

CALIPSO: A Differentiable Solver for Trajectory Optimization with Conic and Complementarity Constraints

Taylor A. Howell¹, Kevin Tracy², Simon Le Cleac’h¹, and Zachary Manchester²

¹ Stanford University, Stanford CA 94305, USA,
thowell@stanford.edu,

WWW home page: <http://roboticexplorationlab.org>

² Carnegie Mellon University, Pittsburgh PA 15213, USA

Abstract. We present a new solver for non-convex trajectory optimization problems that is specialized for robotics applications. CALIPSO, or the Conic Augmented Lagrangian Interior-Point Solver, combines several strategies for constrained numerical optimization to natively handle second-order cones and complementarity constraints. It reliably solves challenging motion-planning problems that include contact-implicit formulations of impacts and Coulomb friction, thrust limits subject to conic constraints, and state-triggered constraints where general-purpose nonlinear programming solvers like SNOPT and Ipopt fail to converge. Additionally, CALIPSO supports efficient differentiation of solutions with respect to problem data, enabling bi-level optimization applications like auto-tuning of feedback policies. Reliable convergence of the solver is demonstrated on a range of problems from manipulation, locomotion, and aerospace domains. An open-source implementation of this solver is available.

Keywords: trajectory optimization, robotics, contact dynamics

1 Introduction

Trajectory optimization is a powerful tool for offline generation of complex behaviors for dynamic systems, as well as online as a planner or feedback controller within model-predictive-control frameworks. The use of constraints greatly enhances the ability of a designer to generate desirable solutions, enforce safe behaviors, and model physical phenomena. Unfortunately, many constraint types that have important and direct applications to robotics are poorly handled by existing general-purpose nonlinear programming (NLP) solvers [31] or differential dynamic programming (DDP) algorithms [21].

Second-order cones [9], which commonly appear as friction-cone [30] or thrust-limit [7] constraints, present difficulties for NLP solvers due to their nondifferentiability at commonly occurring states, like when the friction or thrust forces are zero. Common reformulations of these constraints for solvers like SNOPT [15] and Ipopt [47] are typically non-convex and fail to work well in practice [45].

Contact dynamics, including impact and friction, are naturally modeled with complementarity constraints [35]. This formulation constrains contact forces to only take on non-zero values when the distance between objects is zero. State-triggered constraints [38], in which constraints switch on or off in different regions of the state space, can similarly be modeled with complementarity constraints. However, these constraints violate the linear independence constraint qualification (LICQ), a fundamental assumption in the convergence theory of standard NLP solvers [31, 6].

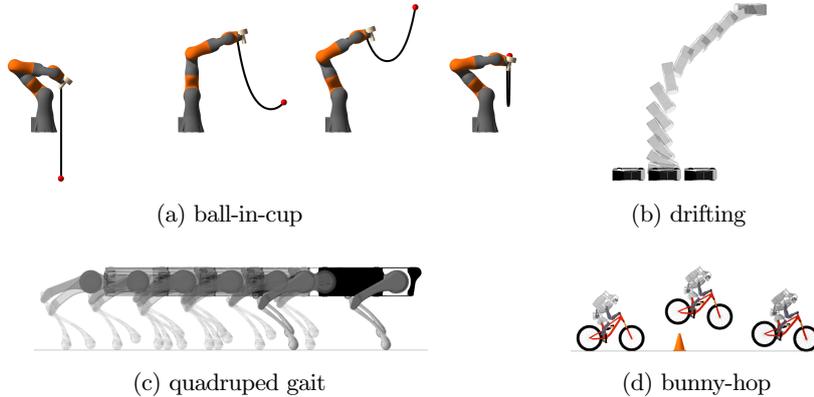


Fig. 1: Contact-implicit trajectory optimization examples optimized with CALIPSO. (a) Ball attached to a string is swung into a cup by optimizing end-effector positions and forces. (b) Autonomous car plans a drifting maneuver in order to parallel park. (c) Gait for a quadruped is optimized via a single step and loop constraint. (d) Bicycle robot performs a bunny hop over an obstacle to reach a goal state.

In this work, we present a new solver for trajectory optimization, CALIPSO: Conic Augmented Lagrangian Interior-Point Solver. The development of this solver is motivated by challenging non-convex motion-planning problems that require second-order cone and complementarity constraints; in particular, contact-implicit trajectory optimization [33, 28] for locomotion and manipulation. CALIPSO combines a number of ideas and algorithms from constrained numerical optimization to solve these difficult problems reliably.

Second-order cones are handled using an interior-point method [44] that exploits the convexity of these nonlinear constraints for strict enforcement without the need for linear approximations. All equality constraints are handled using an augmented Lagrangian method [4], which does not require the constraints to satisfy LICQ and is robust to the degeneracies that can arise from complementarity constraints [18]. Additionally, both the interior-point and augmented-Lagrangian methods are formulated in a primal-dual fashion to enhance the numerical robustness and performance of the solver. The computation of Newton steps on the combined primal-dual augmented-Lagrangian and interior-point KKT conditions are reformulated as a symmetric linear system to enable the use of fast, direct linear algebra methods [10]. Finally, the implicit-function theorem [11, 2] is utilized to efficiently differentiate through solutions, enabling bi-level optimization applications such as auto-tuning of model-predictive-control policies.

Our specific contributions are:

- A differentiable trajectory-optimization solver with native support for second-order cones and reliable handling of complementarity constraints
- A novel, combined, interior-point and augmented-Lagrangian algorithm for non-convex optimization
- A symmetric reformulation of the combined method’s KKT system for fast symmetric linear-system solvers
- A collection of benchmark robot motion-planning problems that contain second-order cone and complementarity constraints

Table 1: Comparison of feature support for trajectory-optimization solvers.

solver	method	accuracy	second-order	complementarity	differentiable
ALTRO [16]	indirect	medium	✓	×	×
Trajax [43]	indirect	medium	×	×	✓
GuSTO [8]	direct	high	✓	×	×
FORCES [49]	direct	high	×	×	×
Drake [42]	direct	high	×	×	×
CALIPSO	direct	high	✓	✓	✓

– An open-source implementation of the solver written in Julia

In the remainder of this paper, we first provide an overview of related work on trajectory optimization methods in Section 2. Then, we provide the necessary background on complementarity constraints, LICQ, augmented-Lagrangian methods, and interior-point methods in Section 3. Next, we present CALIPSO and its key algorithms and routines in Section 4. We then demonstrate CALIPSO on a collection of robot motion-planning examples in Section 5. Finally, we conclude with a discussion and closing remarks in Section 6.

