ATTITUDE GUIDANCE USING ON-BOARD OPTIMISATION

P. Bodin [(1)], K. Lindqvist [(1)], D. Seelbinder [(2)], A. Makarow [(2)], J. Garrido [(2)], M. DiCarlo [(3)], A. Visintini [(3)], A. Hyslop [(4)], V. Preda [(4)]

[(1)]*OHB Sweden, Kista, Sweden, per.bodin@ohb-sweden.se*

[(2)]*DLR, Bremen, Germany, david.seelbinder@dlr.de*

[(3)]*OHB System, Bremen, Germany, alessandro.visintini@ohb.de*

[(4)]*ESTEC/ESA, Noordwijk, The Netherlands, andrew.hyslop@esa.int*

ABSTRACT

The on-board ability to autonomously plan and execute constrained attitude manoeuvres is expected to play an important role in many future space missions. The work presented in this paper summarizes the results from a recently completed ESA study in which such functionality was examined. The study included the application of on-board embedded optimization techniques to solve constrained attitude guidance problems. Different heritage methods not based on on-board optimisation were also developed and applied for comparison. The study demonstrated the capabilities in a number of test cases associated with two benchmark problems based on the Comet Interceptor and Theseus mission studies. The performance was examined within Monte Carlo simulations as well as within execution on a flight-like hardware platform.

1 INTRODUCTION

On-board autonomous planning for execution of constrained trajectories is expected to be an important capability for many future space missions and will potentially allow for efficient handling of failure scenarios, dynamic environments or complex objectives.

Autonomous planning typically relies on the ability to make model-based predictions about the future evolution of on-board states and how these evolve as a result of actuation and on-board autonomous decisions. Target objectives can be formulated and solved as constrained optimization problems.

Applications within attitude guidance can e.g. involve the minimization of errors, transfer time, or propellant consumption during a slew manoeuvre while avoiding certain pointing directions e.g. to protect payload instruments from Sun illumination. The prediction model will act as additional constraints in the associated optimization problem, which generally becomes non-linear and non-convex such that guaranteeing a globally optimal solution becomes computationally very demanding.

The problem can be tackled by applying various convexification techniques to arrive at an approximation that is possible to solve in a practical way and that is suitable for implementation on an embedded system. In recent years, significant efforts have been made to provide such efficient solutions to several different spacecraft engineering problems [1] and many supporting tools are available to support deployment on embedded flight systems.

The results presented in this paper are based on a recently completed ESA study that examined the use of on-board optimization for attitude guidance applied to two benchmark problems based on two ESA mission studies for which on-board optimised attitude guidance can bring advantages, the Comet

Interceptor (COMET-I) mission [2] and the Transient High Energy Sky and Early Universe Surveyor (THESEUS) [3]. The application of optimization methods to the COMET-I mission study was examined in [4], where also the background of and motivation for using optimization methods are provided. In the case of THESEUS, optimization-based attitude guidance can potentially provide efficient spacecraft reorientation in the presence of pointing avoidance constraints.

The study consisted of a literature study and implementation of optimization functionality suitable for embedded systems. In parallel, corresponding attitude guidance functionality was developed based on heritage techniques, designed as alternative rule-based methods with a significantly lower computational complexity. The heritage methods were used as comparison with the optimization techniques in terms of performance and computational complexity, as well as in terms of development effort. The study included Monte Carlo simulations of the benchmark problems as well as the execution of the optimization methods on a flight-like hardware platform.

The paper is organized as follows. Section 2 provides a brief description of the state-of-the-art of convex programming and describes how the optimization technique for the study was selected. Section 3 defines the benchmark scenarios and test cases used in the study and describes how the scenarios are transcribed into optimization problems. Section 4 summarizes the design of the heritage solutions and Section 5 describes the embedded optimization design and tuning. Section 6 summarizes the results from the performance analysis while Section 7 provides the conclusions from the study.

2 SELECTION OF OPTIMIZATION TECHNIQUE

2.1 State-of-the-art in Convex Programming

The use of numerical optimization has experienced a huge acceleration over the last years. Convex optimization is appealing mainly since the local optimum is also the global optimum and since very efficient general-purpose and highly dedicated solvers exist. In addition, the dependency on initial guesses is completely lifted.

Convex optimization methods are flexible enough to permit the modelling of a huge number of problems of practical interest. A particularly successful category of problems, the Second-Order Cone Programming (SOCP) problem is a generalization of quadratic programming that includes the possibility of embedding conic constraints in the formulation. This type of formulation has been widely used in many fields, and with particular success in GNC, especially for guidance applications (e.g., powered-descent and landing guidance [5]–[7], and atmospheric re-entry). More notably, this technology has been successfully employed on several rockets and vehicles, including the Falcon 9 [8] and the experimental DLR vehicle EAGLE [9], [10].

Part of the success of convex optimization is certainly due to the availability of modern numerical methods to deal with it. Beyond the already mentioned CVXOPT and qpOASES, it is worth mentioning ECOS [11]. ECOS implements a standard primal-dual Mehortra predictor-corrector solver, but with search directions found by solving a symmetric indefinite KKT system. The system of equations is solved by applying state-of-the-art LDL factorization and elimination of the numerical determination of pivoting sequence for improving the performance of the solver. An even more customized solver for SOCPs to be employed in embedded solutions has been proposed by Dueri et al. [12]. Given the known structure of the problem the interior-point method could be carefully tailored for the specific problem to be solved, leading to drastically lower times if compared to general-purpose embedded solvers.

2.2 Handling of Non-Convex Constraints

In this study, it was compared how different methods prepare, augment or transform nonlinear constraints before linearization is performed: The most categorical difference is whether nonlinear constraints are addressed during the transcription of an optimal control problem, before it is transformed into a static optimization problem, or if the nonlinear constraints enter the static optimization problem directly. The former case leads to the recently surfacing methods of successive convexification [13]–[15] and successive convex programming [16], [17] (we use the terms successive convexification and successive convex programming interchangeably, abbreviated as SCVX). The latter leads to a nonlinear programming problem (NLP).

SCVX is designed for solving discretized nonlinear optimal control problems. Nonlinear constraints are linearized at the operating point given by the last iteration, in that regards SCVX is no different from generic NLP methods. But the artificial infeasibility that can arise from this linearization is treated through systematically relaxing the constraints through slack variables which are included in an augmented cost function, such that they are driven to zero.

2.3 Discretization Techniques for Optimal Control Problems

Properties and empirical results for commonly used OCP discretization techniques are compared in [18] and [9]. The method is concerned with discretization of a function or a functional over an interval of an independent variable.

