Differentiable Convex Modeling for Robotic Planning and Control

Kevin Sledge Tracy

CMU-RI-TR-24-68

December 2024

School of Computer Science
The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee

Zachary Manchester, Chair	*Carnegie Mellon University*
J. Zico Kolter	*Carnegie Mellon University*
Changliu Liu	*Carnegie Mellon University*
Tom Erez	*Google DeepMind Robotics*

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Keywords: motion planning, trajectory optimization, convex optimization, collision detection, guidance, navigation, control

To Haley.

Abstract

Robotic simulation, planning, estimation, and control, have all been built on top of numerical optimization. In this same time, modern convex optimization has matured into a robust technology delivering globally optimal solutions in polynomial time. With advances in differentiable optimization and custom solvers capable of producing smooth derivatives, convex modeling has become fast, reliable, and fully differentiable. This thesis demonstrates the effectiveness of convex modeling in areas such as Martian atmospheric entry guidance, nanosatellite space telescope pointing, collision detection, contact dynamics of point clouds, online model learning, and finally, a derivative-free method for trajectory optimization that leverages modern parallelized simulation. In all of these domains, the reliability and speed of differentiable convex optimization enables real-time algorithms that are rigorous, performant, and easy to understand and modify.

Acknowledgments

I came to graduate school with no interest in doing research until I met my advisor Zac Manchester. At the time I was learning everything that I could about spacecraft guidance, navigation, and control, and I got a hold of Zac's lecture notes on attitude control. I spent an entire summer pouring over these notes and meeting with him every other Friday with a laundry list of questions. In doing this, I realized that time spent with Zac was the most valuable learning opportunity I would ever get.

Fast forward six years, a cross-country move, and a shift from aerospace to robotics, Zac has maintained this incredibly positive influence on me and my life. He has always been there to introduce me to new problems, help me when I get stuck, and talk me through the big decisions. I could have never asked for a more supportive and engaging advisor, and the impact Zac has had on me and the way I think is immeasurable.

I also feel very privileged to have been a part of the REx Lab during the early days. The time I spent with Brian Jackson and Taylor Howell in those first few years were foundational. I stay in close contact with these two about work and personal developments, and likely will for the rest of my life. After moving to CMU, our lab grew to include the awesome Alex Bouman, Chiyen Lee, Benj Jensen, Jacob Willis, JJ Lee, and Mitch Fogelson, all contributing to an amazing lab culture.

I was also fortunate enough to enjoy several great internships during my time as a graduate student. Drew Calhoun and the hypersonics team at Lockheed Martin, Roshena MacPherson and the GNC team at Astranis, Dan Morgan and the Starshield team at SpaceX, and finally Stefan Schaal with the research team at Intrinsic. During each of these opportunities, I was exposed to entirely new problem domains and ways of thinking that I carry with me to this day.

Most importantly, I could not have achieved any of this without the never-ending support from my family. My parents raised me with the freedom and the privilege to pursue whatever it was that interested me, and were always in my corner no matter where that took me. To my three siblings and my three best friends, each of you has continually supported and inspired me. I am so proud of my siblings and what they have accomplished that I am often embarrassed to tell people about them in detail.

And finally, I am eternally grateful to my wife Gabrielle for being there for me during this whole process. When we first started dating at 17, I don't think either of us would have guessed that I had 12 more years of school ahead of me. Nonetheless, she has been supportive, encouraging, and by my side as this journey has brought me all over the country.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

This dissertation introduces novel methods for robotic simulation, planning, and control based on differentiable convex modeling. By formulating algorithms in these domains in an optimization-first framework, we are often able to simplify the complexity of the algorithm and offload computational complexity to highly specialized and efficient solvers. This dissertation focuses on extending many of these methods with advances in modern convex modeling, where solvers are capable of delivering globally optimal solutions to convex optimization problems while maintaining full differentiability. Since these solvers are fast, robust, and differentiable, they can be treated the same as more traditional numerical linear algebra routines such as those used to solve linear systems [24]. Using convex modeling as a building block for novel algorithm development enables simple, performant, and flexible algorithms.

In the period between 1939 and 1948, Leonid Kantorovic, George Dantzig, and John Von Neumann, introduced the fundamental concepts in constrained optimization surrounding linear programming and duality [39]. Around the same time, the Karush–Kuhn–Tucker (KKT) conditions were established that gave clarity to what it means for a solution to a constrained optimization problem to be "optimal" [24]. While practictioners focused on implementing numerical optimization algorithms on newly introduced computers, theorists turned their attention towards a useful taxonomy for the world of constrained optimization. In 1983, Soviet scientists Nemirovski and Yudin made the first formal argument that there was, in fact, a substantial difference in the problem complexity between solving convex optimization problems and general nonlinear optimization problems [22]. The 1980s also saw the advent of robust interior-point methods that were capable of solving general convex optimization problems in polynomial time [116]. With advances in numerical linear algebra, algorithmic developments, and faster computers, by the 2000's convex optimization was mature and ready for everyday use. Modeling tools such as CVX [63], CVXPY [40] and Convex.jl [179] made convex optimization accessible by taking problems described in natural mathematical syntax and solving them with commercial and open source solvers. In the late 2010s, the works of [7] and [5] enabled the differentiation of well-defined convex optimization problems with respect to generic problem parameters. Today, convex optimization solvers can be treated as well-understood and reliable differentiable functions for

use in a wide range of applications.

In Chapter 4, a classical guidance framework for atmospheric entry dating back to the Apollo program is updated to incorporate convex optimization that results in state-of-the-art performance. The dynamics of an entry vehicle in an arbitrary atmosphere is discussed in 4.2 in a form that is less popular in the literature but more amenable to numerical optimization. From this, the novel Convex Predictor-corrector Entry Guidance (CPEG) algorithm for atmospheric guidance is detailed that is built on convex optimization in 4.4. The performance of CPEG is demonstrated on a realistic set of initial conditions, and convergence is then validated in 4.5.

Chapter 5 expands on Chapter 4 by introducing an updated variant of CPEG that is capable of directly reasoning about atmospheric uncertainty during entry. The guidance framework present in CPEG is enhanced with an estimator capable of atmospheric estimation and adjustments are made to the controller-estimator stack to enable robust real-time control of the vehicles. Results for this algorithm are shown on realistic Martian atmospheres and an ablation study validates the importance of the atmospheric adaptation.

Chapter 6 utilizes a convex optimization-based motion planner that enables precise control of nanosatellite space telescopes through actuation of weighted booms. Traditionally, spacecraft control pointing with onboard reaction wheels. These rotors are spun up and down to transfer angular momentum to and from the nanosatellite. Due to imperfections in these wheels, vibrations are transmitted to the telescope resulting in poor image quality. This chapter introduces a new actuation strategy based on slow-spinning long booms, a noted departure from the more common fast-spinning reaction wheels. Armed with the knowledge of the nanosatellite's orbit and the Earth's magnetic field, a convex motion planner is able to directly reason about future disturbances to enable precise attitude using just the booms. This technique is demonstrated on a precise space telescope that must take long exposure during eclipse periods without the booms hitting their actuation limits.

Chapter 7 presents a new approach for collision detection that enables a uniform framework and smooth differentiability. Traditionally, collision detection between two convex shapes is performed by solving for the closest point between the shapes. This problem is well defined and easy to solve for many of the common convex primitives, but is non-differentiable and is force to default to a different algorithm when the shapes are in contact. Our method for detecting these collisions utilizes a different framework: the minimum uniform scaling between the two shapes that results in an intersection is solved for with convex optimization. This problem is small, guaranteed to be well defined, and has no degenerate cases. The solution to the resulting convex optimization problem is smooth and differentiable no matter the configuration of the objects, and is used to specify collision avoidance constraints in representative motion planning examples.

Chapter 8 introduces an extension to Chapter 7 by exploiting the same framework for use in continuous collision detection. Discrete collision detection checks two static convex shapes for collisions, whereas continuous collision detection must reason about the same problem when the shapes are moving. Traditionally, the transition from discrete to continuous collision detection involves significant modifications to the algorithm with added limitations. Alternatively, the framework from Chapter 7 is augmented with a time parameter, and another well-defined convex optimization problem is used to solve for con-

tinuous collision information. Again, this method is fully differentiable and demonstrated on collision-free motion planning examples where discrete collision detection is insufficient for avoiding contact.

Chapter 9 details a simple and efficient framework for learning generalized linear models online as it relates to connector insertion. In this application, the insertion of a male connector into a female socket presents a challenging environment to model in simulation due to the deformation of the materials, the tightness of the fit, and the unknown friction properties. To avoid learning a full dynamics simulator, this work simply learns the relationship between the control signal, the estimated state, and a wrist-mounted force torque sensor. This relationship is represented with a generalized linear model, making the model learning problem convex and well defined. To solve this problem online in a recursive fashion, the Linear Model Learning (LML) algorithm is developed to learn the globally optimal estimated model using only matrix-vector operations. In the absence of any matrix inversions, this algorithm scales very favorably on a GPU, enabling fast and efficient model learning even with large-scale linear models.

Chapter 10 examines the performance of quasistatic simulation on rigid point clouds, and extends a common formulation to include the necessary torsional friction for realistic contact-rich manipulation. When contact with a rigid point cloud is made in rigid-body dynamics, the single point contact does not allow for the true torsional friction that comes from the type of patch contacts present in reality. To account for this, a modification is made to an optimization-based simulation framework to naturally account for this with a torsional friction term that is proportional to the normal force. The resulting simulation steps are computed with convex optimization and are therefore fully differentiable. The resulting simulator is used to grasp high-fidelity point clouds with realistic contact dynamics.

Chapter 11 introduces the trajectory bundle method for highly accurate trajectory optimization with black-box simulators, costs, and constraints. Existing methods for model-based trajectory optimization have access to derivatives of all cost, dynamics, and constraint functions presented in the problem. In many cases, this is a reasonable assumption, but for many challenging robotics tasks such as those involving nonsmooth contact interactions, these derivatives can be unavailable, expensive to compute, or unreliable. To combat this, the trajectory bundle method approximates these functions by interpolating samples centered on the current iterate. This method of approximation by interpolation is derivative-free and results in a linearization different from a standard first-order Taylor-series. Using this approximation, a convex optimization problem is posed that minimizes over these interpolants to compute a step direction, and the process is repeated until convergence.

With differentiable convex optimization as a robust technology, this dissertation is able to contribute simple and performant algorithms to multiple domains within robotics with guarantees of predictable and safe performance. By reformulating these problems in an optimization-first way, the resulting algorithms are often simpler and more performant than without, all while being modular and configurable.

1.1 Summary of publications

The content of Chapter 3 appears in:

> [176] Kevin Tracy and Zachary Manchester. *On the Differentiability of the Primal-Dual Interior-Point Method*. June 2024. arXiv: 2406.11749 [math]. (Visited on 11/11/2024)

The content of Chapter 4 appears in:

> [173] Kevin Tracy and Zachary Manchester. "CPEG: A Convex Predictor-corrector Entry Guidance Algorithm". In: *2022 IEEE Aerospace Conference (AERO)*. Big Sky, MT, USA: IEEE, Mar. 2022, pp. 1–10. ISBN: 978-1-66543-760-8. DOI: 10.1109/AERO53065.2022.9843641. (Visited on 09/13/2022)

The content of Chapter 5 appears in:

> [168] Kevin Tracy, Giusy Falcone, and Zachary Manchester. "Robust Entry Guidance with Atmospheric Adaptation". In: *AIAA SciTech Forum and Exposition*. National Harbor, Maryland: AIAA, Jan. 2023

The content of Chapter 6 appears in:

> [177] Kevin Tracy, Zachary Manchester, and Ewan Douglas. "Ultra-Fine Pointing for Nanosatellite Telescopes With Actuated Booms". In: *2022 IEEE Aerospace Conference (AERO)*. Big Sky, MT, USA: IEEE, Mar. 2022, pp. 1–8. ISBN: 978-1-66543-760-8. DOI: 10.1109/AERO53065.2022.9843485. (Visited on 09/13/2022)

The content of Chapter 7 appears in:

> [170] Kevin Tracy, Taylor A. Howell, and Zachary Manchester. "Differentiable Collision Detection for a Set of Convex Primitives". In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. London, United Kingdom: IEEE, May 2023, pp. 3663–3670. DOI: 10.1109/ICRA48891.2023.10160716. (Visited on 03/08/2024)

The content of Chapter 9 appears in:

> [178] Kevin Tracy, Zachary Manchester, Ajinkya Jain, Keegan Go, Stefan Schaal, Tom Erez, and Yuval Tassa. *Efficient Online Learning of Contact Force Models for Connector Insertion*. Dec. 2023. arXiv: 2312.09190 [cs]. (Visited on 06/12/2024)

The content of Chapter 10 appears in:

> [172] Kevin Tracy and Zachary Manchester. "Convex Quasi-Dynamic Simulation of Rigid Point Clouds with Torsional Friction". In: *IROS 2023 Workshop on Leveraging Models for Contact-Rich Manipulation*. Detroit, MI, USA: IEEE, Oct. 2023

Non-thesis research: During my Ph.D., I have conducted research on topics that do not appear in this thesis. Publications involving these topics are below.

> [175] Kevin Tracy and Zachary Manchester. "Model-Predictive Attitude Control for Flexible Spacecraft During Thruster Firings". In: *AAS/AIAA Astrodynamics Specialist Conference*. Lake Tahoe, CA, Aug. 2020
>
> [174] Kevin Tracy and Zachary Manchester. "Low-Thrust Trajectory Optimization Using the Kustaanheimo-Stiefel Transformation". In: *AAS/AIAA Space Flight Mechanics Meeting*. Charlotte, NC, Feb. 2021
>
> [167] Kevin Tracy. *A Square-Root Kalman Filter Using Only QR Decompositions*. Aug. 2022. arXiv: 2208.06452 [cs, eess]. (Visited on 12/03/2022)
>
> [171] Kevin Tracy, Taylor A. Howell, and Zachary Manchester. "DiffPills: Differentiable Collision Detection for Capsules and Padded Polygons". In: *http://arxiv.org/abs/2207.00202* (July 2022). DOI: 10.48550/arXiv.2207.00202. arXiv: 2207.00202. (Visited on 09/12/2022)

I have also been fortunate enough to work on the following publications alongside my collaborators as a non-primary author.

> Ewan S. Douglas, Kevin Tracy, and Zachary Manchester. "Practical Limits on Nanosatellite Telescope Pointing: The Impact of Disturbances and Photon Noise". In: *Frontiers in Astronomy and Space Sciences* 8 (Aug. 2021), p. 676252. ISSN: 2296-987X. DOI: 10.3389/fspas.2021.676252. (Visited on 08/25/2021)
>
> Brian E Jackson, Tarun Punnoose, Daniel Neamati, Kevin Tracy, Rianna Jitosho, and Zachary Manchester. "ALTRO-C: A Fast Solver for Conic Model-Predictive Control; ALTRO-C: A Fast Solver for Conic Model-Predictive Control". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)* (2021). ISSN: 9781728190778. DOI: 10.1109/ICRA48506.2021.9561438
>
> Brian E. Jackson, Kevin Tracy, and Zachary Manchester. "Planning With Attitude". In: *IEEE Robotics and Automation Letters* (2021), pp. 1–1. ISSN:

2377-3766, 2377-3774. DOI: 10.1109/LRA.2021.3052431. (Visited on 08/25/2021)

Max Holliday, Kevin Tracy, Zachary Manchester, and Anh Nguyen. "The V-R3x Mission: Towards Autonomous Networking and Navigation for CubeSat Swarms". In: *4S Symposium*. Vilamoura, Portugal, May 22

Arun L. Bishop, John Z. Zhang, Swaminathan Gurumurthy, Kevin Tracy, and Zachary Manchester. "ReLU-QP: A GPU-Accelerated Quadratic Programming Solver for Model-Predictive Control". In: *International Conference on Robotics and Automation (ICRA)*. Yokohama, Japan, 2024. (Visited on 02/13/2024)

Chapter 2

Background

This chapter provides a thorough description of constrained optimization as it relates to the ideas introduced in this dissertation. Standard formulations for general nonlinear programs, as well as a range of convex optimization problems are detailed. Methods for differentiating these problems is then introduced, followed by an algorithmic description for the primal-dual interior-point solver that much of this dissertation is built upon.

2.1 Constrained Optimization

A mathematical optimization problem can be defined as the following:

$$\begin{array}{ll} \underset{x}{\text{minimize}} & f(x) \\ \text{subject to} & g(x) = 0, \\ & h(x) \leq 0 \end{array} \tag{2.1}$$

where the primal variable being optimized is $x \in \mathbf{R}^n$, the objective function to be minimized is $f : \mathbf{R}^n \to \mathbf{R}$, and the constraint functions $g : \mathbf{R}^n \to \mathbf{R}^m$ and $h : \mathbf{R}^n \to \mathbf{R}^p$ dictate the feasible values for x. If there is no solution x that satisfies the constraints, the problem is said to be infeasible.

2.1.1 Optimization Taxonomy

Based on the nature of the functions present in (2.1), solving for the globally optimal solution to a constrained optimization problem ranges from trivial to impossible. In order to make these distinctions, a rough taxonomy of constrained optimization is presented.

Linear Programs

If functions f, g, and h are all linear functions, problem (2.1) is said to be a Linear Program (LP). Linear programs can be solved for their global optimums (if one exists) in polynomial

time, with robust and mature numerical routines. The original and most notable method for solving LPs is the Simplex method, which solves LPs of the following form:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & c^T x \\ \text{subject to} \quad & Ax = b, \\ & x \geq 0 \end{aligned} \tag{2.2}$$

Any LP can be cast into this standard form, though this process often involves the introduction of slack variables.

Convex Optimization

If the functions $f(x)$ and $h(x)$ are convex, and $g(x)$ is affine, then problem (2.1) is a convex optimization problem. LPs denote a subset of convex optimization problems. This is a powerful distinction to make, since convex optimization problems have a set of qualities that make them very desirable in practice:

- Any locally optimal solution in a convex optimization problem is the globally optimal solution. This means that any solution that satisfies the first-order necessary conditions shown in 2.1.2 is a globally optimal solution.
- If the constraints in the problem are such that there are no feasible solutions, a solver can identify this and provide a certificate of infeasibility.
- Convex optimization solvers do not require an initial guess for the primal or dual variables. For some solvers, an initial guess can be provided (this is called "warm starting"), but this is not required to find the globally optimal solution.
- Convex optimization problems can be solved in polynomial time.

These qualities that make convex optimization attractive for real-time optimization where reliability is important.

Nonlinear Programs

Technically if any of the functions f, g, and h in problem (2.1) are nonlinear, then the optimization problem is considered to be a NonLinear Program (NLP). In the optimization literature, convex optimization (which can contain nonlinear functions) is typically excluded from this, and NLPs are usually referring to nonconvex problems. Unlike convex optimization, NLPs have the potential to be impossible to solve, and very little can be guaranteed about the performance on a particular problem. Also unlike convex optimization, the initialization of the solver is critical to the performance and even success of the solve. In the best case, NLP solvers are able to converge to a locally optimal solution, but are unable to guarantee anything about global optimality.

While solving NLPs require more complexity and risk, they are still very common in practice and, with careful engineering, can be an appropriate solution. Many common optimization-based algorithms in robotic perception, control, and planning utilize NLPs with success.

2.1.2 Optimality Conditions

The first-order necessary conditions for optimality, also known as the Karush–Kuhn–Tucker (KKT) conditions, are able to certify a solution as locally optimal. By introducing dual variables $\lambda \in \mathbf{R}^m$ for the equality constraints and and $\mu \in \mathbf{R}^p$ for the inequality constraints, these optimality conditions can be described as the following:

$$\nabla_x f(x) + \left[\frac{\partial g}{\partial x}\right]^T \lambda + \left[\frac{\partial h}{\partial x}\right]^T \mu = 0 \qquad \text{stationarity,} \tag{2.3}$$

$$g(x) = 0 \qquad \text{primal feasibilit,y} \tag{2.4}$$

$$h(x) \leq 0 \qquad \text{primal feasibility,} \tag{2.5}$$

$$\mu \geq 0 \qquad \text{dual feasibility,} \tag{2.6}$$

$$\mu \circ h(x) = 0 \qquad \text{complementarity,} \tag{2.7}$$

where $\circ$ denotes element-wise multiplication. If a primal-dual solution (x^*, λ^*, μ^*) satisfies the KKT conditions, it satisfies the first-order necessary conditions for local optimality. These optimality conditions are useful for both certifying solutions and deriving algorithms to get to these conditions.

2.2 Primal-Dual Interior-Point Method

One of the most common convex optimization problems is the Quadratic Program (QP). Similar to an LP, QPs have linear equality and inequality constraints, but are capable of including both linear and quadratic terms in the objective function. There is less of an agreement on a "standard form" for a quadratic program, but here is a popular description of a QP:

$$\begin{array}{ll} \underset{x}{\text{minimize}} & \frac{1}{2}x^T Q x + q^T x \\ \text{subject to} & Ax = b, \\ & Gx \leq h \end{array} \tag{2.8}$$

$$\Rightarrow$$

$$\begin{array}{ll} \underset{x,\, s}{\text{minimize}} & \frac{1}{2}x^T Q x + q^T x \\ \text{subject to} & Ax = b, \\ & Gx + s = h, \\ & s \geq 0 \end{array} \tag{2.9}$$

where $Q \in \mathbf{S}^n_+$ is a positive semidefinite matrix, $q \in \mathbf{R}^n$ is the linear cost term, $A \in \mathbf{R}^{m \times n}$ and $b \in \mathbf{R}^m$ describe the equality constraint and $G \in \mathbf{R}^{p \times n}$ and $h \in \mathbf{R}^p$ describe the inequality constraint. The standard form in (2.8) is often converted to the equivalent form in (2.9) with the addition of a nonnegative slack variable $s \in \mathbf{R}^p$.

With the introduction of dual variables $z \in \mathbf{R}^p$ for the inequality constraint and $y \in \mathbf{R}^m$

for the equality constraint, the KKT conditions for (2.9) are the following:

$$Qx + q + G^T z + A^T y = 0, \tag{2.10}$$
$$z \circ s = 0, \tag{2.11}$$
$$Gx + s = h, \tag{2.12}$$
$$Ax = b, \tag{2.13}$$
$$s \geq 0, \tag{2.14}$$
$$z \geq 0, \tag{2.15}$$

2.2.1 Predictor-corrector Algorithm

Primal-dual interior-point methods are capable of solving small to medium QPs in a fast and robust fashion. While there are many variations, presented here is a standard Mehrotra predictor-corrector primal-dual interior-point method that has been detailed in [116, 183, 41, 127, 9, 126, 115]. This algorithm is effectively treating the KKT conditions of (2.10)-(2.13) and solving it as a rootfinding problem with Newton's Method. The challenges here come from the nonnegativity constraints in (2.14) and (2.15), and the nonlinearity present in the complementarity condition (2.11). To deal with these, a predictor-corrector method is used where two different step directions are computed: an affine step direction that is calculated with a standard Newton step, and a centering-plus-corrector step that modifies this step direction to account for the nonnegativity constraints in (2.14) and (2.15), and the nonlinearity present in the complementarity condition (2.11). This is an iterative algorithmn, and a single iteration will be detailed below.

Affine Step

The algorithm can be initialized with any value of x and y, so long as $s, z > 0$. From this, the residual function $r : \mathbf{R}^{n+2p+m} \to \mathbf{R}^{n+2p+m}$ is defined as the following:

$$r\left(\begin{bmatrix} x \\ s \\ z \\ y \end{bmatrix}\right) = \begin{bmatrix} Qx + q + A^T y + G^T z \\ s \circ z \\ Gx + s - h \\ Ax - b \end{bmatrix}. \tag{2.16}$$

By solving for the root of the linearized residual function at an iterate (x, s, z, y), a standard Newton step is used to calculate the "affine" step direction:

$$\begin{bmatrix} Q & 0 & G^T & A^T \\ 0 & D(z) & D(s) & 0 \\ G & I & 0 & 0 \\ A & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta x^{\text{aff}} \\ \Delta s^{\text{aff}} \\ \Delta z^{\text{aff}} \\ \Delta y^{\text{aff}} \end{bmatrix} = \begin{bmatrix} -\left(Qx + q + A^T y + G^T z\right) \\ -s \circ z \\ -(Gx + s - h) \\ -(Ax - b) \end{bmatrix}, \tag{2.17}$$

where $D(\cdot)$ creates a diagonal matrix from a vector.