2 Related work

Trajectory-optimization problems are solved with methods that are classically categorized as either indirect or direct [5]. Indirect methods include shooting methods, DDP [21], and iterative LQR (iLQR) [27]. These methods exploit the temporal structure of the problem and utilize a Riccati backward pass to compute updates for control variables followed by forward simulation rollouts to update the states. Classically, apart from the dynamics, these methods do not include support for constraints. In recent years, various approaches have extended these methods to handle constraints [41, 16, 17, 29, 36, 22, 20]. However, reliable constraint handling and solution accuracy for these methods is still challenging in many scenarios.

Direct methods, in contrast, directly transcribe the trajectory-optimization problem as a constrained nonlinear program, with both states and controls as decision variables [37]. The transcription is provided to a general-purpose solver such as SNOPT or Ipopt. This approach is generally reliable and enables robust constraint handling. However, these NLP solvers do not exploit the temporal structure of the trajectory-optimization problem like indirect methods, and historically have been thought to converge to solutions more slowly as a result. To address this limitation, direct solvers tailored for the underlying trajectory-optimization problem structure have been proposed [48] and are available as commercial tools [49].

While providing reliable constraint handling for equality and inequality constraints in many scenarios, direct methods lack support for second-order cones [9] and exhibit difficulties handling complementarity constraints [35]. The exact handling of second-order cones as inequality constraints with NLP solvers results in poor practical performance because the nondifferentiable point of the cone is frequently visited in robotics applications (e.g., an object resting on a surface with zero friction force). Classic reformulations are non-convex and similarly exhibit poor and unreliable convergence [45]. Recently, sequential convexification approaches have been developed that are

able to handle second-order cones by iteratively solving convex approximations of the original problem using general-purpose cone solvers [40, 8].

The complementarity constraints that can arise in robotics problems are generally difficult for NLP solvers to handle because they violate LICQ. To overcome this, a number of problem reformulations and constraint relaxation techniques have been proposed and explored for interior-point [6, 34] and augmented Lagrangian [19] methods, but these approaches largely remain *ad hoc* and are unavailable in state-of-the-art solvers for trajectory optimization, which we compare in Table 1.

3 Background

In this section, we first provide a brief overview of trajectory optimization, followed by background on complementarity constraints, LICQ, augmented Lagrangian methods, and interior-point methods for constrained optimization. Finally, we compare approaches for differentiating through a solver.

3.1 Trajectory optimization

Direct methods for trajectory optimization transcribe problems into standard NLP instances:

$$\begin{aligned} & \underset{x}{\text{minimize}} && c(x) \\ & \text{subject to} && g(x) = 0, \\ & && h(x) \in \mathcal{K}, \end{aligned} \tag{1}$$

with decision variables $x \in \mathbf{R}^n$, objective $c: \mathbf{R}^n \rightarrow \mathbf{R}$, equality constraints $g: \mathbf{R}^n \rightarrow \mathbf{R}^m$, and inequality constraints $h: \mathbf{R}^n \rightarrow \mathbf{R}^p$ in cone \mathcal{K} . The functions are assumed to be smooth and twice differentiable, and the cone is the Cartesian product of convex cones (e.g., standard positive-orthant inequalities and second-order cones) [9].

Trajectory optimization problems,

$$\begin{aligned} & \underset{X_{1:T}, U_{1:T-1}}{\text{minimize}} && C_T(X_T) + \sum_{t=1}^{T-1} C_t(X_t, U_t) \\ & \text{subject to} && F_t(X_t, U_t) = X_{t+1}, && t = 1, \dots, T-1, \\ & && E_t(X_t, U_t) = 0, && t = 1, \dots, T, \\ & && H_t(X_t, U_t) \in \mathcal{K}_t, && t = 1, \dots, T, \end{aligned} \tag{2}$$

are instances with special temporal structure for a dynamical system with state $X_t \in \mathbf{R}^{n_t}$, control inputs $U_t \in \mathbf{R}^{m_t}$, time index t , discrete-time dynamics $F_t: \mathbf{R}^{n_t} \times \mathbf{R}^{m_t} \rightarrow \mathbf{R}^{n_{t+1}}$, and stage-wise objective $C_t: \mathbf{R}^{n_t} \times \mathbf{R}^{m_t} \rightarrow \mathbf{R}$, equality constraints $E_t: \mathbf{R}^{n_t} \times \mathbf{R}^{m_t} \rightarrow \mathbf{R}^{p_t}$, and cone constraints $H_t: \mathbf{R}^{n_t} \times \mathbf{R}^{m_t} \rightarrow \mathbf{R}^{d_t}$, over a planning horizon T .

3.2 Complementarity constraints

Contact-implicit trajectory optimization [33] optimizes trajectories for systems that make and break contact with their environments and represents dynamics using complementarity constraints. For example, optimizing motion over a single time step for an actuated particle in a single dimension modeled with impact such that it cannot

pass through the floor:

$$\begin{aligned}
 & \underset{z,u,\gamma}{\text{minimize}} && (z-z_{\text{goal}})^2+u^2 \\
 & \text{subject to} && m((z-\bar{z})/h-\bar{v}-gh)+\gamma+u=0, \\
 & && z\cdot\gamma=0, \\
 & && z,\gamma\geq 0,
 \end{aligned} \tag{3}$$

with current and next positions $\bar{z}, z \in \mathbf{R}$, current velocity $\bar{v} \in \mathbf{R}$, control input $u \in \mathbf{R}$, contact impulse $\gamma \in \mathbf{R}$, mass $m \in \mathbf{R}_{++}$, gravity $g \in \mathbf{R}_+$, time step $h \in \mathbf{R}_+$, and goal $z_{\text{goal}} \in \mathbf{R}$. These constraints are derived from a constrained discrete Lagrangian [28]. The set of constraints on z and γ are collectively referred to as a complimentary constraint, and are sometimes abbreviated $z \perp \gamma$. This formulation does not require pre-specified contact-mode sequences or hybrid dynamics since the solver is able to optimize physically correct contact dynamics at each time step.