Optimal control solvers such as DIDO or GPOPS-II first solve the OCP on a fixed given grid and after the solution has been obtained, the grid is iteratively adjusted by introducing additional discretization points. While this strategy is safe and preferable in a desktop environment, in an embedded environment it has the disadvantage, that the number of optimization variables changes, and with that the structure and non-zero entries in the problem matrices. This does not match well with static memory and requires customized solvers to solve the optimization problem.

Pseudospectral methods gained attention due to their straightforward implementation and their properties, such as a pseudospectral (i.e., quasi-exponential) convergence of the discrete problem's solution towards the continuous one, as the number of nodes is increased. The methods can be generalized to break the nonlinearity in the dynamics by separating the problem into multiple intervals which are connected by differential defect constraints, very similar to multiple shooting [19]. This increases the problem size, but also the sparsity of the problem matrices, making it an attractive method to pair with sparse solvers. The degree of the polynomial used in each interval can be controlled to trade sparsity/computation time against local discretization error.

There is no clear 'one wins all' situation for discretization methods as also reported in [18]. Because of prior experience and familiarity, we prefer the trapezoidal method with first order control interpolation for general prototyping of new OCP applications. Based on the problem specific impacts of problem matrix size, sparsity and discretization error, a choice was made to implement a generalized hp-pseudospectral method.

2.4 Software Selection Logic and Trade-off Analysis

For safety critical or very high-cost applications, such as space flight, reliability is the dominating criterion for making design choices. As such it is axiomatic to rely on convex methods whenever possible because of their global optimality and well-understood convergence properties. If the desired objective cannot be fulfilled by a purely convex method, the approach with the best understood and numerically tested convergence properties becomes the first choice. With this in mind we evaluated the given benchmark problems [2]–[4] and acknowledged the presence of conic constraints, nonlinear dynamics and a time optimality objective. Having in mind the cascade of problem complexity classes it follows that:

1. The chosen method had to be able to handle second-order cone constraints in natural convex form, without the need to linearize or approximate the cone. This reduced the “fitting” problem classes for the study to second-order-cone programming (SOCP) and semi-definite programming (SDP). As there are no semi-definite cone constraints present in either benchmark problem, the less computationally expensive SOCP was the obvious choice.
2. Nonlinear dynamics and especially time optimality cannot be expressed within a (single) convex problem, thus required to compromise enough on convexity to include a strategy that can handle nonlinear/nonconvex problems.
3. Because of the complexity and development effort for numerical optimization algorithms, the development of solver software from scratch was out of scope for the study. Thus, we needed to evaluate the candidate methods in combination with the availability and maturity of an existing solver software implementation that was able to eventually fulfil safety-critical real-time software requirements.

After a filtering taking into account the above three drivers, three candidates remain that fulfil the major driving selection criteria: PANOC+OpEn, FORCES Pro, and ESE + SCP (DLR software).

In conclusion, the selected baseline strategy was to combine second-order cone programming with successive convexification into sequential convex programming, inspired by the work of Mao and Bonalli [14], [16]. The DLR Embedded Solver Engine (ESE) is a tried and tested tool, which generates static memory. The SOCP problem class perfectly fits the constraints required to model the benchmark scenarios and the DLR experience in using SCP for VTVL vehicles shows, that it is a reliable, fast convergent method.

3 BENCHMARK SCENARIO DEFINITION

This section defines the respective COMET-I and THESEUS benchmark scenarios and demonstrates how the scenarios are transcribed into optimization problems.

3.1 COMET-I

The Comet Interceptor (COMET-I) benchmark scenario is a based on the Comet Interceptor mission which is an upcoming mission that has just completed preliminary design review.

The COMET-I benchmark scenario consists of a comet fly-by at 1000 km from the target with a relative velocity of 70 km/s. The analysis is performed for a duration of 200 s centred around the point of closest approach. The scenario assumes a visual camera with a FoV of 0.92°×0.92° and an IR camera with a FoV of 10°×10°. The comet is kept in the field of view of the cameras by slewing a mirror or the S/C (no longer in mission baseline) during the fly-by.

The S/C is equipped with four reaction wheels for attitude control and during the flyby, the S/C is controlled with reaction wheels only in order to avoid contamination of scientific in-situ measurements of the comet’s dust environment. (Note that the baseline mission has waived this constraint.) Of relevance for the problem is also the solar arrays that increase the area exposed to risk of being hit by a dust particle. The high slew rate, dust environment, as well as the “one-shot” characteristics of the mission profile constitute the main fundamental challenges.

Simulation Cases

The results from two COMET-I simulation cases are included in this report:

- Test case 1: Nominal fly-by case with all equipment functional
- Test case 2: Fly-by with large dust particle impact near closest approach, leading to saturation of one reaction wheel, loss of attitude and loss of comet tracking.

Constraints and Objectives

The constraints and objectives associated with the optimal control problem for the COMET-I benchmark scenario are summarized in Table 1. The four objectives are combined in a weighted sum.

Table 1: Summary of COMET-I constraints and objectives.

Constraint/objective	Mean
Constraints	
Dynamic constraints	Rigid body dynamics and attitude kinematics. Angular momentum from reaction wheels.
Reaction wheel constraints	Unit level constraints for angular momentum and torque. Zero loss torque assuming wheel internal compensation.
Star tracker pointing constraint	The Star Tracker pointing constraints are omitted since it is assumed that the attitude is obtained from a gyro-stellar estimator.
Instrument pointing constraint	Sun exclusion angles of 30° are introduced for the two payload cameras.
Objectives	
Instrument FoV objective	Minimize deviation outside of the conic FoV of the instruments.
Instrument LoS objective	Minimize the angle between the instrument LoS and the vector to the comet.
Solar array pointing objective	Minimize the deviation of the solar array rotational axis.
Command energy objective	Minimize reaction wheel command energy.

3.2 THESEUS

The reference orbit of THESEUS is a circular LEO with an altitude of 600 km and 5.4° inclination.

The payload of the spacecraft is an infrared (IR) telescope, with a Sun exclusion angle of 60°. THESEUS is expected to autonomously slew the telescope towards the target source location. The target source is different in case of internally (on-board) or externally (ground) triggered target sources. In the case of internally triggered target source, the target is within ±30.5° pitch and ±58.5° yaw from the initial attitude; in case of externally triggered source, the target is on the –Z axis of the J2000 reference frame.

