A Tutorial on Real-time Convex Optimization Based Guidance and Control for Aerospace Applications

Conference Paper · June 2018

DOI: 10.23919/ACC.2018.8430984

3 authors:

Yuanqi Mao
University of Washington Seattle
7 PUBLICATIONS **54** CITATIONS
SEE PROFILE

Michael Szmuk
University of Washington Seattle
22 PUBLICATIONS **121** CITATIONS
SEE PROFILE

Behçet Açıkmeşe
University of Washington Seattle
166 PUBLICATIONS **1,282** CITATIONS
SEE PROFILE

A Tutorial on Real-time Convex Optimization Based Guidance and Control for Aerospace Applications*

Yuanqi Mao[1], Michael Szmuk[1] and Behçet Açıkmeşe[1]

***Abstract*— Challenging control problems are ubiquitous in aerospace engineering applications. Such applications include reusable rockets, spacecraft rendezvous and docking, satellite constellation management for terrestrial imaging, and many other ones where vehicles are required to perform reliably while adhering to physical and mission constraints. Physical constraints are due to limitations like finite fuel, whereas mission constraints can arise from sensor pointing requirements or safety keep-out zones. Often, mission success necessitates that these constraints be satisfied in real-time, using limited on-board computational resources. In this paper, we present a tutorial on how to formulate some aerospace control problem examples in an optimization based control framework. Specifically, we decompose the control problem into two levels: guidance (trajectory optimization), and feedback control. In guidance, we use recent convexification results to formulate non-convex trajectory optimization problems as finite-dimensional convex optimization problems, which we solve using fast and reliable Interior Point Method (IPM) algorithms. We discuss the current state of the art of convexification techniques, and outline the context in which these techniques should be used. Lastly, we present an overview of synthesis methods for feedback control laws. These control laws are used to track the guidance trajectories within specified error bounds, and in the presence of model uncertainties and environmental disturbances.**

I. INTRODUCTION

With the recent release of *NASA Space Technology Roadmaps and Priorities*, next-generation Guidance, Navigation, and Control (GN&C) technologies will be of paramount importance for future planetary explorations [1]. Spacecraft GN&C technologies involves complex sciences and have been evolving since the launch of the first rocket. Simply put, *guidance* is the onboard generation of desired trajectories. *Navigation* is to determine the current states (position, velocity, and attitude) of the vehicles. *Control* is to track guidance commands while maintaining stability, by the manipulation of vehicle steering. The main focus of this tutorial is on guidance technologies, and we will also briefly touch upon feedback control synthesis.

Guidance and control technologies have progressed throughout aerospace history. To meet the performance standards of future applications, recent years have seen the advent of new technologies to enable and enhance a wide range of capabilities.

For example, autonomous precision rocket landing is one such capability that is key to enabling the reuse of rockets [2]. Convexification based technologies have been developed for the 3-DoF cases [3], [4], while 6-DoF quaternion based formulations with attitude control have also been addressed in recent years [5], [6].

Autonomous rendezvous and docking is another key enabling technology that presents many challenges. Guidance and control technologies have been developed, for instance, via conic optimization [7], and via Model Predictive Control (MPC) [8]. Furthermore, the complexity and safety of such operations can be enhanced by considering the attitude control problem. Using a quaternion based formulation, non-convex quadratic constraints can be convexified, and the problem can be solved via semidefinite programming [9]. Other techniques may also be applicable, such as the special MPC design proposed in [10].

As a final example, consider quadrotors, systems that are becoming increasingly ubiquitous in today's world. Obstacle avoidance is one key challenge faced by these systems [11]. To address this issue, a sampling based real-time guidance framework was proposed recently in [12], within which simplified quadrotor dynamics were used. For robust real-time embedded applications, a convexification based real-time guidance for quadrotors was proposed in [13], and included non-convex obstacle avoidance constraints.

In this tutorial we will focus on convex optimization [14] based guidance technologies. Due to recent advances, the solutions to a large class of convex optimization problems can be computed in polynomial time [15] using either generic Second-Order Cone Programming (SOCP) solvers [16], or customized solvers that take advantage of specific problem structures [17], [18]. Moreover, if a feasible solution exists, these algorithms are guaranteed to find the global optimal solution in a bounded number of iterations. Conversely, they provide a certificate of infeasibility if no feasible solution exists.

However, most real-world guidance problems are inherently non-convex. The performance and guarantees of convex optimization algorithms are the primary reasons researchers are motivated to formulate non-convex guidance problems in a convex framework. This practice is referred to as *convexification*. In addition to the applications we mentioned above, convexification based approaches have also been applied to swarm formation flying [19], and planetary entry [20]. While the results of these numerical experiments look promising, few theoretical proofs are provided. Inspired by the powered descent guidance for planetary landing problem [3], recent results have introduced a procedure known as *lossless convexification* to handle a large class of non-

*This work was supported partly by Office of Naval Research grant N00014-16-1-3144, National Science Foundation grant NSF-CMMI-1613235, and Air Force Research Laboratory grant FA9453-15-1-0303.

[1]University of Washington, William E. Boeing Department of Aeronautics and Astronautics, Seattle, WA 98105, USA yqmao@uw.edu, mszmuk@uw.edu, behcet@uw.edu

convex control constraints [21]. It proves that these non-convex optimal control problems can be posed as equivalent convex optimization problems without sacrificing feasibility or optimality. More recently, an iterative procedure known as *successive convexification* [22], [23] solves problems with nonlinear dynamics and non-convex state constraints, and provides proofs of global convergence. These two convexification techniques are rigorous attempts to solve complex non-convex optimal control problems in real-time, and they comprise the main topic of this tutorial.