3.3 Linear-independence constraint qualification

General-purpose NLP solvers that rely on Newton’s method to compute search directions (e.g., SNOPT and Ipopt) assume that the constraints provided by the user satisfy the LICQ in the neighborhood of solutions. Certain classes of constraints, including complementarity conditions that naturally arise in contact dynamics, often do not satisfy this assumption. We demonstrate how this assumption is violated with a simple contact-implicit trajectory-optimization problem (3). The Lagrangian for the problem is:

$$L(z,u,\gamma,a,b,c,d) = (z-z_{\text{goal}})^2+u^2 + a\left(m((z-\bar{z})/h-\bar{v}-gh)+\gamma+u\right) + bz\gamma - cz - d\gamma, \tag{4}$$

where $a,b,c,d \in \mathbf{R}$ are the Lagrange multipliers, or “dual variables,” associated with constraints. The KKT system is:

$$\begin{bmatrix} 2 & b & 0 & m/h & \gamma & -1 & 0 \\ b & 0 & 0 & 1 & z & 0 & -1 \\ 0 & 0 & 2 & 1 & 0 & 0 & 0 \\ m/h & 1 & 1 & 0 & 0 & 0 & 0 \\ \gamma & z & 0 & 0 & 0 & 0 & 0 \\ c & 0 & 0 & 0 & 0 & z & 0 \\ 0 & d & 0 & 0 & 0 & 0 & \gamma \end{bmatrix} \begin{bmatrix} \Delta z \\ \Delta \gamma \\ \Delta u \\ \Delta a \\ \Delta b \\ \Delta c \\ \Delta d \end{bmatrix} = - \begin{bmatrix} 2(z-\bar{z})+am/h+b\gamma-c \\ a+bz-d \\ 2u+a \\ m((z-\bar{z})/h-\bar{v}-gh)+\gamma+u \\ z\cdot\gamma \\ c\cdot z \\ d\cdot\gamma \end{bmatrix}, \tag{5}$$

with first-order necessary (KKT) conditions (right-hand side) and KKT matrix (left-hand side). The Newton step that a standard NLP solver would take to drive these KKT conditions to zero is computed by solving this system (5).

In the scenario where the particle is above the surface (i.e., $z > 0$) the contact impulse must be zero (i.e., $\gamma = 0$). As a result, the fifth and seventh rows of the KKT matrix will be linearly dependent, resulting in non-unique optimal dual variables and violation of LICQ. A similar result occurs when $\gamma > 0$ and $z = 0$. Consequently, the Newton step is not well defined in these cases, causing difficulties for the solver. While a myriad of *ad hoc* heuristics exist to alleviate this problem, we pursue a more rigorous approach in the following sections based on ideas from constrained numerical optimization.

3.4 Augmented Lagrangian methods

We now consider equality constrained problems of the form,

$$\begin{aligned} & \underset{x}{\text{minimize}} && c(x) \\ & \text{subject to} && g(x) = 0. \end{aligned} \quad (6)$$

The augmented Lagrangian method transforms this problem (6) into an unconstrained problem by introducing dual variables, $\lambda \in \mathbf{R}^m$, and a quadratic penalty parameterized by $\rho \in \mathbf{R}_+$:

$$L_{\mathcal{A}}(x; \lambda, \rho) = c(x) + \lambda^T g(x) + \frac{\rho}{2} g(x)^T g(x), \quad (7)$$

where we refer to $L_{\mathcal{A}}$ as the *augmented Lagrangian* for the problem (6).

Primal method: The classic method alternates between minimizing the augmented Lagrangian (7) and performing outer updates on the dual variables and penalty:

$$\lambda \leftarrow \lambda + \rho g(x), \quad \rho \leftarrow \phi(\rho), \quad (8)$$

until a solution to the original problem (6) is found [4]. This typically requires ten or fewer outer updates and a simple update, $\phi: \mathbf{R}_+ \rightarrow \mathbf{R}_+$, that scales the penalty by a constant value, works well in practice. Throughout, subscripts are used to denote derivatives and we drop the variable dependence of the functions for clarity. The KKT system for this method is:

$$\left[c_{xx} + \rho g_x^T g_x + \sum_{i=1}^m (\lambda^{(i)} + \rho g^{(i)}) g_{xx}^{(i)} \right] \Delta x = - \left[c_x + g_x^T (\lambda + \rho g) \right]. \quad (9)$$

Search directions $\Delta x \in \mathbf{R}^n$ are computed by solving the linear system (9) for fixed values of the dual variables and penalty. Newton's method with a line search is utilized to compute iterates that satisfy the KKT conditions, or residual (9), to a desired tolerance. Importantly, the KKT matrix becomes increasingly ill-conditioned as the penalty is increased in order to achieve better satisfaction of the equality constraints, degrading the quality of the Newton step.

Primal-dual method: To address the deficiencies of the primal method, a primal-dual method introduces additional dual variables, $y \in \mathbf{R}^m$ and constraints:

$$y = \lambda + \rho g(x), \quad (10)$$

in order to utilize an alternative KKT system with better numerical properties.

Combining these constraints (10) with the primal KKT system (9) and performing a simple manipulation yields the *primal-dual* augmented-Lagrangian KKT system:

$$\begin{bmatrix} c_{xx} + \sum_{i=1}^m y^{(i)} g_{xx}^{(i)} & g_x^T \\ g_x & -\frac{1}{\rho} I \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = - \begin{bmatrix} c_x + g_x^T y \\ g + \frac{1}{\rho} (\lambda - y) \end{bmatrix}. \quad (11)$$

In contrast to the primal system (9), this system (11) does not become ill-conditioned as the penalty is increased because this term does not appear in the KKT matrix—only its inverse appears (i.e., $-\frac{1}{\rho} I$), which actually enhances the conditioning of the system by performing dual regularization [24, 14, 3]. Further, this method does not require LICQ because the KKT matrix remains full rank even in cases where g_x is rank deficient as a result of the dual regularization [18].

Additionally, the KKT conditions now contain relaxed constraints (i.e., $g(x) + \frac{1}{\rho} (\lambda - y)$) that are particularly helpful for complementarity formulations since they are only satisfied in the convergence limit as outer updates are performed. In the contact-dynamics

setting, this relaxation corresponds to “soft” contact models that are iteratively updated to become “hard” as the algorithm converges.