Centering-plus-corrector Step

In order to form the centering-plug-corrector step, a linesearch that guarantees nonnegativity is defined. To detail this, an arbitrary variable $w > 0$ and step direction Δw are considered. The largest step length $\alpha \in [0, 1]$ that ensures $w + \Delta w \geq 0$ can be written as the following:

$$\text{linesearch}(w, \Delta w) = \sup\{\alpha \in [0, 1] \mid w + \alpha \Delta w \geq 0\}, \tag{2.18}$$

which can be solved for in closed form:

$$\text{linesearch}(w, \Delta w) = \min\left(1, \min_{i:\Delta w_i < 0} -\frac{w_i}{\Delta w_i}\right). \tag{2.19}$$

Using this, the affine step size is computed with the minimum of the maximum step sizes for both s and z,

$$\alpha^{\text{aff}} = \min(\text{linesearch}(s, \Delta s^{\text{aff}}), \text{linesearch}(z, \Delta z^{\text{aff}})). \tag{2.20}$$

The centering-plus-corrector step can then be solved for with another linear system:

$$\begin{bmatrix} Q & 0 & G^T & A^T \\ 0 & D(z) & D(s) & 0 \\ G & I & 0 & 0 \\ A & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta x^{\text{cc}} \\ \Delta s^{\text{cc}} \\ \Delta z^{\text{cc}} \\ \Delta y^{\text{cc}} \end{bmatrix} = \begin{bmatrix} 0 \\ \sigma\mu\mathbf{1} - \Delta s^{\text{aff}} \circ \Delta z^{\text{aff}} \\ 0 \\ 0 \end{bmatrix}, \tag{2.21}$$

where $\mu = s^T z / p$ and

$$\sigma = \left(\frac{\left(s + \alpha \Delta s^{\text{aff}}\right)^T \left(z + \alpha \Delta z^{\text{aff}}\right)}{s^T z}\right)^3. \tag{2.22}$$

From here, these two step directions are combined into a single step direction,

$$(\Delta x, \Delta s, \Delta z, \Delta y) = (\Delta x^{\text{aff}}, \Delta s^{\text{aff}}, \Delta z^{\text{aff}}, \Delta y^{\text{aff}}) + (\Delta x^{\text{cc}}, \Delta s^{\text{cc}}, \Delta z^{\text{cc}}, \Delta y^{\text{cc}}), \tag{2.23}$$

And a linesearch is used to ensure the positivity (not just nonnegatvity) of s and z with

$$\alpha = \min\left(1, 0.99 \min(\text{linesearch}(s, \Delta s), \text{linesearch}(z, \Delta z)\right). \tag{2.24}$$

The iteration is then concluded with an update to the primal-dual values with:

$$(x, s, z, y) = (x, s, z, y) + \alpha(\Delta x, \Delta s, \Delta z, \Delta y). \tag{2.25}$$

Algorithm 1 Primal-Dual Interior-Point Method

function solve_qp(Q, q, A, b, G, h)
$x, s, z \leftarrow$ initialize(Q, q, A, b, G, h)
for $i \leftarrow 1$: `max_iters` **do**
    `/* evaluate KKT conditions*/`
    $v_1 \leftarrow Qx + q + A^T y + G^T z$
    $v_2 \leftarrow s \circ z$
    $v_3 \leftarrow Gx + s - h$
    $v_4 \leftarrow Ax - b$

    `/* check convergence */`
    if $\min(\|v_1\|_\infty, \|v_2\|_\infty, \|v_3\|_\infty, \|v_4\|_\infty) <$ `tol` **then**
        return: x, s, z, y
    end if

    `/* calculate affine step direction */`
    $(\Delta x^{\text{aff}}, \Delta s^{\text{aff}}, \Delta z^{\text{aff}}, \Delta y^{\text{aff}}) \leftarrow$ solve_step$(-v_1, -v_2, -v_3, -v_4)$ ▷ 2.2.2

    `/* calculate centering-plus-corrector step direction */`
    $\alpha^{\text{aff}} = \min(\text{linesearch}(s, \Delta s^{\text{aff}}),\ \text{linesearch}(z, \Delta z^{\text{aff}}))$ ▷ (2.19)
    $\mu \leftarrow s^T z / \text{len}(s)$
    $\sigma \leftarrow [(s + \alpha^{\text{aff}} \Delta s^{\text{aff}})^T (z + \alpha^{\text{aff}} \Delta z^{\text{aff}})/(s^T z)]^3$
    $r_2 \leftarrow \sigma\mu \mathbf{1} - \Delta s^{\text{aff}} \circ \Delta z^{\text{aff}}$
    $(\Delta x^{\text{aff}}, \Delta s^{\text{aff}}, \Delta z^{\text{aff}}, \Delta y^{\text{aff}}) \leftarrow$ solve_step$(0^n, r_2, 0^p, 0^m)$ ▷ 2.2.2

    `/* combine step directions and perform linesearch */`
    $(\Delta x, \Delta s, \Delta y, \Delta z) = (\Delta x^{\text{aff}}, \Delta s^{\text{aff}}, \Delta y^{\text{aff}}, \Delta z^{\text{aff}}) + (\Delta x^{\text{cc}}, \Delta s^{\text{cc}}, \Delta y^{\text{cc}}, \Delta z^{\text{cc}})$
    $\alpha \leftarrow \min\left(1, 0.99 \min(\text{linesearch}(s, \Delta s),\ \text{linesearch}(z, \Delta z)\right)$ ▷ (2.19)
    $(x, s, z, y) \leftarrow (x, s, z, y) + \alpha(\Delta x, \Delta s, \Delta z, \Delta y)$
end for

2.2.2 Solving the Linear System

The majority of the computational effort in Alg. 1 is in solving the linear systems in (2.17) and (2.21). Both of these linear systems have the same matrix, sharing the general form:

$$\begin{bmatrix} Q & 0 & G^T & A^T \\ 0 & D(z) & D(s) & 0 \\ G & I & 0 & 0 \\ A & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta s \\ \Delta z \\ \Delta y \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}. \tag{2.26}$$

The matrix in this linear system changes each iteration as s and z are updated, but are the same between the affine step computation and the centering-plus-correction step computation. This means the matrix only has to be factorized once at each iteration. How this is done is dependent on the size and sparsity of the matrix.

Sparse Options

For large problems where any of Q, G, or A are dense, it often makes sense to solve the linear system in (2.26) with a sparse direct solver. The first option is to simply factorize the matrix in (2.26) with a sparse LU factorization and solve it as it is written.

Alternatively, if the second row of (2.26) is left-multiplied by $D(s)^{-1}$, the linear system takes this form:

$$\begin{bmatrix} Q & 0 & G^T & A^T \\ 0 & D(s)^{-1}D(z) & I & 0 \\ G & I & 0 & 0 \\ A & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta s \\ \Delta z \\ \Delta y \end{bmatrix} = \begin{bmatrix} u_1 \\ D(s)^{-1}u_2 \\ u_3 \\ u_4 \end{bmatrix}, \tag{2.27}$$

which is a symmetric quasisemidefinite matrix, meaning the upper left hand block is symmetric positive semidefinite and the lower right hand block is symmetric negative semidefinite. This matrix can be regularized with a positive regularizer on the top left block and a negative regularizer on the bottom right block and the regularized matrix can be factorized with a sparse LDL decomposition [115]. The solution to the linear system in (2.27) can be recovered with iterative refinement, which involves only a few more backsolves with the already factorized linear system. For large sparse systems, the cost of an LU decomposition of (2.26) can be higher than that of an LDL decomposition of (2.27) followed by iterative refinement.

Dense Option

For small dense problems, the most efficient method of solving the linear system is done with a blockwise elimination. The majority of the computational cost in this scenario is performing two Cholesky decompositions at each iteration. Unlike the symmetrized sparse option, there is no regularization or iterative refinement required. This technique is demonstrated in 3.

2.3 Differentiable Optimization

With the advent of automatic differentiation tools, forming derivatives through complex routines is easier than ever. When one of these routines includes an iterative routine, one solution is to unroll the iterations into a computational graph and differentiate it as if it were just a normal function. For routines with a limited number of iterations, this solution may be appropriate. That being said, there are a few reasons why this approach could fail. The first reason is that sometimes the iterations involve non-smooth operations or logical branching that stop the "flow" of gradients, or are otherwise nondifferentiable. The second reason involves the numerical conditioning through procedures with many iterations, where each iteration through which the derivatives are propagated results in a loss of numerical precision.

2.3.1 Implicit Function Theorem

When an iterative routine can be interpreted as finding a root or an equilibrium point to an implicit function, the implicit function theorem can be used to form the required derivatives. Given variables $y \in \mathbf{R}^a$ and $\theta \in \mathbf{R}^b$, an implicit function is defined as:

$$r(y^*, \theta) = 0, \tag{2.28}$$

at an equilibrium point y^*. This implicit function can be linearized about the equilibrium point resulting in the following:

$$\frac{\partial r}{\partial y}\delta y + \frac{\partial r}{\partial \theta}\delta\theta \approx 0, \tag{2.29}$$

which can simply be re-arranged to solve for the Jacobian of y with respect to θ:

$$\frac{\partial y}{\partial \theta} = -\left(\frac{\partial r}{\partial y}\right)^{-1}\frac{\partial r}{\partial \theta}. \tag{2.30}$$

This enables the differentiation of routines that solve for equilibrium points without the need for unrolling the iterations themselves.

2.3.2 Full Jacobians

To apply the implicit function theorem to optimization, a parametrized optimization problem of the following form will be considered:

$$\begin{array}{ll} \underset{x}{\text{minimize}} & f_\theta(x) \\ \text{subject to} & h_\theta(x) \leq 0 \end{array} \tag{2.31}$$

where $x \in \mathbf{R}^n$ is the primal variables, $\mu \in \mathbf{R}^p$ is the dual variable for the inequality constraints, and $\theta \in \mathbf{R}^b$ are the problem parameters that are used to construct the cost or

constraint functions. The KKT conditions for this problem are the following:

$$\nabla_x f_\theta(x) + \left[\frac{\partial h_\theta}{\partial x}\right]^T \mu = 0, \tag{2.32}$$

$$\mu \circ h_\theta(x) = 0, \tag{2.33}$$

$$h_\theta(x) \leq 0, \tag{2.34}$$

$$\mu \geq 0. \tag{2.35}$$

A numerical solver for problem (2.31) will solve for a primal-dual solution (x^*, μ^*) that satisfies the KKT conditions. By treating these primal-dual variables as a single variable $y = [x^T, \mu^T]^T$, the conditions in (2.32) and (2.33) can be viewed as an implicit function:

$$r(y^*, \theta) = \begin{bmatrix} \nabla_x f_\theta(x) + \left[\frac{\partial h_\theta}{\partial x}\right]^T \mu \\ \mu \circ h_\theta(x) \end{bmatrix} \approx 0, \tag{2.36}$$

where y^* contains the approximate primal-dual solution. Using the implicit function theorem as shown in (2.30), the derivatives of the optimal primal-dual variables can be constructed with the following:

$$\frac{\partial y}{\partial \theta} = -\left(\frac{\partial r}{\partial y}\right)^{-1} \frac{\partial r}{\partial \theta}, \tag{2.37}$$

$$\frac{\partial y}{\partial \theta} = \begin{bmatrix} \nabla_x^2 f_\theta(x^*) + \partial\left(\left[\frac{\partial h_\theta}{\partial x^*}\right]^T \mu^*\right)/\partial x & \left[\frac{\partial h_\theta}{\partial x^*}\right]^T \\ D(\mu^*)\frac{\partial h_\theta}{\partial x} & D(h_\theta(x^*)) \end{bmatrix}^{-1} \frac{\partial r}{\partial \theta}, \tag{2.38}$$

thus giving the Jacobian of the primal and dual variables with respect to the problem parameters. In the context of backward mode automatic differentiation, [7] details a way to form these gradients as Jacobian vector products without explicitly forming any large Jacobians.

2.3.3 Objective Value Gradient

In the case where the only derivative of interest is that of the optimal objective value $f_\theta(x^*)$ with respect to the problem parameters, there is a more efficient way of computing this gradient without the implicit function theorem. Consider the Lagrangian of (2.31),

$$\mathcal{L}_\theta(x, \mu) = f_\theta(x) + mu^T h_\theta(x). \tag{2.39}$$

As shown in [33], the gradient of the optimal objective value with respect to the problem parameters is simply the gradient of the Lagrangian at a primal-dual solution with respect to the problem parameters:

$$\nabla_\theta f_\theta(x^*, \mu^*) = \nabla_\theta \mathcal{L}_\theta(x^*, \mu^*), \tag{2.40}$$

resulting in a fast and efficient way to compute this gradient without the need for the implicit function theorem or a linear system solve.

Chapter 3

On the Differentiability of the Primal-Dual Interior-Point Method

Primal-Dual Interior-Point methods are capable of solving constrained convex optimization problems to tight tolerances in a fast and robust manner. The derivatives of the primal-dual solution with respect to the problem matrices can be computed using the implicit function theorem, enabling efficient differentiation of these optimizers for a fraction of the cost of the total solution time. In the presence of active inequality constraints, this technique is only capable of providing discontinuous subgradients that present a challenge to algorithms that rely on the smoothness of these derivatives. This paper presents a technique for relaxing primal-dual solutions with a logarithmic barrier to provide smooth derivatives near active inequality constraints, with the ability to specify a uniform and consistent amount of smoothing. We pair this with an efficient primal-dual interior-point algorithm for solving an always-feasible ℓ_1-penalized variant of a convex quadratic program, eliminating the issues surrounding learning potentially infeasible problems. This parallelizable and smoothly differentiable solver is demonstrated on a range of robotics tasks where smoothing is important. An open source implementation in JAX is available at `www.github.com/kevin-tracy/qpax`.

The contents of this chapter are on arXiv: Tracy and Manchester [176].

3.1 Introduction

Convex optimization has seen widespread use in modern robotics, where the guarantees of global optimality and polynomial time complexity have enabled algorithms that span control [94, 21], state estimation [184, 197], actuator allocation [178, 91], collision detection [59, 170], and simulation [10, 133].

For years, practitioners have exploited domain-specific knowledge to craft convex optimization problems that enjoy fast and reliable convergence for both offline and online use. In the era of data-driven robotics, differentiable optimization has enabled automatic tuning or "learning" of optimization problems directly from data. Although this may not

replace domain-specific knowledge, the ability to build complex tuneable functions with embedded optimization problems is well suited for a variety of tasks.

The sensitivity analysis of linear programs has been studied for decades [24], but recent advances in differentiable convex optimization and parametrized convex layers in deep networks are gaining traction [8, 4, 5]. This has resulted in convex modeling tools such as CVXPY layers that enable easy incorporation of differentiable convex optimization into common workflows [40].

There are two main issues preventing the widespread adoption of differentiable optimization in robotics as it exists today: the subgradient problem and the infeasibility problem. For active inequality constraints in optimization problems, the derivatives propagated through these solvers are restricted to subgradients when there is no uniquely defined gradient. This results in routines getting "stuck" near these inequalities due to their nonsmooth nature. The second issue arises when infeasible problem instances are created during auto-tuning. While infeasibility detection is commonplace in convex optimization, detecting an infeasible problem does not address the root cause of the infeasibility, nor provide informative gradient information to encourage feasibility.

In this paper, we propose solutions to both of these problems. Our contributions include:

- A rigorous method for returning unique and smoothed derivatives of convex optimization problems through the use of a relaxed logarithmic barrier
- A primal-dual interior-point algorithm for solving an always-feasible "elastic" quadratic program with minimal computational overhead

Together, these two advances are presented in an open source software package written in JAX and demonstrated on the relevant robotics tasks of contact dynamics and collision detection.

3.2 Background

This section introduces the notation for a standard form of the convex Quadratic Program (QP), common solution methods, and extensions to differentiable optimization.

3.2.1 Quadratic Programming

In this paper, we focus on the convex QP as a fundamental problem specification. Modern algorithms such as the Primal-Dual Interior-Point (PDIP) method solve these problems globally in a fast and efficient manner [115]. A standard form quadratic program and its equivalent with a slack variable are as follows:

$$\begin{array}{ll}\underset{x}{\text{minimize}} & \frac{1}{2}x^TQx + q^Tx \\ \text{subject to} & Ax = b, \\ & Gx \leq h,\end{array} \tag{3.1}$$

$$\Rightarrow$$

$$\begin{array}{ll}\underset{x,\,s}{\text{minimize}} & \frac{1}{2}x^TQx + q^Tx \\ \text{subject to} & Ax = b, \\ & Gx + s = h, \\ & s \geq 0,\end{array} \tag{3.2}$$

with a primal variable $x \in \mathbf{R}^n$, cost terms $Q \in \mathbf{S}^n_+$ and $q \in \mathbf{R}^n$, equality constraints described with $A \in \mathbf{R}^{m\times n}$ and $b \in \mathbf{R}^m$, and inequality constraints with $G \in \mathbf{R}^{p\times n}$ and $h \in \mathbf{R}^p$. Dual variables are introduced to enforce the constraints, with $y \in \mathbf{R}^m$ associated with the equality constraint, and $z \in \mathbf{R}^p$ with the inequality constraints [24]. The slack variable $s \in \mathbf{R}^p$ is introduced for algorithmic simplicity. The Lagrangian for this problem is then

$$\mathcal{L}(x, s, z, y) = \frac{1}{2}x^TQx + q^Tx + y^T(Ax - b) + z^T(Gx - h), \tag{3.3}$$

resulting in the following KKT conditions for optimality:

$$Qx + q + G^Tz + A^Ty = 0, \tag{3.4}$$
$$z \odot s = 0, \tag{3.5}$$
$$Gx + s = h, \tag{3.6}$$
$$Ax = b, \tag{3.7}$$
$$s, z \geq 0, \tag{3.8}$$

where $\odot$ denotes elementwise multiplication, and $\oslash$ denotes elementwise division. A primal-dual solution (x^*, s^*, y^*, z^*) is globally optimal if it satisfies (3.4)-(3.8).

3.2.2 Primal-Dual Interior-Point Methods

PDIP methods solve (3.2) by treating a modified version of the system of equations in (3.4)-(3.7) as a root finding problem, and then using Newton's method to find a solution while restricting $(s, z) > 0$. As a result, the majority of the computation time is spent solving linear systems of the following form:

$$\begin{bmatrix} Q & 0 & G^T & A^T \\ 0 & D(z) & D(s) & 0 \\ G & I & 0 & 0 \\ A & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta s \\ \Delta z \\ \Delta y \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}, \tag{3.9}$$

where $D(\cdot)$ denotes the diagonal matrix constructor from a vector. Using block reduction techniques, the linear system in (3.9) can be efficiently solved with Alg. (2). This technique for solving linear systems of this form is useful for both the solving and differentiation of the PDIP method.

After the step directions are computed with the linear system, a linesearch is used to ensure the nonnegativity of (s, z) . For an arbitrary variable v and step direction Δv, a linesearch that solves for the largest $\alpha \leq 1$ that keeps $v + \Delta v \geq 0$ is solved in closed form:

$$\alpha = \text{linesearch}(v, \Delta v) = \min\left(1, \min_{i:\Delta v_i<0} -\frac{v_i}{\Delta v_i}\right). \tag{3.10}$$

This linesearch is performed for both s and z, and the step length is simply the minimum of the two. For a complete algorithmic specification and implementation details, we refer the reader to [115] for a PDIP method for which Alg. (2) can be used to solve the linear systems.

Algorithm 2 Primal-Dual Linear System Solver

function solve_kkt(u_1, u_2, u_3, u_4) ▷ assume access to Q, G, A, s, z
$P \leftarrow D(s \oslash z)$
$H \leftarrow Q + G^T P^{-1} G$ ▷ cacheable Cholesky decomposition
$F \leftarrow AH^{-1}A^T$ ▷ cacheable Cholesky decomposition
$r_2 \leftarrow u_3 - u_2 \oslash z$
$p_1 \leftarrow u_1 + G^T P^{-1} r_2$
$\Delta y \leftarrow F^{-1}(AH^{-1}p_1 - u_4)$
$\Delta x \leftarrow H^{-1}(p_1 - A^T \Delta y)$
$\Delta s \leftarrow u_3 - G\Delta x$
$\Delta z \leftarrow (u_2 - z \odot \Delta s) \oslash s$
return $(\Delta x, \Delta s, \Delta z, \Delta y)$

3.2.3 Differentiable Optimization

Modern automatic differentiation tools have made forming derivatives through complex functions easier than ever. When one of these functions includes an iterative routine, unrolling the iterations into a sequential computational graph and proceeding with differentiation is not always possible. The first concern is nonsmooth operations or logical branching in the iterations that can stop the "flow" of the derivatives through the routine, and the second concern is the rapidly decaying numerical precision inherent in differentiating an iterative process.

To address both of these shortcomings, iterative routines such as numerical optimizers are differentiated with methods that do not require propagating derivatives through the iterations themselves. Instead, the optimization problem is solved as normal, and the derivatives are then constructed directly from the solution to the problem.

Implicit Function Theorem

When an iterative routine can be interpreted as finding a root or an equilibrium point to an implicit function, the implicit function theorem can be used to form the derivatives of

interest. Given variables $w \in \mathbf{R}^a$ and parameters $\theta \in \mathbf{R}^b$, an implicit function is defined as

$$r(w^*, \theta) = 0, \tag{3.11}$$

at an equilibrium point w^*. This implicit function can be linearized about this point resulting in the following first-order Taylor series:

$$\frac{\partial r}{\partial w}\delta w + \frac{\partial r}{\partial \theta}\delta\theta = 0, \tag{3.12}$$

which can simply be re-arranged to solve for

$$\frac{\partial w}{\partial \theta} = -\left(\frac{\partial r}{\partial w}\right)^{-1}\frac{\partial r}{\partial \theta}. \tag{3.13}$$

The implicit function theorem enables the differentiation of routines that solve for equilibrium points without the need to unroll and differentiate the iterations themselves.

By treating the KKT conditions from (3.4)-(3.7) as a residual function of the primal-dual solution (x, s, z, y) and problem parameters θ, the implicit function theorem is used to form the derivatives of the optimizer without unrolling the iterations. The linear system from (3.13) applied to this residual function is of the same form as that in Alg. (2), enabling fast and easy computation of derivatives using a framework already available in the solver.

Efficient Computation of Gradients

Using the implicit function theorem to compute the Jacobians of the primal-dual solution with respect to the problem parameters results in the need to directly form potentially large Jacobians. In many cases, it is not these specific Jacobians that are of interest, but rather the left matrix-vector product with a backward pass vector.

For a loss function of the primal variable $\ell(x)$ that takes as input the optimal primal variable from (3.1), reverse-mode automatic differentiation will have constructed $\nabla_x \ell$ by the time it comes to the QP solver in the backward pass of the computational graph. Instead of forming the Jacobians of the primal variable with respect to each of the problem matrices directly, it is instead desirable to form the left matrix-vector products $\frac{\partial \ell}{\partial x}\frac{\partial x}{\partial \square}$, where $\square$ simply denotes any of the problem matrices [7].

As shown in Alg. (3), the gradients of this loss function $\nabla_\square \ell$ with respect to the problem parameters of the QP can be computed by once again utilizing our interior-point linear system solver.

3.3 Logarithmic Barrier Smoothing

Primal interior-point methods work by replacing constraints of the form $h(x) \leq 0$ with a logarithmic barrier penalty in the form of $\phi(x) = -\kappa \sum \log(-h(x))$, where $\kappa \in \mathbf{R}^+$ is referred to as the central path parameter. This barrier function is a smooth approximation of the indicator function, where feasible values of x result in no penalty, and infeasibility

Algorithm 3 Computing Gradients Through a QP

function compute_qp_grads($Q, q, A, b, G, h, x, s, z, y, \nabla_x \ell$)
$dx, ds, d\tilde{z}, dy \leftarrow$ solve_kkt$(-\nabla_x \ell, 0, 0, 0)$ ▷ compute differentials with kkt system
$dz = d\tilde{z} \oslash z$
$\nabla_Q \ell \leftarrow \big((dx)x^T + x(dx)^T\big)/2$
$\nabla_q \ell \leftarrow dx$
$\nabla_A \ell \leftarrow (dy)x^T + y(dx)^T$
$\nabla_b \ell \leftarrow -dy$
$\nabla_G \ell \leftarrow z \odot \big((dz)x^T + z(dx)^T\big)$
$\nabla_h \ell \leftarrow -z \odot dz$
return $\nabla_Q \ell, \nabla_q \ell, \nabla_A \ell, \nabla_b \ell, \nabla_G \ell, \nabla_h \ell$

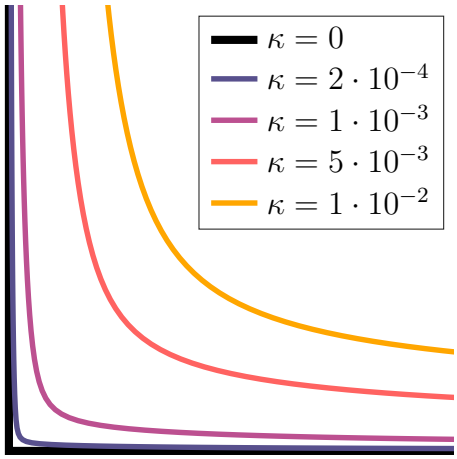


Figure 3.1: A sharp corner in a square as smoothed with the logarithmic barrier at varying central path parameters κ. As $\kappa \to 0$, the corner becomes more pronounced until it assumes a true 90° corner at $\kappa = 0$. The logarithmic barrier effectively smooths out any sharp corners of the feasible set enabling smooth differentiation in the presence of such discontinuities.

results in an infinite penalty [24]. By solving a sequence of unconstrained problems as $\kappa \to 0$, the logarithmic barrier becomes a closer and closer approximation of the indicator function until acceptable convergence is achieved. This barrier function is only defined for feasible values of x, hence the name "interior-point".

For a quadratic program in standard form (3.2), the optimality conditions for a barrier subproblem given a central path parameter κ are almost identical to the original KKT conditions (3.4)-(3.8) with the exception of the complementarity condition (3.5) replaced with $z \odot s - \kappa = 0$. This relaxed complementarity condition allows for a certain amount of smoothing over the feasible set, where larger values of κ have a stronger smoothing effect. The optimality conditions for this barrier subproblem are referred to as the perturbed or relaxed KKT conditions.

3.3.1 Relaxing Primal-Dual Solutions

Algorithm 4 Relaxing a Quadratic Program

function relax_qp$(Q, q, A, b, G, h, x, s, z, y, \kappa)$
for $i \leftarrow 1 :$ `max_iters` **do**
    `/* evaluate relaxed KKT conditions and check convergence*/`
    $r_1 \leftarrow Qx + q + A^T y + G^T z$
    $r_2 \leftarrow s \odot z - \kappa$ ▷ relaxed complementarity
    $r_3 \leftarrow Gx + s - h$
    $r_4 \leftarrow Ax - b$
    if $\|(r_1, r_2, r_3, r_4, r_5, r_6)\|_\infty <$ `tol` **then**
        return: x, t, s_1, s_2, z_1, z_2
    end if

    `/* calculate and take Newton step */`
    $\Delta x, \Delta s, \Delta z, \Delta y \leftarrow$ solve_kkt$(-r_1, -r_2, -r_3, -r_4)$ ▷ Alg. (2)
    $\alpha \leftarrow 0.98 \min\big(\text{linesearch}(s, \Delta s), \text{linesearch}(z, \Delta z)\big)$ ▷ (3.10)
    $(x, s, z, y) \leftarrow (x, s, z, y) + \alpha(\Delta x, \Delta s, \Delta z, \Delta y)$
end for

When differentiating a quadratic program near an active inequality constraint, the implicit function theorem can produce subgradients that are potentially uninformative as a result of rank deficient linear systems [4]. In order to ensure smooth and continuous gradients in the presence of sharp corners in the feasible set, proposed here is a relaxation method that exploits the smoothing of the logarithmic barrier to effectively round out any nonsmoothness. Specifically, the idea is to take a primal-dual solution that is optimal to some low κ_{low}, and relax it to a specified κ_{high}.