II. PROBLEM STATEMENT

The real-time autonomous guidance problem can be posed as a finite horizon optimal control problem with dynamics describing the motion of the spacecraft as well as constraints on the vehicle state and controls [1]. This can be expressed generically as follows.

Problem 1 (General Guidance Problem)**.**

$$\min_{u,t_f} \quad J(x,u,t) = \varphi(x(t_f),t_f) + \int_{t_0}^{t_f} L(x(t),u(t),t)\,\mathrm{d}t \quad \text{(1a)}$$

$$\text{subject to} \quad \dot{x}(t) = f(x(t),u(t),t), \quad \text{(1b)}$$

$$u(t) \in \mathscr{U}, \quad \text{(1c)}$$

$$x(t) \in \mathscr{X}, \ \text{ for all } t_0 \le t \le t_f, \quad \text{(1d)}$$

where t_0 and t_f denote the initial and (finite) final time, respectively, $x(t) \in \mathbb{R}^n$ represents the vehicle state, and $u(t) \in \mathbb{R}^m$ is the control input. In (1a), $\varphi : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$ is the terminal cost, $L : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R} \to \mathbb{R}$ is the running cost, and both can be assumed convex, as is the case for most guidance applications. In (1b), $f : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R} \to \mathbb{R}^n$ is the system dynamics defined by a control-state mapping, which is nonlinear in general, and assumed to be continuously differentiable. In (1c) and (1d), $\mathscr{U} \subseteq \mathbb{R}^m$ and $\mathscr{X} \subseteq \mathbb{R}^n$ are set-valued maps defining control and state constraints, and they are in general non-empty non-convex sets. The reader is referred to [22] for a more rigorous treatment of Problem 1 in Banach space.

Due to the existence of system dynamics and state and control constraints, Maximum Principle based methods fall short, and the problem must be solved numerically via optimization algorithms after a proper discretization [24]. To meet the guidance challenges of next-generation space missions, onboard algorithms will need to meet the following specifications [1]:

- **Optimality**: Given that feasible solutions exist, an optimal solution that minimizes (at least locally) the cost function J is desired.
- **Verifiability**: There must be design metrics that accurately describe the performance, convergence, and robustness of GN&C algorithms, with accompanying methods for verifying these metrics.
- **Real-time implementability**: Algorithms must be implemented and executed on real-time processors in a reasonable amount of time.

Among many numerical methods, convex optimization studies the problem of minimizing convex functions over convex sets. As we mentioned in Section I, convex optimization lends itself well to achieve the above goals, thanks to its polynomial-time solvability and global optimality of its solutions. Using simplified models can sometimes make the problem convex. For example, if we only consider linear system dynamics, upper-bounded control input, and a conic approaching corridor as shown in Figure 1, the spacecraft rendezvous problem becomes convex. However for many

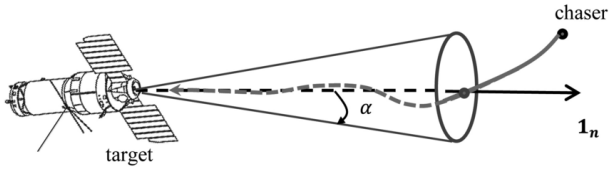


Fig. 1: Conic approach corridor for rendezvous with the docking axis $\mathbf{1}_n$ and apex half-angle α [7].

aerospace applications, it is critical to use high fidelity models to fully capture the inherent non-convexity. To solve the resulting non-convex problem (as in the form of Problem 1), we need to convexify it upfront and then let convex programming solver do the heavy lifting. In this tutorial, we will cover some of these convexification technologies, namely *lossless convexification* and *successive convexification*. Each method is able to handle specific sources of non-convexity in Problem 1. The methods can be applied independently, or simultaneously with precautions.

III. LOSSLESS CONVEXIFICATION

The first technology we consider is called *lossless convexification*. It is primarily used to handle a class of non-convex control constraints, which applies to a broad range of aerospace vehicles. The method introduces a relaxation to the non-convex control constraints through the use of a slack variable. More importantly, it guarantees that if the original problem is feasible, then the optimal solution of the relaxed (convex) problem is also optimal for the original (non-convex) one.

Lossless convexification was introduced in [3] for a minimum fuel planetary landing problem, and a more general result was obtained in [21] for a large class of optimal control problems with convex cost, linear dynamics, convex state constraints, and a specific class of non-convex control constraints, as follows.

Problem 2 (Guidance Problem with a Class of Non-convex Control Constraints)**.**

$$\min_{u} \quad J(x,u) = h(t_f,x(t_f)) + k\int_{t_0}^{t_f} g_0(u(t))\,\mathrm{d}t \quad \text{(2a)}$$

$$\text{subject to} \quad \dot{x}(t) = A(t)x(t) + B(t)u(t) + E(t)w(t), \quad \text{(2b)}$$

$$u(t) \in \mathscr{U}, \quad \text{(2c)}$$

$$x(t) \in \mathscr{X}, \ \text{ for all } t_0 \le t \le t_f, \quad \text{(2d)}$$

where both h and g_0 are again assumed convex. $k \ge 0$ is a weighting scalar. $A : \mathbb{R}_+ \to \mathbb{R}^{n\times n}$, $B : \mathbb{R}_+ \to \mathbb{R}^{n\times m}$, and $E : \mathbb{R}_+ \to \mathbb{R}^{n\times p}$ are piecewise analytic functions of time t. $w(t) \in$