3.5 Interior-point methods

To handle problems with cone constraints (i.e., inequalities and second-order cones) that commonly occur in robotics applications, for example torque limits or friction cones, we employ interior-point methods. To illustrate the approach, we now consider problems with inequality constraints:

$$\begin{aligned} & \underset{x}{\text{minimize}} && c(x) \\ & \text{subject to} && h(x) \geq 0. \end{aligned} \quad (12)$$

An unconstrained problem is formed by introducing a logarithmic barrier, relaxed by a central-path parameter $\kappa \in \mathbf{R}_+$:

$$L_{\mathcal{B}}(x; \kappa) = c(x) - \kappa \sum_{i=1}^p \log(h^{(i)}(x)), \quad (13)$$

where we refer to $L_{\mathcal{B}}$ as the *barrier Lagrangian* for the problem (12).

Primal method: The classic method alternates between minimizing the barrier Lagrangian (13) and outer updates to the central-path parameter until a solution to the original problem (12) is found [9]. An effective strategy for the update is to decrease the parameter by a constant factor.

The KKT system for this method is:

$$\left[c_{xx} - \kappa \sum_{i=1}^p \left(\frac{1}{h^{(i)}} h_{xx}^{(i)} - \frac{1}{(h^{(i)})^2} (h_x^{(i)})^2 \right) \right] \Delta x = - \left[c_x - \kappa \sum_{i=1}^p \frac{1}{h^{(i)}} h_x^{(i)} \right]. \quad (14)$$

As the central-path parameter is decreased, the logarithmic barrier becomes a closer approximation to the indicator function, which has an infinite cost if a constraint is violated and is otherwise zero [9]. While simple, this approach suffers from numerical ill-conditioning, similar to the primal augmented Lagrangian method, as the central-path parameter approaches zero, degrading a solver’s ability to find accurate solutions to the original problem (12).

Primal-dual method: To address the conditioning issues of the primal method, additional dual variables, $z \in \mathbf{R}^p$, and constraints, $z^{(i)} = \kappa/h^{(i)}(x)$ for $i = 1, \dots, p$, are introduced to form a new KKT system:

$$\begin{bmatrix} c_{xx} - \sum_{i=1}^p z^{(i)} h_{xx}^{(i)} & h_x^T \\ \mathbf{diag}(z) h_x & \mathbf{diag}(h) \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = - \begin{bmatrix} c_x - h_x^T z \\ z \circ h - \kappa \mathbf{e} \end{bmatrix}, \quad (15)$$

$$z, h \geq 0. \quad (16)$$

Similar to the primal-dual augmented Lagrangian method, this KKT matrix does not depend on the central-path parameter, resulting in significantly better numerical conditioning than primal methods. The complementarity constraints in the KKT conditions are relaxed (i.e., $z \circ h(x) - \kappa \mathbf{e}$), only being satisfied in the limit as the central-path parameter is decreased to zero. Here, the target \mathbf{e} is a vector of ones and the cone product \circ denotes an element-wise product.

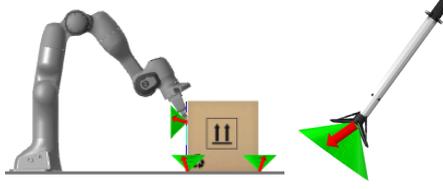


Fig. 2: Robotics applications with second-order cone constraints include Coulomb friction at contact points in manipulation tasks (left) and thrust limits on rockets (right). Cone constraints are shown in green, while force vectors are shown in red.

Second-order cone constraints: The barrier formulation accommodates second-order cone constraints,

$$a \in \mathcal{Q}_l = \{(a^{(1)}, a^{(2:l)}) \in \mathbf{R} \times \mathbf{R}^{l-1} \mid \|a^{(2:l)}\|_2 \leq a^{(1)}\}, \quad (17)$$

of dimension l , which frequently appear in robotics applications as friction cones and thrust limits (Fig. 2). In this setting, the barrier contains squared cone variables,

$$\frac{1}{2} \log \left((a^{(1)})^2 - (a^{(2:l)})^T a^{(2:l)} \right), \quad (18)$$

and remains convex; the cone product and target are:

$$a \circ b = (a^T b, a^{(1)} b^{(2:l)} + b^{(1)} a^{(2:l)}), \quad \mathbf{e} = (1, 0_{l-1}). \quad (19)$$

The inequality constraints (16) are replaced with their second-order cone counterparts (17) [12, 44].

3.6 Differentiable optimization

Solvers can be differentiated by unrolling the algorithm and utilizing the chain rule to differentiate through each iterate [13]. However, in practice, this approach requires truncating the number of iterates, which can lead to low-accuracy solutions. Additionally, the approach can be plagued by numerical issues that lead to exploding or vanishing gradients. A more efficient approach is to utilize the implicit-function theorem at a solution point in order to compute the sensitivities of the solution [2, 1].

An implicit function, $R: \mathbf{R}^a \times \mathbf{R}^b \rightarrow \mathbf{R}^a$, is defined as

$$R(w^*; \theta) = 0, \quad (20)$$

for solutions $w^* \in \mathbf{R}^a$ and problem data $\theta \in \mathbf{R}^b$. At an equilibrium point, $w^*(\theta)$, the sensitivities of the solution with respect to the problem data, i.e., $\partial w / \partial \theta$, can be computed under certain conditions [11] as:

$$\frac{\partial w}{\partial \theta} = - \left(\frac{\partial R}{\partial w} \right)^{-1} \frac{\partial R}{\partial \theta}. \quad (21)$$

In the case that $\partial R / \partial w$ is not full rank, an approximate solution, e.g., least-squares, can be computed. A similar approach for differentiating through DDP-based solvers using a Riccati approach has been proposed [23]. In the context of trajectory optimization, the decision variables w contain trajectories of states and controls and the problem data includes terms like control limits or parameters of the system like friction coefficients.