THESEUS is equipped with four star trackers (STS), characterized by an Earth-limb exclusion angle of 20° and a Sun exclusion angle of 27°. The star trackers impose a limitation on the maximum angular rate of the spacecraft, whose value should not be higher than 5°/s. Reaction wheels (RW) are used for actuation.

Simulation Cases

The results from two out of four THESEUS simulation cases are included in this report:

- Test case 2: Minimization of slew time with external trigger source and one STS failure.
- Test case 3: Minimization of slew time with external trigger source and one RW failure.

The objective and constraints are analysed in more details in the next sections. The considered slew is an externally triggered target source slew with 180° rotation of the infrared telescope boresight, while maintaining the correct orientation of the Sun-shield at the end of the slew.

Constraints and Objectives

The constraints and objectives associated with the optimal control problem for the THESEUS benchmark scenario are summarized in Table 2.

Table 2: Summary of THESEUS constraints and objectives.

Constraint/objective	Mean
Constraints	
Dynamic constraints	The dynamic constraints include the rigid rotational dynamics and the attitude kinematics. Reaction wheels are included as pure integrations.
Reaction wheel constraints	Unit level constraints for angular momentum and torque. Zero loss torque assuming wheel internal compensation.
Maximum slew rate of the spacecraft	The spacecraft is constrained to a maximum slew rate, determined by the star trackers.
Forbidden zone for the telescope	The IR telescope should avoid pointing towards the Sun, within its exclusion angle.
Forbidden zones for the star trackers	For THESEUS, one star tracker must be available at all times. Therefore, simultaneous Earth and Sun blinding of all the star trackers should be prevented.
Objectives	
Slew time	Minimization of the slew time

3.3 Scenario Transcription

Discretization

The Radau pseudospectral discretization method is used for the optimization procedure. Based on the properties of pseudospectral methods, a differentiation matrix $\mathbf{D}$ is computed which can be used to enforce first order time derivative constraints. This matrix allows us to approximate the continuous derivative using discrete samples. With this matrix we can convert the differential equation

$$\dot{x}(t) = f\big(x(t), u(t)\big)$$

into the discrete and purely algebraic constraint

$$f(x_k, u_k) - \frac{2}{t_f - t_0}\sum_i D_{ki} x_i = 0$$

In this way, we have now ways to compute integrals as well as derivatives by means of operating with discrete quantities. That is, we have the Gaussian quadrature which approximates integrals, and we have the Differentiation matrix which approximates local time derivatives.

A good choice of sample points is a proportional mapping of the roots of Legendre-based polynomials, such as the Radau polynomial. The polynomial is the sum two Legendre polynomials of consecutive order:

$$R_N(\tau) = P_N(\tau) + P_{N-1}(\tau), \quad \tau \in [-1,1]$$

where $P_N(\tau)$ is the Legendre polynomial of order N.

Linearization

Nonlinear constraints can be enforced in a successive-convexification algorithm by providing the convex solver with an affine approximation of the constraints. In this way, enforcing a nonlinear constraint is a matter of enforcing the linear approximation of said constraint in an iterative manner until convergence is achieved.

4 HERITAGE SOLUTION DESIGN

The detailed design of the heritage solutions is summarized below for the respective COMET-I and THESEUS benchmark problems.

4.1 COMET-I

The guidance functions associated with the COMET-I heritage solutions is divided into separate strategies for the Nominal and Contingency attitude cases.

Nominal attitude case (Test case 1)

Test case 1 is a nominal case where the S/C follows its nominal attitude profile but is limited by its actuation constraints. Assuming that the roll and pitch errors are small, the slew motion is characterized by the yaw-angle and the S/C slews around its y-axis to track the centre of comet (CoC). The main strategy, when the nominal attitude profile is not possible to follow is to saturate the rate profile at the corresponding maximum rate and to transition from the nominal attitude to the maximum rate through a linear rate profile. The strategy is illustrated with Figure 1. The torque and rate capacities are tracked depending on the reaction wheel configuration and the current reaction wheel speed distribution.

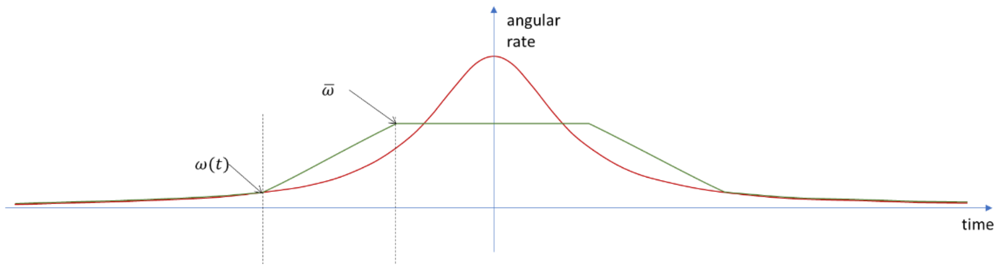


Figure 1: Basic constrained approximation strategy for the LoS angle.

Contingency attitude case (Test case 2)

Test case 2 is a contingency attitude case that includes a large dust particle impact that leads to saturation of a reaction wheel and loss of S/C attitude. The main strategy is that, when a sudden change in momentum is detected, a new guidance profile is generated that takes the S/C back to the nominal attitude profile. The nominal profile is obtained as described in the previous section. The slew-back problem is in this way a non-rest to non-rest slew problem. In addition, the final angle and rate depends on the transfer time, which in its turn will depend on the final conditions. The solution is obtained numerically taking into account the numerical evolution of the nominal angular and rate profiles.

4.2 THESEUS

The chosen heritage guidance algorithm for THESEUS is the eigenaxis slew algorithm (EGS). Alternative heritage guidance methods that were evaluated for THESEUS are artificial potential function and path planning algorithms which both were rejected because of their computational complexity.

5 EMBEDDED OPTIMIZATION DESIGN AND TUNING

The key observations that enable the real-time use of the SCVX algorithm are the following. After the user has completed the problem implementation the following is true:

1. The type, size, structure (and therefore memory use) of all components of the discretized optimal control problem and user defined data are known or can be automatically derived, such that when the transcription to the SOCP interface is performed, also the size and structure of all components of the resulting SOCP problem are known.
2. The type, size and structure of the SOCP problem's real-time inputs do not change between different SCVX iterations.

It follows that only one SOCP sparsity structure is required for an entire run of the SCVX algorithm. We capitalize on that by using a static memory SOCP solver that is optimized for the given sparsity structure. This results into a performance advantage over a general-purpose solver because some algebraic operations can be performed before runtime, and thus reduce the required execution time.

