ESA GNC & ICATT Conference 2026
28 Sep. – 2 Oct. 2026, Sevilla, Spain
https://atpi.eventsair.com/gncicatt/

A GPU-accelerated Implementation for Goal-Oriented Trajectories About Minor Bodies

Carmine Giordano[1,*], Massimo Casasco[2], and Francesco Topputo[1]

[1]*Department of Aerospace Science and Technology, Politecnico di Milano, via La Masa, 34, 20155, Milan, Italy*
[2]*European Space Research and Technology Centre, European Space Agency, Keplerlaan 1, 2201 AZ Noordwijk, The Nederlands*
[*]*Corresponding author: carmine.giordano@polimi.it*

ABSTRACT

The design of goal-oriented trajectories for spacecraft in close-proximity operations around small Solar System bodies requires the solution of complex problems, based on reachability sets or mixed-integer programming problems. Their onboard implementation could be cumbersome due to tight onboard computational budgets. To overcome this limitation, in this work, a GPU-accelerated implementation of the Alternating Direction Method of Multipliers (ADMM) tailored to these trajectory optimization problems is presented. This solver has been benchmarked both a laptop workstation GPU and an embedded Jetson platform, comparing against a CPU-only implementation on equivalent hardware. Our results show that the GPU-accelerated algorithm on the Jetson has similar runtimes with a generic SCP on a laptop computer, enabling so branch-and-bound global-optimality loops on onboard-like hardware in similar wall-clock time. This performance gain opens the door to on-board re-planning and autonomous decision making in future SmallSat and CubeSat missions.

1 Introduction

In recent years, interest in the Solar System minor bodies has surged, driven by their capacity to shed light on life origins [31], illuminate the processes that shaped planetary formation [3], and potentially supply resources critical for sustainable human endeavors both on Earth and in space [22]. Conversely, small bodies present a collision hazard, with impacts with the Earth capable of inflicting global-scale devastation [11]. Consequently, planetary scientists are prioritizing to perform comprehensive investigations of these bodies surface compositions, internal make-up, and physical characteristics [37]. While telescopic surveys can only provide rudimentary estimates of size, shape, and spin, obtaining high-resolution, in-situ measurements via robotic missions is essential to accurately determine their material properties and dynamical behaviors [9]. To this end, a succession of spacecraft have been dispatched to explore both asteroids, such as NEAR Shoemaker [35], OSIRIS-REx [24], and Hayabusa2 [45], and comets, including Stardust [6] and Rosetta [20]. Upcoming missions like Psyche [26] and RAMSES [23] promise further insights. In parallel, the joint NASA–ESA Asteroid Impact and Deflection Assessment (AIDA) initiative was conceived to evaluate kinetic impact strategies for neutralizing potentially hazardous Near-Earth Objects [14]. Under this program, NASA's DART probe successfully collided with Dimorphos in 2022 to test deflection techniques [10], and ESA's Hera will revisit the Didymos binary in 2027 to assess the long-term effects of that collision on the system's dynamics and physical state [28]. Beyond traditional missions, the rise of CubeSats and other SmallSats has begun to democratize deep-space exploration, delivering significant scientific output at a fraction of traditional mission costs [32, 43]. Both DART and Hera exemplify this trend, each deploying tiny companion satellites, LICIACube [15] from DART, and Juventas and Milani from Hera [21, 16], to augment observational capabilities. Stand-alone small spacecraft bound for asteroids are also under development, though their stringent mass constraints typically necessitate high-efficiency electric propulsion to minimize onboard propellant [29, 7]. Historically, trajectory design for close-proximity operations around minor bodies has fallen into

© *C. Giordano*

two main paradigms: identifying long-lived, low-maintenance orbital paths [21, 26] or stitching together bespoke waypoint sequences based on scientifically valuable viewpoints [17, 1, 2]. The former offers operational simplicity but often struggles with the chaotic gravitational environments and constraints on lighting and range needed for passive remote sensing. The latter provides scientific flexibility at the cost of laborious, time-intensive human planning. To overcome both limitations, a new class of goal-driven algorithms has emerged: by encoding high-level mission objectives, these frameworks autonomously generate trajectories through continuous replanning, using tools like reachability maps, spatial diagrams of maneuver feasibility grounded in propulsion limits, to navigate complex environments [39]. Recent extensions of this approach include global surface mapping of asteroids [8] and mission planning tailored to binary systems under tight engineering and science requirements [36].
Despite their potential, current goal-oriented approaches depend on reachability maps. While robust and suitable to be applied to onboard planning, they can be limiting for platforms equipped with continuous thrust systems. To overcome these limitations, a new methodology, based on mixed-integer programming and sequential convex programming (SCP), has been proposed recently [18]. However, both the approaches need relevant computation resources, and their onboard implementation for spacecraft autonomy could be really cumbersome. In this work, this challenge is addressed by developing a GPU-accelerated ADMM solver for goal-oriented trajectory design. Specifically, a GPU-compatible ADMM is implemented in order to explore faster and more efficiently the nodes generated by the branch-and-bound algorithm used to solve the MIP problem obtained for platforms equipped with continuous thrust systems.
Results show that the algorithm exploiting the GPU-compatible ADMM is able to solve the full goal-oriented problem on a power-constrained onboard-like hardware, represented by a nVidia Jetson, employing the same runtime of the CPU-based algorithm running on a consumer laptop. This outcome opens the door to on-board re-planning and autonomous decision making in future SmallSat and CubeSat missions.

2 Background

2.1 Goal-oriented trajectory design

The goal-oriented trajectory design challenge involves determining a path around a minor body that fulfills specified scientific requirements within a fixed mission duration. In this framework, each scientific target is expressed as a high-level condition that must be met along the spacecraft route to increase the scientific return. Such targets vary with mission objectives, e.g., a close flyby at particular altitudes and latitudes for gravity or radio-science experiments, or high-phase-angle limb passes to probe atmospheric layers. In this study, passive remote-sensing tasks aimed at imaging surface features are considered. The scientific outcome is evaluated by counting how many times the spacecraft field of view, represented as a nadir-pointing cone whose half-angle matches the instrument aperture, captures a feature of interest. This model fixes the spacecraft attitude toward nadir and excludes movable optics like mirrors or gimbals. These simplifications, commonly adopted in goal-based trajectory planning [39, 36], are especially pertinent for small satellites, which favor passive sensors to reduce power demands and avoid the mass and mechanical complexity inherent to articulated payloads.
As consequence, each scientific target can mathematically be represented as conical frusta (Fig. 1), which allows to observe the specific feature 1) within a given range, to respect a desired ground sampling distance and avoid safety concerns; 2) within a certain angle from the normal, to have a clear view of the feature.