4 CALIPSO

CALIPSO is a differentiable primal-dual augmented Lagrangian interior-point solver for non-convex optimization problems with second-order cone and complementarity

constraints. Its standard problem formulation is:

$$\begin{aligned} & \underset{x}{\text{minimize}} && c(x;\theta) \\ & \text{subject to} && g(x;\theta)=0, \\ & && h(x;\theta)\in\mathcal{K}, \end{aligned} \quad (22)$$

with decision variables $x\in\mathbf{R}^n$, problem data $\theta\in\mathbf{R}^d$, equality constraints $g:\mathbf{R}^n\times\mathbf{R}^d\rightarrow\mathbf{R}^m$, and constraints $h:\mathbf{R}^n\times\mathbf{R}^d\rightarrow\mathbf{R}^p$ in cone $\mathcal{K}=\mathbf{R}_{++}^q\times Q_{l_1}^{(1)}\times\cdots\times Q_{l_j}^{(j)}$ comprising a q -dimensional inequality and j second-order cones, each of dimension l_i . Internally, problem (22) is reformulated and additional slack variables $r\in\mathbf{R}^m$ and $s\in\mathbf{R}^p$, associated with the equality and cone constraints, respectively, are introduced, and the following modified problem is formed:

$$\begin{aligned} & \underset{x,r,s}{\text{minimize}} && c(x;\theta)+\lambda^T r+\frac{\rho}{2}r^T r-\kappa\sum_{i=1}^p\log(s^{(i)}) \\ & \text{subject to} && g(x;\theta)-r=0, \\ & && h(x;\theta)-s=0, \\ & && s\in\mathcal{K}, \end{aligned} \quad (23)$$

The modified problem's Lagrangian is:

$$\begin{aligned} L(w;\theta,\lambda,\rho,\kappa) &= c(x;\theta)+y^T(g(x;\theta)-r)+z^T(h(x;\theta)-s) \\ & \quad +\lambda^T r+\frac{\rho}{2}r^T r-\kappa\sum_{i=1}^p\log(s^{(i)}). \end{aligned} \quad (24)$$

For convenience we denote the concatenation of all of the solver's variables as $w=(x,r,s,y,z,t)$. The KKT system is:

$$\begin{bmatrix} L_{xx}+\epsilon_p I & 0 & 0 & g_x^T & h_x^T & 0 \\ 0 & (\rho+\epsilon_p)I & 0 & -I & 0 & 0 \\ 0 & 0 & \epsilon_p I & 0 & -I & -I \\ g_x & -I & 0 & -\epsilon_d I & 0 & 0 \\ h_x & 0 & -I & 0 & -\epsilon_d I & 0 \\ 0 & 0 & P_s & 0 & 0 & P_t-\epsilon_d I \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta r \\ \Delta s \\ \Delta y \\ \Delta z \\ \Delta t \end{bmatrix} = - \begin{bmatrix} c_x+g_x^T y+h_x^T z \\ \lambda+\rho r-y \\ -z-t \\ g-r \\ h-s \\ s\circ t-\kappa\mathbf{e} \end{bmatrix}, \quad (25)$$

$$J\Delta w = -R,$$

where and $P_s, P_t\in\mathbf{S}^p$ are the cone-product Jacobians, and $\epsilon_p, \epsilon_d\in\mathbf{R}_+$ are additional primal and dual regularization terms, respectively.

4.1 Search direction

The nominal KKT system (25) without regularization (i.e., $\epsilon_p, \epsilon_d=0$) is non-symmetric, potentially with undesirable eigenvalues that will not return a descent direction. The solver modifies this system for faster computation and a more reliable search direction.

Symmetric KKT system. To be amenable to fast solvers for symmetric linear systems, the nominal KKT system is reformulated:

$$\begin{bmatrix} L_{xx}+\epsilon_p I & g_x^T & h_x^T \\ g_x & -\left(\frac{1}{\rho+\epsilon_p}I\right) & 0 \\ h_x & 0 & -\left(\epsilon_d I+\left((P_s+\epsilon_p\bar{P}_t)^{-1}\bar{P}_t\right)\right) \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix} = - \begin{bmatrix} L_x \\ \bar{L}_y \\ \bar{L}_z \end{bmatrix}, \quad (26)$$

using Schur complements [9], where $\bar{L}_y = L_y + \frac{1}{\rho + \epsilon_p} L_r$, $\bar{L}_z = L_z + (P_s + \epsilon_p \bar{P}_t)^{-1} (\bar{P}_t L_s + L_t)$, and $\bar{P}_t = P_t - \epsilon_d I$. The remaining search directions,

$$\Delta r = (\Delta y + L_r) / (\rho + \epsilon_p), \quad (27)$$

$$\Delta s = (P_s + \epsilon_p \bar{P}_t)^{-1} (\bar{P}_t \Delta z + \bar{P}_t L_s + L_r), \quad (28)$$

$$\Delta t = -\Delta z + \epsilon_p \Delta s - L_s, \quad (29)$$

are recovered from the solution (26). Iterative refinement [31] is performed to improve the quality of the search directions computed using the symmetric system (26 - 29).

Inertia correction. To ensure a unique descent direction, the left-hand side of the symmetric KKT system (26) is corrected to have an inertia of n positive, $m+p$, negative, and no zero-valued eigenvalues. This is accomplished via adaptive regularization that increases the regularization terms, and subsequently reduces the values when possible to limit unnecessary corrections, using a heuristic developed for Ipopt [47].

4.2 Line search

Cone variables are initialized being strictly feasible at the start of each solve. A filter line search that is a slight modification of the one used by Ipopt [47] is performed to ensure an improvement to the solution is achieved at each step of the algorithm and step sizes are initially chosen using a fraction-to-the-boundary rule to ensure that the cone constraints remain strictly satisfied. Additionally, a separate step size is computed for the candidate cone dual variables t to avoid unnecessarily restricting progress of the remaining variables.

After satisfying the cone constraints, a filter line search is performed with the merit function:

$$\varphi(x, r, s; \theta, \lambda, \rho, \kappa) = c(x; \theta) + \lambda^T r + \frac{\rho}{2} r^T r - \kappa \sum_{i=1}^p \log(s^{(i)}), \quad (30)$$

and violation metric:

$$\eta = \|(g(x) - r, h(x) - s)\|_1 / (m + p). \quad (31)$$

The step size is further decremented until either the merit or violation metric is decreased [46].

4.3 Outer updates

Convergence of a subproblem (23) occurs for fixed values of λ , ρ , and κ , when the criteria: $\|R\|_\infty \leq \gamma_\kappa \kappa$, is met for $\gamma_\kappa \in \mathbf{R}_+$. This criteria does not require strict satisfaction of subproblems and decreases the total number of iterations required by the solver. Outer updates on the central-path parameter and the penalty value are subsequently performed:

$$\kappa \leftarrow \max\left(\kappa_{\min}, \min(\psi_\kappa \cdot \kappa, \kappa^{\zeta_\kappa})\right), \quad \rho \leftarrow \min\left(\rho_{\max}, \max(\phi_\rho \cdot \rho, 1/\kappa)\right). \quad (32)$$

The updates are clipped to prevent unnecessarily small/large values.