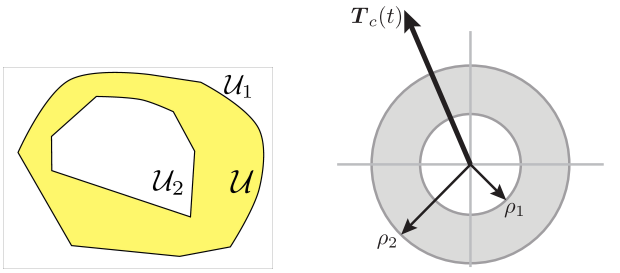
$\mathbb{R}^p$ is a known exogenous input. $\mathscr{X} \subseteq \mathbb{R}^n$ is the set of feasible states, which is assumed convex in this case. $\mathscr{U} \subseteq \mathbb{R}^m$ is the set of feasible control inputs (shown in Figure 2a), and it represents a specific class of non-convex sets that satisfy

$$\mathscr{U} = \mathscr{U}_1 \setminus \mathscr{U}_2, \quad \mathscr{U}_2 = \bigcap_{i=1}^{q} \mathscr{U}_{2,i} \subset \mathscr{U}_1, \tag{3}$$

where $\mathscr{U}_1$ and $\mathscr{U}_2$ are, respectively, compact convex and open convex sets with

$$\mathscr{U}_{2,i} = \{u \in \mathbb{R}^m : \ g_i(u) < 1\}, \quad i = 1,...,q,$$

where $g_i, i = 1,...,q$, are convex functions that are bounded on $\mathscr{U}_1$, that is, there exists some $\bar{g} \in \mathbb{R}$ such that $g_i(u) \leq \bar{g}, \forall u \in \mathscr{U}_1, \ i = 1,...,q$. As an example, Figure 2b gives the thrust bound constraints, $\rho_1 \leq \|T_c(t)\| \leq \rho_2$, which is quite broadly applicable to rockets or other space vehicles.



(a) Set $\mathscr{U}$ in general, shaded

(b) Example of set $\mathscr{U}$, thrust bound constraints

Fig. 2: Non-convex set of feasible control inputs [21].

Apparently the main difficulty of Problem 2 is the non-convexity of $\mathscr{U}$, and lossless convexification was introduced to address exactly that. The intuition behind lossless convexification is best captured by geometric insight. Using planetary soft landing as an example, we have $\mathscr{U} = \{u \in \mathbb{R}^2 : 1 \leq \|u\| \leq \rho\}$, which is a two dimensional non-convex annulus. However, we can lift it to a convex cone by introducing a third dimension σ, and extending the annulus in this direction (see Figure 3). We denote the resulting set as $\mathscr{V}$, and in the planetary landing case $\mathscr{V} = \{(u,\sigma) \in \mathbb{R}^3 : \sigma \geq 1, \ \|u\| \leq \min(\rho,\sigma)\}$. Clearly, $\mathscr{V}$ is in a higher-dimensional space and $\mathscr{U} \subset \mathscr{V}$, which means we are solving a relaxed problem by doing this "lifting". Note that the set $\mathscr{V}$ also contains control inputs that are not feasible for Problem 2. We will see that by carefully designing the convex relaxation (as in Problem 3), the equivalence of solutions between these two sets can in fact be established.

Problem 3 (Convex Equivalent of Problem 2)**.**

$$\min_{u,\sigma} \quad J(x,u) = h(t_f, x(t_f)) + k\,\xi(t_f) \tag{4a}$$

$$\text{subject to} \quad \dot{x}(t) = A(t)x(t) + B(t)u(t) + E(t)w(t), \tag{4b}$$

$$\dot{\xi}(t) = \sigma(t), \tag{4c}$$

$$(u(t), \sigma(t)) \in \mathscr{V}, \tag{4d}$$

$$x(t) \in \mathscr{X}, \ \text{ for all } t_0 \leq t \leq t_f, \tag{4e}$$

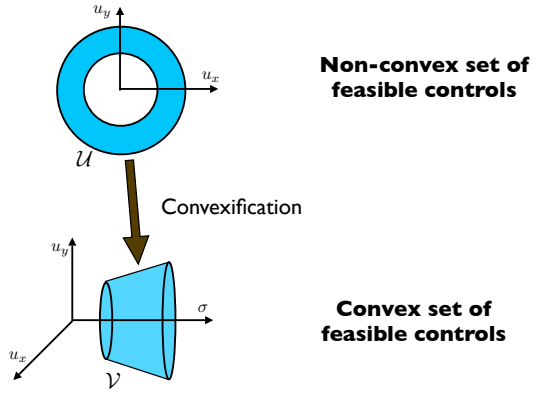


Fig. 3: Convexification of the Control Magnitude Constraint for Planetary Soft Landing. The annulus represents the actual non-convex control constraints $\mathscr{U}$ in (u_x, u_y) space, which is lifted to a convex cone $\mathscr{V}$ in (u_x, u_y, σ) space [21].

where $\sigma(t) \in \mathbb{R}$ is a slack control variable, and $\xi(t) \in \mathbb{R}$ is the corresponding state variable. $\mathscr{V} \subseteq \mathbb{R}^{m+1}$ is the set of relaxed feasible control inputs,

$$\mathscr{V} = \left\{(u,\sigma) \in \mathbb{R}^{m+1} : \sigma \geq 1 \ \text{ and } \ u \in \mathscr{U}_1 \cap \mathscr{V}_2(\sigma)\right\} \tag{5}$$

with $\mathscr{V}_2(\sigma) = \bigcap_{i=i_o}^{q} \mathscr{V}_{2,i}(\sigma)$ where

$$\mathscr{V}_{2,i}(s) := \{u \in \mathbb{R}^m : g_i(u) \leq s\}, \ i_o = \begin{cases} 0 & \text{for} \quad k > 0 \\ 1 & \text{for} \quad k = 0 \end{cases}.$$

This $\mathscr{V}$ set is a generalization of the convex cone in Figure 3, and it can easily be shown to be convex, too. Therefore, Problem 3 is a convex optimal control problem, which can be solved to global optimality in polynomial time [15]. More importantly, under a few other minor assumptions, the following equivalence holds.