5.1 Overcoming the Duality Between Rapid-Prototyping and Real-Time Environments

To be able to use the same codebase for rapid-prototyping and real-time application we desire to circumvent that high level "convenience" code from the "User Interface Layer" and most code from the "Transcription Layer" become part of the real-time software at all. To facilitate this the software is built such, that the complete internal state of the framework can be "snapshotted", saved and loaded again later to seamlessly resume operation from the snapshot point. Source code that was run prior to the snapshot and that is not required anymore afterwards can be completely replaced by loading the snapshot, and the dependency to this source code is eliminated.

We utilize this pattern to create a snapshot after the transcription layer has been executed. At this point the user has defined the problem (User Interface Layer) and the software has deduced all sizes and the structure of all matrices. In particular the software has deduced the knowledge which matrix entries have to be updated in every iteration step and which entries stay constant.

From the perspective of the real-time application, the snapshot data is a compile time constant, with one exception: the real-time parameters. To resolve this, note the following: The snapshot data is a true compile time constant with respect to type and size, only the numeric value of the real-time parameters may change. The snapshot already contains all memory required to hold the real-time data, but the new numeric values must be written to the correct location. Because of the general nature of the framework this task cannot be performed by the software automatically, additional user input is required. The user must write a wrapper function that receives the snapshot and the real-time parameters as input and outputs the snapshot updated with the real-time data, which will then be used in the SCVX loop.

5.2 SCVX Real-Time Strategy

The transcription and the SCVX logic are implemented in MATLAB using dense algebra, the SOCP solver is implemented in C++ using sparse algebra. The problem transcription is performed offline, before run-time and sparsity patterns and index structures are prepared as compile time constants.

At run-time, the SCVX solver receives the real-time inputs which are not known ahead of time, i.e. the state and other quantities depending on measurements or the on-board computing system. The problem matrices which have already been prepared in memory during an initialization phase, are updated with the real-time inputs.

To cover the sparsity structures for the THESEUS mission two-separate static SCVX solvers are required: The first solver is used for control with the reaction wheels, the second static solver is used for control with the thrusters. For the Comet-I mission the sparsity structure of all test cases is compatible with each other such that a single static solver is sufficient.

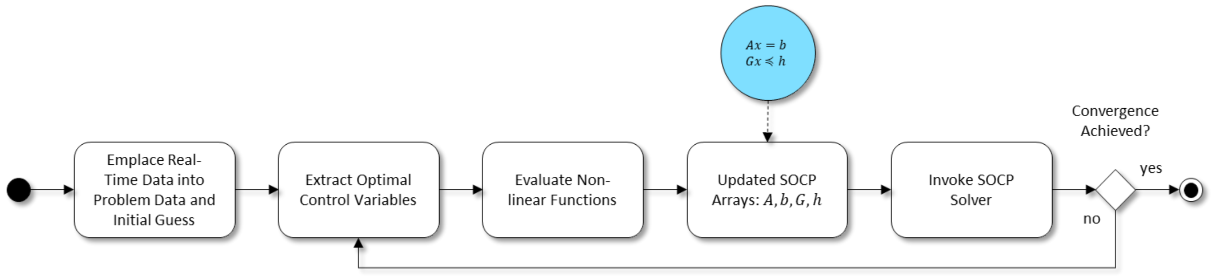
The pre-computed transcription data includes the sizes of all memory objects including the sparsity patterns of all problem matrices. Using this information, the source code of a static memory solver that is dedicated for the given sparsity structure is generated (the static SOCP solver and its generator were not part of the software developed during the project). The dedicated solver is then linked with the SCVX layer to produce a static memory SCVX-Optimal-Control Solver.

In each SCVX iteration a SOCP of the form

$$
\begin{aligned}
\text{minimize} \quad & \mathbf{c}^\top \mathbf{x} \\
\text{subject to} \quad & \mathbf{A}\,\mathbf{x} = \mathbf{b} \\
& \mathbf{G}_{\text{ineq}}\,\mathbf{x} \leq \mathbf{h}_{\text{ineq}} \\
& \mathbf{G}_{\text{cone}}\,\mathbf{x} \preceq \mathbf{h}_{\text{cone}}
\end{aligned}
$$

is solved. It minimizes a linear function of a set of decision variables subject to affine equality, inequality and second-order-conic constraints.

This process requires for some constraints to be expressed in a convex way at the level of the formulation of the optimal control problem, but it also needs every nonlinear function associated with the problem to be linearized in each iteration. The goal is to successively solve local convex approximations of the nonlinear problem. Explicit trust region constraints are usually enforced in order to guarantee that the next solution stays within proximity of the previous one and, therefore, that the linear approximation model is adequate. For the benchmark problems implemented in this project, the trust region constraints remain unchanged throughout the optimization process. Figure 2 shows a simplified flowchart of the main procedure of the SCVX algorithm.



$Ax = b$
$Gx \preccurlyeq h$
Convergence Achieved?
Emplace Real-Time Data into Problem Data and Initial Guess
Extract Optimal Control Variables
Evaluate Non-linear Functions
Updated SOCP Arrays: A, b, G, h
Invoke SOCP Solver
yes
no

Figure 2: SCVX loop top-level logic.

6 PERFORMANCE ANALYSIS

The main closed-loop Monte Carlo results are summarized in Sections 6.1 and 6.2.

6.1 COMET-I

This section shows some results from the test case 1 and 2 as defined in Section 3.1.

Test Case 1: Nominal fly-by

The results show that the planned S/C rate and the corresponding angular momentum fully utilizes the available 3.2 Nms. SCvx plans a slight deviation in the slew axis to find the optimal slew with maximal rate and minimum attitude deviation. The Heritage solution is more conservative and plans the slew around Y-axis only.