In PDIP methods, much care is taken to solve the perturbed KKT conditions for decreasing $\kappa \to 0$. The most common strategy in PDIP methods is a Mehrotra predictor-corrector method that adaptively updates the target κ until it is below the convergence criteria [116]. While solving this system for a sequence where $\kappa \to 0$ is challenging, going the other direction from $\kappa_{\text{low}} \to \kappa_{\text{high}}$ is actually quite trivial. For this case, standard Newton steps on the perturbed KKT conditions with a linesearch to ensure $(s, z) > 0$ is able to converge to κ_{high} in only a few steps. This algorithm is shown in Alg. (4), where once again the linear system solver from Alg. (2) is used.

Once the primal-dual solution to the quadratic program has been relaxed, Alg. (3) is used to calculate gradients that benefit from the logarithmic barrier smoothing. The full sequence for the solving, relaxation, and differentiation of a quadratic problem is as follows:

1. Solve the quadratic program to a specified tolerance and return the solution
2. Relax the primal-dual solution to a target κ
3. Form the derivatives of interest at the relaxed primal-dual solution

This means that the solver itself can return high quality solutions to tight tolerances while still returning smooth gradients evaluated at the relaxed solution. In modern automatic differentiation frameworks, this sequence is written into a custom forward and backward pass through the function with limited overhead.

The choice of a target κ is left to the specifics of the problem at hand. In some scenarios, only a little bit of smoothing is required, making a lower κ appropriate. For very sharp corners in the feasible set (like the tip of a triangle), a larger value of κ can be used to provide even more smoothing. In either case, computing the derivatives of the solver with this technique allows both the tolerance of the solver and the relaxed κ to be specified exactly and independently.

3.4 Elastic Quadratic Program Solver

While the convexity of a quadratic program guarantees a globally optimal solution when one is available, there is generally no guarantee of feasibilty. This infeasibility occurs when the quadratic program has a set of constraints that are impossible to satisfy, something that can easily happen in practice if the constraint matrices are learned. In the event of an infeasible problem, standard PDIP methods are unable to return a useful solution.

Traditionally, infeasibility in convex optimization problems can be handled with a homogenous self-dual embedding that allows for the computation of a certificate of infeasibility [42, 183, 152]. This ensures that, even in the event of an infeasible problem, the solver can "gracefully" fail in a way that simply returns certification that there are no solutions that satisfy the constraints. While this approach is useful for determining if a given problem is feasible, ideally we would set up problems such that infeasibility is not possible.

Given a convex quadratic program in a standard inequality-only form,

$$\begin{array}{ll} \underset{x}{\text{minimize}} & \frac{1}{2}x^T Q x + q^T x \\ \text{subject to} & Gx \leq h, \end{array} \tag{3.14}$$

it is possible there is no x that satisfies $Gx \leq h$. In order to convert the optimization problem in (3.14) into one in which there is always a solution, the hard constraints are converted into penalties. An ℓ_1-penalty on the constraint violation is chosen because the ℓ_1-norm encourages sparsity in the constraint violation, translating into a penalty that encourages the solver to satisfy as many of the constraints as possible. This is a common technique in nonlinear programming [130] for the handling of infeasible subproblems. This "elastic" mode, as it is referred to in the solver SNOPT [61], is a highly effective method to guarantee that a problem always has a solution without sacrificing the quality and utility of the solution. The problem from (3.14) is converted into its elastic form as follows:

$$\underset{x}{\text{minimize}} \quad \frac{1}{2}x^T Q x + q^T x + \|\rho \odot \max(0, Gx - h)\|_1, \tag{3.15}$$

such that feasible values of x do not contribute to the cost function but infeasibility is penalized with $\rho \in \mathbf{R}^p$. While (3.15) is an unconstrained convex optimization problem, it

is nonsmooth and does not pair well with PDIP methods. We therefore reformulate it as:

$$\begin{array}{ll} \underset{x,t}{\text{minimize}} & \frac{1}{2}x^TQx + q^Tx + \rho^Tt \\ \text{subject to} & Gx - h \leq t, \\ & t \geq 0, \end{array} \tag{3.16}$$

where $t \in \mathbf{R}^p$ is a slack variable containing the constraint violation, and a simple linear cost term is used to recover the ℓ_1-penalty from (3.15).

Solving the elastic quadratic program in (3.16) with a PDIP method normally comes at an added computational expense since we have increased the number of primal-dual variables from $n + 2p$ to $n + 5p$. To avoid the cubic complexity in the increase primal-dual dimension, we introduce a custom algorithm for solving these problems that exploits the sparsity of the constraints such that the time to solve the elastic version of the problem is only a slight (5–20%) increase compared to the time to solve the original problem.

As before, the PDIP method for solving (3.16) is dominated by the factorization and solving of linear systems in the following form:

$$\begin{bmatrix} Q & 0 & 0 & 0 & 0 & G^T \\ 0 & 0 & 0 & 0 & -I & -I \\ 0 & 0 & Z_1 & 0 & S_1 & 0 \\ 0 & 0 & 0 & Z_2 & 0 & S_2 \\ 0 & -I & I & 0 & 0 & 0 \\ G & -I & 0 & I & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta t \\ \Delta s_1 \\ \Delta s_2 \\ \Delta z_1 \\ \Delta z_2 \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \\ r_5 \\ r_6 \end{bmatrix}. \tag{3.17}$$

As shown in Alg. (6), this linear system can be solved with block-wise elimination where the only matrix factorization required is that of a positive definite matrix the size of the primal variable x. This routine is used in the full PDIP algorithm for the elastic QP as shown in Alg. (5).

The elastic QP is fully differentiable in the same way the original QP is. To do this, Alg. (7) is used to take a primal-dual solution and relax it to a specified κ, and Alg. (6) for solving the linear systems is re-used for fast and efficient relaxation of the elastic problem. From this, the gradients of a downstream loss function with respect to the problem parameters can be constructed with Alg. (8), again using the same linear system solver.

The solving, relaxation, and differentiation of the elastic mode QP are all only a slight increase in computational complexity compared to the original QP, with the benefit of guaranteed feasibility. This enables the inclusion of always-feasible quadratic programming in learned pipelines where feasibility cannot be guaranteed by construction.

3.4.1 Elastic Initialization

In order to initialize the PDIP method shown in Alg. (5), the only requirement is that $s, z > 0$. In practice, a more advanced initialization technique can both reduce the number

Algorithm 5 PDIP Method for Elastic Quadratic Programs

function solve_qp_elastic(Q, q, G, h, ρ)
$x, s_1, s_2, z_1, z_2 \leftarrow$ initialize(Q, q, G, h, ρ) ▷ 3.4.1
for $i \leftarrow 1$: `max_iters` **do**
    `/* evaluate KKT conditions and check convergence*/`
    $r_1 \leftarrow Qx + q + G^T z_2$
    $r_2 \leftarrow -z_1 - z_2 + \rho$
    $r_3 \leftarrow s_1 \odot z_1$
    $r_4 \leftarrow s_2 \odot z_2$
    $r_5 \leftarrow -t + s_1$
    $r_6 \leftarrow Gx - t + s_2 - h$
    if $\|(r_1, r_2, r_3, r_4, r_5, r_6)\|_\infty <$ `tol` **then**
        return: x, t, s_1, s_2, z_1, z_2
    end if

    `/* calculate affine step direction Alg.` (6)`*/`
    $\Delta x^{\mathrm{a}}, \Delta t^{\mathrm{a}}, \Delta s_1^{\mathrm{a}}, \Delta s_2^{\mathrm{a}}, \Delta z_1^{\mathrm{a}}, \Delta z_2^{\mathrm{a}} \leftarrow$ elastic_kkt$(-r_1, -r_2, -r_3, -r_4, -r_5, -r_6)$
    $\Delta s^{\mathrm{a}}, \Delta z^{\mathrm{a}} \leftarrow (\Delta s_1^{\mathrm{a}}, \Delta s_2^{\mathrm{a}}), (\Delta z_1^{\mathrm{a}}, \Delta z_2^{\mathrm{a}})$
    $s, z \leftarrow (s_1, s_2), (z_1, z_2)$
    `/* calculate centering-plus-corrector step direction */`
    $\alpha^{\mathrm{a}} = \min(\text{linesearch}(s, \Delta s^{\mathrm{a}}), \text{linesearch}(z, \Delta z^{\mathrm{a}}))$ ▷ (3.10)
    $\mu \leftarrow s^T z / \operatorname{len}(s)$
    $\sigma \leftarrow [(s + \alpha^{\mathrm{a}} \Delta s^{\mathrm{a}})^T (z + \alpha^{\mathrm{a}} \Delta z^{\mathrm{a}}) / (s^T z)]^3$
    $r_3 \leftarrow r_3 - \sigma\mu \mathbf{1} + \Delta s_1^{\mathrm{a}} \odot \Delta z_1^{\mathrm{a}}$
    $r_4 \leftarrow r_4 - \sigma\mu \mathbf{1} + \Delta s_2^{\mathrm{a}} \odot \Delta z_2^{\mathrm{a}}$
    $\Delta x, \Delta t, \Delta s_1, \Delta s_2, \Delta z_1, \Delta z_2 \leftarrow$ elastic_kkt$(-r_1, -r_2, -r_3, -r_4, -r_5, -r_6)$

    `/* update with linesearch */`
    $\alpha \leftarrow 0.98 \min(\text{linesearch}(s, \Delta s), \text{linesearch}(z, \Delta z))$
    $(x, t, s_1, s_2, z_1, z_2) \leftarrow (x, t, s_1, s_2, z_1, z_2) + \alpha(\Delta x, \Delta t, \Delta s_1, \Delta s_2, \Delta z_1, \Delta z_2)$
end for

Algorithm 6 Elastic Primal-Dual Linear System Solver

function elastic_kkt($r_1, r_2, r_3, r_4, r_5, r_6$) ▷ assume access to $Q, G, A, s_1, s_2, z_1, z_2$
$w_1 \leftarrow r_3 \oslash z_1$
$w_2 \leftarrow r_4 \oslash z_2$
$p_1 \leftarrow r_5 - r_6 + w_2 - w_1 - (s_1 \odot r_2) \oslash z_1$
$A_3 \leftarrow \text{diag}(a_1 + a_2)$
$\Delta x \leftarrow (Q + G^T A_3^{-1} G)^{-1}(r_1 - G^T A_3^{-1} p_1)$ ▷ Cholesky factorization (cacheable)
$\Delta z_2 \leftarrow A_3^{-1}(p_1 + G\Delta x)$
$\Delta z_1 \leftarrow -r2 - \Delta z_2$
$\Delta s_1 \leftarrow (r_3 - s_1 \odot \Delta z_1)/z_1$
$\Delta s_2 \leftarrow (r_4 - s_2 \odot \Delta z_2)/z_2$
$\Delta t \leftarrow \Delta s_1 - r_5$
return Δx, Δs_1, Δs_2, Δz_1, Δz_2

of iterations required for convergence and dramatically improve the robustness of the solver. The initialization from [183] and [115] is adapted for use in the elastic case.

First, the solution to an easier but similar quadratic program is computed analytically:

$$\begin{aligned} \underset{x,\,t,\,s_1,\,s_2}{\text{minimize}} \quad & \frac{1}{2}x^T Q x + q^T x + \frac{1}{2}(s_1^T s_1 + s_2^T s_2) + \rho^T t \\ \text{subject to} \quad & s_1 - t = 0, \\ & Gx - t + s_2 = h, \end{aligned} \tag{3.18}$$

where the primal and dual solutions are the solution to the linear system,

$$\begin{bmatrix} Q & 0 & 0 & G^T \\ 0 & 0 & -I & -I \\ 0 & -I & -I & 0 \\ G & -I & 0 & -I \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta t \\ \Delta z_1 \\ \Delta z_2 \end{bmatrix} = \begin{bmatrix} -q \\ \rho \\ 0 \\ h \end{bmatrix}, \tag{3.19}$$

which can be solved with a dense block reduction:

$$x = (Q + \frac{1}{2}G^T G)^{-1}\big(-q - \frac{1}{2}G^T(\rho - h)\big), \tag{3.20}$$

$$z_2 = \frac{1}{2}(Gx + \rho - h), \tag{3.21}$$

$$z_1 = \rho - z_2, \tag{3.22}$$

$$t = -z_1. \tag{3.23}$$

This method requires a single Cholesky factorization of a matrix the size of x. From here, we stack $z = (z_1, z_2)$ and initialize $s = (s_1, s_2)$ with

$$\alpha_p = -\min(-z), \tag{3.24}$$

$$s = \begin{cases} -z, & \alpha_p < 0 \\ -z + (1 + \alpha_p \mathbf{1}), & \alpha_p \geq 0 \end{cases}. \tag{3.25}$$

The dual variable for the inequality constraint is then initialized with

$$\alpha_d = -\min(-z), \tag{3.26}$$

$$z = \begin{cases} z + (1 + \alpha_d \mathbf{1}), & \alpha_d \geq 0 \\ z, & \alpha_d < 0 \end{cases}, \tag{3.27}$$

finishing the initialization of the primal and dual variables.

Algorithm 7 Relaxing an Elastic Quadratic Program

function relax_qp_elastic($Q, q, A, b, G, h, \rho, x, t, s_1, s_2, z_1, z_2, \kappa$)
for $i \leftarrow 1$: `max_iters` **do**
    `/* evaluate KKT conditions and check convergence*/`
    $r_1 \leftarrow Qx + q + G^T z_2$
    $r_2 \leftarrow -z_1 - z_2 + \rho$
    $r_3 \leftarrow s_1 \odot z_1 - \kappa$ ▷ relaxed complementarity
    $r_4 \leftarrow s_2 \odot z_2 - \kappa$ ▷ relaxed complementarity
    $r_5 \leftarrow -t + s_1$
    $r_6 \leftarrow Gx - t + s_2 - h$
    if $\|(r_1, r_2, r_3, r_4, r_5, r_6)\|_\infty <$ `tol` **then**
        return: x, t, s_1, s_2, z_1, z_2
    end if

    `/* calculate and take Newton step*/`
    $\Delta x, \Delta t, \Delta s_1, \Delta s_2, \Delta z_1, \Delta z_2 \leftarrow$ elastic_kkt$(-r_1, -r_2, -r_3, -r_4, -r_5, -r_6)$
    $\Delta s,\ \Delta z \leftarrow (\Delta s_1, \Delta s_2),\ (\Delta z_1, \Delta z_2)$
    $s,\ z \leftarrow (s_1, s_2),\ (z_1, z_2)$
    $\alpha^{\text{a}} = 0.98 \min(\text{linesearch}(s, \Delta s),\ \text{linesearch}(z, \Delta z))$ ▷ (3.10)
    $(x, t, s_1, s_2, z_1, z_2) \leftarrow (x, t, s_1, s_2, z_1, z_2) + \alpha(\Delta x, \Delta t, \Delta s_1, \Delta s_2, \Delta z_1, \Delta z_2)$
end for

3.5 Numerical Experiments

The utility of the proposed approaches are demonstrated on the common optimization-based robotics tasks of contact mechanics simulation and collision detection. In each of these tasks, a QP makes up a core part of the algorithm and smooth differentiation through the inherent nonsmoothness proves useful.

3.5.1 Contact Mechanics

For a block at rest on a table with both gravity and friction, the nonsmooth contact dynamics can be represented as the solution to a convex quadratic program. For a full treatment of optimization-based dynamics, the reader is referred to [10] and [78].

Algorithm 8 Computing Gradients Through an Elastic QP

function compute_elastic_qp_grads($Q, q, G, h, x, t, s_1, s_2, z_1, z_2, \nabla_x \ell$)
`/* compute differentials with same linear system */`
$dx, dt, ds_1, ds_2, d\tilde{z}_1, d\tilde{z}_2 \leftarrow \text{elastic\_kkt}(-\nabla_x \ell, 0, 0, 0, 0, 0)$
$p, s, z \leftarrow (x, t), (s_1, s_2), (z_1, z_2)$
$dp, ds, dz \leftarrow (dx, dt), (ds_1, ds_2), (d\tilde{z}_1, d\tilde{z}_2) \oslash (z_1, z_2)$
`/* create indices (one-based) and form gradients from differentials */`
$ix \leftarrow 1 : \text{len}(q)$
$is \leftarrow (\text{len}(q) + 1) : (\text{len}(q) + \text{len}(h))$
$\nabla_Q \ell \leftarrow \left[\left((dp)p^T + p(dp)^T\right)/2\right]_{ix,ix}$
$\nabla_q \ell \leftarrow [dp]_{ix}$
$\nabla_G \ell \leftarrow \left[z \odot \left((dz)p^T + z(dp)^T\right)\right]_{is,ix}$
$\nabla_h \ell \leftarrow [-z \odot dz]_{is}$
return $\nabla_Q \ell, \nabla_q \ell, \nabla_A \ell, \nabla_b \ell, \nabla_G \ell, \nabla_h \ell$

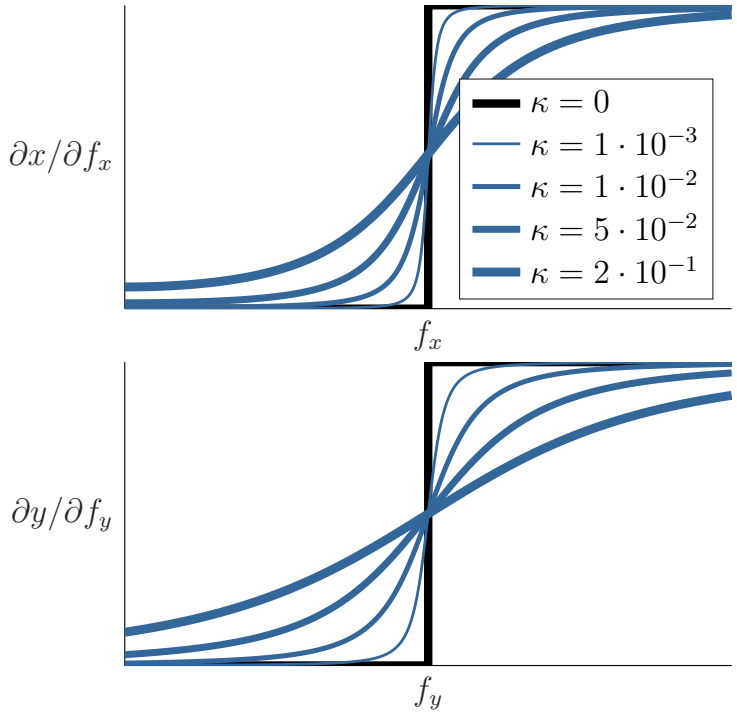


Figure 3.2: Contact dynamics for a two-dimensional block as modeled with a quadratic program. When a horizontal force f_x is applied to the block, it must overcome the friction with the ground before it moves, and when a vertical force f_y is applied, it must overcome gravity. Despite these discontinuities in the dynamics, the relaxed gradients from the differentiable quadratic program solver are able to provide smooth and continuous derivative information before and after the block begins to move.

In the simplest case, a block is stationary until it is acted upon by a force that exceeds

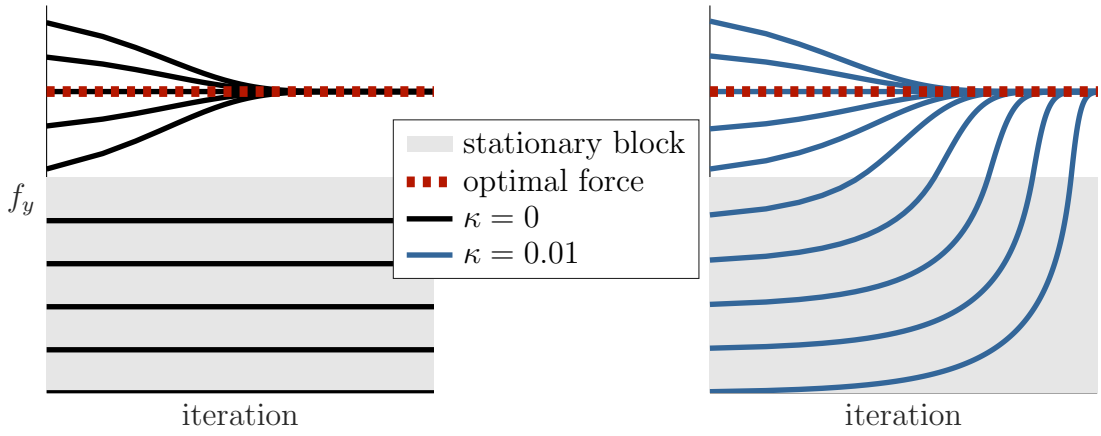


Figure 3.3: Using the gradients from the block pushing example in Fig. 3.2, a vertical force is optimized to accelerate the block to a set value from multiple different initial force values. When the force is below the threshold for the block to move and $\kappa = 0$, the gradient is zero and the optimizer fails to make progress. In the case of $\kappa = 0.01$, even before the block moves there is gradient information that pushes the optimizer to converge on the optimal force regardless of the initial force value.

the static frictional force in the horizontal direction, or the gravitational force in the vertical direction. Until the applied forces exceed these two thresholds, the block does not move. This exercise is demonstrated in Fig. 3.2, where forces are applied in each of the two directions and the true derivatives of these dynamics at $\kappa = 0$ have a discontinuity as soon as the block begins to move. When these derivatives are taken with a relaxed $\kappa > 0$, the discontinuous derivative is smoothed, allowing for an informative gradient about the impending motion of the block before it moves.

This derivative information is used in an optimization routine in Fig. 3.3, where an optimizer attempts to solve for an applied force that produces the desired motion from the block. This optimizer is initialized at multiple different force values, and uses either exact gradients with $\kappa = 0$, or smoothed gradients with $\kappa = 0.01$. In the case of exact gradients, for initial forces where the block does not move, the gradient is zero and the solver is unable to find a descent direction. When κ is relaxed, the smooth gradients provide information about the motion of the block even before the block itself begins to move, allowing the optimizer to converge on the true solution for each initial force. This is a simple yet expressive demonstration of the impact these smooth gradients have on optimization routines in the presence of discontinuous subgradients.

3.5.2 Collision Detection

Collision detection between convex shapes can be formulated as a convex optimization problem, both in terms of the closest point between shapes [60], and in terms of the minimum scale factor [170]. For the former, we introduce two points in a world frame $p_i \in \mathbf{R}^3$, and two polytopes described with $A_i p_i \leq b_i$. By constraining each point to be

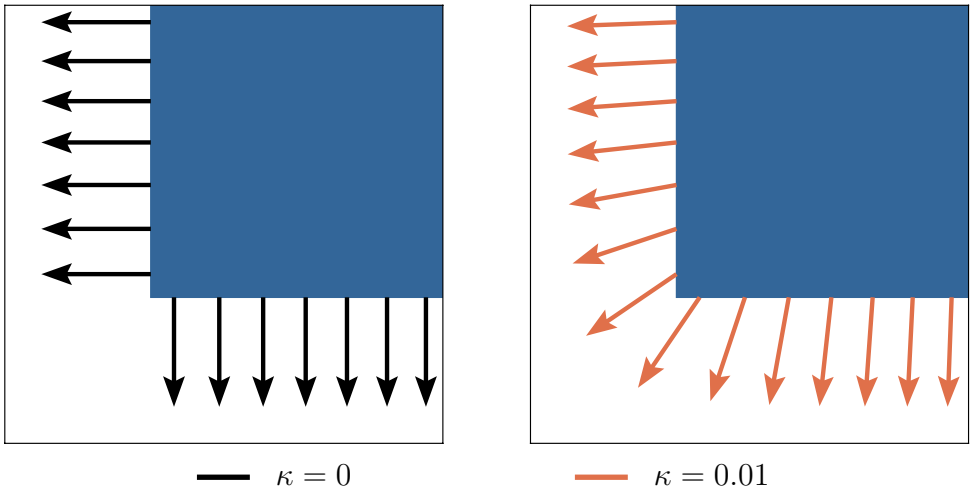


Figure 3.4: Contact normal vectors from an optimization-based differentiable collision detection routine with and without relaxed differentiation. With no relaxation ($\kappa = 0$), the direction of the contact normal switches immediately as the closest point moves from one face to another. When relaxed gradients are used ($\kappa = 0.01$), the contact normal smoothly transitions between the faces.

within a polytope, a QP is used to solve for the closest point between these two shapes,

$$\begin{aligned} \underset{p_1, p_2}{\text{minimize}} \quad & \|p_1 - p_2\|_2^2 \\ \text{subject to} \quad & A_1 p_1 \leq b_1, \\ & A_2 p_2 \leq b_2. \end{aligned} \tag{3.28}$$

Using a differentiable QP solver, the gradient of the objective value with respect to the positions of the polytopes results in the contact normal vectors.

In this example, we examine collision detection between two squares and the behavior of these contact normals in the presence of sharp corners. As shown in Fig. 3.4, the contact normals are evaluated at a strict $\kappa = 0$ and a relaxed $\kappa = 0.01$. In the case of $\kappa = 0$, the contact normals are (correctly) exactly normal to the surface, and as soon as the closest point shifts from one face to the other, the contact normals immediately rotate 90°. While this is expected behavior, this discontinuity in the gradient can prove troublesome for simulation and control algorithms that rely on these contact normals not changing too quickly. Alternatively, with a relaxed $\kappa = 0.01$, the contact normal smoothly rotates the 90 degrees as the face of the closest point changes. This is a result of the logarithmic barrier smoothing out the sharp corner, and allows for continuous and smooth gradients even in the presence of the discontinuity.

3.6 Conclusions

In this paper, we outline shortcomings with existing differentiable optimization tools, namely the nonsmoothness of the gradients near inequality constraints and the inability to

handle infeasible problems, and propose solutions to both of these problems. By relaxing the solution to an optimization problem from tight tolerances to an intentionally relaxed logarithmic barrier, unique and smooth gradients can be computed even from sharp edges in the feasible set. This relaxation is straightforward and leverages existing routines within existing primal-dual interior-point solvers. We also introduce an always-feasible quadratic program where hard constraints are converted into ℓ_1-penalties, and devise a customized algorithm for solving problems of this form with limited added computational overhead. With both of these innovations, consistent and reliable smooth gradients are demonstrated in common robotic tasks where smoothness is a priority. Our fully differentiable and parallelizable solver written in JAX is available at www.github.com/kevin-tracy/qpax.