Theorem 1 (Equivalence of Solutions between Problem 2 and Problem 3 [21])**.** *If the linear system in* (2b) *is controllable, then the optimal solution of Problem 3 is also an optimal solution of Problem 2.*

This result enables us to solve a much harder non-convex problem by solving its convex relaxation in a "lossless" fashion. Thus it truly opens up the gate to convex optimization based guidance for a variety of aerospace systems that have similar non-convex control constraints that can be characterized by the set $\mathscr{U}$ as in (3) and Figure 2a.

IV. SUCCESSIVE CONVEXIFICATION

Successive convexification is an iterative framework by which a non-convex problem is solved via a sequence of convex subproblems. In particular, the `SCvx` algorithm is a recent development based on the successive convexification framework. It is an iterative algorithm designed to tackle the non-convexity resulting from nonlinear dynamics [22], and non-convex state constraints [23]. It falls in a class of control technologies known as *algorithmic control* [25].

The `SCvx` algorithm solves a sequence of convex subproblems, specifically SOCP subproblems. While similar ideas

in both finite dimensional optimization problems [26], [27] and optimal control problems [28], [29] have long been tried, few convergence results were reported. The SCvx algorithm, on the other hand, presents a systematic procedure. More importantly, through mathematical analysis, we guarantee that the SCvx algorithm will converge globally, and the solution it converges to will recover local optimality for the original problem. Among similar Sequential Convex Programming (SCP) based work (see e.g. [30], [31]), the convergence result accompanying the SCvx algorithm marks one of the first rigorous attempts on this subject.

The SCvx algorithm proposed in [22] focuses on nonlinear dynamics, and thus it directly aims to solve Problem 1, with $\mathscr{U}$ and $\mathscr{X}$ assumed to be convex or at least already convexified (by e.g. lossless convexification). One way to convexify the nonlinear system dynamics is through successive linearization by using their first order Taylor approximations. Assume the $(i-1)^{th}$ succession gives us a solution $(x^{i-1}(t), u^{i-1}(t))$. Let $A(t), B(t), C(t)$ be the partial derivative of $f(x^{i-1}(t), u^{i-1}(t), t)$ with respect to x, u and t respectively, $d(t) = x(t) - x^{i-1}(t)$, and $w(t) = u(t) - u^{i-1}(t)$, then the first order Taylor expansion will be

$$\dot{d}(t) = A(t)d(t) + B(t)w(t) + C(t) + H.O.T.. \tag{6}$$

This is a linear system with respect to $d(t)$ and $w(t)$, which are our new states and control respectively in the convex subproblems. This procedure is repeated until guaranteed convergence [22]. The linearization gets us the benefit of convexity, but it also introduces some new issues.

Firstly, the linearization procedure can sometimes generate infeasible problems, even if the original nonlinear problem itself is feasible. This issue is called the *artificial infeasibility*. In such scenarios, the undesirable infeasibility obstructs the iteration process and prevents convergence. The most evident example of this arises when the problem is linearized about an unrealistically short time horizon, so that there is no feasible control input that can satisfy the prescribed dynamics and constraints. To prevent this, we may introduce an additional control input $v(t)$, called *virtual control*, to the linear dynamics (6) (omitting the higher order terms):

$$\dot{d}(t) = A(t)d(t) + B(t)w(t) + E(t)v(t) + C(t), \tag{7}$$

where $E(t)$ can be chosen based on $A(t)$ such that the pair $A(\cdot), E(\cdot)$ is controllable. Then, since $v(t)$ is unconstrained, any state in the feasible region becomes reachable in finite time. Since we only resort to the *virtual control* when necessary, it will be penalized via an additional term $\lambda\gamma(Ev)$ in the cost, where λ is the penalty weight, and $\gamma(\cdot)$ is an exact penalty function, usually chosen to be the 1-norm $\|\cdot(t)\|_1$. Now the penalized cost after linearization of system dynamics will be defined as

$$L(d,w) \coloneqq J(x,u) + \lambda\gamma(Ev), \tag{8}$$

while the penalized cost before linearizing can be formulated in a similar way:

$$P(x,u) \coloneqq J(x,u) + \lambda\gamma(\dot{x} - f), \tag{9}$$

where in kth succession $\dot{x}$ is computed by $f(x^k, u^k) + d^k$, while f means $f(x^{k+1}, u^{k+1})$.

Another concern in SCvx algorithm is that the linear approximation becomes inaccurate when large deviation occurs. To mitigate this risk, we ensure that the linearized trajectory does not deviate significantly from the nominal one obtained in the previous succession via a *trust region* on the new control input, $\|w(t)\|_2 \leq \Delta$, and thus the new state will be restricted as well given the system dynamics. The rationale is that we only trust the linear approximation inside the trust region.