4.4 Solution derivatives

The solution $w^*(\theta)$ returned by CALIPSO is differentiable with respect to its problem data θ . At a solution point, the residual is approximately zero (20), and sensitivities (21) are computed using $\partial R / \partial w = J$, which has already been computed and factorized, and:

$$\partial R / \partial \theta = (L_{x\theta}, 0_m, 0_p, g_\theta, h_\theta, 0_p). \quad (33)$$

Algorithm 1 CALIPSO

```

procedure CALIPSO( $x, \theta, c, g, h, \mathcal{K}, \gamma$ )
  parameters:  $\kappa = 1, \rho = 1, \lambda = 0$ 
  initialize:  $r, s, y, z, t$ 
  until  $\|R(w; \theta, y, \infty, 0)\|_\infty < \gamma R$ 
    until  $\|R(w; \theta, \lambda, \rho, \kappa)\|_\infty < \gamma \kappa$  do
      inertia correction:  $\epsilon_p, \epsilon_d$  ▷ Eq. (25)
      search direction:  $\Delta w = (\Delta x, \Delta r, \Delta s, \Delta y, \Delta z, \Delta t)$  ▷ Eqs. (26 - 29)
      cone line search:  $\alpha, \alpha_t$ 
      candidate:  $\hat{x} = x + \alpha \Delta x, \hat{r} = r + \alpha \Delta r, \hat{s} = s + \alpha \Delta s$ 
      filter:  $\hat{\varphi}, \hat{\eta}$ 
      update:  $w \leftarrow (\hat{x}, \hat{r}, \hat{s}, y + \alpha \Delta y, z + \alpha \Delta z, t + \alpha_t t)$ 
      outer update:  $\lambda, \rho, \kappa$  ▷ Eq. (32, 8)
    differentiate:  $\partial w / \partial \theta$  ▷ Eq. 21
  return  $w, \partial w / \partial \theta$ 
end procedure

```

Additionally, the sensitivity of the solution with respect to each element of the problem data can be computed in parallel.

4.5 Implementation

The CALIPSO solver is summarized in Algorithm 1. An open-source implementation of the solver, `CALIPSO.jl`, written in Julia, is provided. The solver, and the following examples, are available at: <https://github.com/thowell/CALIPSO.jl>.

5 Examples

We highlight the capabilities of CALIPSO by optimizing a collection of motion-planning problems from manipulation, locomotion, and aerospace domains that require second-order cone and complementarity constraints while exactly transcribing constraints without reformulation. Next, we demonstrate the ability to differentiate through the solver and auto-tune policies for nonlinear underactuated robotic systems. Additional details about the experimental setups are available in the open-source implementation.

5.1 Contact-implicit trajectory optimization

CALIPSO is utilized to optimize contact-implicit trajectory-optimization problems. The contact dynamics [33] are directly transcribed without modification. Comparisons are performed with Ipopt using the default MUMPS linear-system solver, and results are summarized in Table 2.

Ball-in-cup. The position and applied forces of a robotic manipulator’s end-effector are optimized to swing a ball into a cup (Fig. 1a). A string between the end-effector and ball is modeled with inequality and complementarity constraints. CALIPSO finds a physically realistic motion plan that is verified with inverse kinematics in simulation, while the Ipopt solution is of poor quality and violates physics by applying nonnegligible force to the ball while the string is slack.

Table 2: Comparison between CALIPSO and Ipopt for final objective value, constraint violation, and total iterations on contact-implicit trajectory-optimization problems. Cases that failed to converge (i.e., Ipopt falling back to restoration mode) are highlighted in red. Without user-tuned smoothing and problem reformulations, Ipopt performs poorly on these examples and returns solutions that are unusable for robotics applications. In contrast, CALIPSO returns high-quality solutions and we verify their performance in simulation with inverse-kinematics planning for the ball-in-cup and model-predictive control for the quadruped gait examples.

solver	objective	violation	iterations	solver	objective	violation	iterations
Ipopt	68.18	2.60e-2	205	Ipopt	8.66	1.00e-1	194
CALIPSO	11.96	4.86e-5	131	CALIPSO	0.24	1.38e-5	189
(a) ball-in-cup				(b) drifting			
solver	objective	violation	iterations	solver	objective	violation	iterations
Ipopt	1855.18	1.13e-1	2000	Ipopt	1503.42	8.96e-8	1409
CALIPSO	574.84	5.32e-4	178	CALIPSO	462.61	1.76e-6	101
(c) quadruped gait				(d) bunny-hop			

Drifting. A parallel-park maneuver is planned that requires an autonomous vehicle to drift (i.e., plan a trajectory with both sticking and sliding contact) into a goal configuration between two parked vehicles (Fig. 1b). The system is modeled as a Dubins car [25] with Coulomb friction applied to the wheels. Ipopt exhibits extremely poor converge and violates the friction cones to solve this problem, whereas CALIPSO finds a high-quality solution that leverages the nonlinear friction cone to slide into the narrow parking spot.

Quadruped gait. A gait is planned for a planar quadruped by optimizing a single step with a loop constraint (Fig. 1c). Ipopt struggles to converge, returning a solution with large complementarity violations and a reference that is unusable for online tracking. In contrast, CALIPSO finds a reference trajectory that we are able to track in a hard-contact simulation using a model-predictive-control policy that is reliable in the presence of model mismatch and disturbances [26].

Bunny-hop. A bicycle robot performs a bunny-hop over an obstacle (Fig. 1d). The rider is modeled as a mass with actuated prismatic joints attached to the bike at each wheel. CALIPSO is able to converge to a trajectory where the bicycle hops over the obstacle by manipulating the rider mass, while Ipopt takes an order of magnitude more iterations to converge.

In summary, Ipopt struggles to return solutions that are useful for robotics applications, whereas CALIPSO reliably returns high-quality solutions while using exact constraint specifications.