2.2 Problem formulation

The generic low-thrust fuel-optimal problem can be specialized by considering the scientific goals. Given a spacecraft equipped with low thrust propulsion system flying close to a minor body, the goal-oriented minimum-fuel problem has the aim to find a trajectory satisfying all the scientific targets in a given timeframe while minimizing the use of propellant. The goal-oriented minimum-fuel problem can be formalized as [18]

Problem 1 (General deterministic goal-oriented minimum-fuel problem)

$$\underset{\mathbf{u}(t)}{\text{Minimize}}\, J := -m(t_f) - \pi(\mathbf{r}(t))$$

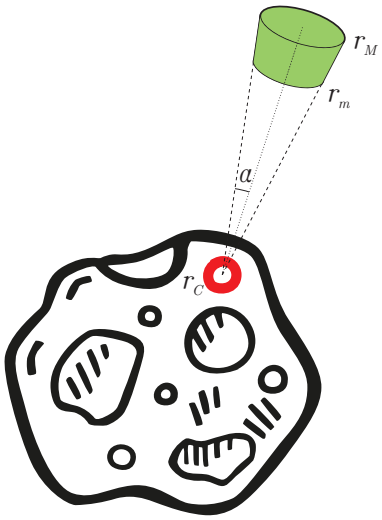


Figure 1 *Representation of scientific goal. The green volume is the target region.*

subject to

$$\begin{cases} \dot{\mathbf{r}} = \mathbf{v} \\ \dot{\mathbf{v}} = \mathbf{f}_f(\mathbf{x}, t) + \frac{\mathbf{T}(t)}{m(t)} \\ \dot{m} = -\frac{\|\mathbf{T}(t)\|}{I_{sp} g_0} \end{cases}, \qquad \mathbf{x}(t_0) = \mathbf{x}_0 \tag{1}$$

and

$$0 \leq \|\mathbf{T}(t)\| \leq T_{\max} \tag{2}$$

and N_g scientific constraints

$$\begin{cases} r_m \leq \|\mathbf{r}(t_k) - \mathbf{r}_{c_i}(t_k)\| \leq r_M & \text{(3a)} \\ \dfrac{\mathbf{r}(t_k) \cdot \mathbf{r}_{c_i}(t_k)}{\|\mathbf{r}(t_k)\| \|\mathbf{r}_{c_i(t_k)}\|} \leq \cos\alpha & \text{(3b)} \end{cases} \qquad \text{for some } t_k \in [t_0, t_f], \qquad \text{for } i = 1, \dots, N_g$$

where $\mathbf{x} = [\mathbf{r}, \mathbf{v}, m]^T$ is the state, embedding the position, the velocity, and the mass, respectively, $\mathbf{T}$ is the thrust vector and $T_{\max}$ its attainable maximum value, I_{sp} the thruster specific impulse, and g_0 gravitational acceleration at sea level. The term $\mathbf{f}_f(\mathbf{x}, t)$ represents the free-body dynamics. The i-th scientific target is described by the position of the feature r_{c_i}, the minimum and maximum distance from the feature, r_m and r_M, respectively, and the semi-aperture of the conical frustum α. The function π is a reward function, that must increase whenever a new scientific target constraint is respected.

The free dynamics term $\mathbf{f}_f$ in Eq. (1) is specialized for close proximity of a minor body. A non-inertial body fixed frame is considered. This choice allows to have the feature positions to be fixed, independent of the time, and ease the implementation of Eqs. (3). Thus, defying $\boldsymbol{\omega}$ as the angular velocity of the main body,

$$\mathbf{f}_f(\mathbf{x}, t) = \mathbf{f}_g(\mathbf{r}, t) + \underbrace{\sum_{k \in \mathcal{P}} -\mu_k \left(\frac{\mathbf{r}_k}{\|\mathbf{r}_k\|^3} + \frac{\mathbf{r} - \mathbf{r}_k}{\|\mathbf{r} - \mathbf{r}_k\|^3} \right)}_{\text{3-body acc.}} + \underbrace{C_r \frac{A}{m} \frac{P_0 d_0^2}{c} \frac{\mathbf{r} - \mathbf{r}_S}{\|\mathbf{r} - \mathbf{r}_S\|^3}}_{\text{SRP}} - \boldsymbol{\omega} \times (\boldsymbol{\omega} \times \mathbf{r}) - 2\boldsymbol{\omega} \times \mathbf{v} \tag{4}$$

where $\mathcal{P}$ is the set containing third bodies acting on the spacecraft and characterized by the gravitational parameter μ_k and relative position with respect to minor body $\mathbf{r}_k$. The solar ration pressure (SRP) is computed exploiting the canonball model, with P_0 ($1367\,\mathrm{W/m^2}$) the solar flux at 1 astronomical unit (AU), c is the speed of light, d_0 is the

Sun–Earth distance (1 AU), C_r is the reflectivity coefficient of the spacecraft, A is its equivalent surface area, and $\mathbf{r}_S$ the position of the Sun.
The term $\mathbf{f}_g(\mathbf{r}, t)$ represents the gravitational acceleration exerted by the main body. Considering the complex shapes of minor bodies, the use of the simple point-body model could produce inaccurate results. For this reason, in this work, the *tri-axial ellipsoid model* is exploited[38]

$$\mathbf{f}_g(\mathbf{r}, t) = -\frac{3}{2}\mu \begin{bmatrix} \tilde{x} \int_\lambda^\infty \frac{1}{(a^2+u)\Delta(u)}\,\mathrm{d}u \\ \tilde{y} \int_\lambda^\infty \frac{1}{(b^2+u)\Delta(u)}\,\mathrm{d}u \\ \tilde{z} \int_\lambda^\infty \frac{1}{(c^2+u)\Delta(u)}\,\mathrm{d}u \end{bmatrix} \tag{5}$$

where a, b, and c are the semi-axes of the ellipsoid, in decreasing order, $\tilde{x}$, $\tilde{y}$, and $\tilde{z}$ the Cartesian components of position relative vector with respect to the ellipsoid (i.e., $\tilde{\mathbf{r}} = \mathbf{r} - \mathbf{r}_{ell}$), and $\Delta(u) = \sqrt{(a^2+u)(b^2+u)(c^2+u)}$.

2.3 The goal-oriented convex problem

The general problem can be translated in a convex programming problem to be solved. In order to do so, first the state and control are decoupled by means of two new control variables and a different metric for the mass[44]

$$\Gamma(t) = \frac{\|\mathbf{T}(t)\|}{m(t)}, \quad \boldsymbol{\tau}(t) = \frac{\mathbf{T}(t)}{m(t)}, \quad z(t) = \ln(m(t))$$

Then, following the SCP paradigm, dynamics is linearized about a reference solution $(\bar{\mathbf{x}}, \bar{\mathbf{u}})$, with the control vector defined as $\mathbf{u} = [\boldsymbol{\tau}, \Gamma]^T$, and a trust-region constraint is added to keep the linearization valid [27]. Lastly, the problem is discetized by subdividing the timespan $[t_0, t_f]$ into N segments, and both the state and the control are projected into this time grid to have finite-dimensional variables, while the scientific constraints are handled by using the Big M method.
Hence, the generic problem is converted in the following mixed-integer problem