Now we are ready to state the final formulation of the convex subproblem as follows.

Problem 4 (Convex Subproblem in SCvx)**.**

$$\min_{w,v} \quad L(d,w) = J(x,u) + \lambda\gamma(Ev) \tag{10a}$$

$$\text{s.t.} \quad \dot{d}(t) = A(t)d(t) + B(t)w(t) + E(t)v(t) + C(t), \tag{10b}$$

$$\|w(t)\|_2 \leq \Delta, \tag{10c}$$

$$u(t) + w(t) \in \mathscr{U}, \tag{10d}$$

$$x(t) + d(t) \in \mathscr{X}, \ \text{ for all } t_0 \leq t \leq t_f. \tag{10e}$$

The SCvx algorithm (see Algorithm 1) solves this convex subproblem in each succession, and uses a typical strategy to update the trust region radius.

In Step 2, the ratio ρ^k is used as a metric for the quality of linear approximations. A desirable scenario is when ΔP^k agrees with ΔL^k, i.e. ρ^k is close to 1. Hence if ρ^k is above ρ_2, which means our linear approximation predicts the cost reduction well, then we may choose to enlarge the trust region in Step 3, i.e., we put more faith in our approximation. Otherwise, we may keep the trust region unchanged, or contract its radius if needed. The most unwanted situation is when ρ^k is negative, or close to zero. The current step will be rejected in this case, and one has to contract the trust region and re-optimize at (x^k, u^k).

One important distinction of SCvx algorithm is the fact that its subproblems are solved to full optimality at each succession, thanks to fast enough customized solvers [18]. Conventional trust region algorithms perform a line search along the Cauchy arc to achieve a "sufficient" cost reduction [32]. As a result, it takes significantly less successions for SCvx to converge, by achieving more cost reduction at each succession.

We would also like to point out the differences between the SCvx algorithm and the Sequential Quadratic Programming (SQP) [33] type of methods. SQP based methods use the second order information in the sense that they need to approximate the Hessian matrix of the cost function (and sometimes constraints as well). This practice requires techniques such as Broyden–Fletcher–Goldfarb–Shanno (BFGS) update, which could be computationally expensive and thus not suitable for real-time applications. Even with BFGS update, one still need some other condition to ensure the convexity being preserved [33]. The SCvx algorithm, one the other hand, only uses the first order information, so that 1) the subproblems are guaranteed to be convex; 2) it

Algorithm 1 Successive Convexification (`SCvx`)

Input Select initial state x^0 and control u^0 s.t. $x^0 \in \mathscr{X}$ and $u^0 \in \mathscr{U}$. Initialize trust region radius with positive Δ^0 and lower bound Δ_l. Select positive penalty weight λ, and parameters $0 < \rho_0 < \rho_1 < \rho_2 < 1$ and $\alpha > 1$.

Step 1 At each succession k, solve Problem 4 at (x^k, u^k, Δ^k) to get an optimal solution (d^k, w^k).

Step 2 Compute the *actual* change of $P(x,u)$ in (9):

$$\Delta P(x^k, u^k) := \Delta P^k = P(x^k, u^k) - P(x^k + d^k, u^k + w^k),$$

and the *predicted* change by linear approximation in (8):

$$\Delta L(d^k, w^k) := \Delta L^k = P(x^k, u^k) - L(d^k, w^k).$$

if $\Delta L^k = 0$ **then**
  Stop, and **return** (x^k, u^k);
else
  Compute the ratio $\rho^k = \Delta J^k / \Delta L^k$.
end if

Step 3

if $\rho^k < \rho_0$ **then**
  Reject this step, contract the trust region radius, i.e., $\Delta_k \leftarrow \Delta_k / \alpha$, and go back to **Step 1**;
else
  Accept this step, i.e. $x^{k+1} \leftarrow x^k + d^k$, $u^{k+1} \leftarrow u^k + w^k$, and update Δ^{k+1} by

$$\Delta^{k+1} = \begin{cases} \Delta^k / \alpha, & \text{if } \rho^k < \rho_1; \\ \Delta^k, & \text{if } \rho_1 \le \rho^k < \rho_2; \\ \alpha \Delta^k, & \text{if } \rho_2 \le \rho^k. \end{cases}$$

end if
$\Delta^{k+1} \leftarrow \max\{\Delta^{k+1}, \Delta_l\}$, $k \leftarrow k+1$, and go to **Step 1**.

requires significantly less computational effort per iteration. The approximation may be less accurate, but that error is properly regulated by the trust region updating mechanism as mentioned before.

The global convergence of the `SCvx` algorithm (Algorithm 1) was shown in [22], and here we state the final result without proofs.

Theorem 2. *If Algorithm 1 generates an infinite sequence $\{x^k\}$, then $\{x^k\}$ has limit points. Furthermore, if any limit point $\bar{x}$ is feasible for Problem 1, then it is also a local optimum of Problem 1.*

As for non-convex state constraints, [23] proposed an even faster version of the `SCvx` algorithm, called `SCvx-fast`, whose convergence does not rely on the trust region updating mechanism as in Algorithm 1. As a result, more aggressive steps can be taken, and numerical evidence suggests that convergence indeed occurs in fewer iterations. The `SCvx-fast` algorithm can handle state constraints that resemble a union of convex keep-out zones (Figure 4). It has some new features such as a *Project-and-Linearize* step, and its global convergence is proved as well. The reader is referred to [23] for more details.