5.2 State-triggered constraints

A trigger condition $\Gamma: \mathbf{R}^n \rightarrow \mathbf{R}$ encodes the logic: $\Gamma(x) > 0 \implies h(x) \geq 0$, that a constraint is enforced only when the trigger is satisfied. Such state-triggered constraints

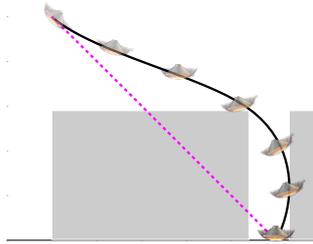


Fig. 3: Entry-vehicle soft-landing plan that must avoid elevated regions to the left and right of the landing zone, represented as state-triggered constraints. The unconstrained and constrained solutions are shown in magenta and black respectively.

are utilized within various aerospace applications [38, 39, 40] and commonly utilize a non-smooth formulation (left):

$$\min(0, -\Gamma(x)) \cdot h(x) \leq 0 \quad \rightarrow \quad \begin{aligned} \Gamma_+ - \Gamma_- &= \Gamma(x) \\ h_+ - h_- &= h(x) \\ \Gamma_+ \cdot h_- &= 0 \\ \Gamma_+, \Gamma_-, h_+, h_- &\geq 0, \end{aligned} \quad (34)$$

that linearizes poorly and can violate LICQ. In this work, we employ an equivalent complementarity formulation (right) to land an entry-vehicle in an environment with keep-out zones adjacent to the landing site. The resulting solution (Fig. 3) demonstrates CALIPSO’s ability to find solutions that satisfy complementarity constraints. Ipopt returns a solution that violates the keep-out zone for the majority of the trajectory.

5.3 Model-predictive-control auto-tuning

CALIPSO is utilized offline to plan reference trajectories and online as the tracking controller in a model-predictive-control (MPC) policy. The policy aims to track the reference, given an updated state estimate from a simulation, by re-planning over a reduced horizon to compute a new (feedback) control that is then applied to the system.

The policy’s cost weights are treated as problem data to be optimized and the solution computed by the policy (i.e., the re-optimized control inputs) are differentiated with respect to these parameters (21) in order to compute gradients that are utilized to automatically tune the policy. The metric for tuning the policy consists of quadratic costs on tracking the reference. Gradient descent with a line search is used to update the cost weights by differentiating through rollouts of the policy.

The policy weights are initialized with all ones and we compare the performance of the auto-tuned policy after 10 gradient steps with open-loop and untuned policies on swing-up tasks for cart-pole and acrobot systems. For both systems, the auto-tuned policy outperforms the baselines; results are summarized in Table 3.

6 Discussion

In this work, we have presented a new solver for trajectory optimization problems with second-order cone and complementarity constraints: CALIPSO. The solver prioritizes reliability and numerical robustness, and offers specialized constraint support

Table 3: Comparison of tracking error between open-loop and both untuned and auto-tuned model-predictive-control policies in simulation. By differentiating through CALIPSO, gradient-based optimizing is able to rapidly improve controller performance without requiring designer input.

	open-loop MPC (untuned)		MPC (tuned)
cart-pole	5.11e4	15.06	0.79
acrobot	1.38e4	439.26	0.04

that enables planning for challenging tasks arising in manipulation, locomotion, and aerospace applications while enabling the designer to exactly transcribe constraints without requiring problem reformulations. Additionally, the solver is differentiable with respect to its problem data, allowing it to be called by efficient, gradient-based, upper-level optimization routines for applications like policy auto-tuning. To the best of our knowledge, no existing solver offers this collection of unique features.

Future work will explore support for additional cones, which naturally fit within CALIPSO’s interior-point framework, including semidefinite cones that are of interest in many control applications [32]—particularly settings with nonlinear dynamics. Additionally, extending the Julia implementation to C or C++ will potentially enable real-time performance of MPC policies onboard robots with limited computing hardware. The solver has potential to support state-triggered constraints online in safety-critical applications or be utilized in feedback loops for contact-implicit model-predictive control [26] with systems that make and break contact with their environments.

Acknowledgments

The authors would like to thank Shane Barratt for helpful discussions related to cone programming. This work was supported by Frontier Robotics, Innovative Research Excellence, Honda R&D Co., Ltd. and an Early Career Faculty Award from NASA’s Space Technology Research Grants Program (Grant Number 80NSSC21K1329).

References

- [1] Agrawal A, Barratt S, Boyd S, Busseti E, Moursi WM (2019) Differentiating through a cone program. arXiv preprint arXiv:190409043
- [2] Amos B, Kolter JZ (2017) OptNet: Differentiable optimization as a layer in neural networks. In: International Conference on Machine Learning, PMLR, pp 136–145
- [3] Argáez M, Tapia RA (2002) On the global convergence of a modified augmented Lagrangian linesearch interior-point Newton method for nonlinear programming. *Journal of Optimization Theory and Applications* 114(1):1–25
- [4] Bertsekas DP (2014) *Constrained optimization and Lagrange multiplier methods*. Academic Press
- [5] Betts JT (1998) Survey of numerical methods for trajectory optimization. *Journal of Guidance, Control, and Dynamics* 21(2):193–207
- [6] Biegler LT (2010) *Nonlinear programming: concepts, algorithms, and applications to chemical processes*. SIAM