Problem 2 (Convex programming problem for the goal-oriented minimum-fuel problem)
Find $\mathbf{y} = [\mathbf{x}_k, \mathbf{u}_k, \boldsymbol{\nu}_k, \eta_k, \gamma_k],\ k = 1, \dots, N$, *such that*

$$J := -z_N + \sum_{i=0}^{N} \lambda \|\boldsymbol{\nu}_k\|_1 + \sum_{i=0}^{N} \lambda \max(0, \eta_k) - \lambda_g \sum_{i=1}^{N_g} \min\left(1, \sum_{k=0}^{N} \gamma_{i,k}\right) \tag{6}$$

is minimized, subject to

$$\boldsymbol{\Delta} = \hat{A}\mathbf{x} + \hat{B}\mathbf{u} + \mathbf{q} + \hat{N}\boldsymbol{\nu} = \mathbf{0} \tag{7}$$

the control constraints

$$0 \le \Gamma_k \le T_{\max} e^{-\bar{z}_k}(1 - z_k + \bar{z}_k) + \eta_{1,k} \tag{8}$$

$$\|\boldsymbol{\tau}_k\| \le \Gamma_k \tag{9}$$

and the goal constraints

$$\begin{cases} -\dfrac{(\bar{\mathbf{r}}_k - \mathbf{r}_{c_i}) \cdot \mathbf{r}_k}{\|\bar{\mathbf{r}}_k - \mathbf{r}_{c_i}\|} - \left(r_m - \|\bar{\mathbf{r}}(t_i) - \mathbf{r}_{c_i}\|\right) - \eta_{2,k} \le M(1 - \gamma_{i,k}) & \text{(10a)} \\ \|\mathbf{r}_k - \mathbf{r}_{c_i} - r_M\| \le M(1 - \gamma_{i,k}) \qquad \textit{for } i = 1, \dots, N_g & \text{(10b)} \\ \|\mathbf{r}_k - \mathbf{r}_{c_{i,k}}\| - \left(\dfrac{1}{\cos\alpha}(\mathbf{n}^T\mathbf{r}_k - \mathbf{n}^T\mathbf{r}_{c_i})\right) \le M(1 - \gamma_{i,k}) & \text{(10c)} \end{cases}$$

and the trust region constraint

$$\|\mathbf{x} - \bar{\mathbf{x}}\|_1 \le R \tag{11}$$

Eq. (7) represents the general formulation of the dynamical defects related to a generic discretization method, with $\mathbf{q}$ *being a constant value. The variables* $\boldsymbol{\nu}$ *and* η *are the virtual controls and are added to avoid artificial infeasibility,*

while R is the radius of the thrust region.
The variable $\gamma_{i,k}$ is the index binary variable for the single scientific target, i.e., it is 1 if the target i is respected at point k, and 0 otherwise, with M a sufficiently large value. The last term in Eq. (6) is added as a reward function to maximize the number of sceintific targets visited during the spacecraft path.

In this work, the dynamics is discretize by means of a first-order hold (FOH) method. The control history is approximated as a piecewise affine function, that is [27]

$$\mathbf{u}(t) = \frac{t_{k+1} - t}{h}\mathbf{u}_k + \frac{t - t_k}{h}\mathbf{u}_{k+1} = \lambda^-(t)\mathbf{u}_k + \lambda^+(t)\mathbf{u}_{k+1} \tag{12}$$

In this case, the linearized dynamics becomes

$$\dot{\mathbf{x}} = A\mathbf{x} + B\lambda^-\mathbf{u}_k + B\lambda^+\mathbf{u}_{k+1} + (\mathbf{f}_f(\bar{\mathbf{x}}, \bar{\mathbf{u}}) - A\bar{\mathbf{x}}) \tag{13}$$

In each segment, Eq. (13) could be integrated by exploiting the linear systems theory. Thus, the dynamical constraint for each segment become [27]

$$\mathbf{x}_{k+1} - \left(\hat{A}_k\mathbf{x}_k + \hat{B}_k^-\mathbf{u}_k + \hat{B}_k^+\mathbf{u}_{k+1} + \mathbf{q}_k + \boldsymbol{\nu}_k\right) = \mathbf{0} \tag{14}$$

where

$$\begin{aligned}
\hat{A}_k &= \Phi(t_k, t_{k+1}) \\
\hat{B}_k^- &= \hat{A}_k \int_{t_k}^{t_{k+1}} \Phi^{-1}(t_k, t) B\lambda^- \, \mathrm{d}t \\
\hat{B}_k^+ &= \hat{A}_k \int_{t_k}^{t_{k+1}} \Phi^{-1}(t_k, t) B\lambda^+ \, \mathrm{d}t \\
\mathbf{q}_k &= \hat{A}_k \int_{t_k}^{t_{k+1}} \Phi^{-1}(t_k, t) \left(\mathbf{f}_f(\bar{\mathbf{x}}, \bar{\mathbf{u}}) - A_k\bar{\mathbf{x}}\right) \mathrm{d}t
\end{aligned}$$

where $\Phi(t_k, t)$ is the state transition matrix between time t_k and time t.

3 The GPU-accelerated Programming Problem

Problem 2 could be solved by means of general a mixed-integer programming (MIP) solver. While previous works show that this approach is efficient and robust to find a solution [18, 19], MIPs are notorious NP-hard problems and thus, generally speaking, it is can not be solved in polynomial time. Additionally, increasing the discretization points N or the number of targets N_g increases the search space exponentially, hindering so its use in case of long timespans or large target databases. Furthermore, the NP-hardness makes this algorithm hardy suitable on board to increase spacecraft autonomy.
However, the algorithm used to solve Problem 2 is based on a branch-and-bound method which works by exploring thousands of nodes out of the solutions tree. Thus, the process can be sped up by exploring more nodes at the same time. Consequently, solutions to Problem 2 can be found faster if massive parallelization can be exploited. While CPUs typically have only a handful of powerful cores optimized for sequential tasks, GPUs excel at highly parallel workloads because they contain thousands of smaller, efficient cores designed to execute the same instruction across many data elements simultaneously. Additionally, GPUs offer much higher memory bandwidth and specialized hardware for common parallel operations (e.g., matrix math), allowing them to move and process large volumes of data far more quickly when the workload can be split into many concurrent threads. Using GPUs to solve multiple SCP problems at once has been already studied for fast Monte Carlo simulations[12], showing a great reduction of computational times. Hence, it is expected that Problem 2 can be solved faster by exploiting GPUs for parallel exploration of the nodes of the solution tree. It should be noted, though, that single GPU threads have limited computation capability and available memory and, thus, reducing operations and allocations must

be taken into account in the implementation.
The architecture of the GPU-accelerated solver for Problem 2 is given in Fig. 2. After the problem is initialized on the CPU, the following loop is solved iteratively, until the convergence is reached: 1) On the CPU, the branch-and-bound (B&B) algorithm selects some node to be explored (fixing the value of γ); 2) Each problem is initialized in the CPU and moved to the GPU memory; 3) On the GPU, each core solves a different problem in parallel; 4) Solutions are copied back to the general memory and analyzed by the CPU. If convergence is not reached, the B&B will select new nodes to be explored, and the loop starts again.

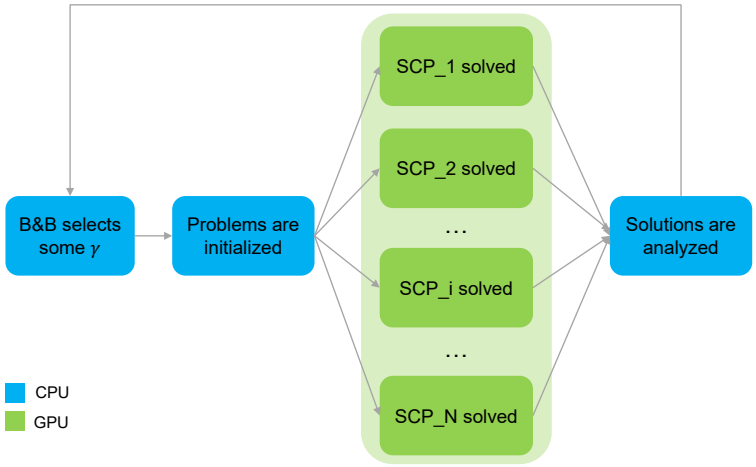


Figure 2 *Algorithm architecture of GPU-accelerated solver for the goal-oriented problem.*

Thus, in order to accelerate Problem 2 solution on the GPU, an SCP problem must be solved for each GPU thread. This means that a convex problem is solved iteratively on each GPU thread. Hence, it become necessary to implement a GPU-compatible convex solver. It could be noted, however, that this need is significantly different from the other GPU-accelerated convex solvers (as, e.g., CuClarabel [13]). In those cases, in fact, only the most computationally expensive, highly parallelizable, parts of the convex solver are moved to the GPU, while in this work, the full convex solver must be run on each thread.