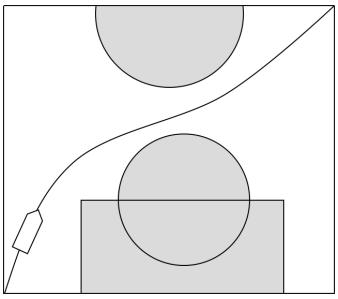


Fig. 4: Convex shaped keep-out zones as state constraints for the `SCvx-fast` algorithm.

6-DoF Planetary Landing Example

Here, we provide an example demonstrating how *successive convexification* can be used to generate a 6-DoF rocket landing trajectory for an in-plane scenario [6].

Figure 5 shows the convergence history of the trajectory. The solution process is initialized using a straight line from the initial position to the final position. The rest of the state vector time-history (i.e. consisting of the attitude quaternion, body-rates, velocities, and mass) is initialized in a similarly manner, thus producing an initial trajectory that is simple but in general **NOT** dynamically feasible. The dots represent the discretization points used to solve the problem numerically. The solid lines represent the trajectory produced by propagating the nonlinear 6-DoF dynamics (external to the optimization problem) using the controls generated by the current subproblem.

One can observe from Figure 5 that as the iterations progress, the *virtual control* goes to zero, and the solution refines its linearization until the propagated solution matches the results of the numerical optimization. In practice, the process typically converges in under ten iterations.

V. REAL-TIME COMPUTATION

In this section, we will briefly discuss methods that solve the resulting convex optimal control problems in real-time and onboard embedded systems. The first step is to discretize the continuous time systems [24]. Inter-sample feasibility assurance is provided in [34]. The discretized optimal control problem is then converted into canonical form: $x^* = \arg\min\{c^T x \,|\, Ax = b, x \in \mathscr{K}\}$, where $\mathscr{K} \subset \mathbb{R}^n$ is formed by the Cartesian product of convex cones. Once in canonical form, Interior Point Method (IPM) solvers (e.g. SDPT3, SeDuMi, ECOS [16], Gurobi, CPLEX) can be used to solve the convex optimization problem to global optimality. In the case of successive convexification, this process is repeated for each convex subproblem at each iteration.

While computationally efficient, these generic solvers may not always perform as needed in real-time. Moreover, many applications require solving *similar* optimization problems repeatedly, while varying only some problem parameters

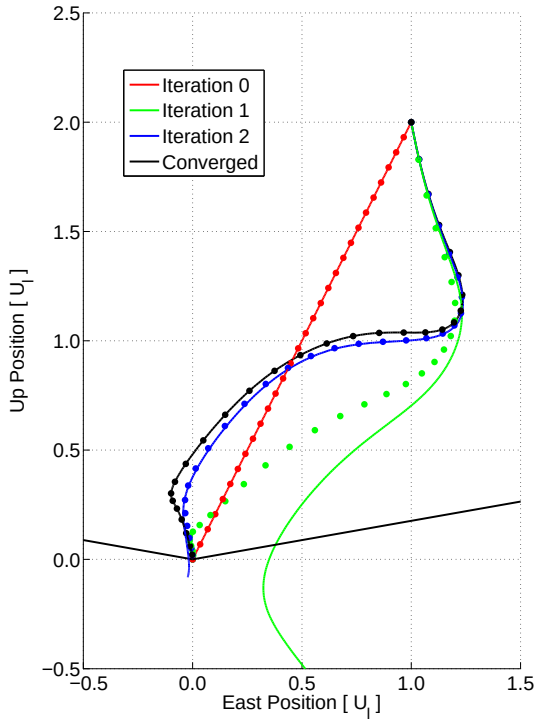


Fig. 5: Convergence history of successive convexification process for a 6-DoF rocket landing problem.

such as initial conditions. Such applications can benefit from an increase in speed and software verifiability through the use of customized solvers [17], [18].

We begin by noting that optimal control problems naturally translate into sparse optimization problems. That is, the A matrix is typically over 90% sparse (i.e., only 10% of the elements are non-zero). In order to classify problems according to their sparsity structure, we formally define a problem class [18].

Definition 1. *Given $A_0 \in \mathbb{R}^{p\times n}$, $b_0 \in \mathbb{R}^p$, $c_0 \in \mathbb{R}^n$, and $\mathscr{K}_0 \subset \mathbb{R}^n$, a problem class, $\mathscr{P}$, is defined as:*

$$\mathscr{P} = \{A \in \mathbb{R}^{p\times n}, b \in \mathbb{R}^p, c \in \mathbb{R}^n, \mathscr{K} \subset \mathbb{R}^n : \\ \text{str}(A) \leq \text{str}(A_0), \text{str}(b) \leq \text{str}(b_0), \\ \text{str}(c) \leq \text{str}(c_0), \mathscr{K} = \mathscr{K}_0\},$$

where the $\leq$ operator denotes element-wise inequality, and str() maps any non-zero element to a 1 and leaves 0 elements undisturbed, thereby forming the sparsity structure of its input. Thus, $(A,b,c,\mathscr{K}) \in \mathscr{P}$ if any zero element in $(A_0, b_0.c_0, \mathscr{K}_0)$ is also a zero element in $(A,b,c,\mathscr{K})$. Consequently, a problem class can be interpreted as an upper bound on the sparsity structure of (A,b,c).