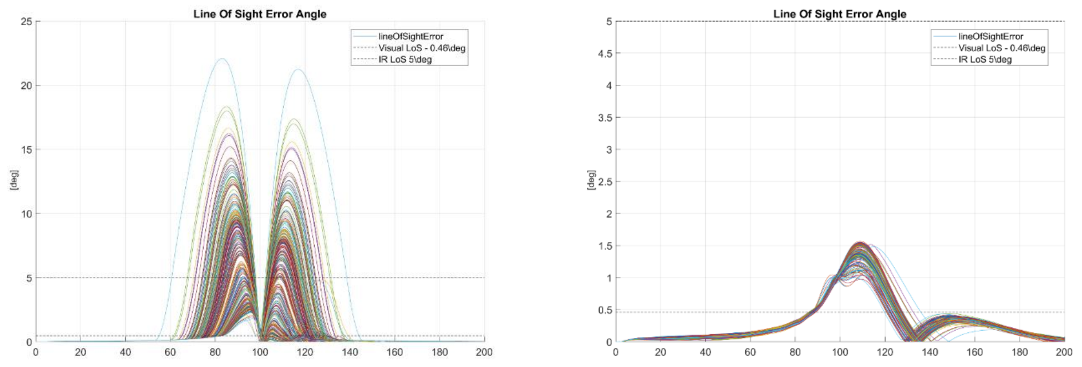


Figure 3: LoS error angle, test case 1. Left: Heritage. Right: SCvx.

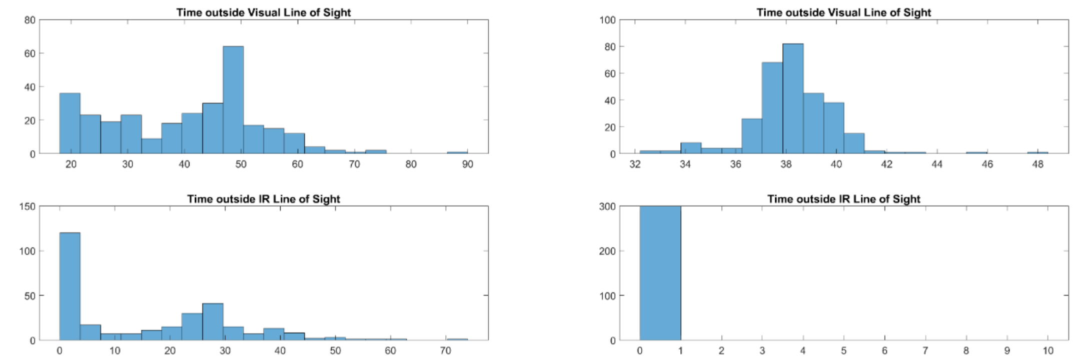


Figure 4: LoS metrics, test case 1. Left: Heritage. Right: SCvx.

Test case 2: Wheel saturation

The results from test case 2 demonstrate significantly better results for SCvx than for the heritage solution. The Heritage solution was designed for early impact and tries to get back to the nominal trajectory. This strategy does not work well in the case of a late impact. SCvx plans the late impact as well as the early even though the late impact is often more challenging. For the early impact case, comparing the solutions shows that the SCvx can plan a higher yet achievable rate than the heritage solution, which is more conservative.

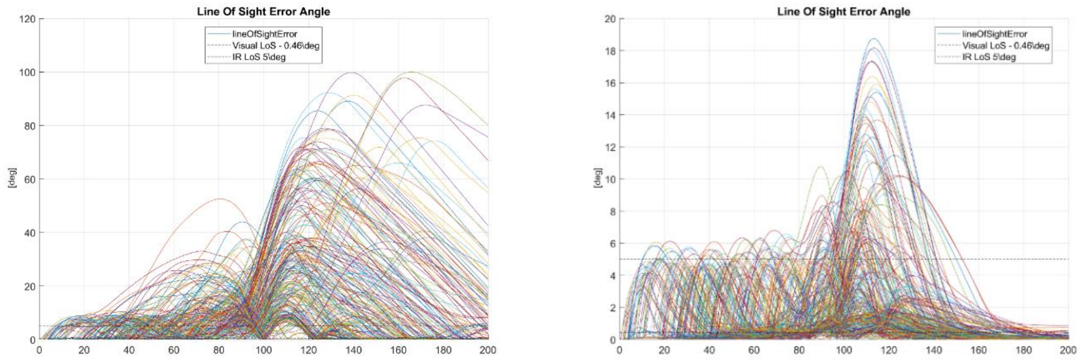


Figure 5: LoS error angle, test case 2. Left: Heritage. Right: SCvx.

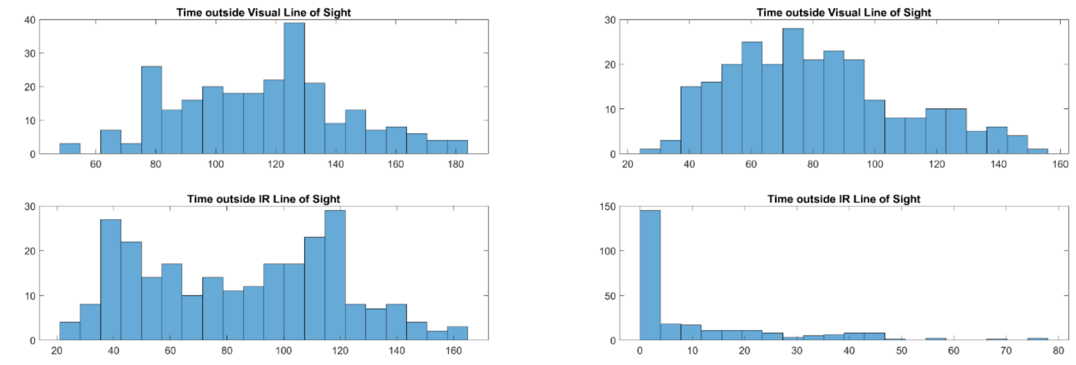


Figure 6: LoS metrics, test case 2. Left: Heritage. Right: SCvx.

Summary of Results

Metrics based on the Monte Carlo results are summarized in Table 3.

Table 3: Test result metrics for COMET-I.

Case	Time outside of Visual Camera FoV [s]		Time outside of IR Camera FoV [s]		LoS Error [°]	Control Effort [(Nm)^2]	Guidance Computation Time [s]		
	Mean	Max	Mean	Max	Mean	Mean	Mean	Min	Max
Heritage									
Test case 1	21.0	53.9	0.0	11.1	0.2	15.2	0.032	0.022	0.053
Test case 2	104.4	162.5	63.0	121.7	7.0	47.7	0.65	0.027	1.87
Test case 2*	114.8	157.4	46.5	106.4	3.1	58.4	0.52	0.38	1.73
SCvx									
Test case 1	26.6	49.9	0.0	0.0	0.2	7.3	2.65	1.41	9.93
Test case 2	85.1	143.9	9.8	78.8	1.1	37.4	1.88	0.50	20.82

*Results for the cases with early dust impact (<10 s, 29 cases in total) within design scope of heritage algorithm.