3.1 The GPU-compliant convex solver

A quick overview of convex solvers

When faced with a convex optimization problem, the choice of solver typically comes down to three broad classes, each offering a different balance of speed, accuracy, memory usage, and robustness:

1) **Interior-Point Methods**. These methods solve the Karush–Kuhn–Tucker system directly at each iteration, often via successive barrier-augmented Newton steps [5]. Because they leverage Hessian information, interior-point solvers converge in very few iterations and attain high numerical precision, making them the workhorse for moderate-scale, high-accuracy applications. Their downside is that each iteration requires solving large linear systems, which can become a bottleneck both in runtime and memory footprint as problem size grows into the tens or hundreds of thousands of variables [33]. Additionally, they have poor warm-start capability, that can increase runtimes in case of successive convex problems to be solved.
2) **First-Order Methods**. By contrast, first-order approaches, including gradient descent, proximal-gradient methods, and coordinate descent, use only gradient or subgradient information. This makes each iteration extremely lightweight and well suited to very large-scale or streaming data problems, where memory-efficiency and per-iteration cost dominate [46]. A powerful subclass employs operator-splitting ideas, most prominently the Alternating Direction Method of Multipliers (ADMM). ADMM alternates between solving two easy subproblems (i.e., a linear solve plus a projection) and updating Lagrange multipliers, often converging to modest accuracy in few cheap iterations. This makes it ideal for embedded or very large-scale applications where medium-precision answers suffice and computational resources are limited[4]. Challenges remain in parameter selection (step-sizes and penalties) and infeasibility detection, though

techniques like homogeneous self-dual embeddings help mitigate these issues.

3) **Active-Set Algorithms**. Active-set methods maintain and update a working guess of which inequality constraints are active (i.e., tight) at the optimum. At each iteration they solve a smaller equality-constrained problem over the current active set, then add or remove constraints based on the gradient and dual multipliers [30]. Because they only solve reduced systems, warm-starting from a nearby solution can dramatically reduce work. However, in the worst case one may have to explore exponentially many candidate active sets, so these methods can become impractical as the number of constraints grows [25].

In summary, for small to medium problems where high accuracy and robust convergence are paramount, and where memory is ample, interior-point solvers remain the gold standard. For large-scale tasks, first-order methods offer unmatched scalability, albeit with lower solution precision. In practice, a hybrid approach is common: initialize a solution with a fast first-order to obtain a reasonably accurate estimate, then polish to high precision via an interior-point solver on the reduced, warm-started subproblem. Since in this work, several large-scale convex problem must be solved employing a limited memory footprint, the natural choice is using a first-order method. Specifically, given its properties and the characteristics of the problems to solve, ADMM is considered to be the most suitable methodology.

A glimpse on the Alternating Direction Method of Multipliers

As anticipated, the Alternating Direction Method of Multipliers is a methodology able to solve large-scale convex problems in a fast and mildly accurate manner. It is based on solving alternatively two easy subproblems, namely, a linear system and a conic projection. Specifically, consider a general quadratic conic programming problem

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\mathbf{x}^T P\mathbf{x} + \mathbf{q}^T\mathbf{x} && && (15a)\\ \text{subjected to} \quad & A\mathbf{x} = \mathbf{b} && (\mathcal{D}) && (15b)\\ & \mathbf{x} \in \mathcal{C} && && (15c) \end{aligned}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the decision variable, the cost is defined by a semidefinite matrix $P \in \mathbb{R}^{n\times n}$ and a vector $\mathbf{q} \in \mathbb{R}^n$, the set of linear constraints $\mathcal{D}$ is defined by a matrix $A \in \mathbb{R}^{m\times n}$ and the known terms $\mathbf{b} \in \mathbb{R}^m$, and the convex constraints to a nonempty, closed and convex set $\mathcal{C}$.
Following the ADMM paradigm, an additional decision variable $\mathbf{z} \in \mathbb{R}^n$ is introduced, and the constraints are split, so that $\mathbf{x}$ must respect the constraints in $\mathcal{D}$, while $\mathbf{z}$ the ones in $\mathcal{C}$. Thus, the general conic problem is rewritten as

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\mathbf{x}^T P\mathbf{x} + \mathbf{q}^T\mathbf{x} + I_{\mathcal{C}}(\mathbf{z}) && && (16a)\\ \text{subjected to} \quad & \mathbf{x} - \mathbf{z} = \mathbf{0} && (\mathcal{D}) && (16b)\\ & A\mathbf{x} = \mathbf{b} && (\mathcal{D}) && (16c)\\ & \mathbf{z} \in \mathcal{C} && && (16d) \end{aligned}$$

where the $I_{\mathcal{C}}$ is the indicator function of the closed convex set $\mathcal{C}$, i.e., $I_{\mathcal{C}}(\mathbf{z}) = 0$ if $\mathbf{z} \in \mathcal{C}$, $I_{\mathcal{C}}(\mathbf{z}) = \infty$ otherwise.
The problem in Eqs. (16) is solved first on $\mathbf{x}$, keeping $\mathbf{z}$ constant, then in $\mathbf{z}$, keeping $\mathbf{x}$ constant. Exploiting the augmented Langrangian metodology in a reduced formulation [4], the steps to be solved iteratively are

$$\begin{aligned} \mathbf{x}_{k+1} &= \arg\min_{\mathbf{x}} \left(\frac{1}{2}\mathbf{x}^T P\mathbf{x} + \mathbf{q}^T\mathbf{x} + \frac{\rho}{2}\|\mathbf{x} - \mathbf{z}_k + \mathbf{u}_k\|^2 \right) && (17a)\\ \mathbf{z}_{k+1} &= \Pi_{\mathcal{C}}\left(\mathbf{x}_{k+1} + \mathbf{u}_k\right) && (17b)\\ \mathbf{u}_{k+1} &= \mathbf{u}_k + \mathbf{x}_{k+1} - \mathbf{z}_{k+1} && (17c) \end{aligned}$$

where $\mathbf{u}$ is the Langrangian multiplier and $\Pi_{\mathcal{C}}$ is the conical projection function, while ρ is a coefficient, that can change during iterations, to balance the original cost function and the $\mathbf{x} - \mathbf{z}$ constraint.
The step on $\mathbf{x}$ in Eq. (17a) can be solved by means of an indirect formulation

$$\begin{bmatrix} P + \rho I & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x}_{k+1} \\ \boldsymbol{\nu} \end{bmatrix} = \begin{bmatrix} \rho(\mathbf{z}_k - \mathbf{u}_k) - \mathbf{q} \\ \mathbf{b} \end{bmatrix} \tag{18}$$

where $\boldsymbol{\nu}$ is the Lagrangian multiplier. Thus, it is required to solve a (potentially sparse) linear problem.
On the other hand, the step on $\mathbf{z}$ in Eq. (17b), requires to perform a conical projection, i.e., to solve the optimization problem

$$\mathbf{z}_{k+1} = \arg\min_{\mathbf{v}} \|\mathbf{v} - (\mathbf{x}_{k+1} + \mathbf{u}_k)\|^2 \qquad \text{subjected to } \mathbf{v} \in \mathcal{C}$$

that is to find the vector closest to $\mathbf{x}_{k+1} + \mathbf{u}_k$ inside the cone $\mathcal{C}$. As an example, consider the projection problem represented in Fig. 3, which represents the convex constraint (in green) $|x| \leq ky$. If we want to project the generic point (u, v) onto the cone, there are three distinct cases:

1) If $|u| \leq kv$, the convex constraint is already respected (i.e., the point is already in the green cone, as the blue point). So, there is no operation needed, and the projection is (u, v);
2) If $|u| \leq -kv$ (i.e., the point is inside the grey cone), the closest point is the origin, so the projection is $(0, 0)$;
3) Otherwise (as the red dot), the projection is $(\alpha k|u|, \alpha y)$, with α provided by the projection formula.

Even if this example illustrates a simple projection process, not all the projections onto convex sets enjoy a closed formula. Additionally, if more convex sets constraint a single variable, a more difficult problem, with no closed solution, must be solved (e.g., by means of Djekstra's algorithm). Thus, it is generally easier and useful having projection onto sets having a closed formula, and contemporarily having a single set for each variable.

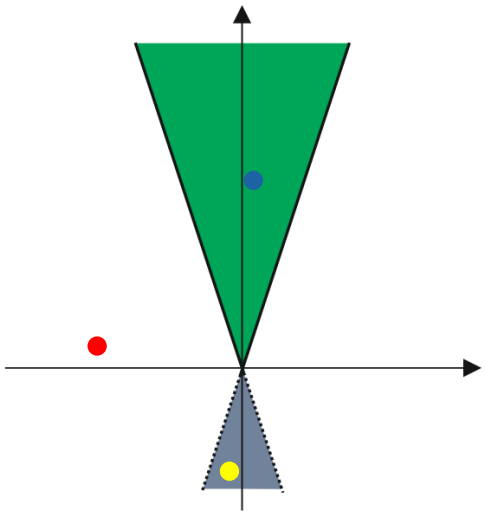


Figure 3 *Example of the projection onto a cone.*

3.2 The GPU-accelerated goal-oriented minimum-fuel problem

Following the needs set by the ADMM algorithm, Problem 2 must be modified accordingly. The single convex problem to be solved within Problem 2, with γ selected by the B&B algorithm and so fixed is

Problem 3
Given a fixed value for γ, find $\mathbf{y} = [\mathbf{x}_k, \mathbf{u}_k, \boldsymbol{\nu}_k, \eta_k]$, $k = 1, \dots, N$, such that

$$J := -z_N + \sum_{i=0}^{N} \lambda_i \|\boldsymbol{\nu}_k\|_1 + \sum_{i=0}^{N} \mu_i \max(0, \eta_k) \tag{19}$$

is minimized, subject to

$$
\begin{aligned}
&A\mathbf{x} + B\mathbf{u} + N\boldsymbol{\nu} = \mathbf{q} && \textit{Dynamics}\\
&\mathbf{x}_0 = \bar{\mathbf{x}}_0 && \textit{I.C.}\\
&T\mathbf{x} + V\mathbf{u} - \eta \leq K && \textit{Lin. control}\\
&\|\boldsymbol{\tau}_k\| \leq \Gamma_k && \textit{SOCC control}\\
&\|\mathbf{x} - \bar{\mathbf{x}}\|_1 \leq R && \textit{Trust region}\\
&\|\mathbf{r}_k - \mathbf{r}_{c_{i,k}}\| - \left(\frac{1}{\cos\theta}\mathbf{n}^T(\mathbf{r}_k - \mathbf{r}_{c_{i,k}})\right) \leq 0 && \textit{Goal constraints}
\end{aligned}
$$

where all the dependences on γ have been dropped for simplicity's sake. The ADMM-compatible goal-oriented problem, thus, becomes

Problem 4 (ADMM-compatible programming problem for the goal-oriented minimum-fuel problem)

Given a fixed value for γ, *find* $\mathbf{y} = [\mathbf{x}_k, \mathbf{u}_k, \boldsymbol{\nu}_k, \eta_k, \boldsymbol{\sigma}_k, \boldsymbol{\kappa}_i, g_i]$, $k = 1, \ldots, N$, *such that*

$$
J := -z_N + \lambda^2 \boldsymbol{\nu}^T \boldsymbol{\nu} + \mu^2 \boldsymbol{\eta}^T \boldsymbol{\eta} \tag{20}
$$

is minimized, subject to

$$
\begin{aligned}
&A\mathbf{x} + B\mathbf{u} + N\boldsymbol{\nu} = \mathbf{q} && \mathcal{D}\\
&\mathbf{x}_0 = \bar{\mathbf{x}}_0 && \mathcal{D}\\
&T\mathbf{x} + V\mathbf{u} - \eta - K = \boldsymbol{\sigma} && \mathcal{D}\\
&\boldsymbol{\sigma} \leq \mathbf{0} && \mathcal{C}\ \textit{(neg. orth.)}\\
&\|\boldsymbol{\tau}_k\| \leq \Gamma_k && \mathcal{C}\ \textit{(SOCC)}\\
&\|\mathbf{x} - \bar{\mathbf{x}}\|_1 \leq R && \mathcal{C}\ (\ell_1\textit{-ball})\\
&\mathbf{r}_k - \mathbf{r}_{c_{i,k}} = \boldsymbol{\kappa}_i && \mathcal{D}\\
&\frac{1}{\cos\theta}\mathbf{n}^T(\mathbf{r}_k - \mathbf{r}_{c_{i,k}}) = g_i && \mathcal{D}\\
&\|\boldsymbol{\kappa}_i\| \leq g_i && \mathcal{C}\ \textit{(SOCC)}
\end{aligned}
$$

Red highlights the changes with respect to the convex problem in Problem 2. First of all, the virtual controls are considered in the cost function in a quadratic fashion. This helps in having a quadratic problem that is better handled by the ADMM, while not changing the structure of the problem itself. Then, the linearized control constraint is split into a linear constraint and a negative orthant convex constraint. This change has two advantages on the conic projection: it moves $\mathbf{x}$ and $\mathbf{u}$ out of a conic constraint, so that they will appear once and giving a conic constraint on $\boldsymbol{\sigma}$ that is easier to implement. A similar change is also applied to the goal constraints, where the variable $\boldsymbol{\kappa}$ an g are introduced to have $\mathbf{x}$ appearing only once in the conic constraints and a simpler SOCC for the projection.