Embedded systems typically operate in environments with stringent time requirements and limited memory. With this in mind, suppose that a given embedded system solves a set of problems $\mathscr{P}_0 \subseteq \mathscr{P}$. Then, the problem class and an upper bound on its size is known at compile time. Thus, the exact amount of memory that is necessary to solve $\mathscr{P}$ is statically allocated, removing the need for dynamic memory allocation altogether. This property is highly valued in the development of software for safety-critical systems because it eliminates memory leaks and more importantly, reduces the complexity of flight software verification.

The real-time capability of the customized solver is demonstrated via a set of quadrotor flight experiments [13]. Timing results are reported in [13], and videos of these experiments can be found on our YouTube channel [35]. Note that the quadrotors in the experiments are running the presented convexification methods onboard, and performming a variety of tasks autonomously in real-time.

VI. FEEDBACK CONTROL SYNTHESIS

While the aforementioned methods may suffice in the absence of disturbance and plant uncertainties, in practice they are usually used to design nominal trajectories. Solving the optimal control problem (Problem 1) can be done independently or as a part of MPC [36]. In both cases, feedback control synthesis is used to maintain the actual system states within an invariant "tube" about the nominal trajectory [37], [38]. The tube provides robustness to model uncertainties and environmental disturbances. In particular, [37] devised a method to generate the feedback control policy by solving Linear Matrix Inequalities (LMI). The method can be summarized as follows.

1) Specify a tube radius, and corresponding sets $\mathscr{U}_f$ and $\mathscr{X}_f$. Obtain sets $\mathscr{U}_0$ and $\mathscr{X}_0$ such that $\mathscr{U}_0 + \mathscr{U}_f \subseteq \mathscr{U}$, and $\mathscr{X}_0 + \mathscr{X}_f \subseteq \mathscr{X}$, where the "+" means Minkowski sum of two sets.
2) Solve the guidance problem (Problem 1) with feasible sets $\mathscr{U}_0$, $\mathscr{X}_0$ to get nominal control u_0 and nominal trajectory x_0.
3) Let $u = u_0 + \kappa_f(x,x_0)$, and $x = x_0 + \delta x$, where κ_f is designed by e.g. LMI synthesis, such that $\kappa_f(x,x_0) \in \mathscr{U}_f$ and $\delta x \in \mathscr{X}_f$, for all uncertainties $\theta \in \Theta$.

Figure 6 provides a visualization of sets $\mathscr{X}$, $\mathscr{X}_0$, and $\mathscr{X}_f$, demonstrating the robustness of this method. More details about the design procedure can be found in [37].

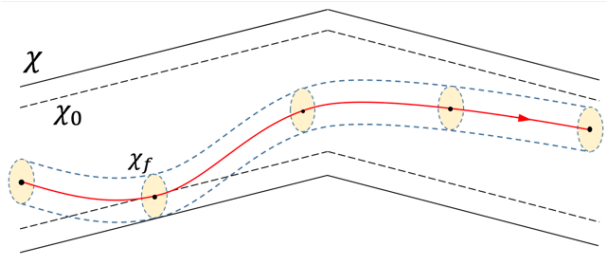


Fig. 6: Robust nominal trajectory tracking with "tube" based feedback control synthesis.

VII. CONCLUDING REMARKS

In this paper, we presented a tutorial on how to apply convex optimization to solve aerospace control problems. We

proposed a two tiered approach, consisting of a guidance tier and a feedback tier. For the guidance tier, we discussed two convexification techniques: *lossless convexification*, and *successive convexification*. The former addresses a class of non-convex control constraints, while the latter addresses non-convexities that arise from nonlinear dynamics and non-convex state constraints. These techniques can be used to convert realistic non-convex problems in to convex ones.

Further, as system autonomy, precision, and performance requirements continue to grow, convexification based guidance and control technology provides unique advantages. It not only offers methods and algorithms that can be implemented in real-time, but also provides theoretical guarantees (i.e. proofs of equivalence or convergence), which are extremely valuable for aerospace applications, and that can enable missions that were previously out of reach.

ACKNOWLEDGMENT

The authors gratefully acknowledge Ilya Kolmanovsky of University of Michigan for organizing this timely tutorial session.

REFERENCES

[1] J. A. Starek, B. Açıkmeşe, I. A. Nesnas, and M. Pavone, "Spacecraft autonomy challenges for next-generation space missions," in *Advances in Control System Technology for Aerospace Applications*. Springer, 2016, pp. 1–48.

[2] L. Blackmore, "Autonomous precision landing of space rockets," *The Bridge on Frontiers of Engineering*, vol. 4, no. 46, pp. 15–20, 2016.

[3] B. Açıkmeşe and S. R. Ploen, "Convex programming approach to powered descent guidance for Mars landing," *AIAA Journal of Guidance, Control and Dynamics*, vol. 30, no. 5, pp. 1353–1366, 2007.

[4] B. Açıkmeşe, J. Carson, and L. Blackmore, "Lossless convexification of non-convex control bound and pointing constraints of the soft landing optimal control problem," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 6, pp. 2104–2113, 2013.

[5] U. Lee and M. Mesbahi, "Constrained autonomous precision landing via dual quaternions and model predictive control," *Journal of Guidance, Control, and Dynamics*, vol. 40, pp. 292–308, 2017.

[6] M. Szmuk, U. Eren, and B. Açıkmeşe, "Successive convexification for mars 6-dof powered descent landing guidance," in *AIAA Guidance, Navigation, and Control Conference*, 2017, p. 1500.

[7] P. Lu and X. Liu, "Autonomous trajectory planning for rendezvous and proximity operations by conic optimization," *Journal of Guidance, Control, and Dynamics*, vol. 36, no. 2, pp. 375–389, 2013.