The results shows that the SCvx and heritage solutions are comparable for test case 1 but that the SCvx solution is significantly better for test case 2. The control effort for SCvx is significantly lower than for the heritage solution for both test cases. In particular for test case 2, the effort for the heritage solution is higher since it struggles more to achieve the objective but does not really succeed as well as the SCvx. The average guidance computation time is significantly lower for test case 1. For test case 2, the time is comparable to the time for the SCvx solution since in this case, the heritage guidance computation involves an iterative secant search.

6.2 THESEUS

In order to validate and verify the implemented guidance algorithms, a Monte Carlo campaign with 300 runs for test case 2 and 1000 runs for test case 3 was performed.

Test case 2

The left plot of Figure 7 shows the simulation slew time and the resulting significant gain in terms of slew time when using SCvx. The right plot shows the guidance angular rate obtained with the heritage and SCvx algorithm for a single run of the MC campaign. The left plot of Figure 8 shows the wheel torque profiles while the right plot shows the wheel angular momentum profile from simulation. The computational times for the heritage solution is between 0.39 and 3.6 ms with a mean value of 0.67 ms. For the SCvx solution the computational time is between 3.1 and 5.4 s with a mean value of 3.9 s. The statistics are based on a Monte Carlo campaign performed with 300 runs.

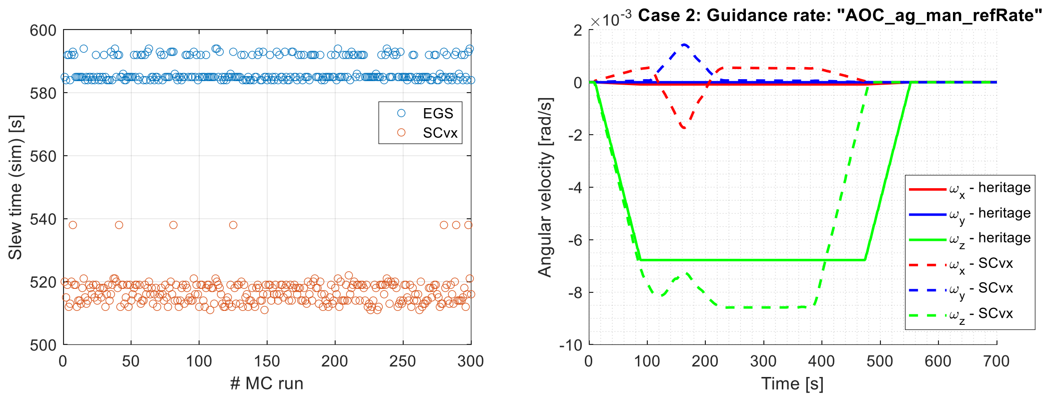


Figure 7: THESEUS test case 2. Left: Slew time. Right: Guidance angular rate.

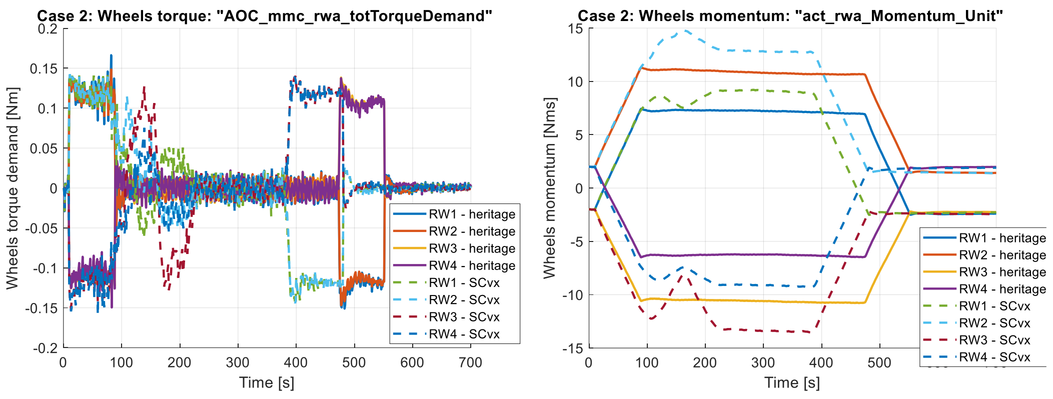


Figure 8: THESEUS test case 2. Left: Wheel torque. Right: Wheels angular momentum.

Test case 3

The left plot of Figure 9 shows the simulation slew time and the overall gain in terms of slew time when using SCvx. The right plot shows the guidance angular rate obtained with the heritage and SCvx algorithm for a single run of the MC campaign. The left plot of Figure 10 shows the wheel torque profiles and the right plot shows the wheels angular momentum profiles from simulation. The computational times for the heritage solution is between 0.39 and 5.9 ms with a mean value of 0.85 ms. For the SCvx solution the computational time is between 13.4 and 25 s with a mean value of 17.1 s. The statistics are based on a Monte Carlo campaign performed with 1000 runs.

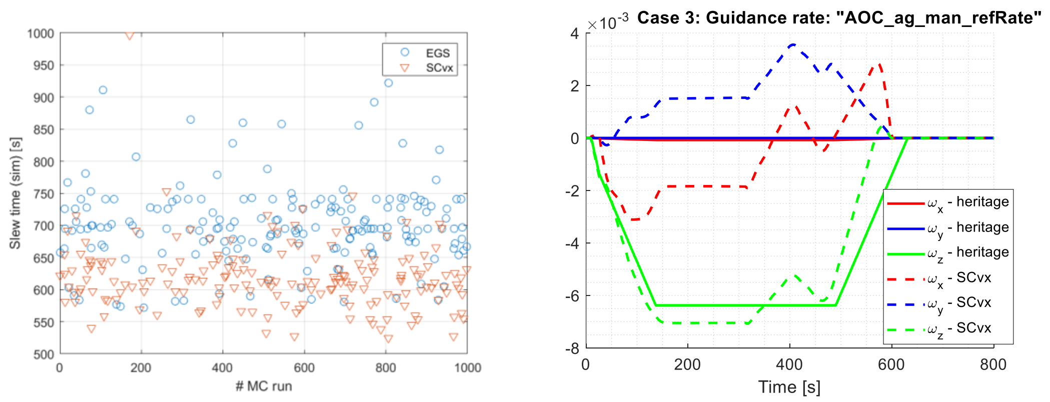


Figure 9: THESEUS test case 3. Left: Slew time. Right: Guidance angular rate.

