined as

$$h_{\text{max, lin}}^{(k)} = \max_{1 \le i \le N_k} \left\| \begin{bmatrix} \mathbf{h}_{i,\text{lin,dynamics}}^{(k)} \\ \mathbf{h}_{i,\text{lin,control1}}^{(k)} \\ \mathbf{h}_{i,\text{lin,control2}}^{(k)} \end{bmatrix} \right\|_\infty \tag{24}$$

Defining the maximum desired constraint violation as ϵ_c and setting

$$h_{\text{max}}^{(k)} = \max\left(h_{\text{max,nonlin}}^{(k)}, h_{\text{max,lin}}^{(k)}\right), \tag{25}$$

the number of collocation points to be added $p_{\text{add,M1}}$ is determined using [26]

$$p_{\text{add,M1}}^{(k)} = \left\lceil \log_{N_k}\left(\frac{h_{\text{max}}^{(k)}}{\epsilon_c}\right) \right\rceil \tag{26}$$

Mesh Refinement Method 2

This mesh refinement procedure is based on [26] and adapted to the flipped Radau discretization and our convex optimization formulation. The idea is to compare the optimized states with a solution that contains a larger number of nodes. When the states are smooth, an approximation with more collocation points is expected to be more accurate with respect to the nonlinear dynamics. Thus, the difference between the original solution and the one with higher accuracy is used to define the error of the mesh. For each segment, we define $\tilde{N}_k = N_k + 1$ as the new collocation points with higher accuracy. We introduce the old mesh as $S^{(k)} = \left(\sigma_1^{(k)}, \sigma_2^{(k)}, \ldots, \sigma_{N_k}^{(k)}\right)$ and new mesh as $\tilde{S}^{(k)} = \left(\tilde{\sigma}_1^{(k)}, \tilde{\sigma}_2^{(k)}, \ldots, \tilde{\sigma}_{\tilde{N}_k}^{(k)}\right)$ where $\sigma_i^{(k)}$ and $\tilde{\sigma}_i^{(k)}$ denote the corresponding pseudospectral times. Interpolating the current solution $\left[\mathbf{X}(\sigma_i^{(k)}), \mathbf{U}(\sigma_i^{(k)})\right]$ at the new points $\tilde{\sigma}_i^{(k)}$ yields $\left[\mathbf{X}(\tilde{\sigma}_i^{(k)}), \mathbf{U}(\tilde{\sigma}_i^{(k)})\right]$. Defining $\tilde{\mathbf{X}}(\sigma_i^{(k)})$ as the new polynomial of degree $\tilde{N}_k$ whose derivative should match the nonlinear dynamics at the collocation points in $\tilde{S}^{(k)}$, we obtain the following expression using Eq. 13:

$$\tilde{\mathbf{X}}(\tilde{\sigma}_i^{(k)}) = \mathbf{X}(\tilde{\sigma}_0^{(k)}) + \frac{t_{N_k}^{(k)} - t_0^{(k)}}{2} \sum_{j=1}^{\tilde{N}_k} \tilde{I}_{ij}^{(k)} \mathbf{f}_{\text{nonlin}}(\mathbf{X}(\tilde{\sigma}_j^{(k)}), \mathbf{U}(\tilde{\sigma}_j^{(k)})) \qquad i = 1, ..., \tilde{N}_k \tag{27}$$

We emphasize that the nonlinear dynamics are used here. The integration matrix $\tilde{\mathbf{I}}^{(k)}$ corresponds to the mesh $\tilde{S}^{(k)}$ and $\mathbf{X}(\tilde{\sigma}_0^{(k)})$ is the initial (non-collocated) point. The errors are then calculated as

$$\begin{aligned} E_{\text{pos}}(\tilde{\sigma}_i^{(k)}) &= \left\| \tilde{\mathbf{X}}_{\text{pos}}(\tilde{\sigma}_i^{(k)}) - \mathbf{X}_{\text{pos}}(\tilde{\sigma}_i^{(k)}) \right\| \qquad i = 1, ..., \tilde{N}_k \\ E_{\text{vel}}(\tilde{\sigma}_i^{(k)}) &= \left\| \tilde{\mathbf{X}}_{\text{vel}}(\tilde{\sigma}_i^{(k)}) - \mathbf{X}_{\text{vel}}(\tilde{\sigma}_i^{(k)}) \right\| \end{aligned} \tag{28}$$

where the subscripts *pos* and *vel* refer to the position and velocity components, respectively. The maximum error $\mathbf{E}_{\text{max}}^{(k)}$ is then

$$\mathbf{E}_{\text{max}}^{(k)} = \begin{bmatrix} \max_{1 \le i \le \tilde{N}_k} E_{\text{pos}}(\tilde{\sigma}_i^{(k)}) \\ \max_{1 \le i \le \tilde{N}_k} E_{\text{vel}}(\tilde{\sigma}_i^{(k)}) \end{bmatrix} \tag{29}$$