The ADMM problem must be then adapted to be run on a GPU. While it is generally compatible with a GPU kernel (i.e., a function to be run on a GPU), limitations given by the GPU architecture (e.g., no dynamic arrays, memory limitations, warp divergence, no BLAS functions) must be considered. Most of this characteristics are considered within the code implementations. For example, in order to avoid warp divergence, ternary operators and Boolean algebra are heavily exploited. On the other hand, same changes must be performed even on the problem itself. In order to solve the system in Eq. (18), reducing the computational burden and the memory footprint, an LU factorization with caching and no pivoting is applied. Specifically, considering that we have a symmetric block matrix, the LU factorization is

$$
\underbrace{\begin{bmatrix} I & 0 \\ A\hat{P}^{-1} & L_S \end{bmatrix}}_{L}\underbrace{\begin{bmatrix} \hat{P} & A^T \\ 0 & U_S \end{bmatrix}}_{U}\begin{bmatrix} \mathbf{x}_{k+1} \\ \boldsymbol{\nu} \end{bmatrix} = \begin{bmatrix} \rho(\mathbf{z}_k - \mathbf{u}_k) - \mathbf{q} \\ \mathbf{b} \end{bmatrix} \tag{21}
$$

where $\hat{P} = P + \rho I$, and $S = -AP^{-1}A^T$, and L_S and $U_S = DL_S$ the LDL factors of S. Factors are cached and changed only if ρ changes. Specifically, if ρ is changed, the system in Eq. (21) is solved by

1) Update $\hat{P} = \hat{P} + \Delta\rho I$
2) Compute $\hat{P}^{-1}$ by inverting its elements, since it is diagonal
3) Compute S
4) Compute the LDL decomposition of S
5) Compute $\mathbf{x}_{k+1}$ with forward and backward substitution

If ρ is kept constant, only step 5. is computed. In this algorithm, sparsity is exploited to reduce the runtimes. On the other hand, all the required conic projections are made of simple operations, so no adaptation is needed. A notable exception is, however, the thrust region constraint, which is a $\ell_1 - ball$ constraint. Even if a closed formula exists, it takes sorting and divisions, that are complex operations to be performed on a GPU. For this reason it is changed to a $\ell_2 - ball$ constraint which enjoys a simpler projection. Thus, the thrust region is enforced with

$$\|\mathbf{x} - \bar{\mathbf{x}}\|_2 \leq R/\sqrt{n_x} \tag{22}$$

where n_x is the length of the $\mathbf{x}$ vector, inserted to keep consistency with the region radius.

4 Results

The algorithm has been tested on a scenario from [18]. The scenario is based on the ESA's mission SATIS [42]. SATIS is a 12XL CubeSat having the aim to reach the asteroid Apophis before its flyby with the Earth in 2029 and perform scientific activities to characterize the asteroid before and after its close encounter. In this case, the target is modeled as a tri-axial ellipsoid [41]. The CubeSat is considered to be placed on an SSTO, generated with the procedure in Takahashi and Scheeres [40], and perturbed by the SRP and the gravitational perturbatons by the Sun, the Earth, and the Moon. This choice is common in CPOs close to minor bodies due to the stability of this kind of orbit in a high-perturbed environment, like the one of Apophis close to its close encounter.
Table 1 summarizes the main values of both target and spacecraft associated to this scenario. Simulation parameters and the SCP parameters are given in Table 2 and 3, respectively. All the simulations are carried out in Julia on 1) a laptop equipped with a 8-core Intel Core i7-10870H@2.20GHz laptop with 16 GB of RAM and a nVidia RTX3060 Laptop GPU, and 2) a nVidia Orin Jetson equipped with a 12-core ARM Cortex-A78AEwith 64 GB of low-power RAM and a dedicated nVidia Ampere GPU.

Table 1 *Scenario 1 parameters. $\mathcal{U}$ indicates the uniform distribution.*

	Parameter	Value	
Spacecraft	Initial orbit semimajor axis a_0	$\mathcal{U}(0.6\,\text{km}, 1.5\,\text{km})$	
	m_0	25 kg	
	$T_{\max}$	2 mN	
	I_{sp}	1500 s	
	C_r	1.3	
	A	$1.5\,\text{m}^2$	
Main body	Gravity type	Tri-axial ellipsoid	
	μ	$2.736\,463 \times 10^{-9}\,\text{km}^3/\text{s}^2$	
	(a, b, c)	(228.966 m, 159.026 m, 139.798 m)	[41]
	ω	283.46 °/d	[34]
Goals	α	20°	
	r_m	0.5 km	
	r_M	5 km	

A solution of this scenario, following the methodology in [18], is given in Fig. 4, considering ECOS as convex solver. Considering the full branch-and-bound algorithm, the runtime is 302 s.

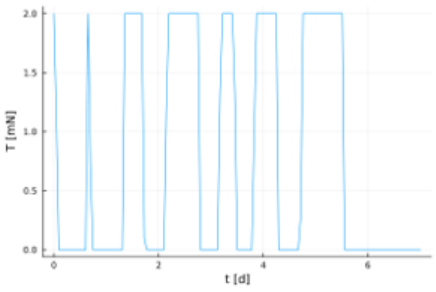
In order to have a proper comparison, first the same problem has been solved with ECOS and the custom ADMM, for a single convex problem (i.e., fixing the γ, without calling the B&B), on the laptop computer. ECOS is able to solve the single problem in 5.2 s, while ADMM takes about 9.5 s. As expected, ADMM is slower than an

Table 2 *Simulation parameters*

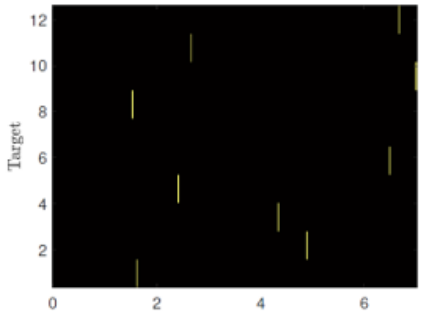
Parameter	Value
Time of Flight	7 d
Initial time t_0	January 1, 2029
Number of goals N_g	12
Maximum runtime	500 s
Maximum iterations	50

Table 3 *SCP parameters*

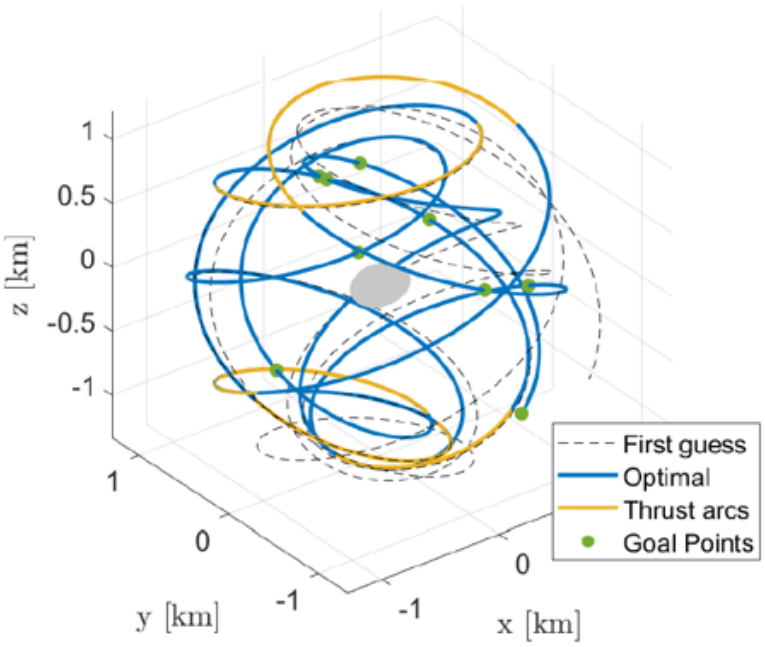
Parameter	Value
λ	100.0
M	10.0
R_0	100.0
(ρ_0, ρ_1, ρ_2)	$(0.01, 0.2, 0.85)$
(α, β)	$(1.5, 1.5)$
λ_g	$1/12$
$(\varepsilon_h, \varepsilon_z, \varepsilon_x, \varepsilon_R)$	$(10^{-6}, 10^{-4}, 10^{-8}, 10^{-4})$