Chapter 4

Atmospheric Entry Guidance

As scientific and crewed payloads have more demanding goals, precise atmospheric entry guidance is playing an increasing role in mission success. State-of-the-art entry guidance algorithms are structured in a predictor-corrector framework, where a simulation is used to predict a trajectory, and corrections are then made to the control inputs. These guidance methods are simple and effective, but current algorithms assume low lift-to-drag entry vehicles, are limited to only bank-angle control, and have a limited ability to guarantee the safety of the vehicle. We propose a new predictor-corrector entry guidance method that formulates the correction step as a convex optimization problem. This allows for more flexibility in specifying the vehicle's dynamics and control inputs, and the ability to explicitly handle safety constraints such as heating, pressure, and acceleration limits. We test the new algorithm in Mars entry scenarios similar to the Mars Science Laboratory with both bank-angle control and bank-angle plus angle-of-attack control, demonstrating both its performance and ability to generalize to future vehicle capabilities.

The contents of this chapter have been previously published at IEEE Aerospace Conference 2021 in Tracy and Manchester [173]

4.1 Introduction

In 1971 the Soviet Union's Mars-2 spacecraft made history by entering the Martian atmosphere before impacting the surface. Nine days later, an identical Mars-3 spacecraft performed the first soft-landing on the Martian surface, ushering in a new era in planetary exploration. NASA followed with successful Mars landings in 1976 with Viking 1 and 2 and has since then landed and operated multiple robotic systems on the Martian surface [97].

Entry vehicle architectures can be divided into three broad categories [97]: 1) Ballistic entry is an uncontrolled descent with drag as the only force, 2) unguided ballistic-lifting entry has an uncontrolled non-zero lift force, and 3) guided ballistic-lifting entry has some control over the vehicle's lift vector. Controlled entry guidance allows for the prioritization of landing locations with scientific merit instead of just those that minimize risk to the

vehicle.

The Mars Science Laboratory (MSL) carrying the Curiosity rover touched down in 2012 as the first Mars entry vehicle with guided ballistic-lifting entry. MSL had control over the vehicle bank angle during entry, enabling control of the direction of the lift vector within the lifting plane. While MSL dramatically reduced the size of the landing ellipse from over 100 km to 10 km, its guidance is still too coarse for pinpoint landings. By developing more performant entry guidance capabilities, entry vehicles could effectively place robotic or crewed landers in desirable science collection areas, including high altitude sites.

Much of the work on guidance for low lift-to-drag entry vehicles originated with the Apollo terminal guidance methods. These algorithms, as described in [64], rely on control of the bank angle with simple switching manuevers to control the cross-range and down-range errors. Slightly modified versions have been developed for use with more recent Mars entry vehicles, such as in [119]. Current research investigates the use of predictor-corrector algorithms [29] to improve the landing accuracy. A popular predictor-corrector formulation that exhibits bank-angle switching behavior is the Fully Numerical Predictor-corrector Entry Guidance (FNPEG) algorithm [103]. In the baseline FNPEG algorithm, Newton's method is used to solve for a static bank-angle that satisfies a terminal downrange distance constraint, and the sign of the bank-angle is modulated to control crossrange errors [101]. Here, the prediction phase is used to generate gradients for the terminal constraint, and corrections are applied to the open-loop commanded bank-angle in an effort to satisfy these terminal constraints. This framework is simple and effective but requires significant added complexity for incorporation of safety constraints or changes to the vehicle control inputs.

Trajectory optimization for offline planning of entry vehicle trajectories has been explored in [189] and [191], where the nonconvex optimal control problem was solved by linearizing the nonlinear dynamics and constraints, solving a conic optimization problem with a trust region, and repeating until convergence. This successive-convexification method was used instead of standard NonLinear Programming (NLP) solvers, like SNOPT [61] or IPOPT [187], because it is able to directly handle second-order cone constraints instead of relying on local linear approximations. Optimal trajectories computed offline were then paired with an optimization-based tracking controller, as described in [189] and [191]. While these formulations are able to stabilize a trajectory, there are no guarantees that safety constraints can be satisfied online. Also, the computational complexity of the trajectory-optimization formulation makes these methods intractable for real-time control onboard an entry vehicle.

The Convex Predictor-corrector Entry Guidance (CPEG) algorithm proposed in this paper combines ideas from trajectory optimization with the predictor-corrector guidance framework by solving a constrained optimization problem during the correction step. First, the dynamics of the entry vehicle with the current control plan are simulated to a target altitude for a predicted trajectory. Next, the vehicle dynamics are linearized about the predicted trajectory and a convex trajectory optimization problem is solved that minimizes landing error. By solving for a correction using convex optimization, CPEG is able to reason about the full state and control history to inform the correction instead of just the final state. This also allows for the vehicle's safety constraints, such as heating, pressure,

and acceleration, to be explicitly included in the correction computation. Our specific contributions in this paper are:

1. A general quasi-linear formulation of entry vehicle dynamics that is well-suited to numerical optimization.

2. A predictor-corrector entry guidance algorithm with a highly generalizable correction step utilizing convex optimization.

3. Customized trust regions and objective functions for entry vehicles with multiple control modalities.

The paper proceeds as follows: In Section 4.2, the classic Vinh entry vehicle dynamics are compared with a more modern Cartesian approach. In Section 4.3, the details of the full nonconvex trajectory optimization problem are discussed. In Section 4.4, the CPEG algorithm is derived. In Section 4.5, CPEG is validated on entry vehicles with bank-angle control, as well as bank-angle and angle-of-attack control. Finally, Section 4.6 outlines our conclusions and potential future research directions.

4.2 Entry Vehicle Dynamics

Despite much of the recent powered-descent guidance literature using Cartesian state representations, entry vehicles are still most often represented in spherical coordinates. In this section, the traditional entry vehicle dynamics denoted below as the "Vinh" model will be discussed, as well as an alternative Cartesian formulation.

4.2.1 The Vinh Model

The classic Vinh model, presented in 1976 in [30] and again a few years Later in Vinh's textbook [185], has been the standard method for simulating entry vehicles for the past 45 years. Parameterizing the entry vehicle in spherical coordinates, the state in the Vinh model contains familiar terms like latitude, longitude, and flight-path angle. Despite being highly nonlinear and prone to scaling issues, it is the most common dynamics model in the literature [30, 185, 186, 192, 190, 101, 53].

The dynamics in the Vinh model are calculated with the angle-of-attack, α, bank-angle, σ, flight-path angle, γ, longitude, θ, latitude, ϕ, and heading angle, ψ. The resulting

equations of motion over a planet that's rotating with a constant angular velocity Ω are,

$$\dot{r} = V \sin\gamma, \tag{4.1}$$
$$\dot{\theta} = V \cos\gamma \sin\psi/(r\cos\phi), \tag{4.2}$$
$$\dot{\phi} = V \cos\gamma \cos\psi/r, \tag{4.3}$$
$$\dot{V} = -D - \sin\gamma/r^2 + \Omega^2 r \cos\phi \sin\gamma \cos\phi - \Omega^2 r \cos\phi \cos\gamma \sin\phi \cos\psi, \tag{4.4}$$
$$\dot{\gamma} = L\cos\sigma/V + \left(V^2 - 1/r\right)\cos\gamma/(Vr) + 2\Omega\cos\phi\sin\psi + \Omega^2 r\cos\phi\cos\gamma\cos\phi/V \tag{4.5}$$
$$+ \Omega^2 r \cos\phi \sin\gamma \sin\phi \cos\psi/V, \tag{4.6}$$
$$\dot{\psi} = L\sin\sigma/(V\cos\gamma) + V\cos\gamma\sin\psi\tan\phi/r - 2\Omega(\tan\gamma\cos\psi\cos\phi - \sin\phi) \tag{4.7}$$
$$+ \Omega^2 r \sin\phi\cos\phi\sin\psi/(V\cos\gamma), \tag{4.8}$$

where r is the normalized radial distance from the center of the planet, V is the normalized planet-relative velocity, and L and D are the magnitudes of the lift and drag accelerations.

This model is highly nonlinear in both the state and the control, even when the planetary motion is ignored. While the planet's angular velocity is assumed to be constant, its inclusion in the dynamics still contributes significant nonlinearities. Because of this, much of the literature ignores the planet's angular velocity [189]. There are also scaling issues present if these equations are naively implemented. Since r and V are not angles, they are usually of a much larger magnitude than the rest of the state. This can lead to poor accuracy in variable time-step integrators, as well as ill-conditioning in numerical trajectory optimization.

4.2.2 Cartesian Entry Dynamics

We have found that entry vehicle dynamics are both simpler to derive and numerically better-conditioned when represented in standard Cartesian coordinates instead of the spherical coordinates used in the Vinh formulation. This state representation is popular with the powered-descent guidance community, albeit without any aerodynamic forces in the dynamics [20, 3, 2].

We assume a planet-fixed frame P is aligned with an inertial frame N along the z axis. The planet spins with angular velocity $\omega \in \mathbb{R}^3$ in the positive z direction, making the velocity of the entry vehicle the following:

$$^P v = {}^N v - \omega \times r, \tag{4.9}$$

where ${}^N v \in \mathbb{R}^3$ is the inertial velocity, ${}^P v \in \mathbb{R}^3$ is the planet relative velocity, and $r \in \mathbb{R}^3$ is the position of the entry vehicle in the planet frame. This expression can be differentiated once more to provide the relationship between the inertial and planet-relative accelerations:

$$^P a = {}^N a - 2(\omega \times {}^P v) - \omega \times (\omega \times r). \tag{4.10}$$

The state of the entry vehicle can be parameterized with the planet-relative position vector r, and planet relative velocity ${}^P v$ denoted as just v, both expressed in the coordinates of

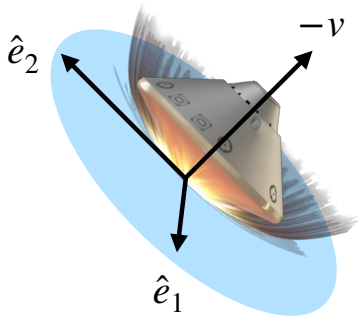


Figure 4.1: The E frame is fixed to the entry vehicle, with $\hat{e}_1$ in the direction of the specific angular momentum vector, and $\hat{e}_2 = \hat{v} \times \hat{e}_1$. When defined in this frame, the lift vector can be expressed using only $\hat{e}_1$ and $\hat{e}_2$.

the planet frame. The Cartesian dynamics can now be written in state space as,

$$\begin{bmatrix} v \\ a \end{bmatrix} = \begin{bmatrix} 0 & I \\ -[\omega\times]^2 & -2[\omega\times] \end{bmatrix} \begin{bmatrix} r \\ v \end{bmatrix} + \begin{bmatrix} 0 \\ a_g + a_D + a_L \end{bmatrix}, \tag{4.11}$$

where $[\omega\times]$ is the skew-symmetric cross product matrix,

$$[\omega\times] = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}. \tag{4.12}$$

One of the main benefits of the dynamics in equation (4.11) is the linear kinematics. This means that linear approximations of the relationship between position and velocity are exact, and the only nonlinearities present are in the accelerations. The gravitational acceleration in the direction of the planet's center is expressed assuming simple spherical gravity:

$$a_g = -\frac{\mu}{\|r\|^3} r, \tag{4.13}$$

where $\mu \in \mathbb{R}$ is the standard gravitational constant for the given planet. The acceleration caused by the drag force is in the direction opposing velocity, and is calculated as,

$$a_D = -\frac{1}{2m} \rho A C_d \|v\| v, \tag{4.14}$$

where $m \in \mathbb{R}$ is the mass of the entry vehicle, $\rho \in \mathbb{R}$ is the atmospheric density, $A \in \mathbb{R}$ is the aerodynamic reference area, and $C_d \in \mathbb{R}$ is the coefficient of drag. In this work, the atmospheric density $\rho \in \mathbb{R}$ will be represented by a piecewise exponential function [53].

For the description of the lift acceleration, a reference frame is defined that describes a plane about the entry vehicle that is orthogonal to the velocity vector. This two-dimensional frame, referred to as the E frame and depicted in Fig. 4.1, has two basis

vectors described by the following:

$$\hat{e}_1 = \frac{r \times v}{\|r \times v\|}, \tag{4.15}$$

$$\hat{e}_2 = \frac{v \times \hat{e}_1}{\|v \times \hat{e}_1\|}. \tag{4.16}$$

The magnitude of the lift vector is calculated as,

$$\|L\| = \frac{1}{2m} C_L \rho(r) A \|v\|^2, \tag{4.17}$$

where $C_L \in \mathbb{R}$ is the coefficient of lift. In the case where the entry vehicle only has control over the bank-angle, the resulting lift acceleration can be described by the magnitude of the lift and the bank-angle:

$$a_L = \|L\|(\sin(\sigma)\hat{e}_1 + \cos(\sigma)\hat{e}_2). \tag{4.18}$$

In the case where the entry vehicle can control both the angle-of-attack as well as the bank-angle, the lift vector can be written as,

$$a_L = \|L\|(\ell_1\hat{e}_1 + \ell_2\hat{e}_2), \tag{4.19}$$

subject to the constraint $||\ell_1^2 + \ell_2^2|| \leq 1$. Here the lift acceleration is a linear function of the control inputs, which is a key feature when this model is linearized in an optimization problem. Both the Vinh model and the Cartesian model are nonlinear, but the Cartesian model behaves significantly better under linearization, making it a far better candidate for trajectory optimization.

4.2.3 State and Control Definitions

In the case where only the bank-angle is controlled, the state is augmented with the bank-angle, and the sole control input is the derivative of this bank-angle with respect to time. This allows for cost functions that specify desired behavior for the derivative of the bank-angle, with the state and control as the following:

$$x = \begin{bmatrix} r^T & v^T & \sigma \end{bmatrix}, \tag{4.20}$$

$$u = \dot{\sigma}. \tag{4.21}$$

These dynamics are now in control-affine form with linear kinematics. For the case with actuation of both the bank angle and angle-of-attack, the state and control are the following:

$$x = \begin{bmatrix} r^T & v^T \end{bmatrix}^T, \tag{4.22}$$

$$u = \begin{bmatrix} \ell_1 & \ell_2 \end{bmatrix}^T, \tag{4.23}$$

where ℓ_1 and ℓ_2 were defined in (4.19).

4.3 Trajectory Optimization

Feedback control laws for entry vehicles suffer in performance due to the severe under-actuation of the vehicle. This is, in part, due to the fact that an entry vehicle has very limited ability to speed up or slow down in the along-track direction. To deal with this, it makes more sense to solve the guidance problem with a holistic planning approach, one that can reason about this limited control authority and plan for it. Therefore, we pose this problem as a trajectory optimization problem, where a locally optimal state trajectory and control plan can be solved for numerically.

4.3.1 Safety Constraints

Three key vehicle safety constraints — heating, pressure, and acceleration — are most dependent on the atmospheric density. Unfortunately, this is also the part of the environment in which there is the largest amount of uncertainty. The atmospheric density is often only known to roughly within a factor of two, with even less known about the wind conditions [53].

The heating constraint has to do with the max allowable heat rate that the ablative heat shield can withstand [46]. This is measured in power per square centimeter, and it is expressed as the following:

$$\dot{Q} = k_q \sqrt{\rho} V^{3.15} \leq \dot{Q}_{max}. \tag{4.24}$$

This function is nonlinear but can be locally approximated with linear functions during the correction step. The next safety constraint is the maximum dynamic pressure on the entry vehicle, which is expressed as the following:

$$q = .5\rho V^2 \leq q_{max}. \tag{4.25}$$

The last safety constraint is the maximum allowable normal load, which is the total aerodynamic force on the entry vehicle. This is expressed as a norm of the lift and drag forces:

$$a = \sqrt{\|L\|^2 + \|D\|^2} \leq a_{max}. \tag{4.26}$$

4.3.2 Full Nonconvex Formulation

In order to formulate a convex correction problem, we first consider the full nonlinear non-convex problem. First, the dynamics described in equation (4.11) are discretized with an explicit integrator like the classic fourth-order Runge-Kutta method [123], giving a discrete-time dynamics model of the form,

$$x_{k+1} = f(x_k, u_k, \Delta t_k). \tag{4.27}$$

No assumptions have been made about the control configuration in this dynamics model: it can account for either bank-angle-only or bank-angle plus angle-of-attack control. The

full nonlinear trajectory optimization problem has the form,

$$\begin{aligned} \underset{x,\,u,\,\Delta t}{\text{minimize}} \quad & \ell_N(x_N, u_N) + \sum_{k=1}^{N-1} \ell_k(x_k, u_k) \\ \text{subject to} \quad & x_{k+1} = f(x_k, u_k, \Delta t_k) \forall k, \\ & g_k(x_k, u_k) \leq 0 \qquad \forall k, \\ & \Delta t_{min} \leq \Delta t_k \leq \Delta t_{max} \; \forall k, \\ & x_N = x_{goal}, \end{aligned} \tag{4.28}$$

where safety constraints (4.24)—(4.26) are included in the inequality constraint function $g_k(x_k, u_k)$. Note that this is a free-final-time problem in which the Δt_k are decision variables in addition to the states and controls. This is necessary due to the inability of the entry vehicle to reach its goal state at an arbitrarily specified time. Problem (4.28) is nonconvex due to both the nonlinear dynamics, as well as the variable time between knot points. It is worth noting that, even with linear continuous-time dynamics, the discrete-time dynamics constraints (4.27) become nonlinear when the time step is made to be a decision variable.

Trajectory optimization problems like (4.28) can be solved with a variety of methods. One standard approach is to use an off-the-shelf NLP solver like IPOPT [187] or SNOPT [61]. Alternatively, more specialized trajectory optimizers like ALTRO can be used [79, 84]. While computationally tractable using one of the described methods, the nonconvexity of the problem means there are no available guarantees for the quality of the solution or convergence of the solver. As a result, running nonconvex trajectory optimization onboard safety-critical aerospace systems is unpopular, explaining the prevalence of simpler heritage methods for entry guidance.

4.4 Convex Predictor-corrector

CPEG combines ideas from numerical trajectory optimization with the classic predictor-corrector guidance framework: It uses a prediction step, in which the vehicle dynamics are simulated until a target altitude is reached, combined with a corrector step that is based on solving a local convex approximation of a nonlinear trajectory optimization problem to steer the vehicle to the desired target. These steps are then repeated until convergence is achieved. This section provides a detailed derivation of the CPEG algorithm.

4.4.1 Prediction and Dynamics Linearization

In the first stage of CPEG, the dynamics of the entry vehicle are simulated with a standard Runge-Kutta method using the current nominal control trajectory, $\bar{U}$, until a target altitude is reached. We denote this predicted trajectory by $\bar{X}$. After the prediction step, the discrete-time nonlinear dynamics are approximated using a first-order Taylor series,

$$\bar{x}_{k+1} + \delta x_{k+1} \approx f(\bar{x}_k, \bar{u}_k) + A_k \delta x_k + B_k \delta u_k, \tag{4.29}$$

where A_k and B_k are the following Jacobians,

$$A_k = \left.\frac{\partial f(x_k, u_k, \Delta t_k)}{\partial x_k}\right|_{\bar{x}_k, \bar{u}_k}, \tag{4.30}$$

$$B_k = \left.\frac{\partial f(x_k, u_k, \Delta t_k)}{\partial u_k}\right|_{\bar{x}_k, \bar{u}_k}. \tag{4.31}$$

Subtracting the dynamics of the reference trajectory from both sides, the local linear dynamics of trajectory corrections can be written as:

$$\delta x_{k+1} = A_k \delta x_k + B_k \delta u_k. \tag{4.32}$$

A crucial distinction between CPEG and sequential convexification methods [189, 106, 107], is that trajectory iterates are always dynamically feasible, thanks to the prediction step. This eliminates the possibility of inconsistent linearizations of the dynamics constraints [130], in which no feasible correction trajectory exists. Specifically, there is always a trivial solution to (4.32) of all zeros for δx and δu.

4.4.2 Cost Function

The cost function used in CPEG is comprised of a term that penalizes the miss distance from the target and a term that penalizes specified control behaviors. For the penalty on miss distance, putting a naive quadratic cost on the error between the final position and the desired position is inappropriate since it also penalizes altitude errors. Instead, only the position error projected onto the landing plane is penalized, effectively ignoring altitude error. Since the altitude target is implicitly satisfied during the prediction step, this allows for the correction to only apply changes to the control plan that minimize the projected miss distance. The cost function for this projected miss distance is the following:

$$\ell_{miss}(\delta X, \delta U) = \|W(r_N + \delta r_N - r_{goal})\|_2^2, \tag{4.33}$$

where $r_N \in \mathbb{R}^3$ is the final position in the reference trajectory, $\delta r_N \in \mathbb{R}^3$ is the correction computed for this position, and $r_{goal} \in \mathbb{R}^3$ is the desired final position for parachute deployment. To project this error onto the landing plane, we define following projection matrix,

$$W = I - pp^T, \tag{4.34}$$

where p is the unit vector normal to the planetary surface at the target position:

$$p = \frac{r_{goal}}{\|r_{goal}\|}. \tag{4.35}$$

The second part of the cost function seeks to shape the control behavior. In the case of bank-angle control, we consider two different control cost functions that produce qualitatively different behavior:

$$\ell_{\sigma,L1}(\delta U) = \lambda \|\dot{\sigma}_k\|_1, \tag{4.36}$$

and

$$\ell_{\sigma,quad}(\delta U) = \lambda \dot{\sigma_k}^2, \tag{4.37}$$

where λ is a scalar tuning parameter. The first cost function (4.36) penalizes the L1 norm of the derivative of the bank-angle, resulting in bank-angle trajectories with a minimum number of discrete switches. The second cost function (4.37) penalizes the square of the bank-angle derivative, resulting in smooth bank-angle trajectories. For the bank-angle plus angle-of-attack case, as described in (4.19), we apply a simple quadratic cost to the norm of the controlled lift vector, effectively penalizing high angles of attack:

$$\ell_{\sigma\alpha}(\delta U) = \lambda \|u_k\|_2^2. \tag{4.38}$$

4.4.3 Constraints

Of the three nonlinear safety constraints, two can be linearized, and the third can be converted to a conservative convex relaxation. For the heating and dynamic pressure constraints (4.24)–(4.25), a Taylor expansion of each is formed, approximating the constraint to first-order. From here, a linearized inequality constraint can be directly included in the convex correction problem. For these constraints, the linearized versions are:

$$[\nabla \dot{Q}(\bar{x}_k)]^T \delta x_k \leq \dot{Q}_{max} - \dot{Q}(\bar{x}_k), \tag{4.39}$$

$$[\nabla q(\bar{x}_k)]^T \delta x_k \leq q_{max} - q(\bar{x}_k). \tag{4.40}$$

The acceleration loading constraint (4.26) is nonlinear, but a conservative convex relaxation can be derived in the form of a second-order cone constraint. First, the kinematics for the velocity can be conservatively approximated as the following:

$$v_{k+1} = v_k + a_k \Delta t, \tag{4.41}$$

$$a_k = \frac{v_{k+1} - v_k}{\Delta t}, \tag{4.42}$$

$$a_k = \frac{\bar{v}_{k+1} + \delta v_{k+1} - \bar{v}_k - \delta v_k}{\Delta t}. \tag{4.43}$$

The maximum loading constraint can then be re-written as,

$$\|\bar{v}_{k+1} + \delta v_{k+1} - \bar{v}_k - \delta v_k\| \leq \Delta t \cdot a_{max}, \tag{4.44}$$

which is in the form of a convex second-order cone, and can be directly incorporated into the correction problem.

The three safety constraints from equations (4.39), (4.40), and (4.44), are stacked into a generic safety constraint function,

$$g_{safety}(\delta x_k, \delta u_k) \leq 0. \tag{4.45}$$

4.4.4 Trust Region

To ensure that corrections are sufficiently small that the dynamics linearizations and constraint approximations remain accurate, a trust-region constraint is added to the convex correction problem. While standard trust-region methods apply norm constraints to δX and δU [130], insight into the entry guidance problem enables a more tailored approach.

The quality of the linearization presented in (4.32) is highly accurate for approximating the vehicle kinematics, gravity, and atmospheric drag, but is much less accurate when applied to the bank-angle in the bank-angle-only control case. Therefore, we design a trust region that restricts corrections to the bank-angle, $\delta\sigma_k$, given the known accuracy of small-angle approximations but allows large corrections to the other states. This approach also allows us to avoid the need to adapt trust regions inside the solver, enabling faster and more reliable convergence. We apply the following trust-region constraints to each corrector problem:

$$\|\delta u_k\|_2 \leq \delta u_{max} \tag{4.46}$$
$$|\delta\sigma_k| \leq \delta\sigma_{max} \tag{4.47}$$

4.4.5 Convex Corrector Problem

For the case where the entry vehicle has control of only the bank-angle as described in (4.18), the convex correction problem can be formulated as,

$$\begin{aligned} \underset{\delta X,\, \delta U}{\text{minimize}} \quad & \ell_{miss}(\delta X, \delta U) + \ell_{\sigma}(\delta U) \\ \text{subject to} \quad & A_k \delta x_k + B_k \delta u_k = \delta x_{k+1}, \\ & g_{safety}(\delta x_k, \delta u_k) \leq 0, \\ & \|\delta u_k\|_2 \leq \delta u_{max}, \\ & |\delta\sigma_k| \leq \delta\sigma_{max}, \end{aligned} \tag{4.48}$$

where the miss cost function is described in (4.33), and the bank-angle cost function can be either (4.36) or (4.37).

For the case where the entry vehicle has control over both bank-angle and angle-of-attack as described in (4.19), the convex correction problem can be posed as:

$$\begin{aligned} \underset{\delta X,\, \delta U}{\text{minimize}} \quad & \ell_{miss}(\delta X, \delta U) + \ell_{\sigma\alpha}(\delta U) \\ \text{subject to} \quad & A_k \delta x_k + B_k \delta u_k = \delta x_{k+1}, \\ & g_{safety}(\delta x_k, \delta u_k) \leq 0, \\ & \|u_k + \delta u_k\|_2 \leq 1. \end{aligned} \tag{4.49}$$

These problems can be solved quickly and reliably by standard conic solvers such as Mosek [124], COSMO [55], and ECOS [42].