The number of collocation points to be added $p_{\text{add,M2}}$ is again determined using [26]

$$p_{\text{add,M2}}^{(k)} = \max\left(\left\lceil \log_{N_k}\left(\frac{E_{\text{max,pos}}^{(k)}}{\epsilon_{\text{c,pos}}} \right) \right\rceil, \left\lceil \log_{N_k}\left(\frac{E_{\text{max,vel}}^{(k)}}{\epsilon_{\text{c,vel}}} \right) \right\rceil \right) \tag{30}$$

The mesh is updated according to Algorithm 1. The number of nodes in each segment is increased if the error defined in Eq. 25 or Eq. 29 is violated. We define a minimum ($\text{colpoints}_{\text{min}}$) and maximum ($\text{colpoints}_{\text{max}}$) number of collocation points per segment and split the segment if needed.

When all nonlinear constraints are satisfied, we determine the error when propagating the nonlinear equations of motion with the obtained controls (step 2). The propagation error for each segment is defined as the difference between the optimized and propagated state at the last collocation point of the segment:

$$\Delta\mathbf{X}^{(k)} = \mathbf{X}_{N_k}^{(k)} - \left[\mathbf{X}_0^{(k)} + \int_{t_0^{(k)}}^{t_{N_k}^{(k)}} \mathbf{f}_{\text{nonlin}}(t, \mathbf{x}, \mathbf{u})\, \mathrm{d}t \right] \tag{31}$$

Algorithm 1 Mesh update method

```
1:  for each segment $k \in 1, ..., K$ do
2:      if $\left[h_{\text{max}}^{(k)} > \epsilon_c\right]$ for method 1 or $\left[E_{\text{max,pos}}^{(k)} > \epsilon_{\text{c, pos}}$ or $E_{\text{max,vel}}^{(k)} > \epsilon_{\text{c, vel}}\right]$ for method 2 then
3:          determine $p_{\text{add}}^{(k)}$
4:          increase $N_k$ by $p_{\text{add}}^{(k)}$
5:          if $N_k >$ colpoints$_{\text{max}}$ then
6:              split segment into two or more segments such that
7:              colpoints$_{\text{min}}$ and colpoints$_{\text{max}}$ are respected
8:          end if
9:          save new collocation points and segments
10:     else
11:         save $N_k$
12:     end if
13: end
14: return collocation points and segments
```

where the integral is computed with an explicit fourth-order Runge-Kutta method using the initial condition $\mathbf{X}_0^{(k)}$. The control values are interpolated linearly. Note that we do not continue the integration with the values obtained in the previous segment, but start with the optimized initial state in the next segment (see Fig. 7). This allows us to precisely determine the segments where the integration error is large without accumulating it. The complete mesh refinement procedure is shown in Fig. 2.

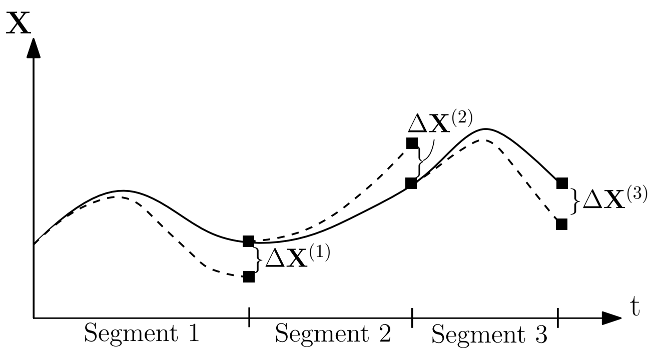
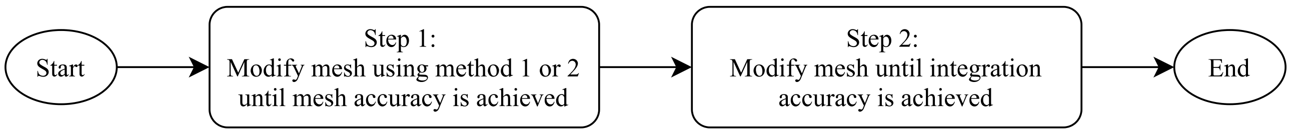


Figure 1: 2D illustration of the integration error $\Delta\mathbf{X}^{(k)}$ for three segments. The solid line represents the optimized and the dashed line the propagated state $\mathbf{X}$.