- [7] Blackmore L, Açikmeşe B, Scharf DP (2010) Minimum-landing-error powered-descent guidance for Mars landing using convex optimization. *Journal of Guidance, Control, and Dynamics* 33(4):1161–1171
- [8] Bonalli R, Cauligi A, Bylard A, Pavone M (2019) GuSTO: Guaranteed sequential trajectory optimization via sequential convex programming. In: 2019 International Conference on Robotics and Automation (ICRA), IEEE, pp 6741–6747
- [9] Boyd S, Vandenberghe L (2004) *Convex optimization*. Cambridge University Press
- [10] Davis TA (2005) Algorithm 849: A concise sparse Cholesky factorization package. *ACM Transactions on Mathematical Software (TOMS)* 31(4):587–591
- [11] Dini U (1907) *Lezioni di analisi infinitesimale*, vol 1. Fratelli Nistri
- [12] Domahidi A, Chu E, Boyd S (2013) ECOS: An SOCP solver for embedded systems. In: 2013 European Control Conference (ECC), IEEE, pp 3071–3076
- [13] Domke J (2012) Generic methods for optimization-based modeling. In: *Artificial Intelligence and Statistics*, PMLR, pp 318–326
- [14] Gill PE, Robinson DP (2012) A primal-dual augmented Lagrangian. *Computational Optimization and Applications* 51(1):1–25
- [15] Gill PE, Murray W, Saunders MA (2005) SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Review* 47(1):99–131
- [16] Howell TA, Jackson BE, Manchester Z (2019) ALTRO: A fast solver for constrained trajectory optimization. In: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, pp 7674–7679
- [17] Howell TA, Le Cleac’h S, Singh S, Florence P, Manchester Z, Sindhvani V (2021) Trajectory optimization with optimization-based dynamics. arXiv preprint arXiv:210904928
- [18] Izmailov AF, Solodov MV, Uskov EI (2012) Global Convergence of Augmented Lagrangian Methods Applied to Optimization Problems with Degenerate Constraints, Including Problems with Complementarity Constraints. *SIAM Journal on Optimization* 22(4):1579–1606, DOI 10.1137/120868359, URL <http://epubs.siam.org/doi/10.1137/120868359>
- [19] Izmailov AF, Solodov MV, Uskov EI (2012) Global convergence of augmented Lagrangian methods applied to optimization problems with degenerate constraints, including problems with complementarity constraints. *SIAM Journal on Optimization* 22(4):1579–1606
- [20] Jackson BE, Punnoose T, Neamati D, Tracy K, Jitosh R, Manchester Z (2021) ALTRO-C: A fast solver for conic model-predictive control. In: 2021 IEEE International Conference on Robotics and Automation (ICRA), IEEE, pp 7357–7364
- [21] Jacobson DH, Mayne DQ (1970) *Differential dynamic programming*. 24, Elsevier Publishing Company
- [22] Jallet W, Bambade A, Mansard N, Carpentier J (2022) Constrained Differential Dynamic Programming: A primal-dual augmented Lagrangian approach, URL <https://hal.archives-ouvertes.fr/hal-03597630>, working paper or preprint
- [23] Jin W, Wang Z, Yang Z, Mou S (2020) Pontryagin differentiable programming: An end-to-end learning and control framework. *Advances in Neural Information Processing Systems* 33:7979–7992
- [24] Kuhlmann R, Büskens C (2018) A primal-dual augmented Lagrangian penalty-interior-point filter line search algorithm. *Mathematical Methods of Operations Research* 87(3):451–483
- [25] LaValle SM (2006) *Planning Algorithms*. Cambridge University Press
- [26] Le Cleac’h S, Howell TA, Schwager M, Manchester Z (2021) Fast contact-implicit model-predictive control. arXiv preprint arXiv:210705616
- [27] Li W, Todorov E (2004) Iterative linear quadratic regulator design for nonlinear biological movement systems. In: *ICINCO* (1), Citeseer, pp 222–229
- [28] Manchester Z, Kuindersma S (2020) Variational contact-implicit trajectory optimization. In: *Robotics Research*, Springer, pp 985–1000

- [29] Mastalli C, Budhiraja R, Merkt W, Saurel G, Hammoud B, Naveau M, Carpentier J, Righetti L, Vijayakumar S, Mansard N (2020) Crocodyl: An efficient and versatile framework for multi-contact optimal control. In: 2020 IEEE International Conference on Robotics and Automation (ICRA), IEEE, pp 2536–2542
- [30] Moreau JJ (2011) On unilateral constraints, friction and plasticity. In: *New Variational Techniques in Mathematical Physics*, Springer, pp 171–322
- [31] Nocedal J, Wright SJ (2006) *Numerical Optimization*, 2nd edn. Springer
- [32] Palan M, Barratt S, McCauley A, Sadigh D, Sindhvani V, Boyd S (2020) Fitting a linear control policy to demonstrations with a Kalman constraint. In: *Learning for Dynamics and Control*, PMLR, pp 374–383
- [33] Posa M, Cantu C, Tedrake R (2014) A direct method for trajectory optimization of rigid bodies through contact. *The International Journal of Robotics Research* 33(1):69–81
- [34] Raghunathan AU, Biegler LT (2003) Mathematical programs with equilibrium constraints (MPECs) in process engineering. *Computers & Chemical Engineering* 27(10):1381–1392
- [35] Scheel H, Scholtes S (2000) Mathematical programs with complementarity constraints: Stationarity, optimality, and sensitivity. *Mathematics of Operations Research* 25(1):1–22
- [36] Singh S, Slotine JJ, Sindhvani V (2021) Optimizing trajectories with closed-loop dynamic SQP. arXiv preprint arXiv:210907081
- [37] von Stryk O (1993) Numerical solution of optimal control problems by direct collocation. In: *Optimal Control*, Springer, pp 129–143
- [38] Szmuk M, Malyuta D, Reynolds TP, Mceowen MS, Acikmese B (2019) Real-Time Quad-Rotor Path Planning Using Convex Optimization and Compound State-Triggered Constraints. In: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, Macau, China, pp 7666–7673
- [39] Szmuk M, Reynolds TP, Acikmese B, Mesbahi M, Carson III JM (2019) Successive Convexification for 6-DoF Powered Descent Guidance with Compound State-Triggered Constraints. arXiv:190102181 URL <http://arxiv.org/abs/1901.02181>
- [40] Szmuk M, Reynolds TP, Açikmeşe B (2020) Successive Convexification for Real-Time Six-Degree-of-Freedom Powered Descent Guidance with State-Triggered Constraints. *Journal of Guidance, Control, and Dynamics* 43(8):1399–1413
- [41] Tassa Y, Mansard N, Todorov E (2014) Control-limited differential dynamic programming. In: 2014 IEEE International Conference on Robotics and Automation (ICRA), IEEE, pp 1168–1175
- [42] Tedrake R, the Drake Development Team (2019) Drake: Model-based design and verification for robotics. URL <https://drake.mit.edu>
- [43] Tu S, Frostig R (2022) Trajax: A Python library for differentiable optimal control on accelerators. URL <https://github.com/google/trajax>
- [44] Vandenberghe L (2010) The CVXOPT linear and quadratic cone program solvers. Online: <http://cvxopt.org/documentation/coneprogpdf>
- [45] Vanderbei RJ, Yurttan H (1998) Using LOQO to solve second-order cone programming problems. *Constraints* 1:2
- [46] Wächter A, Biegler LT (2005) Line search filter methods for nonlinear programming: Motivation and global convergence. *SIAM Journal on Optimization* 16(1):1–31
- [47] Wächter A, Biegler LT (2006) On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming* 106(1):25–57
- [48] Wang Y, Boyd S (2009) Fast model predictive control using online optimization. *IEEE Transactions on Control Systems Technology* 18(2):267–278
- [49] Zanelli A, Domahidi A, Jerez J, Morari M (2020) FORCES NLP: an efficient implementation of interior-point methods for multistage nonlinear nonconvex programs. *International Journal of Control* 93(1):13–29