(a) *Thrust Profile*



(b) *Target evolution. Yellow indicates when a target is respected.*



(c) *Optimal trajectory in 3D*

Figure 4 *Test scenario solution with the original methodology [18].*

IPM, but it requires less than a third of memory, since it makes fewer allocations. Furthermore, when cuADMM (i.e., the ADMM on the GPU) is used, a runtime of 450 s is observed, $\sim$ 50 times the CPU. This is related to the limited computational capabilities on the single process that the GPU has.
The same problem on the Jetson is slightly slower on the CPU, with a runtime of 12.2 s, while faster on the GPU, with a runtime of 325 s. Faster runtimes on the GPU are obtained both due to a higher-performance board and due to the fact that the laptop is equipped with a consumer GPU with gamer drivers, which guarantee worse performances.
If also the B&B algorithm is considered, a total runtime of 1080 s is observed on the laptop, while the Jetson requires only 325.7 s, having a total runtime closer to the CPU one on the laptop. The Jetson is significantly faster than the laptop since it can explore significantly more branches per B&B run, with faster runtimes for each node.

5 Conclusion

In this work, a fully GPU-accelerated implementation of the Alternating Direction Method of Multipliers tailored to the mixed-integer, goal-oriented trajectory optimization problems arising in close-proximity operations about minor bodies, has been presented. By reformulating the sequential convex programming subproblem into an ADMM-compatible quadratic conic program and carefully mapping both the sparse linear-solve and conic-projection steps to CUDA kernels with cached LU/LDL factorizations, this solver exploits thousands of GPU threads to explore branch-and-bound nodes in parallel. Benchmarking on an NVIDIA RTX3060 laptop GPU and an embedded NVIDIA Jetson Orin platform demonstrates that, although the per-iteration cost of the first-order ADMM method is higher than that of an interior-point solver on CPU, the GPU version matches or exceeds CPU performance when integrated into the full branch-and-bound framework. Specifically, on the Jetson board our GPU-accelerated solver completes the global optimality loop in 325 s, closely approaching the CPU runtime on a high-end laptop, and explores more B&B nodes in the same wall-clock time. These results confirm that GPU acceleration can overcome the onboard computational bottleneck for sequential convex MIP-based trajectory design, bringing real-time re-planning and autonomous decision-making within reach for future SmallSat and CubeSat missions.

Acknowledgments

This project is part of CASTOR, a project that has received funding from the European Union's Horizon Europe research and innovation programme under the Marie Skłodowska-Curie grant agreement no. 101103826.

References

[1] Andrea Accomazzo, Paolo Ferri, Sylvain Lodiot, Jose-Luis Pellon-Bailon, Armelle Hubault, Roberto Porta, Jakub Urbanek, Ritchie Kay, Matthias Eiblmaier, and Tiago Francisco. Rosetta operations at the comet. *Acta Astronautica*, 115:434–441, 2015.

[2] Peter G Antreasian, Coralie D Adam, Kevin Berry, Jeroen Geeraert, Kenneth M Getzandanner, Dolan Highsmith, Jason M Leonard, Erik J Lessac-Chenen, Andrew H Levine, James V McAdams, et al. Osiris-rex proximity operations and navigation performance at bennu. In *AIAA SCITECH 2022 Forum*, page 2470, 2022.

[3] Erik Asphaug. Growth and evolution of asteroids. *Annual Review of Earth and Planetary Sciences*, 37(1):413–448, 2009.

[4] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

[5] Stephen P Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

[6] Don Brownlee. The stardust mission: analyzing samples from the edge of the solar system. *Annual Review of Earth and Planetary Sciences*, 42(1):179–205, 2014.

[7] Stefano Campagnola, Naoya Ozaki, Yoshihide Sugimoto, Chit Hong Yam, Hongru Chen, Yosuke Kawabata, Satoshi Ogura, Bruno Sarli, Yasuhiro Kawakatsu, Ryu Funase, et al. Low-thrust trajectory design and operations of PROCYON, the first deep-space micro-spacecraft. In *25th International Symposium on Space Flight Dynamics*, volume 7. German Aerospace Center (DLR) Munich, Germany, 2015.

[8] Francesco Capolupo, Thierry Simeon, and Jean-Claude Berges. Heuristic guidance techniques for the exploration of small celestial bodies. *IFAC-PapersOnLine*, 50(1):8279–8284, 2017.

[9] Julie C Castillo-Rogez, Marco Pavone, Jeffrey A Hoffman, and Issa AD Nesnas. Expected science return of spatially-extended in-situ exploration at small solar system bodies. In *2012 IEEE Aerospace Conference*, pages 1–15. IEEE, 2012.

[10] Nancy L Chabot, Andrew S Rivkin, Andrew F Cheng, Olivier S Barnouin, Eugene G Fahnestock, Derek C Richardson, Angela M Stickle, Cristina A Thomas, Carolyn M Ernst, R Terik Daly, et al. Achievement of the planetary defense investigations of the double asteroid redirection test (dart) mission. *The Planetary Science Journal*, 5(2):49, 2024.

[11] Clark R Chapman. The hazard of near-earth asteroid impacts on earth. *Earth and Planetary Science Letters*, 222(1):1–15, 2004.

[12] Govind M Chari, Abhinav G Kamath, Purnanand Elango, and Behcet Acikmese. Fast Monte Carlo Analysis for 6-DoF Powered-Descent Guidance via GPU-accelerated Sequential Convex Programming. In *AIAA SciTech 2024 Forum*, page 1762, 2024.

[13] Yuwen Chen, Danny Tse, Parth Nobel, Paul Goulart, and Stephen Boyd. Cuclarabel: Gpu acceleration for a conic optimization solver. *arXiv preprint arXiv:2412.19027*, 2024.

[14] Andrew F Cheng, Andrew S Rivkin, Patrick Michel, Justin Atchison, Olivier Barnouin, Lance Benner, Nancy L Chabot, Carolyn Ernst, Eugene G Fahnestock, Michael Kueppers, et al. Aida dart asteroid deflection test: Planetary defense and science objectives. *Planetary and Space Science*, 157:104–115, 2018.

[15] Elisabetta Dotto, Vincenzo Della Corte, Marilena Amoroso, Ivano Bertini, John R Brucato, Andrea Capannolo, Biagio Cotugno, Gabriele Cremonese, Valerio Di Tana, Igor Gai, et al. Liciacube-the light italian cubesat for imaging of asteroids in support of the nasa dart mission towards asteroid (65803) didymos. *Planetary and Space Science*, 199:105185, 2021.

[16] Fabio Ferrari, Vittorio Franzese, Mattia Pugliatti, Carmine Giordano, and Francesco Topputo. Preliminary mission profile of hera's milani cubesat. *Advances in Space Research*, 67(6):2010–2029, 2021.