Start
Step 1:
Modify mesh using method 1 or 2
until mesh accuracy is achieved
Step 2:
Modify mesh until integration
accuracy is achieved
End

Figure 2: Flowchart of the mesh refinement procedure.

4 CLOSED-LOOP GUIDANCE PROCEDURE

In the closed-loop guidance scenario, the spacecraft repeatedly recomputes its reference trajectory. We predefine the vector $\mathbf{t}_{\text{opt}} = [t_0, t_1, \ldots, t_L]^\top$ with $t_L < t_f$ that contains the times when a (new)

trajectory is to be determined. After the l-th reoptimization, the equations of motion in Eq. 4 are propagated for $\Delta t_l = t_{l+1} - t_l$ ($l = 0, 1, \dots, L$ with $t_{L+1} = t_f$) until the next trajectory is to be computed. This process continues for all $t_l \in \mathbf{t}_{\text{opt}}$ until the final time is reached. The complete process is illustrated in Fig. 3. To account for uncertainties and other not modeled errors, the propagated state is perturbed by random values between -10^4 and 10^4 km (position) and -10^{-3} and 10^{-3} km/s (velocity) and then used as the new initial state. Those values are one order of magnitude larger than the typical accuracy of current optical navigation techniques [9].
In this paper, we calculate the first reference trajectory at t_0 with a shape-based method that uses a simple cubic interpolation. In all other cases, the solution of the previous optimization is used as the initial guess. The number of nodes is adjusted and the states and controls are interpolated at this new mesh. The goal is to maintain the accuracy of the initial mesh, that is, the number of nodes per time unit shall be similar. As the time of flight reduces after each reoptimization, we therefore remove a certain number of collocation points in those segments where the propagation error is lower than some threshold. As we propagate only for a very small time portion Δt_l, only this first part of the trajectory is of interest; the remaining part is discarded. The key idea is to ensure a high accuracy during this period. For this reason, we add another segment at the beginning with a sufficient number of collocation points that represents Δt_l. If the calculated trajectory is close to the propagated trajectory, that is, $\Delta \mathbf{X}^{(1)}(\Delta t_l)$ is small, it is expected that the algorithm converges within few iterations only. Our rationale is that the new solution is expected to lie in the neighborhood of the previous solution which is used as the new initial guess. Although a proof of the convergence is beyond the scope of this paper, the numerical simulations strongly support this statement.

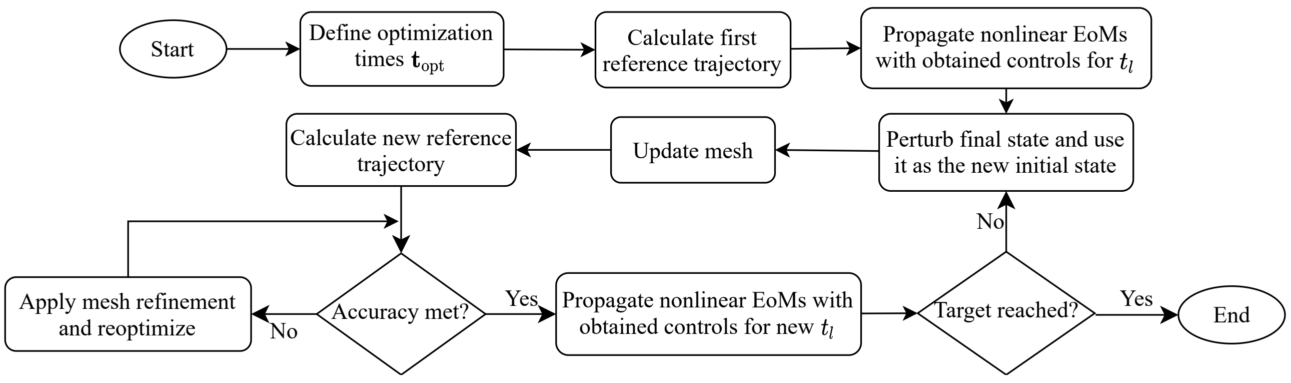

Start
Define optimization times $\mathbf{t}_{\text{opt}}$
Calculate first reference trajectory
Propagate nonlinear EoMs with obtained controls for t_l
Perturb final state and use it as the new initial state
Update mesh
Calculate new reference trajectory
Accuracy met?
Apply mesh refinement and reoptimize
No
Yes
Propagate nonlinear EoMs with obtained controls for new t_l
Target reached?
No
Yes
End

Figure 3: Flowchart of the closed-loop guidance simulation.