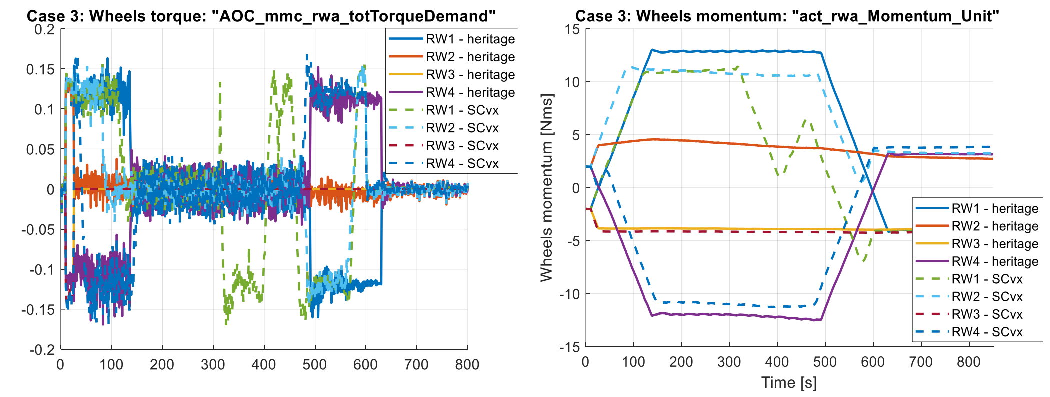


Figure 10: THESEUS test case 3. Left: Wheel torque. Right: Wheels angular momentum.

6.3 Implementation on Flight-Like Hardware

The target hardware is an ARM-Cortex-based development board (ZedBoard), which is supported by the MATLAB Embedded Coder for C code applications. As the SOCP solver is written in C++ we cannot rely on the native support, hence we use the following procedure to facilitate runtime tests: The Embedded Coder is used to generate code of the SCVX algorithm for the ARM-Cortex architecture, including the loading and reading process of the transcription data. Then the GCC cross-compiler builds a standalone executable which is uploaded to the ZedBoard.

The separation of the SCVX software into offline transcription and real-time SCVX loop enables the user to work on problem formulation level mostly unhindered by MATLAB Coder restrictions and benefit from the ease of use of the Matlab environment. However, it has some drawbacks regarding the implementation on embedded hardware: The structure of the transcription data which arises from general OCPs is that of a linked list with elements of nonuniform memory size. With pure static memory allocation MATLAB Coder does not support such a data structure and as a workaround, the data must be padded such that all list elements are of the same size. This currently leads to a significant memory overhead, which is subject to further improvements in the future.

The runtime is characterized with three different execution times. The first time t_{total} denotes the total amount of time required to execute the SCVX algorithm from scratch, including the execution times required for loading and reading the structure array, for instantiating the SCVX object and allocating memory, and for solving the discretized optimal control problem. The SCVX raw solution time is denoted by t_{SCVX} and excludes the previous pre-processing steps. Finally, the time required to solve all recurring SOCPs is denoted by t_{SOCP}. To reduce the influence of run time outliers, all the time metrics represent the median over ten runs. The runtime is measured for the nominal case of both benchmark problems. For comparison, Table 4 also includes the runtime measurements on a common personal computer (CPU Intel Core i9-10900K, max. 3.7 GHz) for solving the same optimization problems (see values in parentheses).

Table 4: ZedBoard computation times for the segments with six collocation points each.

	Theseus			Comet Interceptor		
	t_{total}	t_{SCVX}	t_{SOCP}	t_{total}	t_{SCVX}	t_{SOCP}
Absolute values [s]	34.1 (1.12)	33.48 (1.07)	30.7 (0.94)	42.48 (1.35)	41.44 (1.27)	35.47 (1.08)
Relative values [%]	100 (100)	98.18 (95.54)	90.03 (83.93)	100 (100)	97.55 (94.07)	83.5 (80)

For the Theseus problem, a discretization mesh consisting of ten uniformly distributed segments with five collocation points per segment satisfies the control and state constraints as discussed above. In total, the mesh comprises 50 collocations points. In the nominal case, the SCVX algorithm requires 15 iterations to satisfy the convergence criterion. Table 4 shows that the SCVX algorithm spends most of the computation time, approximately 90%, solving the underlying and recurring SOCPs. Loading the structure array and allocating memory requires less than 2% (5%) of the execution time.

For the Comet Interceptor problem, the discretization mesh comprises ten segments and six collocation points per segment. The SCVX algorithm requires 13 iterations to reach the convergence criterion. The formulation of the Comet Interceptor problem and the resulting conditioning of the SOCP matrices result in a higher computation time compared to Theseus. However, the relative values in the second row of Table 4 are comparable between the two benchmark problems.

The execution time on the ZedBoard is roughly 30 times higher than on the development PC, while the ratio of the CPU tact frequency from development PC to ZedBoard is only 5.5. The impact of the CPU architecture (e.g., caching, instruction set) is thus much higher than the pure CPU tact. If the ZedBoard is assumed as the flight hardware, trajectory computation times of 34 s for Theseus and 42 s for Comet-I might be acceptable to perform onboard trajectory computation before the start of a given maneuver, but the benefit from real-time trajectory re-computation mid-maneuver, as, for example, in the scenario for the dust particle impact compensation, is diminished. A future improvement of the runtime is expected from optimizing the memory use to reduce cache misses.

It can be seen from the runtime analysis that the majority of computation time is spent solving SOCP problems. This is in line with the expectation that the solution of the linear system occurring at the core of the interior point method should take the majority of the computation time. Although the total time exceeds expectation, the SOCP solver is the part of the software with the highest level of optimization and the margin for improvement of the core interior point algorithm and its implementation is deemed low.

More promising strategies for computation time reduction are to reduce memory use and achieve more efficient cache usage. Secondly trying to reduce the overall number of required SCVX iterations, such that a lower number of SOCPs must be solved. The main point of improvement for this is the trust-region strategy. While we have used a fixed trust-region size, we have seen that a dynamic update of the trust-region size can be beneficial in some situations. Another route of improvement is to try to improve the numerical conditioning of the problems, aiming at reducing the interior-point iterations that are required to solve the SOCP problems. Multiple OCP level problem scaling techniques were investigated and found of little to no benefit. A deeper analysis of the SOCP input matrices could however reveal a more successful scaling approach and improve the centring of the problem before applying the interior-point algorithm.

7 CONCLUSIONS

Several conclusions can be drawn from the work performed in this study with in particular the following main areas are of interest:

Performance: The performance observed from the SCvx based guidance is in most of the test cases better than the guidance resulting from the heritage solutions. In general, the SCvx solutions can better utilize the capacity of the RWA or RCS since constraints are taken into account on unit level.