[8] A. Weiss, M. Baldwin, R. S. Erwin, and I. Kolmanovsky, "Model predictive control for spacecraft rendezvous and docking: Strategies for handling constraints and case studies," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 4, pp. 1638–1647, July 2015.

[9] Y. Kim and M. Mesbahi, "Quadratically constrained attitude control via semidefinite programming," *IEEE Transactions on Automatic Control*, vol. 49, no. 5, pp. 731–735, 2004.

[10] U. V. Kalabic, R. Gupta, S. D. Cairano, A. M. Bloch, and I. V. Kolmanovsky, "MPC on manifolds with an application to the control of spacecraft attitude on SO(3)," *Automatica*, vol. 76, pp. 293–300, 2017.

[11] A. Richards, T. Schouwenaars, J. P. How, and E. Feron, "Spacecraft trajectory planning with avoidance constraints using mixed-integer linear programming," *Journal of Guidance, Control, and Dynamics*, vol. 25, no. 4, pp. 755–764, 2002.

[12] R. Allen and M. Pavone, "A real-time framework for kinodynamic planning with application to quadrotor obstacle avoidance," in *AIAA Guidance, Navigation, and Control Conference*, 2016, p. 1374.

[13] M. Szmuk, C. A. Pascucci, D. Dueri, and B. Açıkmeşe, "Convexification and real-time on-board optimization for agile quad-rotor maneuvering and obstacle avoidance," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 4862–4868.

[14] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[15] Y. Nesterov and A. Nemirovskii, *Interior-Point Polynomial Algorithms in Convex Programming*. Society for Industrial and Applied Mathematics, 1994.

[16] A. Domahidi, E. Chu, and S. Boyd, "ECOS: An SOCP solver for embedded systems," in *European Control Conference (ECC)*. IEEE, 2013, pp. 3071–3076.

[17] J. Mattingley and S. Boyd, "Cvxgen: A code generator for embedded convex optimization," *Optimization and Engineering*, vol. 13, no. 1, pp. 1–27, 2012.

[18] D. Dueri, J. Zhang, and B. Açıkmeşe, "Automated custom code generation for embedded, real-time second order cone programming," in *19th IFAC World Congress*, 2014, pp. 1605–1612.

[19] B. Açıkmeşe, D. P. Scharf, E. A. Murray, and F. Y. Hadaegh, "A convex guidance algorithm for formation reconfiguration," in *AIAA Guidance, Navigation, and Control Conference*, 2006, p. 6070.

[20] 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.

[21] B. Açıkmeşe and L. Blackmore, "Lossless convexification of a class of optimal control problems with non-convex control constraints," *Automatica*, vol. 47, no. 2, pp. 341–347, 2011.

[22] Y. Mao, M. Szmuk, and B. Açıkmeşe, "Successive convexification of non-convex optimal control problems and its convergence properties," in *IEEE 55th Conference on Decision and Control (CDC)*, 2016, pp. 3636–3641.

[23] 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, 2017.

[24] D. Hull, "Conversion of optimal control problems into parameter optimization problems," *Journal of Guidance, Control, and Dynamics*, vol. 20, no. 1, pp. 57–60, 1997.

[25] P. Tsiotras and M. Mesbahi, "Toward an algorithmic control theory," *Journal of Guidance, Control, and Dynamics*, vol. 40, pp. 194–196, 2017.

[26] F. Palacios-Gomez, L. Lasdon, and M. Engquist, "Nonlinear optimization by successive linear programming," *Management Science*, vol. 28, no. 10, pp. 1106–1120, 1982.

[27] J. Zhang, N.-H. Kim, and L. Lasdon, "An improved successive linear programming algorithm," *Management science*, vol. 31, no. 10, pp. 1312–1331, 1985.

[28] J. B. Rosen, "Iterative solution of nonlinear optimal control problems," *SIAM Journal on Control*, vol. 4, no. 1, pp. 223–244, 1966.

[29] D. Q. Mayne and E. Polak, "An exact penalty function algorithm for control problems with state and control constraints," *IEEE Transactions on Automatic Control*, vol. 32, no. 5, pp. 380–387, 1987.

[30] F. Augugliaro, A. P. Schoellig, and R. D'Andrea, "Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 1917–1922.

[31] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, 2014.

[32] A. R. Conn, N. I. Gould, and P. L. Toint, *Trust region methods*. Siam, 2000, vol. 1.

[33] R. Fletcher, *Practical methods of optimization, 2nd Edition*. Wiley, 1987, vol. 2.

[34] D. Dueri, Y. Mao, Z. Mian, J. Ding, and B. Açıkmeşe, "Trajectory optimization with inter-sample obstacle avoidance via successive convexification," in *IEEE 56th Conference on Decision and Control (CDC)*, 2017, pp. 1150–1156.

[35] "Autonomous Control Labortory (ACL)." [Online]. Available: https://www.youtube.com/channel/UCZwV0cPCR3QeGn4dSfXxkKw

[36] F. Borrelli, A. Bemporad, and M. Morari, *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[37] B. Açikmese, J. M. Carson, and D. S. Bayard, "A robust model predictive control algorithm for incrementally conic uncertain/nonlinear systems," *International Journal of Robust and Nonlinear Control*, vol. 21, no. 5, pp. 563–590, 2011.

[38] W. Langson, I. Chryssochoos, S. Raković, and D. Q. Mayne, "Robust model predictive control using tubes," *Automatica*, vol. 40, no. 1, pp. 125–133, 2004.