4.1 SCP Algorithm and Auxiliary Variables

The SCP algorithm uses a trust region mechanism with constant parameters to adjust the trust region radius. Details can be found in [22] and [23]. We briefly describe the modifications that are required to write the constraints in standard form. In particular, it is necessary to rewrite the constraints that contain the 1-norm or $\max$ function:

Virtual Control
Instead of minimizing $\sum_{m \in I_{\text{eq}}} \mu_m \cdot \|\boldsymbol{\nu}_m\|_1$ directly, the auxiliary variable $\mathbf{w}_{\text{vc}}$ is added to the optimization problem and the expression $\mu_m \cdot \mathbf{1}^\top \cdot \mathbf{w}_{\text{vc}}$ is minimized subject to $\boldsymbol{\nu}_m \leq \mathbf{w}_{m,\text{vc}}$ and $-\boldsymbol{\nu}_m \leq \mathbf{w}_{m,\text{vc}}$.

Trust Region
The trust region constraint $\|\mathbf{x} - \bar{\mathbf{x}}\|_1 \leq r_{\text{tr}}$ is replaced by three auxiliary constraints:

$$\mathbf{1}^\top \cdot \mathbf{w}_{\text{tr}} \leq r_{\text{tr}} \tag{32a}$$
$$-\mathbf{w}_{\text{tr}} - \mathbf{x} \leq -\bar{\mathbf{x}} \tag{32b}$$
$$\mathbf{x} - \mathbf{w}_{\text{tr}} \leq \bar{\mathbf{x}} \tag{32c}$$

$\mathbf{w}_{\text{tr}}$ is then added to the optimization variables.

***Max* Function**
We minimize $\mathbf{w}_{\text{ineq}}$ instead of $\sum_{m \in I_{\text{ineq}}} \lambda_m \cdot \max(0, \eta_m)$ and add the constraints $-\mathbf{w}_{\text{ineq},j} \leq 0$, $\eta_m - \mathbf{w}_{\text{ineq},m} \leq 0$. The optimization vector is augmented with $\mathbf{w}_{\text{ineq}}$.

The final optimization vector reads $[\mathbf{x}, \mathbf{u}, \boldsymbol{\nu}, \mathbf{w}_{\text{vc}}, \mathbf{w}_{\text{tr}}, \mathbf{w}_{\text{ineq}}]^\top$.

5 NUMERICAL SIMULATIONS

In this section, we validate the mesh refinement method and simulate the closed-loop guidance. We consider a transfer from the Sun-Earth Lagrange point L_2 (SEL$_2$) to asteroid 2000 SG344; this is one of the targets suggested for ESA's Miniaturised Asteroid Remote Geophysical Observer (M-ARGO) mission [27]. Throughout this section, we assume two-body dynamics, a constant specific impulse and no additional perturbations. Moreover, no mission constraints are considered. All calculations are performed in MATLAB version 2018b on an Intel Core i5-6300 2.30 GHz Laptop with four cores and 8 GB of RAM. The second-order cone program defined by Eqs. 7 and 8 is solved using the open-source Embedded Conic Solver (ECOS) [14]. Physical constants of all simulations are given in Table 1 and parameters of the SCP algorithm in Table 2. The algorithm converges when the maximum constraint violation and the decrease of the objective function are smaller than ϵ_c and ϵ_ϕ, respectively. Note that all values are scaled with LU (where 1 LU = 1 astronomical unit), VU, TU, ACU and MU, respectively (see Table 1). The interested reader is referred to [22, 23] for details on the SCP algorithm and trust region parameters ρ_i, α and β. Tables 3 and 4 show the parameters for the transfer and closed-loop guidance, 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	$\sqrt{\mu/\text{LU}}$ km/s
Time unit TU	LU/VU s
Acceleration unit ACU	VU/TU km/s^2
Mass unit MU	$m(t_0)$

Table 2: Parameters of the SCP algorithm [22, 23].