4.4.6 CPEG Algorithm

The full CPEG algorithm is detailed in algorithm 9. The inputs to CPEG are the current position and the current control plan. From here, the dynamics of the entry vehicle are simulated until parachute deployment with the current control plan. This predicted trajectory is then discretized and linearized, resulting in dynamics Jacobians A_k and B_k (equations (4.30)-(4.31)). From here, the convex correction problem is posed given the control configuration and cost strategy. This convex optimization problem is solved, and the correction δU is used to correct the control plan. The prediction-correction steps are repeated until the norm of the correction being made to the control plan is below a specified tolerance.

Algorithm 9 CPEG Algorithm

input x_0, U ▷ nominal control plan
while $\|\delta U\| >$ tolerance **do**
    $\bar{X}, \bar{U} = \text{simulate}(x_0, U)$ ▷ predict trajectory
    $A, B = \text{linearize}(\bar{X}, \bar{U})$ ▷ linearize about prediction
    $\delta X, \delta U = \text{cvx}(\bar{X}, \bar{U}, A, B)$ ▷ solve for correction
    $U \mathrel{+}= \delta U$ ▷ correct control plan
end while
return U ▷ return updated control plan

4.5 Numerical Experiments

Parameters roughly matching those of the Mars Science Laboratory (MSL) [119] were used to test the CPEG algorithm. All scenarios begin at an altitude of 125 km above the Martian surface with a Mars-relative velocity of 5.845 km/second. CPEG was implemented in the Julia programming language [18], using the Convex.jl optimization modeling library [179], and the Mosek [124] and OSQP [152] solvers. CPEG was validated on the following three cases:

- Bank-angle control with L1 cost penalty, denoted σ_{L1}.
- Bank-angle control with quadratic penalty, denoted σ_2.
- Bank-angle plus angle-of-attack control, denoted $\sigma + \alpha$.

For the bank-angle cases, CPEG was arbitrarily initialized with a constant bank-angle of zero, with noise added to the bank-angle derivative. For the bank-angle plus angle-of-attack case, a similar approach was used, but noise was added to the normalized lift vector. The final converged trajectories for the three cases are shown in figures 4.2 and 4.3. In all of the cases, CPEG was able to successfully guide the entry vehicle to the target point at the desired altitude.

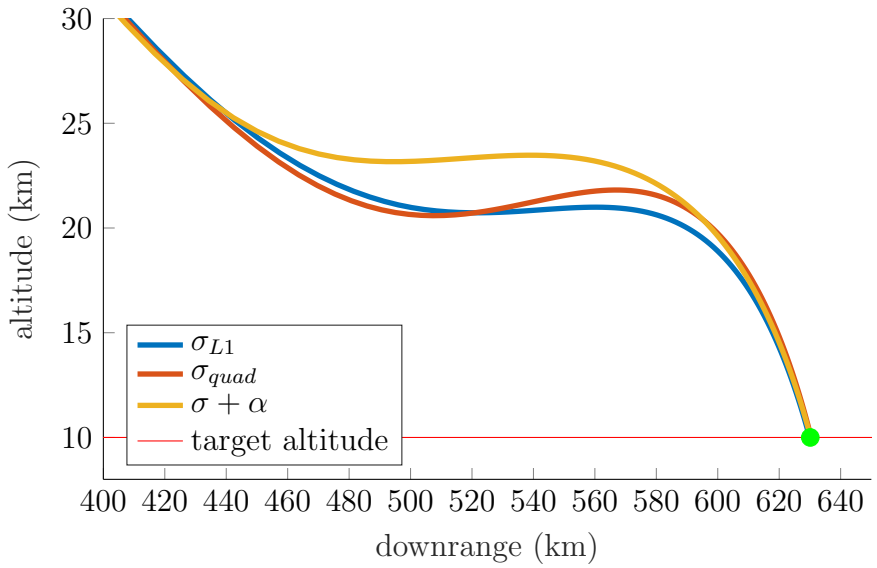


Figure 4.2: Altitude and downrange distance from the converged trajectories from CPEG on the three specified cases. The σ_{L1} case is with bank-angle control and an L1 penalty on bank-angle derivative, σ_{quad} is bank-angle only with a quadratic penalty on bank-angle derivative, and $\sigma + \alpha$ is control over both bank-angle and angle-of-attack. Due to the differences in control authority and cost function, all three converge on different trajectories that hit the target position at parachute deployment.

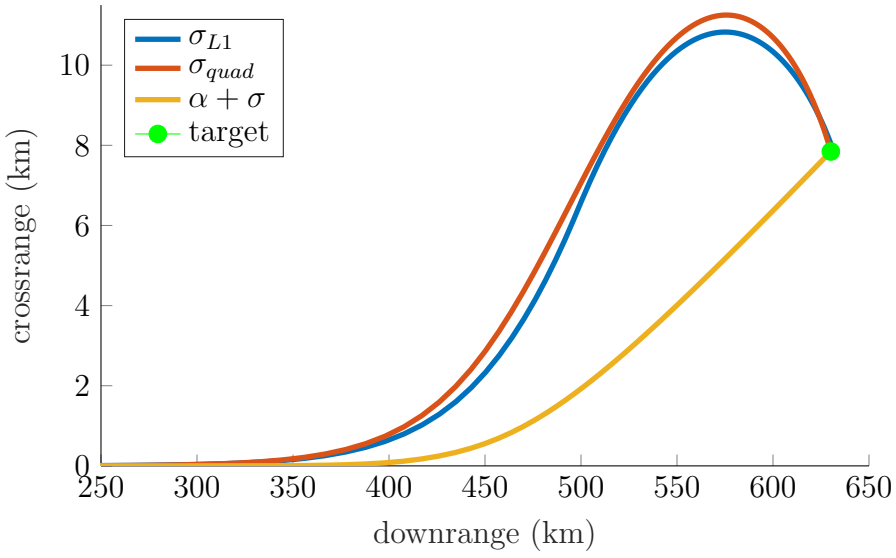


Figure 4.3: Crossrange and downrange trajectory data from the converged trajectories from CPEG on the three specified cases. The cases with only control over the bank-angle have to do a bank reversal to hit the target, whereas the case with control over bank-angle and angle-of-attack is able to leverage the full lift control to avoid the switching.

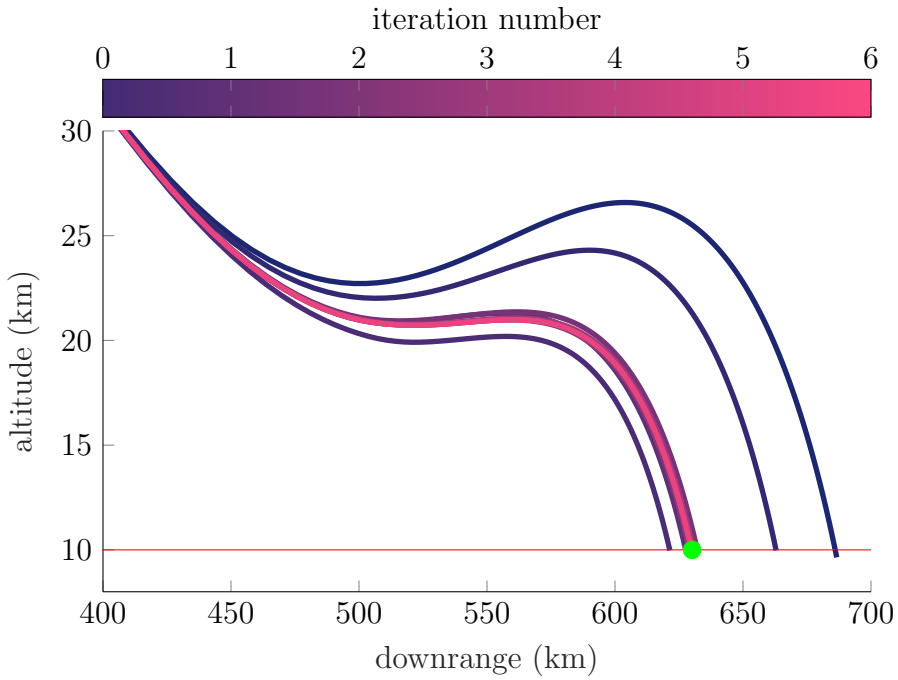


Figure 4.4: Predicted entry vehicle trajectories for the bank-angle only L1 penalty case, as seen by the altitude and downrange data. As the iterates continue, the entry vehicle converges on a trajectory that reaches the target at the 10km altitude mark.

4.5.1 Bank-Angle Control

For the case where the entry vehicle has only bank-angle control, the convergence of CPEG can be observed in figures 4.4 and 4.5 for the case with an L1 cost on the bank-angle derivative, and figures 4.6 and 4.7 with a quadratic cost. These plots show the output of the prediction step of CPEG, where the color of the predictions is blue for the first iteration of the algorithm and turns purple, then pink for later iterations. After convergence, the two bank-angle profiles that CPEG produced for the L1 and quadratic cost functions are shown in figure 4.8. The L1 cost on the derivative of the bank-angle encouraged sparsity in this derivative, resulting in a bank-angle profile that switches between constant bank-angles. For the case with a quadratic cost on the bank-angle derivative, the resulting bank-angle profile is smooth with no discrete switching behavior.

4.5.2 Bank-Angle and Angle-of-Attack Control

As described by the dynamics in equation (4.19), the control input for this case is the lift vector itself in the directions orthogonal to the velocity vector. This allows for manipulation of both the bank-angle and angle-of-attack and is guaranteed to be within the maximum allowable lift by the unit norm constraint in equation (4.49). In this control case, the control input Jacobian is constant and independent of the nominal control plan, making the linearization significantly more accurate than the bank-angle-only case. As a result, the

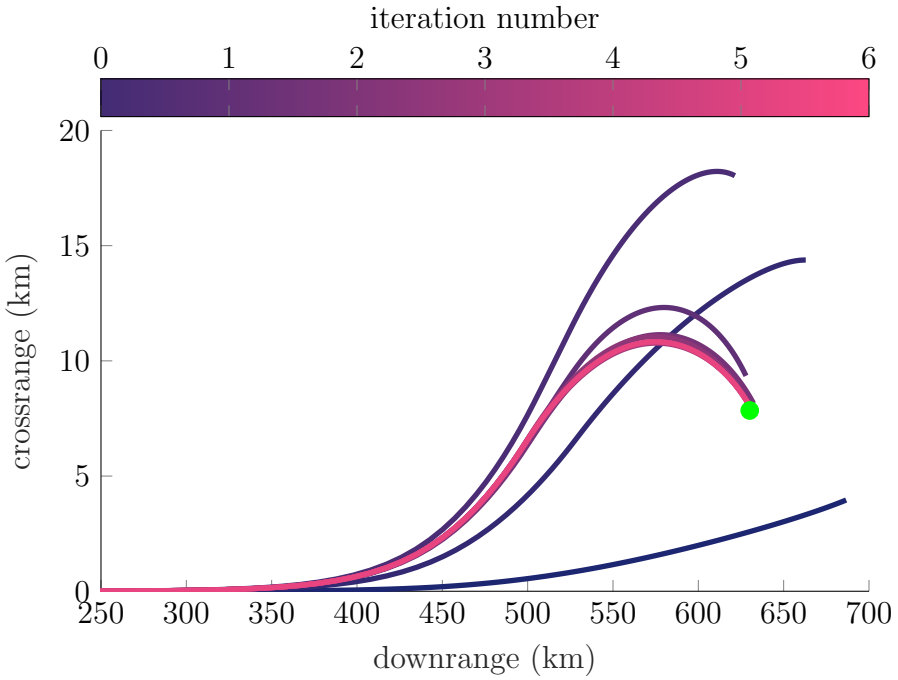


Figure 4.5: Predicted entry vehicle trajectories for the bank-angle only L1 penalty case, as seen by the crossrange and downrange data.

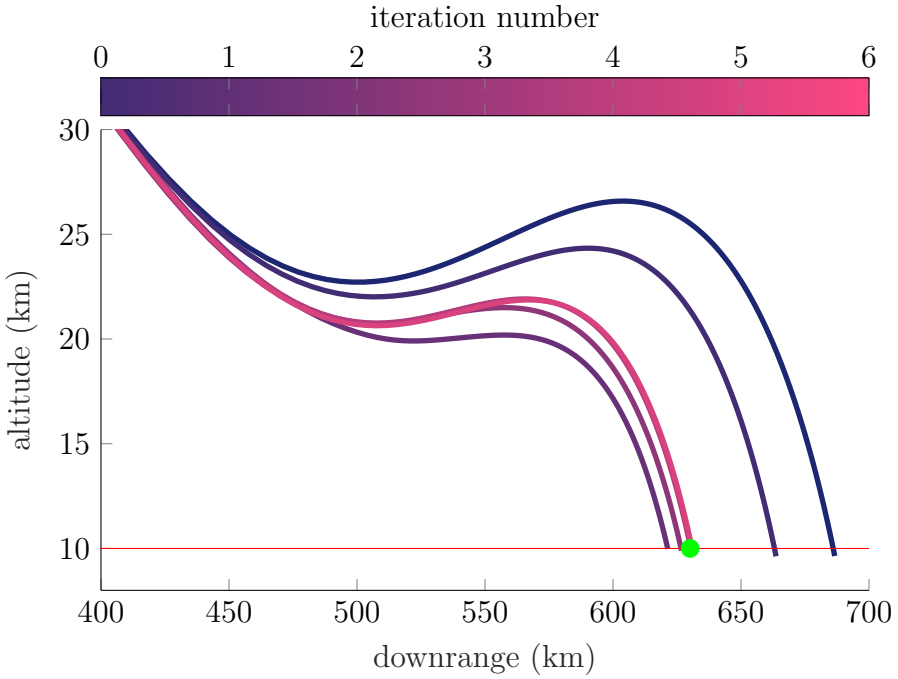


Figure 4.6: Predicted entry vehicle trajectories for the bank-angle only quadratic penalty case, as seen by the altitude and downrange data. The

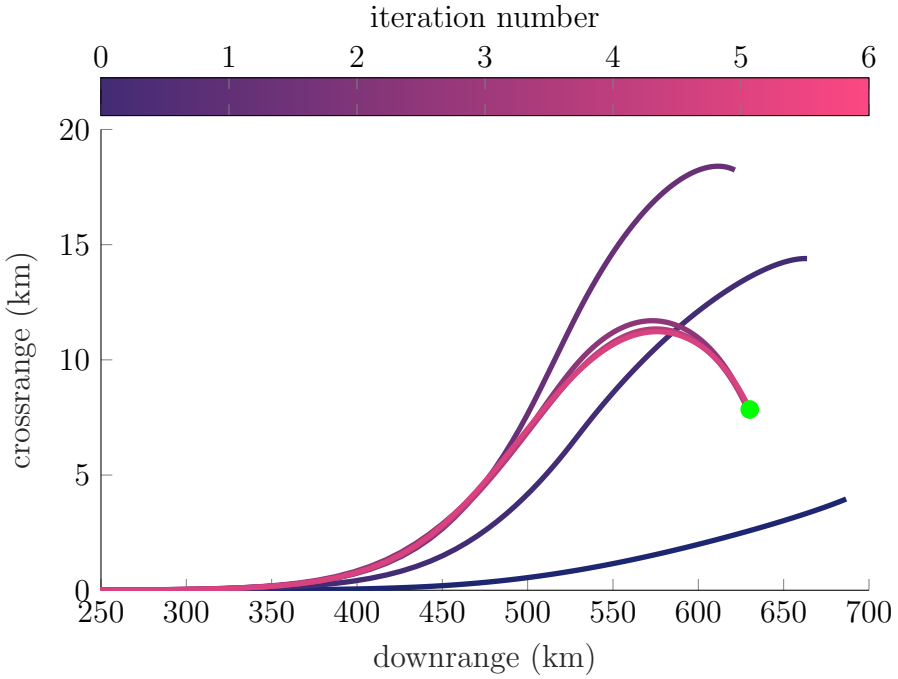


Figure 4.7: Predicted entry vehicle trajectories for the bank-angle only quadratic penalty case, as seen by the crossrange and downrange data.

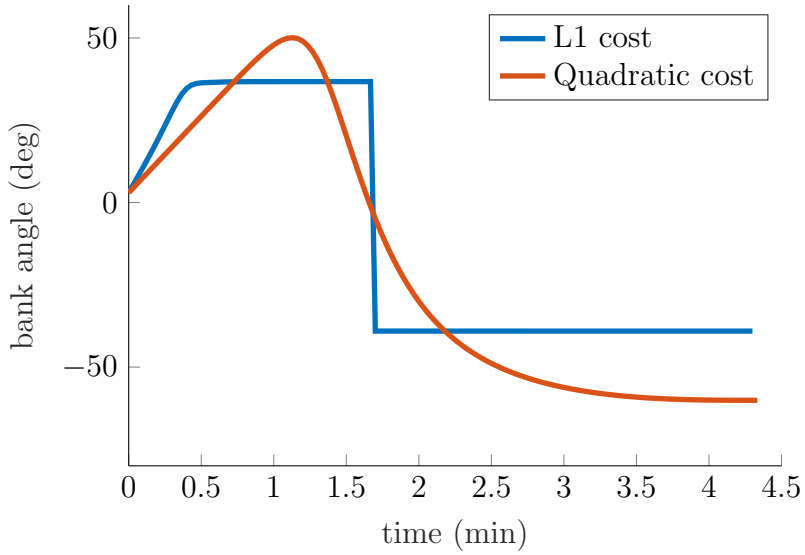


Figure 4.8: Bank-angle only control plans for both the L1 and quadratic cost cases. The L1 cost motivated a bang-bang switching style bank-angle profile. The quadratic cost resulted in a smooth and continuous bank-angle profile.

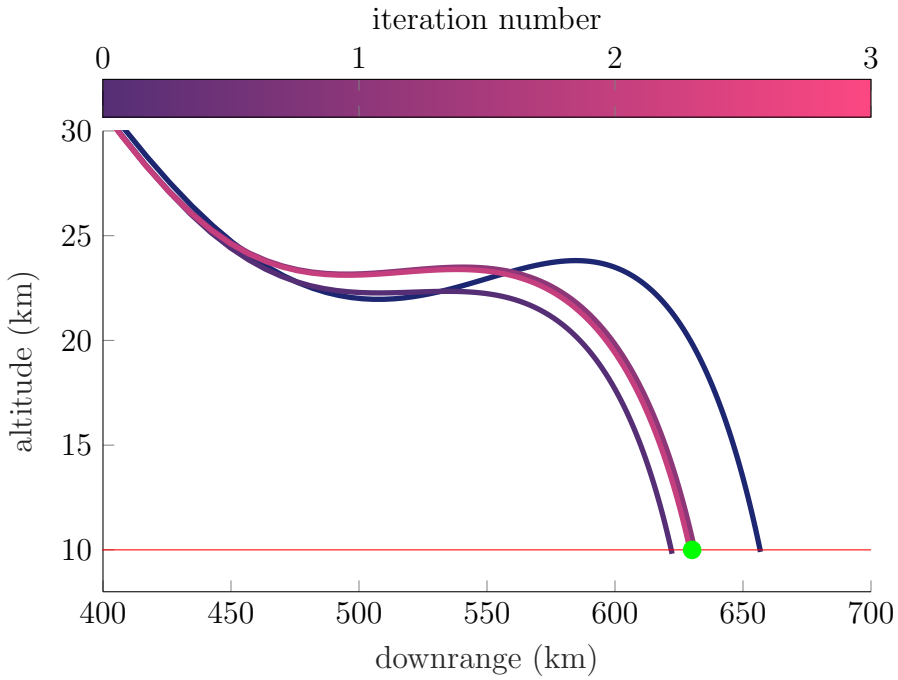


Figure 4.9: Predicted entry vehicle trajectories for the bank-angle and angle-of-attack case, as seen by the altitude and downrange data.

convergence of CPEG with bank-angle plus angle-of-attack control is significantly faster than with the bank-angle alone. The evolution of the predicted trajectories is shown in figures 4.9 and 4.10, with the same coloring scheme as the bank-angle only section. After convergence, the control inputs were converted back into bank-angle and angle-of-attack and shown together in figure 4.11.

4.6 Conclusion

This paper proposes an improved version of the classic predictor-corrector entry guidance scheme in which the correction step is formulated as a convex optimization problem. Two control strategies were tested with CPEG: bank-angle control, and bank-angle plus angle-of-attack modulation. For the bank-angle-only case, cost functions that penalized the derivative with an L1 cost and a quadratic cost were both demonstrated, resulting in dramatically different optimal bank-angle profiles. For the case with both bank-angle and angle-of-attack control, the quality of the dynamics linearization was accurate enough that CPEG was able to converge on an optimal trajectory in just a few iterations. An implementation of CPEG running all of the examples in this paper is available at `https://github.com/RoboticExplorationLab/EntryGuidance.jl`.

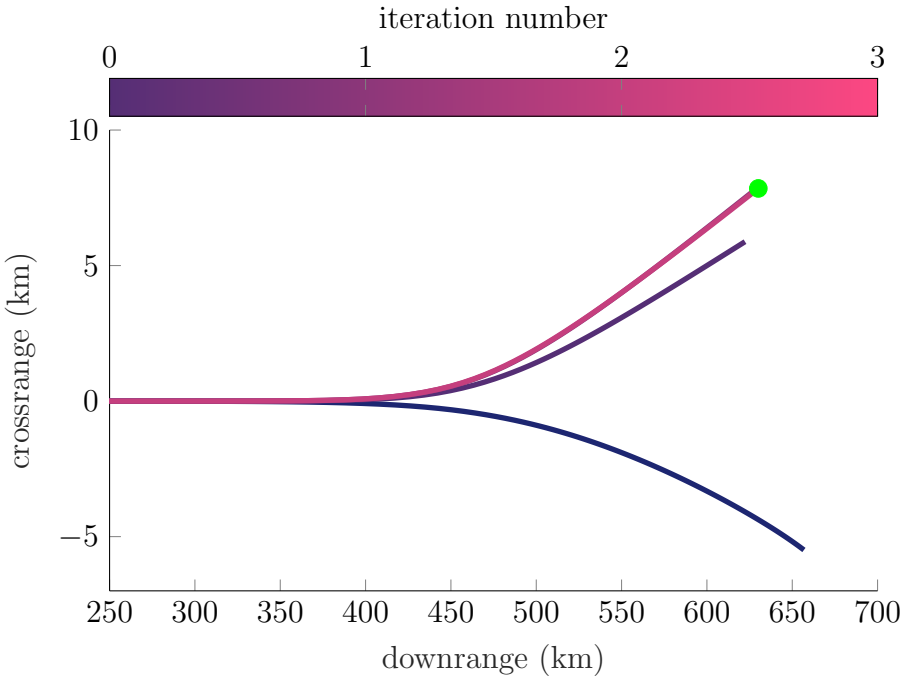


Figure 4.10: Predicted entry vehicle trajectories for the bank-angle and angle-of-attack case, as seen by the crossrange and downrange data.

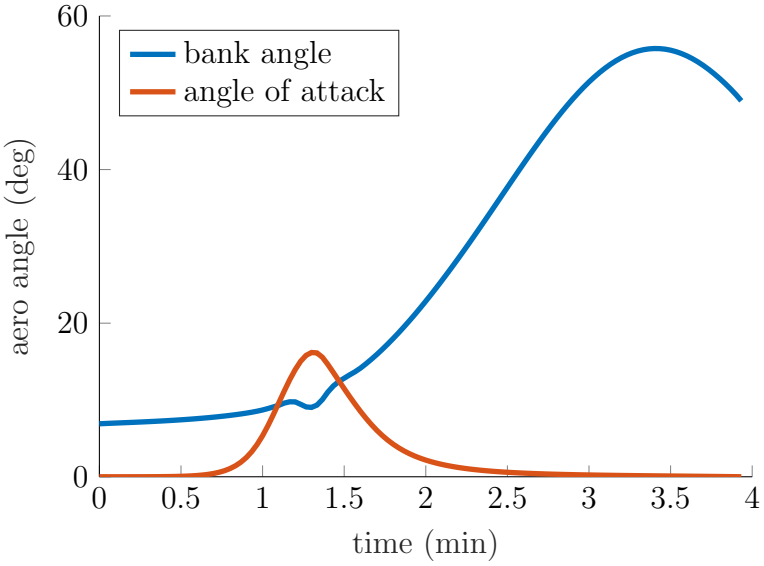


Figure 4.11: Bank-angle and angle-of-attack profiles for the case where both angles are being controlled. CPEG was able to converge on this control plan in just three iterations.

Chapter 5

Robust Entry Guidance with Atmospheric Adaptation

Robust atmospheric entry guidance for blunt-body entry vehicles with bank angle modulation is achieved by combining online atmospheric density estimation with an updated version of the Convex Predictor-corrector Entry Guidance (CPEG) algorithm. During atmospheric entry, a square-root Extended Kalman Filter is used to estimate a ratio between the density of the experienced atmosphere with that of an approximate model, which is spline-fit based on MarsGRAM perturbed data. The information from this filter is used to modify the approximate model used by the guidance algorithm. The proposed update to CPEG includes time as a decision variable, dramatically improving the robustness of the algorithm. CPEG predicts the trajectory at each control call with a nonlinear simulation followed by a single convex trajectory optimization problem that updates the commanded bank angle derivative. The robustness and performance of this estimator and controller guidance architecture are demonstrated on a wide range of realistic Martian atmospheres and is able to achieve state-of-the-art accuracy with respect to altitude-triggered parachute deployment.

The contents of this chapter have been previously published at IEEE Aerospace Conference 2021 in Tracy, Falcone, and Manchester [168]

5.1 Introduction

Entry into the Martian atmosphere refers to the phase of Entry, Descent, and Landing (EDL) that occurs between the point at which the hypersonic entry vehicle first interfaces with the planet's sensible atmosphere and the point at which the supersonic parachute is deployed. It is during this portion of the EDL that the vehicle is subjected to the most demanding flight conditions, during which both peak acceleration and peak heating are experienced [193]. Large uncertainties in models of the Martian atmospheric environment make entry guidance challenging, and significantly degrade landing accuracy, where the landing accuracy is the distance between the landing target site and the actual landing

site. Actual atmospheric density can deviate from predictions by a factor of two or more [85], requiring guidance laws to combat uncertainty with frequent re-planning and reliance on conservative initial entry conditions. These approaches ultimately reduce performance and limit the potential landing sites that can be accessed on Mars [97]. To enable precision landing, which could potentially increase the number of reachable sites of high scientific interest, we propose a new approach to entry guidance that can strategically account for and reduce the atmospheric model uncertainty during guided entry.