[17] Fabio Ferrari, Vittorio Franzese, Mattia Pugliatti, Carmine Giordano, and Francesco Topputo. Trajectory options for hera's milani cubesat around (65803) didymos. *The Journal of the Astronautical Sciences*, 68(4):973–994, 2021.

[18] C Giordano, S Campagnola, and F Topputo. Goal-oriented trajectories about minor bodies using Sequential Convex Programming: A Deterministic Approach. In *35th AAS/AIAA Space Flight Mechanics Meeting*, pages 1–26, 2025.

[19] C Giordano, S Campagnola, and F Topputo. Goal-Oriented Trajectories About Minor Bodies Using Sequential Convex Programming: a Stochastic Approach. In *35th AAS/AIAA Space Flight Mechanics Meeting*, pages 1–18, 2025.

[20] Karl-Heinz Glassmeier, Hermann Boehnhardt, Detlef Koschny, Ekkehard Kührt, and Ingo Richter. The rosetta mission: flying towards the origin of the solar system. *Space Science Reviews*, 128:1–21, 2007.

[21] Hannah R Goldberg, Özgür Karatekin, Birgit Ritter, Alain Herique, Paolo Tortora, Claudiu Prioroc, Borja Garcia Gutierrez, Paolo Martino, and Ian Carnelli. The juventas cubesat in support of esa's hera mission to the asteroid didymos. 2019.

[22] AM Hein, M Saidani, and H Tollu. Exploring potential environmental benefits of asteroid mining. In *69th International Astronautical Congress (IAC), Bremen, Germany, 1–5 October 2018*, 2018.

[23] Michael Kueppers, Paolo Martino, Ian Carnelli, and Patrick Michel. Ramses–esa's study for a small mission to apophis. In *AAS/Division for Planetary Sciences Meeting Abstracts*, volume 55, pages 201–08, 2023.

[24] DS Lauretta, SS Balram-Knutson, Eea Beshore, WV Boynton, C Drouet d'Aubigny, DN DellaGiustina, HL Enos, DR Golish, CW Hergenrother, ES Howell, et al. Osiris-rex: sample return from asteroid (101955) bennu. *Space Science Reviews*, 212:925–984, 2017.

[25] Pierre-Louis Lions and Bertrand Mercier. Splitting algorithms for the sum of two nonlinear operators. *SIAM Journal on Numerical Analysis*, 16(6):964–979, 1979.

[26] Peter Lord, Scott Tilley, David Y Oh, Dan Goebel, Carol Polanskey, Steve Snyder, Greg Carr, Steven M Collins, Gregory Lantoine, Damon Landau, et al. Psyche: Journey to a metal world. In *2017 IEEE Aerospace Conference*, pages 1–11. IEEE, 2017.

[27] Danylo Malyuta, Taylor P Reynolds, Michael Szmuk, Thomas Lew, Riccardo Bonalli, Marco Pavone, and Behçet Açıkmeşe. Convex optimization for trajectory generation: A tutorial on generating dynamically feasible trajectories reliably and efficiently. *IEEE Control Systems Magazine*, 42(5):40–113, 2022.

[28] Patrick Michel, Michael Küppers, Adriano Campo Bagatin, Benoit Carry, Sébastien Charnoz, Julia De Leon, Alan Fitzsimmons, Paulo Gordo, Simon F Green, Alain Hérique, et al. The esa hera mission: detailed characterization of the dart impact outcome and of the binary asteroid (65803) didymos. *The planetary science journal*, 3(7):160, 2022.

[29] Andrea C Morelli, Alessandra Mannocchi, Carmine Giordano, Fabio Ferrari, and Francesco Topputo. Initial trajectory assessment of a low-thrust option for the ramses mission to (99942) apophis. *Advances in Space Research*, 73(8):4241–4253, 2024.

[30] Jorge Nocedal and Stephen J Wright. *Numerical optimization*. Springer, 1999.

[31] J Oro, T Mills, and A Lazcano. Comets and life in the universe. *Advances in Space Research*, 15(3):81–90, 1995.

[32] Armen Poghosyan and Alessandro Golkar. CubeSat evolution: Analyzing CubeSat capabilities for conducting science missions. *Progress in Aerospace Sciences*, 88:59–83, 2017.

[33] Florian A Potra and Stephen J Wright. Interior-point methods. *Journal of Computational and Applied Mathematics*, 124(1-2):281–302, 2000.

[34] P Pravec, P Scheirich, J Ďurech, J Pollock, P Kušnirák, K Hornoch, A Galad, D Vokrouhlickỳ, AW Harris, Emmanuel Jehin, et al. The tumbling spin state of (99942) apophis. *Icarus*, 233:48–60, 2014.

[35] L Prockter, S Murchie, A Cheng, S Krimigis, R Farquhar, A Santo, and J Trombka. The near shoemaker mission to asteroid 433 eros. *Acta Astronautica*, 51(1-9):491–500, 2002.

[36] Antonio Rizza, Carmine Giordano, and Francesco Topputo. A goal-oriented guidance approach for binary asteroids exploration. *Astrodynamics*, 2024. In press.

[37] Small Bodies Assessment Group (SBAG). *Goals and Objectives for the Exploration and Investigation of the Solar System's Small Bodies*.

[38] Daniel Jay Scheeres. Dynamics about uniformly rotating triaxial ellipsoids: applications to asteroids. *Icarus*, 110(2):225–238, 1994.

[39] David A Surovik and Daniel J Scheeres. Adaptive reachability analysis to achieve mission objectives in strongly non-keplerian systems. *Journal of Guidance, Control, and Dynamics*, 38(3):468–477, 2015.

[40] Shota Takahashi and Daniel J Scheeres. Higher-order corrections for frozen terminator orbit design. *Journal of Guidance, Control, and Dynamics*, 43(9):1642–1655, 2020.

[41] Giulia Valvano, Othon Cabo Winter, Rafael Sfair, R Machado Oliveira, G Borderes-Motta, and TS Moura. Apophis–effects of the 2029 earth's encounter on the surface and nearby dynamics. *Monthly Notices of the Royal Astronomical Society*, 510(1):95–109, 2022.

[42] Roger Walker. Overview of esa lunar & interplanetary cubesat missions. In *Interplanetary Small Satellite Conference*, 2023.

[43] Roger Walker, David Binns, Cristina Bramanti, Massimo Casasco, Paolo Concari, Dario Izzo, Davar Feili, Pablo Fernandez, Jesus Gil Fernandez, Philipp Hager, et al. Deep-space CubeSats: thinking inside the box. *Astronomy & Geophysics*, 59(5):24–30, 2018.

[44] Zhenbo Wang and Michael J Grant. Minimum-fuel low-thrust transfers for spacecraft: A convex approach. *IEEE Transactions on Aerospace and Electronic Systems*, 54(5):2274–2290, 2018.

[45] Sei-ichiro Watanabe, Yuichi Tsuda, Makoto Yoshikawa, Satoshi Tanaka, Takanao Saiki, and Satoru Nakazawa. Hayabusa2 mission overview. *Space Science Reviews*, 208:3–16, 2017.

[46] Yue Yu, Purnanand Elango, Ufuk Topcu, and Behçet Açıkmeşe. Proportional–integral projected gradient method for conic optimization. *Automatica*, 142:110359, 2022.