Parameter	Value
Penalty weight λ	1.0
Penalty weight μ	1.0
Trust region r_0	100.0
$[\rho_0, \rho_1, \rho_2]$	$[0.01, 0.25, 0.9]$
α, β	$1.5, 1.5$
ϵ_c	10^{-7}
ϵ_ϕ	10^{-5}
Max. iterations	100

5.1 Mesh Refinement

We test the mesh refinement methods 1 and 2 from Section 3. The initial mesh is [5, 5], that is, two segments with five collocation points each. The results are shown in in Table 5 and Fig. 4. Step 1 (reduce the maximum nonlinear constraint violation) and step 2 (reduce the maximum integration

Table 3: Parameters for SEL_2 to asteroid 2000 SG344 transfer.

Parameter	Value
Initial epoch	04-Feb-2024 12:00:00 UTC
Time of flight t_f	700 days
Initial mass $m(t_0)$	22.6 kg
Final mass $m(t_f)$	free
Maximum thrust T_{max}	2.2519 mN
Specific impulse I_{sp}	3067 s

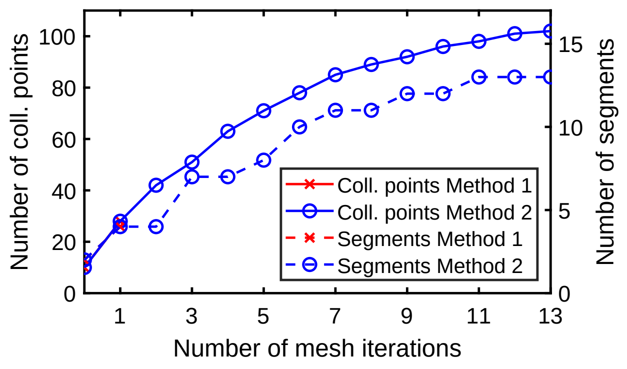
Table 4: Parameters for the closed-loop guidance simulation.

Parameter	Value
Reoptimization period Δt_l	14 days
colpoints$_{\text{min}}$	5
colpoints$_{\text{max}}$	11
Position perturbance	-10^4 to 10^4 km
Velocity perturbance	-10^{-3} to 10^{-3} km/s
Propagation tol.	10^{-5} LU, 10^{-3} VU
Initial collocation points	155
Initial mesh	[5, 10, 10 ..., 10]

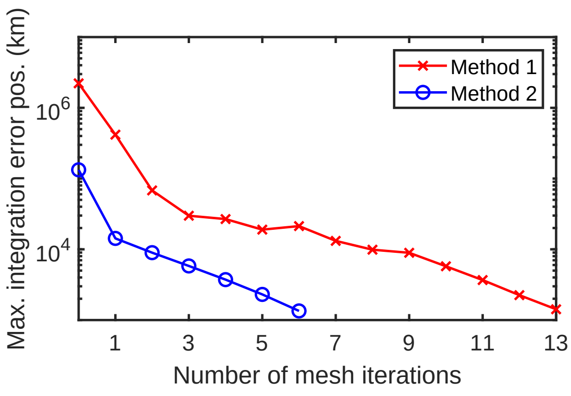
error) refer to the two-step approach described in Section 3. Method 1 and 2 refer to the two mesh refinement methods.
It is evident that there are considerable differences between the two methods. Method 1 aims at reducing the maximum constraint violation as fast as possible and achieves this after one mesh iteration only. The number of collocation points increases only slightly compared to method 2 (see Table 5). Consequently, the maximum integration error per segment is still large. In contrast, method 2 requires many more iterations and converges with a significantly larger number of collocation points as shown in Fig. 4a. This results in an integration error that is one order of magnitude lower. When the integration error is to be reduced in step 2, we observe the opposite behavior: method 1 requires more iterations because of the larger initial error (see Fig. 4b). Nonetheless, the maximum integration error decreased below the desired threshold 10^3 km for both methods.

Table 5: Results of mesh refinement procedure for step 1.

	Mesh iterations	Final coll. points	Max. integration error position (km)
Method 1	1	28	2.2×10^6
Method 2	13	102	1.3×10^5



(a) Mesh step 1: Collocation points and number of segments for each mesh iteration.



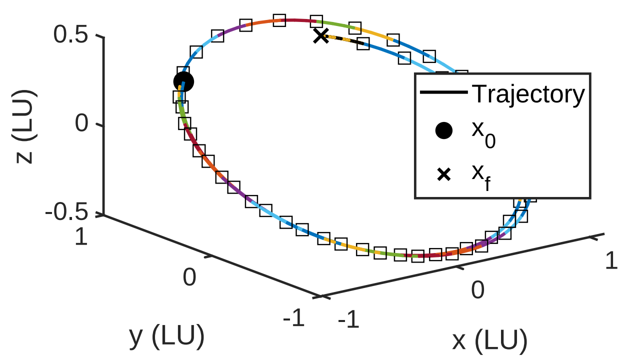
(b) Mesh step 2: Maximum integration error for position (see Eq. 31) for each mesh iteration.