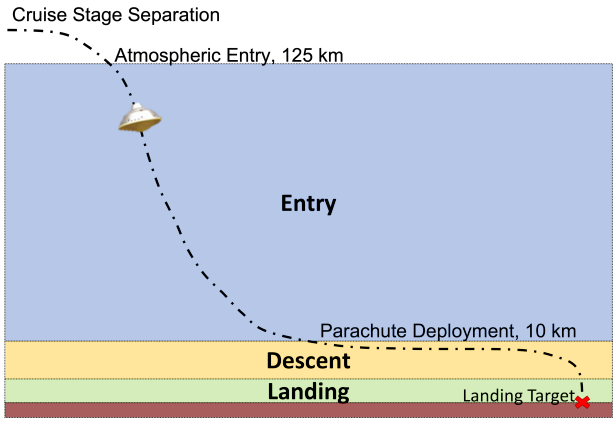


Figure 5.1: Cartoon representation of entry, descent, and landing.

In the last few decades, predictor-corrector guidance algorithms have become the state-of-the-art; these have offered steadily improving landing accuracy with modest computational costs. These methods, like PredGuid [142, 141] and FNPEG [102], predict the trajectory of the entry vehicle with a nonlinear simulation, then use derivative or "sensitivity" information from the simulation to improve or "correct" the control plan [103, 28]. This predictor-corrector framework offers improved landing accuracy compared to Apollo-era entry guidance methods [27] while being relatively simple to implement with low computational complexity. However, a significant drawback of traditional predictor-corrector guidance methods is their inability to generate complex control policies; many such methods are restricted to solving for a static bank angle, which means that the initial bank angle control profile predicted for the whole entry phase is considered constant; however, only during the prediction and correction phase, the guidance uses independent downrange and crossrange control to determine when the sign of the bank angle should flip. This simplicity severely limits the sophistication of the guidance method, and the resulting performance can chatter with multiple aggressive bank angle switches. Another limitation of existing predictor-corrector guidance strategies is their inability to incorporate real-time information about the Martian atmosphere, which is the largest source of uncertainty in the vehicle's dynamics, for prediction and correction purposes.

Recently, there has been a growing interest in trajectory optimization methods for open-loop entry guidance for both Earth and Mars. This approach, in which the guidance problem is formulated as a numerical optimization problem, enables algorithms that can reason about the dynamics of the vehicle and constraints like actuator limits, state bounds,

and heating limits. These optimization problems can be solved by a variety of numerical methods. As an example, sequential convex programming, in which a non-convex optimization problem is iteratively convexified and solved until convergence, has recently become popular [189, 192, 107, 159, 160, 106]. Alternatively, the trajectory optimization problem can be converted into a nonlinear program [17] and solved by a variety of off-the-shelf solvers, like SNOPT [61] or Ipopt [187], or by a more specialized trajectory optimization solver like ALTRO [79, 82]. While these trajectory optimization guidance methods offer more sophisticated and performant control plans than classical predictor-corrector methods, they can be prohibitively complex to implement on resource-constrained flight computers, and are still unable to account for real-time information about the atmosphere.

A middle ground between simple predictor-corrector guidance schemes and offline trajectory optimization is the Convex Predictor-corrector Entry Guidance algorithm (CPEG) [177]. This method uses the same predictor-corrector framework as existing methods, but instead of solving for a simple static bank angle, it forms a convex trajectory optimization problem that solves for the correction to the nominal control plan. This approach enables the direct inclusion of any actuator or safety constraints and the ability to reason about complex control policies and the dynamics of the vehicle over the whole trajectory. By forming the convex trajectory optimization problem with a dynamics model linearized about the prediction rollout, the problem is guaranteed to always be feasible, and the convexity of the problem guarantees that an optimal solution can be computed in polynomial time [24].

This work proposes an improvement to the CPEG algorithm that includes time as an optimization variable, allowing for the generation of trajectories with a free final time. This modification allows for significantly more flexibility when handling scenarios with multiple potentially successful control policies, since many trajectories can all be expressed using the same number of optimization variables. The absence of this feature in the original CPEG algorithm resulted in a lack of robustness when run online with significant model uncertainty. Additionally, this work directly addresses the main source of the uncertainty in the vehicle dynamics – the atmospheric density – by estimating information about the atmosphere online and including it in the approximate model used by CPEG. Our specific contributions include:

1. A free-final-time extension of the Convex Predictor-corrector Entry Guidance algorithm
2. An estimation architecture for atmospheric density and an atmospheric approximate model
3. Validation of the combined controller and estimator on realistic density and wind dispersions from MarsGRAM [85]

This paper proceeds by first describing the dynamics of the entry vehicle in Cartesian coordinates in Section 5.2, followed by the free-final-time version of CPEG in Section 5.3. The atmospheric density estimation approach is detailed in Section 5.4. Numerical experiments using Monte-Carlo sampling of atmospheric parameters from MarsGRAM are presented in Section 5.5. Finally, Section 5.6 summarizes our conclusion.

5.2 Entry Vehicle Dynamics

This work considers a blunt-body entry vehicle in the Martian atmosphere with the ability to modulate its bank angle $\sigma \in \mathbf{R}$. These entry vehicles are traditionally described and simulated in spherical coordinates using the "Vinh" model [30, 185, 186, 192, 190, 102, 53]. While this model is ubiquitous and well-studied, it suffers from numerical scaling issues due to the state vector including angles as well as distance and velocity, and is highly nonlinear in both the state and the control input. These shortcomings make optimization challenging and numerically unreliable. Alternatively, the state and dynamics of the entry vehicle can be equivalently described in a Mars-fixed Cartesian reference frame. This parametrization is more common in the powered-descent guidance literature [20, 3, 2], and has much better numerical scaling, as well as linear kinematics, both of which make it a better fit for optimization-based control methods as detailed in [171]. The dynamics of the entry vehicle with this state representation are the following:

$$\dot{r} = v, \tag{5.1}$$
$$\dot{v} = a_L + a_D + a_g - 2(\omega \times v) - \omega \times (\omega \times r). \tag{5.2}$$

The aerodynamic accelerations from lift and drag are dependent on the vehicle velocity relative to the atmosphere, including wind. The wind vector w is expressed in the Mars-fixed frame, and the relative velocity is simply $v_{rel} = v + w$. The magnitude of the lift and drag accelerations can then be computed as follows:

$$L = \frac{1}{2m}\rho A C_L \|v_{rel}\|^2, \tag{5.3}$$
$$D = \frac{1}{2m}\rho A C_D \|v_{rel}\|^2. \tag{5.4}$$

The hypersonic coefficients, C_L and C_D have been evaluated through the Newtonian flow theory for a blunted body, specifically:

$$C_A = (1 - \sin\delta^4)\frac{r_{nose}^2}{r_{base}^2} + (2\sin\delta^2\cos\alpha^2 + \cos\delta^2\sin\alpha^2)\left(1 - \frac{r_{nose}^2}{r_{base}^2}\cos\delta^2\right), \tag{5.5}$$
$$C_N = \left(1 - \frac{r_{nose}^2}{r_{base}^2}\cos\delta^2\right)\cos\delta^2\sin 2\alpha, \tag{5.6}$$
$$C_L = C_N\cos\alpha - C_A\sin\alpha, \tag{5.7}$$
$$C_D = C_A\cos\alpha + C_N\sin\alpha, \tag{5.8}$$

where C_A is the axial force coefficient, C_N is the normal force coefficient, δ is the cone angle of the blunted-cone vehicle, α is the angle of attack, r_{base} is the base radius, r_{nose} is the nose radius of the hypersonic vehicle.

The direction of the drag acceleration is in the opposite direction of the relative velocity, expressed as:

$$a_D = -D\frac{v_{rel}}{\|v_{rel}\|}. \tag{5.9}$$

The lift acceleration is directed using the bank angle σ, and is modeled and represented with two basis vectors $\hat{e}_1 \in \mathbf{R}^3$ and $\hat{e}_2 \in \mathbf{R}^3$, where together they span the plane orthogonal to the relative velocity of the vehicle in the atmosphere. These aerodynamic basis vectors are constructed as the following:

$$\hat{e}_1 = \frac{r \times v_{rel}}{\|r \times v_{rel}\|}, \tag{5.10}$$

$$\hat{e}_2 = \frac{v_{rel} \times \hat{e}_1}{\|v_{rel} \times \hat{e}_1\|}. \tag{5.11}$$

Using these basis vectors, the lift acceleration can be computed as the following:

$$a_L = L(\sin(\sigma)\hat{e}_1 + \cos(\sigma)\hat{e}_2). \tag{5.12}$$

The acceleration due to gravity can be calculated using any number of established methods ranging in fidelity from simple two-body gravity [123], to a more complex spherical harmonic expansion of the geopotential [71, 57].

The largest amount of uncertainty in the dynamics of the entry vehicle comes from the atmosphere. Both the atmospheric density ρ, as well as the wind vector w are highly uncertain. To visualize this uncertainty, Mars GRAM [85] was used to generate dispersions for these two parameters, which are shown in Fig. 5.2 and Fig. 5.3. A summary of the parameters used to run Mars GRAM is presented in Table 5.1. For the generation of atmospheric data points in highly perturbed environments, Mars GRAM not only uses random Monte Carlo samples that re-initialize the random number seed (NR1) for each sample but also has been set for different zoffsets, which are constant height offsets that modify the atmospheric density. Specifically, positive offsets increase density, while negative offsets decrease density. In addition, rpscale, the random perturbation scale was also set to the maximum, with increasing this factor intensifying the magnitude of the perturbation. The data have also been evaluated in the context of a global dust storm, through INTENS, which changes the dust storm intensity.

Table 5.1: Mars GRAM nominal and dispersion parameters

Parameter	Value	Parameter	Value
Initial Altitude, km	125	NR1	[1,5001]
Date	6 August 2012	zoffset, km	Uniform Dist., [-3.25,3.25]
Time	5:30:00	rpscale	2
Final Altitude, km	10	INTENS	2

Since aerodynamic acceleration is the dominant term in the dynamics at the high speeds an entry vehicle experiences, the uncertainty in the atmospheric density results in a highly uncertain dynamics model. The uncertainty in the wind contributes to the model uncertainty, but less so than the density.

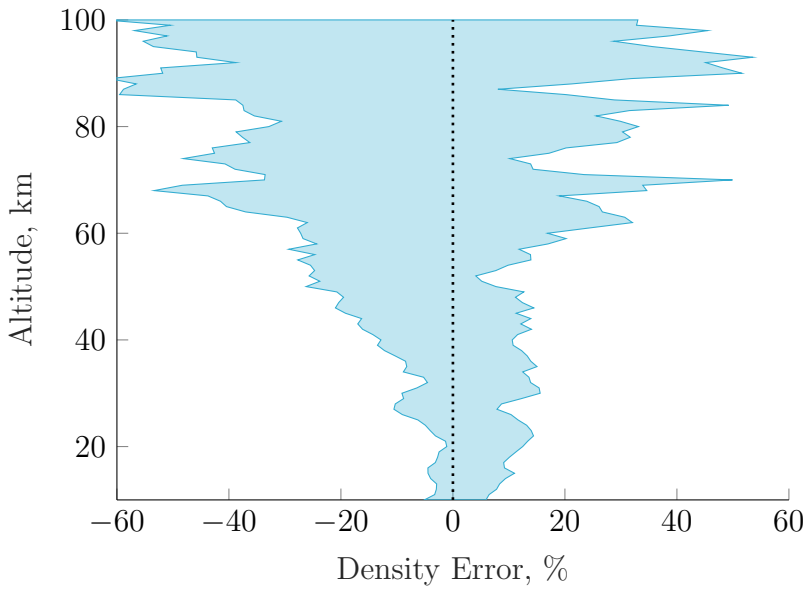


Figure 5.2: Dispersion of the atmospheric density as a function of altitude calculated with Mars GRAM. The dynamics of the entry vehicle are heavily influenced by this density, and the dramatic uncertainty of this value motivates the need for more robust guidance schemes.

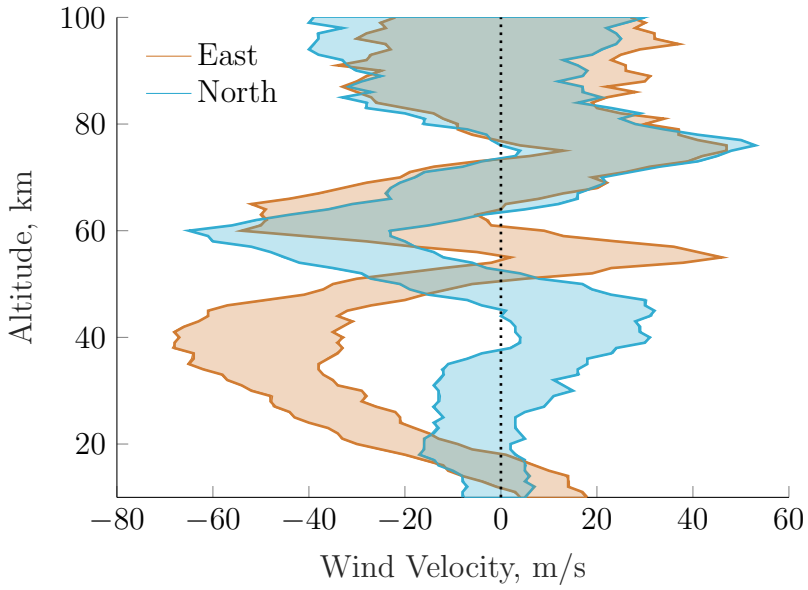


Figure 5.3: Dispersions of the east and north wind velocities in the Martian atmosphere as a function of altitude calculated with Mars GRAM. This range of atmospheric wind profiles is used in the Monte Carlo testing of the proposed entry guidance algorithm.

5.3 CPEG

Predictor-corrector guidance algorithms are a simple and effective way to control underactuated systems like the entry vehicle. These methods all share a general framework in which a nominal control policy is used to simulate a predicted trajectory, then a correction is made to the nominal control policy based on this prediction. Predictor-corrector methods have been the state of the art for entry guidance of blunt-body vehicles, but they are not without their shortcomings. Existing methods like FNPEG are only able to reason about simple static bank angle policies, and rely on independent crossrange and downrange logic to modulate the bank angle [103]. Alternatively, offline trajectory optimization allows for more complex control policies and the direct inclusion of constraints, but is significantly more expensive than predictor-corrector methods. A trajectory optimization problem in its most general form is the following,

$$\begin{aligned} \underset{x_{1:N},\, u_{1:N-1}}{\text{minimize}} \quad & \ell_N(x_N) + \sum_{k=1}^{N-1} \ell_k(x_k, u_k) \\ \text{subject to} \quad & x_{k+1} = f(x_k, u_k) \qquad k = 1 \dots N-1, \\ & g_k(x_k, u_k) \leq 0 \qquad k = 1 \dots N-1, \\ & u_{min} \leq u_k \leq u_{max} \quad k = 1 \dots N-1, \\ & x_{min} \leq x_k \leq x_{max} \quad k = 1 \dots N \end{aligned} \tag{5.13}$$

where x is the state, u is the control, ℓ is the cost function, $f(x, u)$ is the discrete dynamics function, and $g(x, u)$ is a general constraint function. Problem (5.13) can be solved with a multitude of available trajectory optimization solvers, but they are prohibitively complex to run on board the entry vehicle.

5.3.1 Baseline CPEG

The Convex Predictor-corrector Entry Guidance (CPEG) algorithm, introduced in [171], seeks to find a middle ground between simple predictor-corrector guidance methods and offline trajectory optimization. This is achieved by adopting the predictor-corrector framework, but solving a convex trajectory optimization problem for the correction instead of a simple update to a static bank angle like in FNPEG. This approach allows CPEG to reason about sophisticated control trajectories as well as the dynamics of the vehicle throughout the entirety of the trajectory. The convex optimization problem that makes up the correction portion of the algorithm is guaranteed to be feasible, and its optimum can be solved for at real-time rates [115].

In order for problem (5.13) to be convex, the cost and constraint functions must be convex. This means that problems with nonlinear dynamics, like the entry vehicle, must be approximated. We linearize the discrete dynamics function $f(x, u)$ about the predicted trajectory $(\bar{x}, \bar{u})$:

$$\bar{x}_{k+1} + \delta x_{k+1} \approx f(\bar{x}_k, \bar{u}_k) + A_k \delta x_k + B_k \delta u_k, \tag{5.14}$$

where A_k and B_k are the following Jacobians,

$$A_k = \left.\frac{\partial f(x_k, u_k)}{\partial x_k}\right|_{\bar{x}_k, \bar{u}_k}, \tag{5.15}$$

$$B_k = \left.\frac{\partial f(x_k, u_k)}{\partial u_k}\right|_{\bar{x}_k, \bar{u}_k}. \tag{5.16}$$

Since the prediction was dynamically feasible, and $\bar{x}_{k+1} = f(\bar{x}_k, \bar{u}_k)$, equation (5.14) is reduced to

$$\delta x_{k+1} \approx A_k \delta x_k + B_k \delta u_k. \tag{5.17}$$

From here, the convex correction problem is posed as,

$$\begin{aligned}
\underset{\delta x_{1:N},\, \delta u_{1:N-1}}{\text{minimize}} \quad & \ell_N(\bar{x}_N + \delta x_N) + \sum_{k=1}^{N-1} \ell_k(\bar{x}_k + \delta x_k, \bar{u}_k + \delta u_k) \\
\text{subject to} \quad & \delta x_{k+1} = A_k \delta x_k + B_k \delta u_k && k = 1 \ldots N-1, \\
& g_k(\bar{x}_k + \delta x_k, \bar{u}_k + \delta u_k) \leq 0 && k = 1 \ldots N-1, \\
& u_{min} \leq u_k \leq u_{max} && k = 1 \ldots N-1, \\
& x_{min} \leq x_k \leq x_{max} && k = 1 \ldots N, \\
& \delta u_{min} \leq \delta u_k \leq \delta u_{max} && k = 1 \ldots N-1, \\
& \delta x_{min} \leq \delta x_k \leq \delta x_{max} && k = 1 \ldots N,
\end{aligned} \tag{5.18}$$

where the bounds on δx and δu serve as a customizable trust region that ensures the step direction stays within a region of the state space where the linearization is accurate [130]. Since the linearization is evaluated about the dynamically feasible predicted trajectory, this optimization problem is guaranteed to have a feasible solution of all zeros for δx and δu, eliminating one of the largest vulnerabilities in real-time optimization-based control.

The full baseline CPEG algorithm is shown in Algorithm 9, where an initial condition and nominal control policy are used to simulate the entry vehicle until parachute deployment (the prediction), a linearized model is evaluated about this predicted trajectory, and a convex trajectory optimization problem (5.18) is solved for the correction.

5.3.2 Variable-Time CPEG

We now extend CPEG to handle free-final-time problems where time is a decision variable. In the baseline algorithm, the prediction is done with a fixed time step and the simulation simply terminates when a parachute deployment trigger is satisfied. This means that the corrected control policy can result in a subsequent prediction with a different number of time steps. This characteristic of the baseline algorithm results in decreased robustness when there are multiple cost-equivalent trajectories in the cost landscape. In this case, CPEG can chatter between two or more control policies that are of different lengths but have equal cost.

To address this issue, an updated version of CPEG is proposed in this work that treats time as an explicit optimization variable, allowing for the time of parachute deployment to effectively be decided by the optimization solver, instead of solely by the prediction step. This updated version of CPEG is significantly more robust to chattering. The free-final-time modification can be incorporated into the existing CPEG framework by simply including the time step for each knot point as a control input. The resulting state variable $x \in \mathbf{R}^7$ and control variable $u \in \mathbf{R}^2$ are the following:

$$x = [r^T, v^T, \sigma]^T, \tag{5.19}$$

$$u = [\dot{\sigma}, \Delta t]^T, \tag{5.20}$$

where $\Delta t \in \mathbf{R}$ is the size of the time step. We use the following cost function:

$$J(x, u) = \gamma \|r_N - r_{\text{target}}\|^2 + \sum_{k=1}^{N-1} \beta \dot{\sigma}_k^2 + (\Delta t_k - \Delta t_{\text{target}})^2, \tag{5.21}$$

where $\gamma \in \mathbf{R}_+$ and $\beta \in \mathbb{R}_+$ are positive cost weights, $r_{\text{target}} \in \mathbf{R}^3$ is the target position, and $\Delta t_{\text{target}} \in \mathbf{R}_+$ is target time-step size. These weights should be tuned such that the largest penalty is applied to the terminal position accuracy, and a sufficient penalty exists on the time-step size to keep the time steps positive and reasonable.

Each control correction requires the solution of a convex Quadratic Program (QP), of which there are many fast and robust solution methods available. This work utilized a custom implementation of the primal-dual interior point solver detailed in [115] and [183]. At the start of the QP solver, the inequality constraints are temporarily omitted and the closed-form solution to the equality-constrained QP is checked for feasibility. In many cases, the inequality constraints are inactive at the optimum, and this approach is able to solve the QP at the cost of solving a single linear system.

5.4 Atmospheric Estimation

During atmospheric entry guidance, the largest source of uncertainty in the dynamics comes from the atmosphere. The density of the atmosphere is highly variable based on the time of the year, weather, dust conditions, temperature, and space weather. Mars GRAM [85] is the highest fidelity tool available for the simulation of the Martian atmosphere, and it was used to sample 5,000 potential atmospheric density profiles. 1,000 of these profiles are shown in Fig. 5.2, where there is significant uncertainty at higher altitudes, with better certainty at lower altitudes, though still varying by $10 - 20\%$. Because of this uncertainty, open-loop control policies perform poorly.

To manage this uncertainty, we propose a simple and effective method for estimating critical information about the atmospheric density online during the entry process. The density of the Martian atmosphere roughly follows an exponential law, such that the difference in density between the surface and 125 km altitude, which is considered in this study as the entry interface, spans several orders of magnitude. Because of this large variation in

magnitude, estimating the density directly in a Kalman Filter is numerically challenging. Another possible approach to tackle atmospheric uncertainties would be to directly model the atmosphere as an exponential and estimate the surface density and scale height [111], but this approach suffers from severe ill-conditioning because the exponential dramatically amplifies errors at higher altitudes [69, 45].

In place of estimating some parametrized version of the atmospheric density profile from scratch, we use an approach similar to [144] where a Kalman filter for estimating the ratio between the observed and nominal atmospheres:

$$k_\rho = \frac{\rho_{\text{observed}}}{\rho_{\text{nominal}}}, \tag{5.22}$$

where ρ_{nominal} is computed offline and stored. Estimating k_ρ directly is better behaved numerically because the estimator is simply looking to modify an existing approximate density profile, and the estimated value will always be near 1, regardless of altitude. The state and control for this filter are the following:

$$x_{\text{kf}} = [r^T, v^T, \sigma, k_\rho]^T, \tag{5.23}$$

$$u_{\text{kf}} = \dot{\sigma}, \tag{5.24}$$

where the dynamics for this state representation are the same as in section 5.2, with the exception that $\rho = k_\rho \cdot \rho_{\text{approximate}}$. The measurement model is assumed to come directly from the navigation system, with additive Gaussian white noise. To estimate k_ρ effectively in the presence of highly variable dynamic pressure, a Square-Root Extended Kalman Filter (SQEKF) is used that allows for twice the dynamic range of the standard Extended Kalman Filter (EKF) [88]. This increased range and numerical robustness help to manage the potentially ill-conditioned covariance matrices present during entry guidance. A simple version of an SQEKF is demonstrated in [167] that leverages a highly mature QR decomposition routine to handle a majority of the computation [154].

The SQEKF takes a system of the following form:

$$x_{k+1} = f(x_k, u_k) + w_k, \tag{5.25}$$

$$y_k = g(x_k) + v_k, \tag{5.26}$$

where $f(x, u)$ is the discrete time dynamics function, $g(x)$ is the measurement function, and $w \sim \mathcal{N}(0, W)$ and $v \sim \mathcal{N}(0, V)$ are the process and sensor noise terms, respectively. The SQEKF is able to achieve better precision than the standard EKF by representing covariance matrices by their upper-triangular matrix square roots (Cholesky factors) [149, 182, 14]. Using this notation, the SQEKF algorithm from [167] is shown in Algorithm 10, where Γ_W and Γ_V are the upper triangular Cholesky factorizations of W and V, and the function qr_r returns the upper triangular R factor from a QR decomposition [154, 79]. The mean of the estimate is $\mu_{t|t}$, and the covariance Σ is represented with its matrix square root F such that $F^T F = \Sigma$. The performance of this filter on a realistic atmosphere sample from Mars GRAM is shown in Fig. 5.4.