Execution Time: The execution times observed from the tests on the flight like HW are in the range of 30 to 40 s which does not really allow for fast recomputation of the guidance profiles in connection with critical re-configuration of HW or in other cases, where the guidance profile is needed quickly. It is however expected that there is some room for improvement in terms of execution time.

Development Effort: An estimate of development effort shows that the application-specific required, recurrent effort is similar for the development of the optimization-based and the heritage solutions. However, the optimization-based solution also requires a significant initial, non-recurring effort to develop the necessary numeric optimization software. This is estimated to be about 5 times as much as the effort for a single mission application, not counting the development of the core convex solver.

The observations and conclusions summarized above indicate that choosing an SCvx-based attitude guidance solution is not a “magic” universal tool that seamlessly will solve any problem. Significant effort is needed to be able to arrive at a well-posed and well-tuned problem that allows the optimization-based framework to provide a solution. However, with such a problem at hand, the framework is able to provide a versatile solution that seems to be able to better utilize the on-board resources and deliver a solution that provides better performance than the heritage approach.

8 ACKNOWLEDGEMENTS

This work was supported through funding from the European Space Agency's Technology Development Element (TDE) programme (ESA Contract No. 4000133640/21/NL/CRS “Attitude guidance using on-board optimisation”).

9 REFERENCES

[1] X. Liu, P. Lu, and B. Pan, “Survey of Convex Optimization for Aerospace Applications,” *Astrodynamics*, vol. 1, no. 1, pp. 23–40, Sep. 2017, doi: 10.1007/s42064-017-0003-8.

[2] G. Jones and C. Snodgrass, “Comet Interceptor: A proposed ESA Mission to a Dynamically New Comet.,” in *Geophysical Research Abstracts*, 2019.

[3] L. Amati *et al.*, “The THESEUS Space Mission Concept: Science Case, Design and Expected Performances,” *Advances in Space Research*, vol. 62, no. 1, pp. 191–244, Jul. 2018, doi: 10.1016/j.asr.2018.03.010.

[4] V. Preda, A. Hyslop, and S. Bennani, “Optimal Science-Time Reorientation Policy for the Comet Interceptor Flyby via Sequential Convex Programming,” *CEAS Space Journal*, vol. 14, no. 1, pp. 173–186, Jun. 2021, doi: 10.1007/s12567-021-00368-2.

[5] B. Acikmese 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, Sep. 2007, doi: 10.2514/1.27553.

[6] L. Blackmore, B. Acikmese, and D. P. Scharf, “Minimum-Landing-Error Powered-Descent Guidance for Mars Landing Using Convex Optimization,” *Journal of Guidance, Control, and Dynamics*, vol. 33, no. 4, pp. 1161–1171, Jul. 2010, doi: 10.2514/1.47202.

[7] Marco Sagliano, “Pseudospectral Convex Optimization for Powered Descent and Landing,” *Journal of Guidance, Control and Dynamics*, vol. 41, no. 2, 2018, doi: 10.2514/1.G002818.

[8] L. Blackmore, “Autonomous Precision Landing of Space Rockets,” *The Bridge*, vol. 4, no. 46, 2019, [Online]. Available: https://www.nae.edu/164334/Autonomous-Precision-Landing-of-Space-Rockets

[9] A. Wenzel, M. Sagliano, and D. Seelbinder, “Performance Analysis of Real-Time Optimal Guidance Methods for Vertical Take-off, Vertical Landing Vehicles,” in *69th International Astronautical Congress*, in Proceedings of the International Astronautical Congress, IAC. 2018.

[10] M. Dumke, G. F. Trigo, M. Sagliano, P. Saranrittichai, and S. Theil, “Design, Development, and Flight Testing of the Vertical Take-off and Landing GNC Testbed EAGLE,” *CEAS Space Journal*, vol. 12, no. 1, pp. 97–113, Aug. 2019, doi: 10.1007/s12567-019-00269-5.

[11] A. Domahidi, E. Chu, and S. Boyd, “ECOS: An SOCP Solver for Embedded Systems,” in *European Control Conference*, 2013. doi: 10.23919/ECC.2013.6669541.

[12] D. Dueri, B. Acikmese, D. P. Scharf, and M. W. Harris, “Customized Real-Time Interior-Point Methods for Onboard Powered-Descent Guidance,” *Journal of Guidance, Control, and Dynamics*, pp. 1–16, Aug. 2016, doi: 10.2514/1.G001480.

[13] X. Liu, Z. Shen, and P. Lu, “Entry Trajectory Optimization by Second-Order Cone Programming,” *Journal of Guidance, Control, and Dynamics*, vol. 39, no. 2, pp. 227–241, 2015, doi: 10.2514/1.G001210.

[14] Y. Mao, M. Szmuk, X. Xu, and B. Acikmese, “Successive Convexification: A Superlinearly Convergent Algorithm for Non-Convex Optimal Control Problems,” 2018, doi: 10.48550/ARXIV.1804.06539.

[15] Y. Mao, D. Dueri, M. Szmuk, and B. Açıkmeşe, “Successive Convexification of Non-Convex Optimal Control Problems with State Constraints,” *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 4063–4069, Jul. 2017, doi: 10.1016/j.ifacol.2017.08.789.

[16] R. Bonalli, A. Cauligi, A. Bylard, and M. Pavone, “GuSTO: Guaranteed Sequential Trajectory Optimization via Sequential Convex Programming,” Mar. 2019.

[17] Ch. Zillober, K. Schittkowski, and K. Moritzen, “Very Large Scale Optimization by Sequential Convex Programming,” *Optimization Methods and Software*, vol. 19, no. 1, pp. 103–120, Feb. 2004, doi: 10.1080/10556780410001647195.

[18] D. Malyuta, T. Reynolds, M. Szmuk, M. Mesbahi, B. Acikmese, and J. M. Carson, “Discretization Performance and Accuracy Analysis for the Rocket Powered Descent Guidance Problem,” in *AIAA Scitech 2019 Forum*, American Institute of Aeronautics and Astronautics, Jan. 2019. doi: 10.2514/6.2019-0925.

[19] M. Sagliano, “Generalized Hp Pseudospectral-Convex Programming for Powered Descent and Landing,” *Journal of Guidance, Control, and Dynamics*, vol. 42, no. 7, pp. 1562–1570, Jul. 2019, doi: 10.2514/1.g003731.