Figure 4: Overview of the mesh refinement results for both mesh refinement methods.

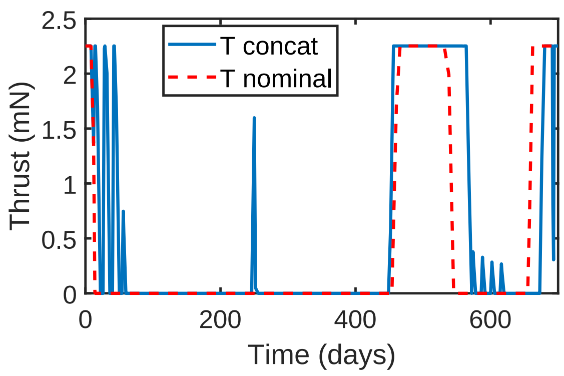
5.2 Closed-Loop Guidance

We use a uniform mesh of 15 segments with ten collocation points each and add an additional segment with five points at the beginning (see Table 4). A new reference trajectory is calculated every 14 days following the approach in Section 4, resulting in a total of 50 reoptimizations considering a time of flight of 700 days (see Table 3). The (poor) first initial guess is determined with a simple cubic interpolation. We use mesh refinement method 1 for the simulation.

Figure 5 shows the concatenated trajectory and thrust magnitude over time. Note that the concatenated thrust curve is slightly different compared to the nominal one (which corresponds to the first trajectory). Even though the first and concatenated trajectory are very similar, the different control histories indicate that the algorithm determined locally optimal solutions that are in the neighborhood of the nominal one. The small spikes around 600 days of travel time in Fig. 5b suggest that the density of collocation points was not sufficient to capture the bang-off-bang control structure perfectly. The obtained final masses of 21.72 kg and 21.51 kg in the first and last optimization, respectively, are also very similar. This confirms that the concatenated trajectory is very close to the nominal, locally optimal one. As we use the previous solution as the new initial guess, the number of iterations and computational times are only relatively high for the first optimization; as illustrated in Fig. 6a, all subsequent ones required only very few iterations and seconds, respectively. The number of collocation points remains constant for the first reoptimizations (see Fig. 6b) because the propagation error for each segment exceeded the tolerance defined in Table 4. The reason is that at the beginning of the flight the influence of the Earth is not negligible. Figure 7 shows the integration errors for the first segment that we obtain when propagating the equations of motion with two-body and n-body dynamics, respectively. Apparently, the perturbing force of the Earth cannot be neglected during the first few reoptimizatons when the spacecraft is still close to Earth. This results in a large mean integration error of 1.1×10^4 km and 1.7×10^{-2} km/s when integrating with n-body dynamics (see Table 6). As the distance between Earth and spacecraft increases over time, the difference between the two-body and n-body propagation becomes smaller. After computing the last trajectory, the spacecraft is able to reach the target with an error of only few hundred kilometers regardless of the considered integration model. This is a very promising result as it demonstrates that a closed-loop guidance based on convex programming can achieve a high accuracy.

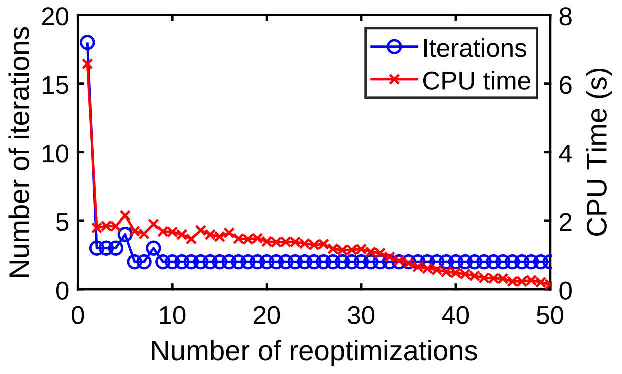


(a) Concatenated trajectory; squares indicate that trajectory is recomputed at these points.