Algorithm 10 Square Root Extended Kalman Filter

function sqkf($\mu_{t|t}, F_{t|t}, u_t, y_{t+1}, \Gamma_W, \Gamma_V$)

$A_t = \partial f/\partial x|_{\mu_{t|t}, u_t}$ ▷ dynamics Jacobian

$\mu_{t+1|t} = f(\mu_{t|t}, u_t)$ ▷ state prediction

$F_{t+1|t} = \mathrm{qr}_\mathrm{r}(F_{t|t}A_t^T, \Gamma_W)$ ▷ covariance prediction

$C_t = \partial g/\partial x|_{\mu_{t+1|t}}$ ▷ measurement Jacobian

$z = y_{t+1} - g(\mu_{t+1|t})$ ▷ measurement innovation

$G = \mathrm{qr}_\mathrm{r}(F_{t+1|t}C_t^T, \Gamma_V)$ ▷ innovation covariance

$L = [G^{-1}(G^{-T}C_t)F_{t+1|t}^T F_{t+1|t}]^T$ ▷ Kalman gain

$\mu_{t+1|t+1} = \mu_{t+1|t} + Lz$ ▷ state update

$F_{t+1|t+1} = \mathrm{qr}_\mathrm{r}(F_{t+1|t}(I - LC_t)^T, \Gamma_V L^T)$ ▷ covariance update

return$(\mu_{t+1|t+1}, F_{t+1|t+1})$

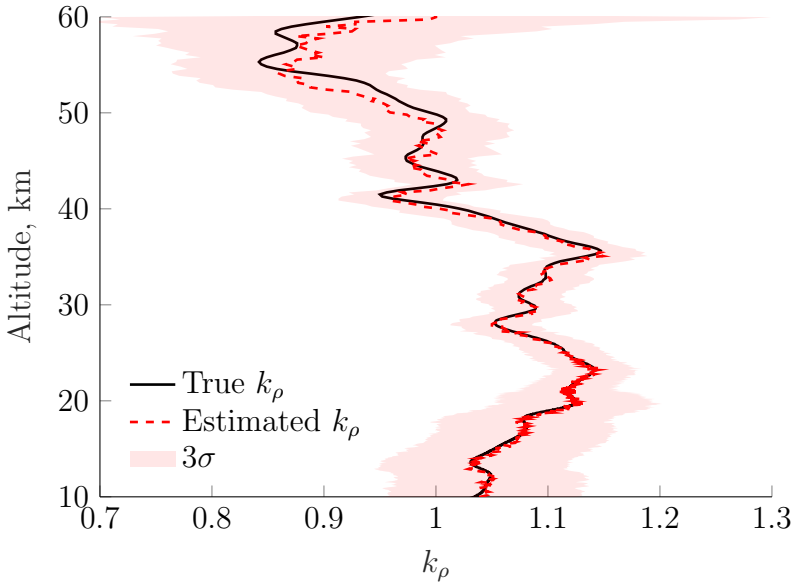


Figure 5.4: Results from the atmospheric density estimator during entry guidance where the ratio between the actual and nominal densities, k_ρ, is shown alongside the true value. There is large uncertainty in the higher altitudes where the atmosphere is very thin, and again at lower altitudes where the velocity of the vehicle is such that aerodynamic forces are not as dominant in the dynamics.

5.5 Numerical Experiments

To validate the combined controller and estimator architecture, 1,000 Monte Carlo runs were used to test the robustness and performance of the algorithms. Each Monte Carlo run consists of an atmospheric density and wind sampled from Mars GRAM, as shown in Fig. 5.2 and Fig. 5.3. An initial position dispersion with a standard deviation of 1 km and initial bank angle dispersion of 5 degrees was used to further disambiguate between runs. The initial conditions coincide with the Mars Sample Return (MSL) initial conditions; specifically, the initial position had a mean altitude of 125 km, with a flight path angle of -15.5 degrees and entry velocity of 5.85 km/s [26, 193]. The target was 632 km down range and 7.9 km cross range, with a parachute deployment triggered at 10 km altitude. Furthermore, the hypersonic vehicle considered has the same characteristics of the MSL vehicle; specifically, the vehicle is a 70 degrees sphere-cone, nose radius equal to 1.125 m, a base radius equal to 2.25 m, and an entry mass of 2800 kg [26].

The variable-time CPEG algorithm was run with a nominal time step size of 2 seconds until the length of the trajectory was less than 50 knot points, at which point the nominal time step was lowered to 0.1 seconds. This algorithm was run at 5 Hz with one convex correction solution at each time step, 94% of which were solved in a single iteration of the QP solver. A trust region was used to keep the updated trajectory in an area of the state space where the linearized dynamics model is still accurate, with $|\delta\sigma| \leq 20^\circ$ and $|\delta\Delta t| \leq 0.1$. The SQEKF was initialized at an altitude of 60 km, and the estimated k_ρ value was used in the dynamics model in CPEG. The measurement model in the SQEKF assumed full state observations from the navigation system, with conservative measurement error standard deviations of 100 meters for the position, and 20 cm/s for the velocity.

All 1,000 Monte Carlo runs were successful in deploying the parachute within 1km of the target, with the trajectories shown in Fig. 5.5 and Fig. 5.6. Depending on the specific atmosphere and initial perturbation, there were two families of trajectories that the entry vehicle traversed. The bank-angle profiles for these trajectories are shown in Fig. 5.8. The terminal errors for these runs are shown in Fig. 5.7, with each individual run shown as well as a 3σ ellipse.

To demonstrate the effectiveness of the atmospheric adaptation, the same 1,000 Monte Carlo runs were executed with and without adaption and the errors are shown in Fig. 5.10. The terminal errors for the runs with adaptation have significantly lower errors than the same runs without adaptation. The statistics from these terminal errors are shown in Table 5.2, showing a 37% decrease in the mean error, a 33% reduction in median error, and a 49% smaller maximum error with the atmospheric adaptation turned on.

Table 5.2: Comparison of the parachute deployment error for variable time CPEG with and without the atmospheric adaptation. With the adaptation on, the mean, median, and maximum errors are all reduced.

	Mean Error	Median Error	Maximum Error
Adaptation Off	0.353 km	0.287 km	1.485 km
Adaptation On	**0.223** km	**0.193** km	**0.758** km

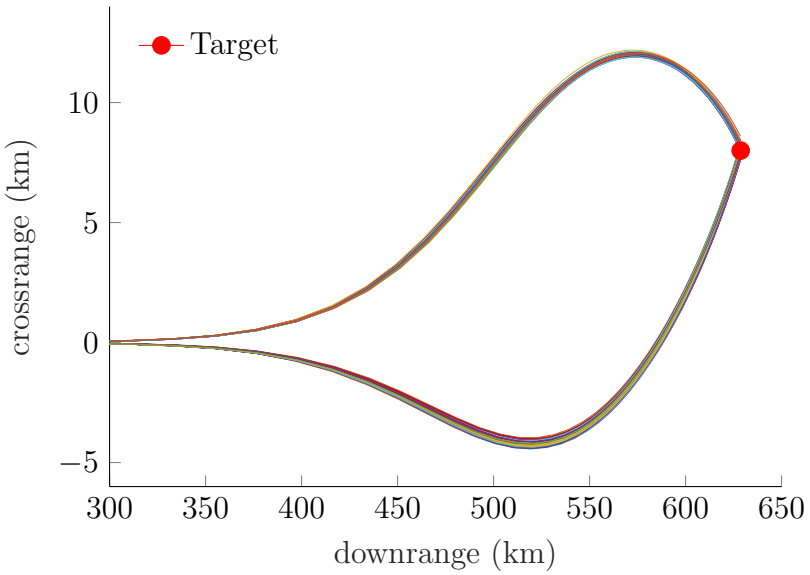


Figure 5.5: Downrange and crossrange distances from the point of atmospheric interface for 1,000 Monte Carlo runs with variable-time CPEG and atmospheric estimation. Depending on the initial conditions and the atmosphere, not all the trajectories take the same route.

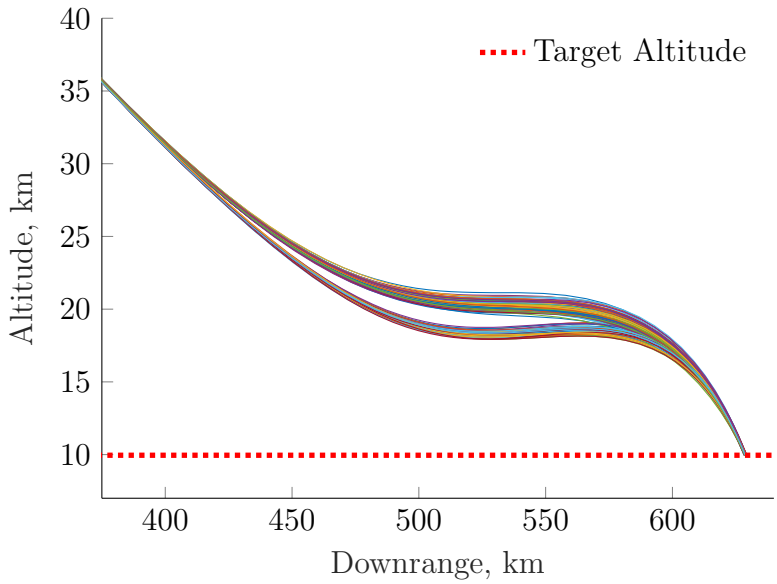


Figure 5.6: Altitude and downrange history for 1,000 Monte Carlo runs that terminate at an altitude of 10km. Based on the initial perturbation and atmosphere, there are two common paths that the entry vehicle takes to get to the target, one maintaining a higher altitude than the other.

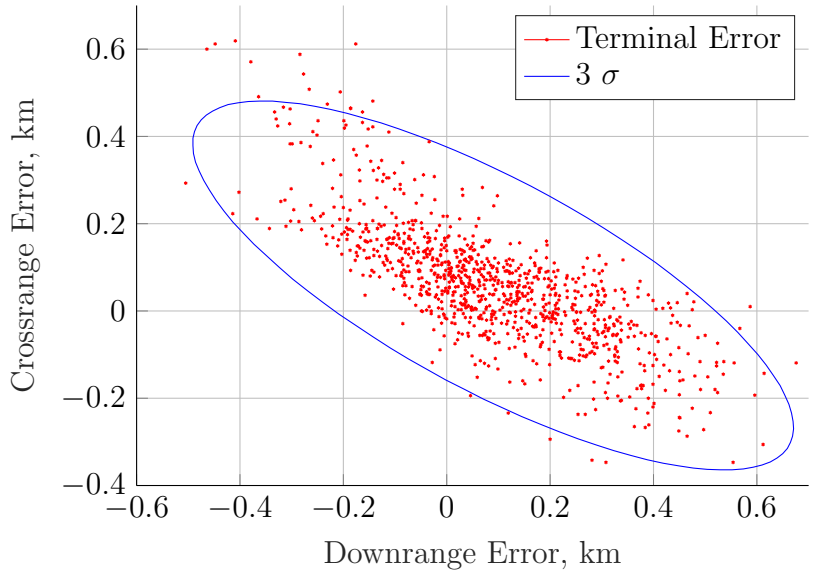


Figure 5.7: Terminal errors for parachute deployment for the 1,000 Monte Carlo runs as shown in downrange and crossrange errors. Each terminal error is shown, as well as an ellipse denoting the 3σ bounds.

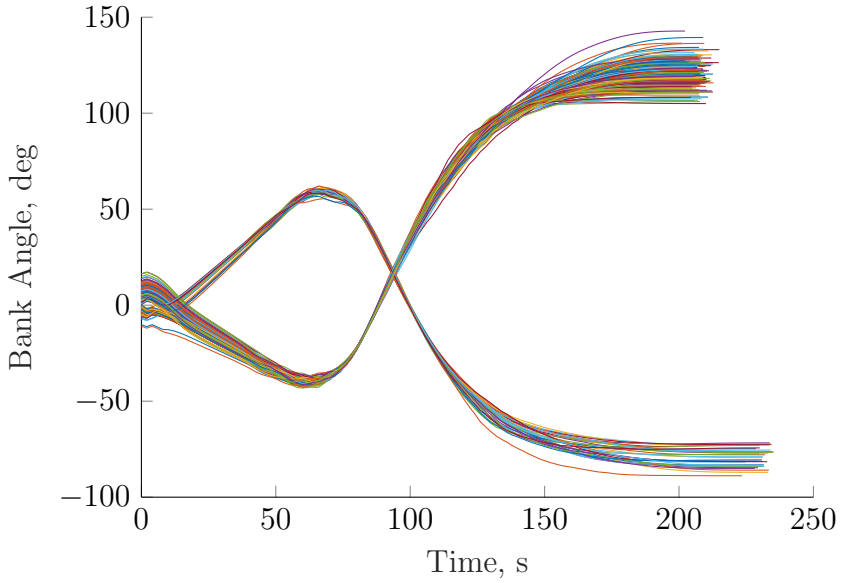


Figure 5.8: Bank-angle profiles for the 1,000 Monte Carlo runs. The initial bank angle varies as it was part of the dispersion, and depending on this initial condition and the atmosphere sampled during the run, one of two families of control approaches was taken.

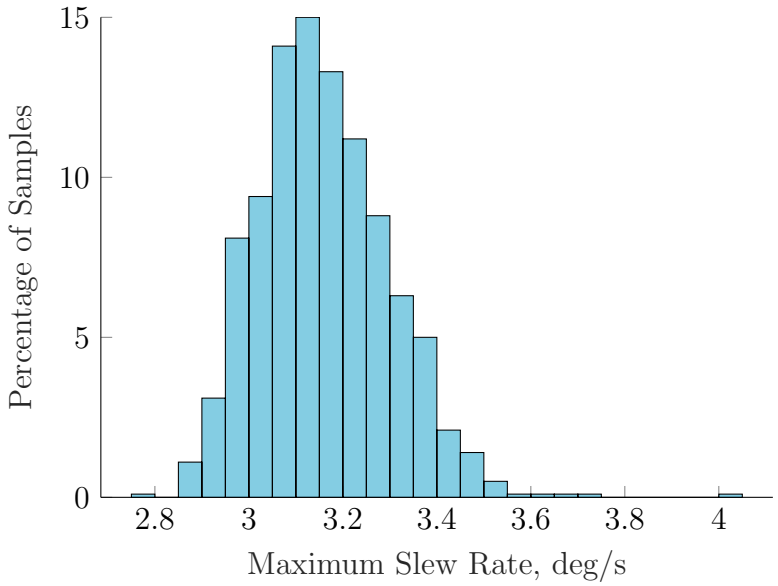


Figure 5.9: Maximum slew rate for each of the 1,000 Monte Carlo runs. These maximum slew rates are tightly coupled around 3.2 deg/s, well below the 20 deg/s constraint.

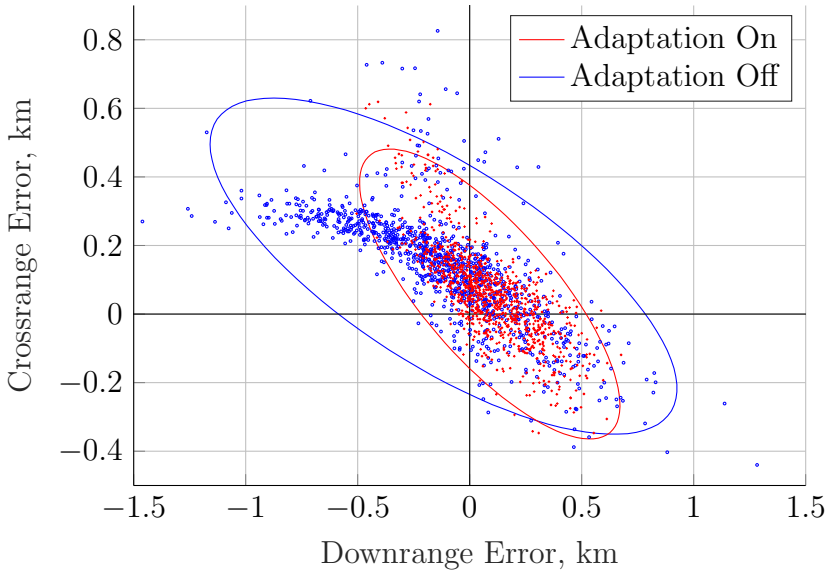


Figure 5.10: Comparison of the terminal position errors from 1,000 Monte Carlo runs where atmospheric adaptation was switched on and off. The spread of the terminal errors is significantly tighter and closer to the origin for the cases with adaptation on.

5.6 Conclusions

This work presents a combined estimator and controller architecture for atmospheric entry guidance in the Martian atmosphere. Atmospheric adaptation is accomplished with a square-root extended Kalman Filter that estimates the ratio of observed atmosphere to a nominal model computed offline, and uses this estimated ratio to modify the atmospheric model used in the controller. The guidance portion was accomplished using a variant of the convex predictor-corrector entry guidance algorithm (CPEG) that has been modified to include time as an optimization variable. This guidance algorithm is based on the predictor-corrector framework in which a nonlinear simulation is performed from the current initial condition and nominal control policy, and information from this prediction is used to correct the nominal control policy. By solving for this correction using convex trajectory optimization, CPEG is able to explicitly reason about the dynamics of the vehicle through the entirety of the trajectory and formulate a sophisticated control plan.

The estimator and controller were validated with a 1,000 Monte Carlo simulations in which atmospheric density and wind profiles were sampled from Mars GRAM and the initial conditions of the entry vehicle were varied. With realistic noise levels, the median miss distance in these 1,000 runs was 193 meters, compared with a median miss distance of 287 meters with the atmospheric adaptation turned off. In 94% of the control calls in these 1,000 runs, the convex optimization problem was able to be solved in closed form with the solution to just one linear system. We believe that this reflects a substantial accuracy improvement over prior entry guidance methods that do not adapt to unmodeled atmospheric uncertainties, with only a modest increase in computational cost.

Chapter 6

Ultra-Fine Pointing for Nanosatellite Telescopes With Actuated Booms

The smallsat revolution has impacted the architecture of most modern satellites with the notable exception of fine-pointing space telescopes. Conventional attitude control hardware scales poorly as the spacecraft gets smaller, resulting in significant mass and performance penalties for nanosatellites with strict pointing requirements. This paper presents a novel attitude actuation and planning strategy that utilizes actuated booms with tip masses and magnetorquers for three-axis pointing and momentum desaturation. The speed of the booms is an appropriate match for the slowly varying environmental disturbance torques encountered in low-Earth orbit. As a result, these booms do not create the high-frequency jitter that reaction wheels do, lessening the need for complex second-stage correction hardware in the payload. An optimization-based motion planner is able to reason about the orbital ephemeris to ensure the booms never exceed their actuation limits, and a Linear Quadratic Gaussian controller is able to maintain fine-pointing during times of payload operation.

The contents of this chapter have been previously published at IEEE Aerospace Conference 2021 in Tracy, Manchester, and Douglas [177].

6.1 Introduction

Space telescopes have been able to explore the universe in ways that terrestrial telescopes cannot through the atmosphere. Current monolithic systems like the Hubble Space Telescope use onboard reaction wheels or Control Moment Gyroscopes (CMGs) to control pointing [15]. These actuators spin weighted rotors onboard the spacecraft to store angular momentum and maintain pointing in the presence of disturbance torques. Unfortunately, the fractional mass of traditional attitude-control hardware grows dramatically as the spacecraft gets smaller. For larger satellites, the actuators take up only a few percent of the total spacecraft mass but, as the spacecraft gets smaller, it can consume 30% or more of the total mass [43].

Figure 6.1: Proposed architecture for a 6U CubeSat space telescope. Each boom has a single degree-of-freedom in linearly independent axes, enabling three-axis attitude control. Magnetorquers are used to desaturate the angular momentum of the booms and keep them within their operating limits.

One issue with modern attitude control hardware on nanosatellites comes from the vibrations present in reaction wheel and CMG operation. These actuators have to spin at high angular velocities to store the required onboard angular momentum, and small defects or imbalances in the rotors cause high-frequency jitter. These vibrations can resonate with structural modes in the satellite and can corrupt payload pointing performance. To deal with image-corrupting jitter, nanosatellite payloads employ second-stage corrections to enable finer pointing performance for the payload than the body of the spacecraft. Common methods for accomplishing this are fast-steering optical mirrors, image plane shifting with lead zirconate titanate (PZT) actuators, and image stabilization [147, 138, 139, 6]. These highly complex electromechanical systems are expensive and must be tailored to a specific payload, increasing costs and payload size, weight, and power (SWaP).

In this paper, a novel actuation strategy for fine-pointing nanosatellites is explored by abandoning high-frequency rotor-based actuators in favor of low-frequency deployable booms, as shown in Figure 6.1. By taking advantage of the squared relationship between boom length and the inertia of the boom, tip-mounted masses provide the control authority required for attitude control without having to accelerate or decelerate the booms too aggressively. A nanosatellite would deploy three booms about linearly independent axes and rotate them *slowly* to reject the *slowly* varying disturbance torques. By better matching the actuators' speed to the frequency content of the disturbances, these booms are able to eliminate jitter and enable high-accuracy body pointing of the nanosatellite. By improving body pointing, payloads are no longer restricted to those that can accommodate second-stage correction, and existing payloads can be simplified.

For nanosatellites in low-Earth orbit, disturbance torques come in the form of drag, solar radiation pressure, magnetic, and gravity-gradient torques. All of these disturbances

vary slowly throughout an orbit and can be predicted or estimated with high accuracy. With knowledge of the incoming disturbance torques, the nanosatellite can formulate a motion plan that accounts for disturbances and keeps the deployable booms from hitting their hard stops by using the onboard magnetorquers to offload angular momentum through interactions with the Earth's magnetic field [56, 111]. Since nominal operations have the nanosatellites inertially pointing during payload operations, linearized attitude dynamics are sufficiently accurate for planning purposes. This allows for a convex formulation of the motion-planning problem, guaranteeing a globally optimal solution in polynomial time [24].

Our primary contributions in this paper include:

1. The introduction of a novel attitude-actuation strategy for fine-pointing nanosatellites.

2. The application of convex optimization to motion planning for nanosatellites with actuated booms.

3. An estimator and controller architecture for handling boom control during payload operations.

In the remainder of this paper, we first provide details on the simulation environment used for this research in Section 6.2. Next, the deployable-boom actuation strategy is discussed in Section 6.3, and a convex motion planner is developed in Section 6.4 to reason about the actuator's constraints. A lower-level estimation and control architecture is then detailed in Section 6.5. Finally, numerical experiments are presented in Section 6.6 to validate the proposed ideas, and results are summarized in Section 6.7.

6.2 Spacecraft Dynamics Model

This section describes the model used to analyze and simulate the dynamics of a nanosatellite space telescope in low-Earth orbit. Since the spacecraft has no propulsion onboard, the orbital and attitude dynamics are decoupled and can be modeled separately. The open-source Julia package *SatelliteDynamics.jl* is used for orbital simulation, taking into account high order gravity, atmospheric drag, solar radiation pressure, and third-body accelerations. For the attitude dynamics, a rigid-body simulation is used that includes the disturbance torques present in low-Earth orbit.

We denote the Earth-Centered Inertial frame (ECI) as $\mathbb{E}$, and the spacecraft body frame as $\mathbb{B}$. Relating these two frames is ${}^{\mathbb{B}}Q^{\mathbb{E}}$, the rotation matrix that takes vectors expressed in $\mathbb{E}$ and resolves them in frame $\mathbb{B}$.

Gravity-Gradient Torque

Gravity varies inversely with the square of the distance from the central body. Because of this, parts of the spacecraft that are farther away from the center of the central body experience a smaller gravitational force than the parts that are closer. The resulting non-uniform gravitational force acting on the spacecraft causes a torque. This torque can be

neatly expressed in terms of the spacecraft's attitude, orbital position, and the inertia [111, 194]. First, the normalized position vector is computed in the body frame:

$$\hat{m} = \frac{{}^{\mathbb{B}}Q^{\mathbb{E}} r_{\mathbb{E}}}{\|r_{\mathbb{E}}\|}, \tag{6.1}$$

then, the gravity-gradient torque can be calculated,

$$\tau_{gg} = \frac{3\mu}{\|r\|^3}(\hat{m} \times J\hat{m}), \tag{6.2}$$

where μ is the standard gravitational parameter for Earth. This calculation only takes into consideration the spherical gravitational term, since the gravity gradient torque from higher-order gravity terms is of negligible magnitude. For inertially pointing spacecraft in pure Keplerian motion, the resulting gravity gradient torque is periodic with the orbit.