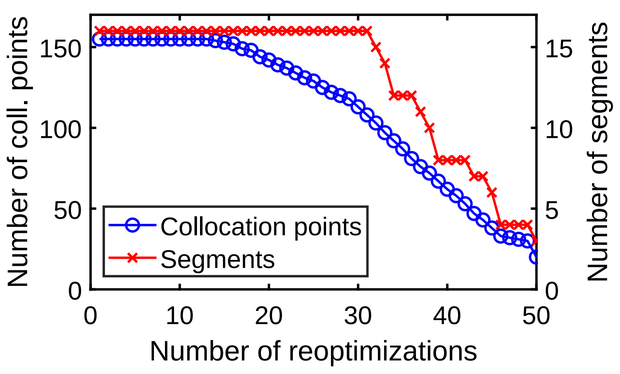


(b) Concatenated thrust magnitude vs. thrust magnitude for nominal trajectory.

Figure 5: Concatenated trajectory and thrust magnitude for the closed-loop guidance simulation for the SEL_2 to asteroid 2000 SG344 transfer.



(a) Number of iterations and CPU time per reoptimization.



(b) Number of collocation points and segments per reoptimization.

Figure 6: Overview of iterations, CPU time, collocation points and segments for the closed-loop guidance simulation for the SEL$_2$ to asteroid 2000 SG344 transfer.

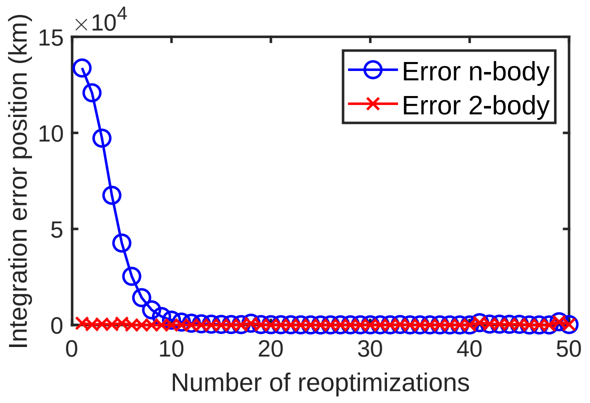


Figure 7: Integration error $\Delta \mathbf{X}^{(1)}$ for the closed-loop guidance simulation after propagating the equations of motions for a period Δt_l $(l = 0, 1, ..., L)$ considering the two- and n-body model, respectively.

Table 6: Integration errors for the closed-loop guidance simulation for the SEL$_2$ to asteroid 2000 SG344 transfer.

Dynamics for integration	Mean integration error for first segment (km, km/s)	Final position/velocity error (km, km/s)
Two-body dynamics	1.7×10^2, 3.1×10^{-4}	3.0×10^2, 4.5×10^{-4}
N-body dynamics	1.1×10^4, 1.7×10^{-2} (*)	2.3×10^2, 3.4×10^{-4}

(*) Large mean error due to the perturbing force of the Earth at the beginning.

6 CONCLUSION

We developed a closed-loop guidance method that repeatedly recomputes the references trajectory. A sequential convex programming algorithm optimizes the trajectory in certain intervals and the spacecraft state is propagated with the obtained controls. A mesh refinement technique ensures the desired accuracy while keeping the computational effort low. A transfer to an asteroid has demonstrated the excellent accuracy and robustness even when large perturbances are considered.
We presented a novel approach that integrates a numerical optimization technique into a closed-loop

guidance method. As the chosen convex programming algorithm has a large convergence radius, we were able to resolve the low-thrust trajectory optimization problem during the whole flight despite the imposed perturbance. Its rapid speed makes it suitable for real-time applications. Even though SCP may in general be less accurate compared with other direct methods, we have shown that the target can be reached with high accuracy when the mesh is adjusted accordingly. The presented guidance method is therefore an appealing choice for real space missions.

REFERENCES

[1] A. Hein, M. Saidani, and H. Tollu, "Exploring Potential Environmental Benefits of Asteroid Mining," in *69th International Astronautical Congress*, 2018.

[2] A. Poghosyan and A. Golkar, "CubeSat evolution: Analyzing CubeSat capabilities for conducting science missions," *Progress in Aerospace Sciences*, vol. 88, pp. 59–83, 2017.

[3] A. Slavinskis and et al., "Nanospacecraft fleet for multi-asteroid touring with electric solar wind sails," in *2018 IEEE Aerospace Conference*, 2018, pp. 1–20.

[4] M. B. Quadrelli and et al., "Guidance, Navigation, and Control Technology Assessment for Future Planetary Science Missions," *Journal of Guidance, Control, and Dynamics*, vol. 38, no. 7, pp. 1165–1186, 2015.

[5] R. Walker and et al., "Deep-space CubeSats: thinking inside the box," *Progress in Aerospace Sciences*, vol. 59, no. 5, pp. 24–30, 2018.

[6] Y. Mao, M. Szmuk, and B. Açıkmeşe, "A Tutorial on Real-time Convex Optimization Based Guidance and Control for Aerospace Applications," in *2018 Annual American Control Conference (ACC)*, 2018, pp. 2410–2416.

[7] D. Dueri, B. Açıkmeşe, D. P. Scharf, and M. W. Harris, "Customized Real-Time Interior-Point Methods for Onboard Powered-Descent Guidance," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 2, pp. 197–212, 2017.

[8] 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, pp. 315–327, 2021.

[9] V. Franzese and F. Topputo, "Optimal Beacons Selection for Deep-Space Optical Navigation," *The Journal of the Astronautical Sciences*, vol. 67, no. 4, p. 1775–1792, 2020.

[10] R. C. Huang, I. Hwang, and M. J. Corless, "Nonlinear Algorithm for Tracking Interplanetary Low-Thrust Trajectories," *Journal of Guidance, Control, and Dynamics*, vol. 35, no. 2, pp. 696–700, 2012.

[11] J. T. Betts, "Survey of Numerical Methods for Trajectory Optimization," *Journal of Guidance, Control, and Dynamics*, vol. 21, no. 2, pp. 193 – 207, 1998.

[12] F. Topputo and C. Zhang, "Survey of Direct Transcription for Low-Thrust Space Trajectory Optimization with Applications," *Abstract and Applied Analysis*, vol. 2014, pp. 1–15, 2014.

[13] X. Liu, P. Lu, and B. Pan, "Survey of Convex Optimization for Aerospace Applications," *Astrodynamics*, vol. 1, no. 1, pp. 23–40, 2017.

[14] A. Domahidi, E. Chu, and S. Boyd, “ECOS: An SOCP Solver for Embedded Systems,” in *European Control Conference*, Zurich, Switzerland, 2013, pp. 3071–3076.

[15] M. Szmuk, U. Eren, and B. Açıkmeşe, “Successive Convexification for Mars 6-DoF Powered Descent Landing Guidance,” in *AIAA Guidance, Navigation, and Control Conference*, Grapevine, Texas, 2017.

[16] H. Yang, X. Bai, and H. Baoyin, “Rapid Generation of Time-Optimal Trajectories for Asteroid Landing via Convex Optimization,” *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 3, 2018.

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

[18] Z. Wang and M. J. Grant, “Minimum-Fuel Low-Thrust Transfers for Spacecraft: A Convex Approach,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 54, no. 5, pp. 2274–2290, 2018.

[19] G. Tang, F. Jiang, and J. Li, “Fuel-Optimal Low-Thrust Trajectory Optimization using Indirect Method and Successive Convex Programming,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 54, no. 4, pp. 2053–2066, 2018.

[20] Y. Song and S. Gong, “Solar-sail deep space trajectory optimization using successive convex programming,” *Astrophysics and Space Science*, vol. 364, 2019.

[21] Y. Kayama, K. C. Howell, M. Bando, and S. Hokamoto, “Low-Thrust Trajectory Design with Convex Optimization for Libration Point Orbits,” in *AAS/AIAA Space Flight Mechanics Meeting*, February 2021, AAS Paper 21-231.

[22] C. Hofmann and F. Topputo, “Rapid Low-Thrust Trajectory Optimization in Deep Space Based On Convex Programming,” *Journal of Guidance, Control, and Dynamics*, 2020.

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

[24] B. Açıkmeşe and S. R. Ploen, “Convex Programming Approach to Powered Descent Guidance for Mars Landing,” *Journal of Guidance, Control, and Dynamics*, vol. 30, no. 5, pp. 1353–1366, 2007.

[25] D. Garg, “Advances in Global Pseudospectral Methods for Optimal Control,” Ph.D. dissertation, University of Florida, 2011.

[26] M. A. Patterson, W. Hager, and A. V. Rao, “A ph Mesh Refinement Method for Optimal Control,” *Optimal Control Applications and Methods*, vol. 36, no. 4, 2014.

[27] F. Topputo and et al., “Envelop of reachable asteroids by M-ARGO CubeSat,” *Advances in Space Research*, vol. 67, no. 12, pp. 4193–4221, 2021.