Atmospheric Drag Torque

To describe the atmospheric drag torque, the relative velocity of the spacecraft with respect to the atmosphere is calculated with the following:

$$v_{rel} = {}^{\mathbb{B}}Q^{\mathbb{E}}(v_{eci} - \omega_{Earth} \times r_{\mathbb{E}}). \tag{6.3}$$

A normalized version of this vector will be represented as $\hat{v}_{rel} = v_{rel}/\|v_{rel}\|$. The spacecraft in this experiment has been parameterized as a box with 6 orthogonal faces. This geometry is described with normal vectors $\hat{n}_i$ for each face, position vectors from the center of mass of the spacecraft to the center of pressure of each face r_i, and the area of each face S_i. Only the faces in the direction of the relative velocity vector are affected, and the force is proportional to the cosine of the angle between the normal face vector and the relative velocity vector. This can also be represented as the dot product between two normalized vectors:

$$w_i = \max(0, v^T[{}^{\mathbb{E}}Q^{\mathbb{B}}\,\hat{n}_i]). \tag{6.4}$$

The force on each face is then calculated with the atmospheric density ρ and the coefficient of drag C_d,

$$F_{aero,i} = -\frac{1}{2}\rho\, C_d \|v_{rel}\| v_{rel} S_i w_i. \tag{6.5}$$

The torque acting on the nanosatelite is then the sum of the moments caused by these forces:

$$\tau_{aero} = \sum_{i=1}^{6} r_i \times F_{aero,i}. \tag{6.6}$$

Solar Radiation Pressure Torque

Similar to the aerodynamic drag torque, the radiation from the sun carries momentum, and can impart a force on the spacecraft. First, the position vector from the spacecraft to the sun is calculated as:

$$^{\mathbb{B}}r^{sun} = {}^{\mathbb{E}}r^{sun} - {}^{\mathbb{E}}r^{\mathbb{B}}, \tag{6.7}$$

after which it is expressed in the body frame and normalized:

$$\hat{s} = \frac{{}^{\mathbb{B}}Q^{\mathbb{E}}\ {}^{\mathbb{B}}r^{sun}}{\|{}^{\mathbb{B}}r^{sun}\|}. \tag{6.8}$$

Based on the spectral and diffuse reflection coefficients, R_{spec} and R_{diff} respectively, the optical properties of the spacecraft material with respect to solar radiation pressure can be calculated. The combined effects of reflection, diffusion, and absorption are captured in our variable q_i,

$$q_i = 2\big[\frac{1}{3}R_{diff} + R_{spec}\hat{s}^T\hat{n}_i\big]\hat{n}_i + (1 - R_{spec})\hat{s}. \tag{6.9}$$

The force caused by radiation pressure on each face is then,

$$F_{srp,i} = -P_{sun}S_i q_i \cdot \max(0, \hat{s}^T[{}^{\mathbb{E}}Q^{\mathbb{B}}\, n_i]), \tag{6.10}$$

and the final torque is the sum of the moments caused by the force on each face:

$$\tau_{srp} = \sum_{i=1}^{6} r_i \times F_{srp,i}. \tag{6.11}$$

Magnetic Torques

We use the International Geomagnetic Reference Field (IGRF) [164] to model the Earth's magnetic field. The IGRF models the scalar potential of the magnetic field with a spherical harmonic expansion and calculates the magnetic field vector as the negative gradient of this potential with respect to the position. This potential is described by a set of time-varying coefficients that account for decades of empirical data and can predict the Earth's magnetic field up to 5 years in the future. This resulting magnetic field vector is a function of position and time, and for spacecraft with no propulsion, can be computed online to predict future magnetometer measurements. The spacecraft can then use the onboard magnetorquers to interact with the Earth's magnetic field in the following way:

$$\tau_{mag} = m \times b_{\mathbb{B}}, \tag{6.12}$$

where $b_{\mathbb{B}}$ is the magnetic field vector expressed in the spacecraft body frame, and m is the spacecraft's magnetic moment in the body frame. The total disturbance torque on the nanosatellite is the sum of the aforementioned torques:

$$\tau = \tau_{gg} + \tau_{aero} + \tau_{srp} + \tau_{mag}. \tag{6.13}$$

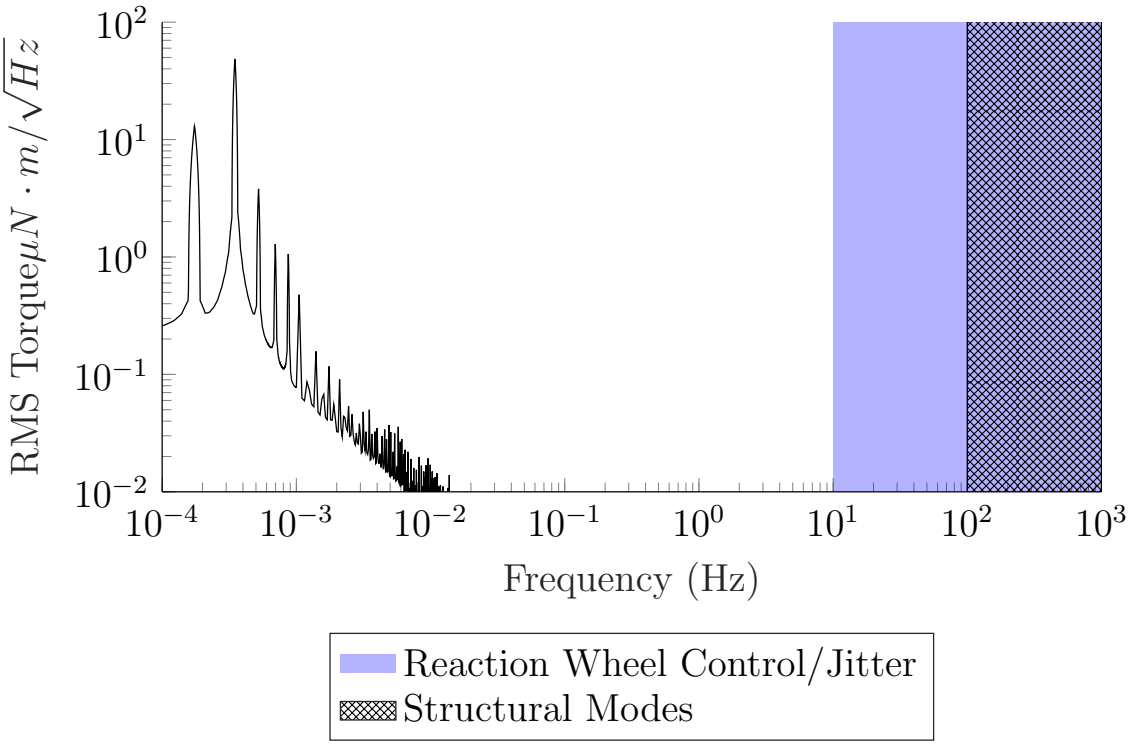


Figure 6.2: The frequency content of disturbance torques on a small satellite in low-Earth orbit. Disturbances are due to atmospheric drag, solar radiation pressure, and the gravity gradient. The frequency content of the disturbances is of significantly lower frequency than both standard reaction wheels and structural modes of the nanosatellites.

6.3 Actuation Strategy

To justify the introduction of a novel actuation strategy, the nature of the disturbance torques was analyzed. A 1000-trial Monte-Carlo simulation of the orbital and attitude dynamics was performed and disturbance torques were collected. The orbits in the simulations were at an altitude of 550 km with inclinations between 0° and 90°, eccentricities between 0 and 0.00002, and epochs between 2014 and 2017. These dispersions served to capture the full range of potential disturbance torques, as well as accurately sample the time-varying atmospheric density. A discrete Fourier transform of the disturbance torque data was taken, and the magnitudes of the terms in the Fourier series were used to plot its power-spectral density in Figure 6.2.

Figure 6.2 displays a clear mismatch between the speed of the disturbances and the speed of the actuators currently used in space telescopes. Jitter from traditional reaction wheels and the structural modes of the spacecraft is 3-4 orders of magnitude faster than the disturbance torques. This mismatch strongly suggests that slower actuators could be used, and as a result, interactions between the actuators and structural modes could be avoided.

The proposed architecture is displayed in Figure 6.1, with three masses mounted to

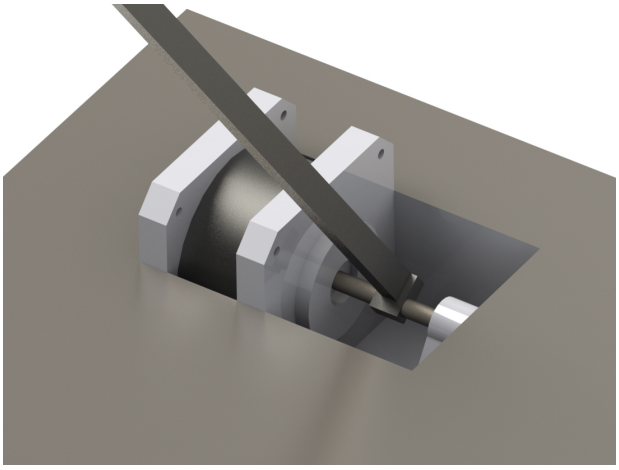


Figure 6.3: Boom actuation with a direct-drive micro-stepper motor. Torques commanded by the micro-stepper will accelerate and decelerate the boom, fully controlling the attitude of the nanosatellite.

deployable booms that will be used for full three-axis attitude control. Each deployable boom has a single-degree-of-freedom actuator at the interface between the boom and the spacecraft that can rotate the boom and mass combination. By moving the booms, angular momentum is transferred from the body of the spacecraft to the masses, allowing for full actuation of the spacecraft's attitude. This actuation strategy is significantly slower than reaction wheels, and will therefore avoid both high-frequency jitter as well as the excitement of the flexible modes of the spacecraft.

Two potential boom-actuator configurations are described in Figures 6.3 and 6.4, with a stepper motor and a linear voice coil respectively. Both of these actuators are able to precisely move the boom, but will also impart constraints on the torque, velocity, and configurations of the boom. These constraints will be accounted for by the onboard motion planner to ensure full attitude actuation is maintained. The boom actuators themselves can also contribute unwanted dynamics, like cogging torques and stiction, into the dynamics of the boom. Care must be taken when designing and building these actuators to ensure that these contributions to the dynamics are well characterized and incorporated into the planner.

6.4 Motion Planner

The spacecraft must maintain pointing through a balance of magnetorquer control and boom control. A holistic planning approach is taken due to the following constraints: The magnetorquers cannot be used during payload operations, and the deployable booms must stay within their allowable ranges in position, velocity, and acceleration. Coarse attitude control has been demonstrated using only magnetorquers [56], but this control strategy is not precise enough for ultra-fine pointing and relies on a varying magnetic field to demonstrate full three-axis actuation. The booms provide full three-degree-of-

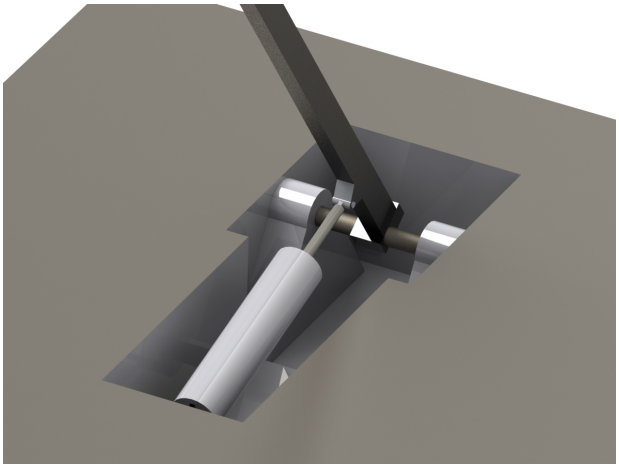


Figure 6.4: Boom actuation with a linear voice-coil actuator. By extending and contracting, the linear force is converted to a torque near the base of the boom. This moment in turn controls the angular acceleration of the boom.

freedom actuation of the attitude but must avoid violating their actuator constraints. These conditions can be combined into a motion planner that balances magnetorquer use for coarse pointing and momentum desaturation while utilizing the booms for precise pointing during periods of payload operation. Similar to the existing nanosatellite telescopes DeMi [6] and ASTERIA [92], this motion planner is tailored to the case where exposures are to be taken during eclipse. This means that the nanosatellite must turn off the magnetorquers for the duration of the $\sim$ 35-minute exposures.

The angular positions of the booms are described with $\theta \in \mathbb{R}^3$, and the angular velocities of the booms as $\dot{\theta} \in \mathbb{R}^3$. The control input responsible for boom actuation is the angular acceleration of the booms, denoted as $\alpha \in \mathbb{R}^3$. The dynamics of the booms themselves are modeled as double integrators with angular acceleration as a control input:

$$\ddot{\theta} = \alpha. \tag{6.14}$$

Assuming a zero-order hold on the commanded angular acceleration and discretizing, we have,

$$\begin{bmatrix} \theta_{k+1} \\ \dot{\theta}_{k+1} \end{bmatrix} = A \begin{bmatrix} \theta_k \\ \dot{\theta}_k \end{bmatrix} + B\alpha_k, \tag{6.15}$$

where A and B are a function of the sample time, dt:

$$A = \begin{bmatrix} I_3 & dt \cdot I_3 \\ 0_3 & I_3 \end{bmatrix}, \tag{6.16}$$

$$B = \begin{bmatrix} \frac{1}{2} dt^2 \cdot I_3 \\ dt \cdot I_3 \end{bmatrix}. \tag{6.17}$$

The planning problem is then posed as a convex optimization problem where the optimal sequence of actuator commands, magnetic moment m and boom angular acceleration α,

are solved for that counter all expected disturbance torques. To ensure that the magnetorquers are never on during payload operations, the optimization formulation conservatively prohibits any magnetorquer usage during periods of eclipse, denoted as indexes $k \in \mathcal{E}$. The full optimization problem can be written as follows:

$$
\begin{aligned}
\underset{m,\alpha,\theta,\dot{\theta}}{\text{minimize}} \quad & \sum_{k=1}^{N} \| [m_k^T,\ \gamma\alpha_k^T,\ \beta\theta_k^T,\ \sigma\dot{\theta}_k^T]^T \|^2 \\
\text{subject to} \quad & \tau_k = m_k \times b_{\mathbb{B}} - J_{boom}\alpha_k \forall k, \\
& \begin{bmatrix} \theta_{k+1} \\ \dot{\theta}_{k+1} \end{bmatrix} = A \begin{bmatrix} \theta_k \\ \dot{\theta}_k \end{bmatrix} + B\alpha_k \qquad \forall k, \\
& m_{min} \leq m_k \leq m_{max} \qquad \forall k, \\
& \alpha_{min} \leq \alpha_k \leq \alpha_{max} \qquad \forall k, \\
& \theta_{min} \leq \theta_k \leq \theta_{max} \qquad \forall k, \\
& \dot{\theta}_{min} \leq \dot{\theta}_k \leq \dot{\theta}_{max} \qquad \forall k, \\
& m_k = 0 \qquad \forall k \in \mathcal{E},
\end{aligned}
\tag{6.18}
$$

where the inertias of the deployable booms are the diagonal entries of J_{boom}, the magnetic field of the Earth expressed in the spacecraft body frame is $b_{\mathbb{B}}$, and the disturbance torque is τ. The torque matching constraint and the kinematics of the boom are both enforced as linear equality constraints, and all the state and actuator limits are expressed as box inequality constraints [152]. The cost function is a quadratic penalty on control usage for the two sets of actuators, with γ, β, and σ as tuning parameters. Since the cost function is quadratic and positive definite and the constraints are all linear, problem 6.18 can be expressed as a convex Quadratic Program (QP). There are many readily available and robust QP solvers available that are able to find the global optimum, as well as many specialized solvers for use on embedded systems [115, 152, 13]. Using one of these tools, a highly performant customized solver can be generated for this specific problem, and can be implemented on compute-constrained flight hardware.

6.5 Estimation and Control

Fine-pointing nanosatellites demand the strictest pointing requirements during periods of payload operation. The boom actuators are solely responsible for attitude control during these periods since the magnetorquers are too coarse. The planner is able to leverage predicted disturbance torques to put the booms in a configuration that allows for full controllability during the exposure, but these predicted disturbance torques are not accurate enough to feed forward to the controller during these sensitive payload operations. Instead, during image captures, these disturbance torques will be estimated online in a recursive filter, and the online estimate of the disturbance torque will be incorporated as a feedforward control input. This section details the state estimator and controller combination that is used to maintain high-accuracy pointing during periods of image capture.

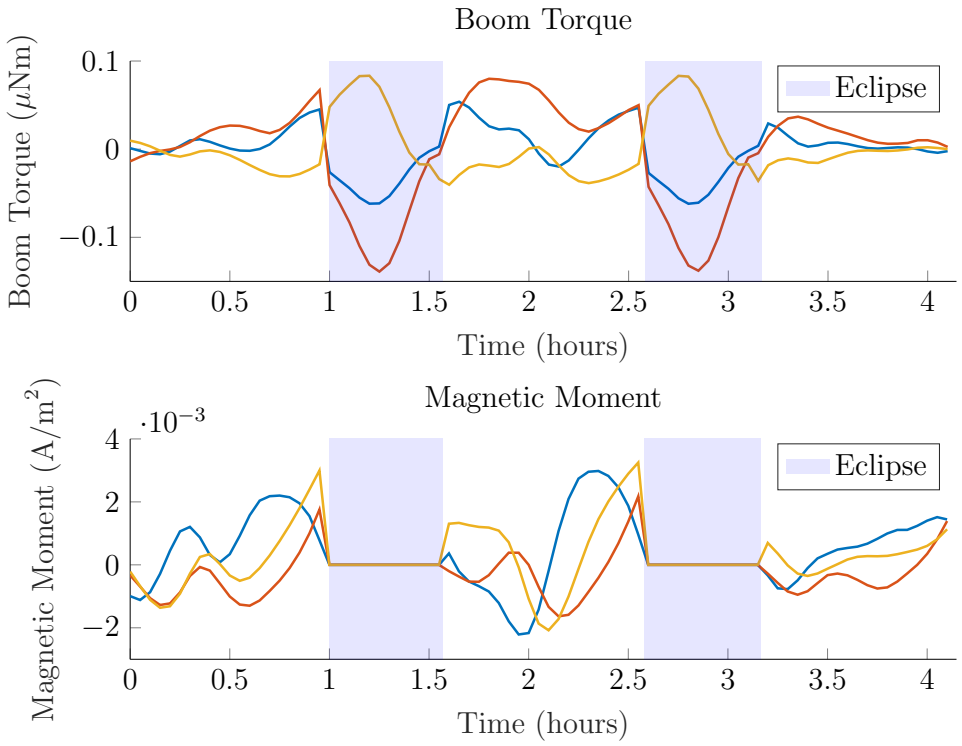


Figure 6.5: Motion plan for a nanosatellite with control over both boom torques and magnetorquers, given estimated future disturbance torques. During eclipse, when payload operations take place, the nanosatellite is constrained to only use the boom torques due to their precision.

6.5.1 State Estimator

The disturbance torques on the nanosatellite are smooth and slowly varying, as shown in Figure 6.6. A Multiplicative Extended Kalman Filter (MEKF) will be used for simultaneous estimation of the attitude, angular velocity, and disturbance torque [111]. Conventional MEKF's on large spacecraft omit the spacecraft's angular velocity from the state due to the accuracy of the onboard gyroscope. Nanosatellites do not have gyroscopes of this caliber and must estimate the angular velocity online as a result. The filter state is denoted z, and is augmented to include both the disturbance torque $\hat{\tau}$, as well as its first derivative:

$$z = \begin{bmatrix} q^T & \omega^T & \hat{\tau}^T & \dot{\hat{\tau}}^T \end{bmatrix}^T, \tag{6.19}$$

where $q \in \mathbb{R}^4$ is the quaternion describing the rotation from $\mathbb{E}$ to the body frame $\mathbb{B}$, $\omega \in \mathbb{R}^3$ is the angular velocity of the nanosatellite, $\hat{\tau} \in \mathbb{R}^3$ is the disturbance torque, and $\dot{\hat{\tau}} \in \mathbb{R}^3$ is its time derivative. Estimating the disturbance torque derivative allows the filter to better predict and anticipate changes in the disturbance torque. The deterministic dynamics model for the MEKF is as follows:

$$\dot{z} = \begin{bmatrix} \frac{1}{2} q \otimes (\omega) \\ J^{-1}(\hat{\tau} - J_{boom}\alpha - \omega \times J\omega) \\ \dot{\hat{\tau}} \\ 0 \end{bmatrix}. \tag{6.20}$$

These dynamics are discretized using an explicit integrator for a given sample time Δt, and additive white Gaussian (AWGN) process noise, ν_x, is added as follows:

$$z_{k+1} = f(z_k, \alpha_k, \Delta t) + \nu_x. \tag{6.21}$$

In the measurement model, we assume full measurements of the attitude and angular velocity,

$$y = \begin{bmatrix} q \\ \omega \end{bmatrix} + \nu_y, \tag{6.22}$$

where ν_y is AWGN sensor noise. From here, the MEKF as described in [111] is implemented with the addition of the angular velocity to the estimator state.

6.5.2 Feedback Controller

By linearizing the dynamics of the nanosatellite about a nominal desired attitude, and replacing the quaternion with an axis-angle vector $\phi \in \mathbb{R}^3$, the local error dynamics can be expressed as the following:

$$\dot{x}_{lqr} = \begin{bmatrix} \dot{\phi} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} 0 & I \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \phi \\ \omega \end{bmatrix} + \begin{bmatrix} 0 \\ -J_{boom} \end{bmatrix} \alpha. \tag{6.23}$$

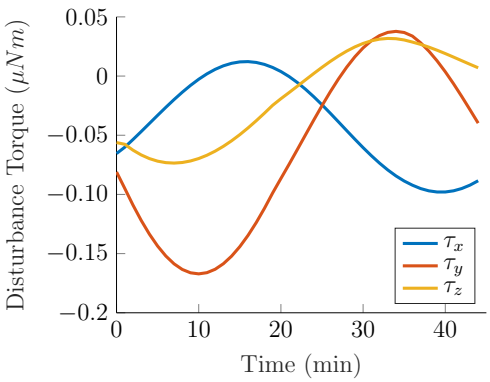


Figure 6.6: Disturbance torques over a 45 minute period. These torques are smooth and slowly varying, making estimation of this torque possible in a Kalman Filter.

From here, the dynamics can be discretized assuming a zero-order hold on α, and a feedback gain K is solved for that minimizes the following infinite-horizon Linear Quadratic Regulator (LQR) cost function:

$$\ell(x,u) = \frac{1}{2}\sum_{k=0}^{\infty} x_k^T Q x_k + \alpha_k^T R \alpha_k, \tag{6.24}$$

where $Q \in \mathbb{R}^{6\times 6}$ and $R \in \mathbb{R}^{3\times 3}$ are positive definite diagonal matrices [153]. The resulting control law takes the form,

$$u = -K x_{lqr} - J_{boom}^{-1}\hat{\tau}, \tag{6.25}$$

where $\hat{\tau}$ is the estimated disturbance torque from the MEKF.

6.6 Numerical Experiments

All experiments were run in Julia [18], using the optimization modeling language JuMP [104] with Mosek [124] as the solver for the motion-planning problem. All of the code used for the experiments is readily available at `https://github.com/RoboticExplorationLab/WiggleSat.jl`.

To test the planning and control algorithms presented in this paper, a nanosatellite in a low-Earth orbit with an altitude of 420 km, inclination of 51.4°, and eccentricity of 0.00108 is considered. The orbit is propagated with accelerations from a high-order gravity model, atmospheric drag, solar radiation pressure, and third body contributions from the Moon and the Sun [123]. The disturbance torques as described in Section 6.2 are then calculated given the desired attitude. The attitude measurement has a standard deviation of 1 arcsecond [43], the gyroscope is modeled after the Honeywell GG1320AN with an angle random walk of 0.0035 deg/$\sqrt{hr}$, and the sample rate on the sensors, filter, and controller is 1 Hz [75].

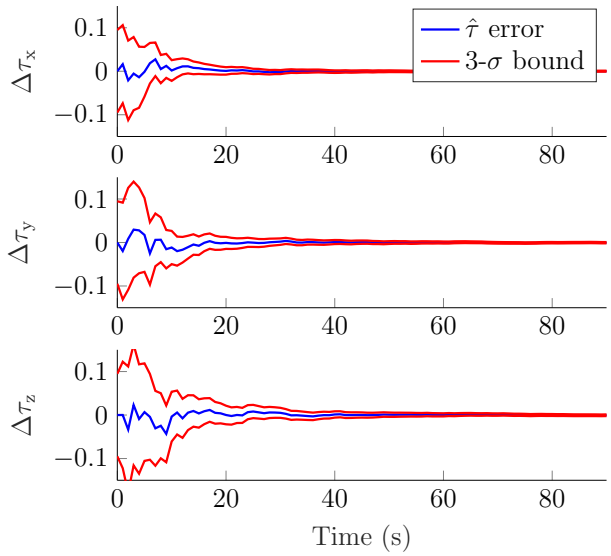


Figure 6.7: MEKF estimation errors for the unknown disturbance torque, as well as 3-σ bounds from the covariance. By modeling the disturbance torques as a double integrator system, where both the torque and its time derivative are estimated, the MEKF is able to converge on accurate estimates of the true torque values in less than 1 minute.

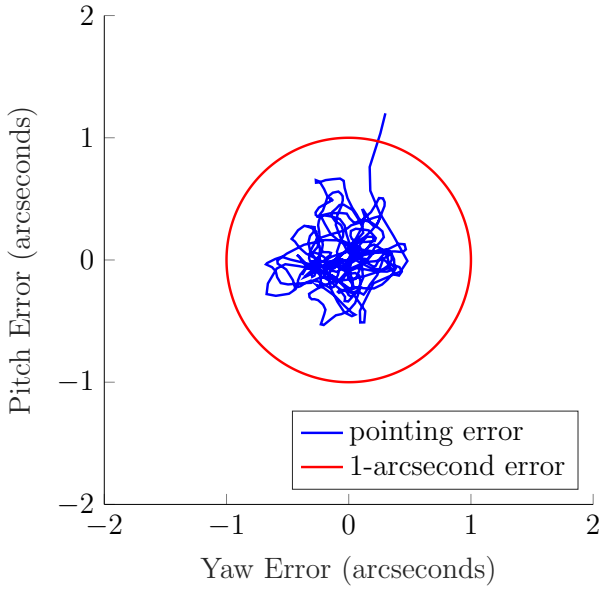


Figure 6.8: Closed-loop yaw and pitch error with an attitude sensor standard deviation of 1 arcsecond. The combined estimator and controller are able to maintain sub-arcsecond pointing even in the presence of the sensor noise and unknown disturbance torques.

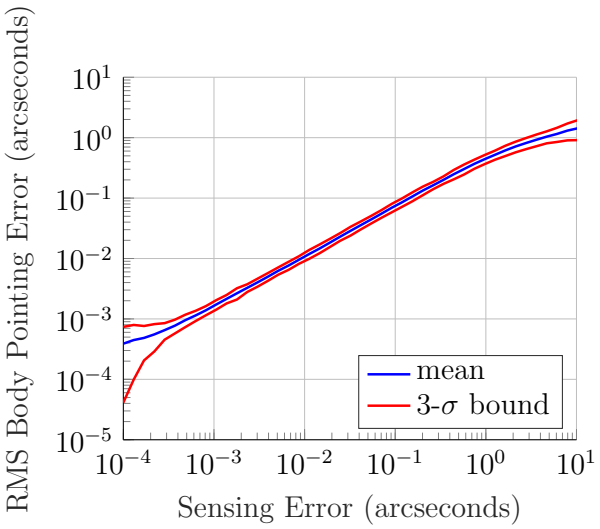


Figure 6.9: For each attitude sensing error, a series of simulations were run to estimate the mean and 3-σ bounds for the RMS body pointing error. Despite all the simulations using the same gyroscope, the estimator and controller combination is able to continue driving down the body pointing error with the sensing error.

The motion planner, as detailed in Section 6.4, is used to calculate a nominal control plan for both the magnetorquers and the booms. With a time horizon of 4 hours, the planner is able to account for two eclipse periods where the magnetorquers are unavailable. The solution from the planner is shown in Figure 6.5, with the nominal control plans for both boom torques as well as commanded magnetic moment. The planner effectively balances momentum management with the requirement to put the arms in a configuration prior to eclipse that allows for full controllability throughout the duration of the eclipse.

During eclipse, the estimator and controller designed in Section 6.5 are used to maintain the desired attitude. Instead of relying on the predicted disturbance torque, an MEKF is used to estimate both the attitude and angular velocity, as well as the disturbance torque and its time derivative. The convergence of this filter on the unknown disturbance torque is shown in Figure 6.7. Even with a poor initialization of all zeros for the estimate of the disturbance torque, the filter is able to converge on the true value within one minute of operation.

The pointing performance of the combined estimator and controller is shown in Figure 6.8. Here, the initial condition starts outside the 1-arcsecond error circle, and the controller is able to keep the error inside the circle for a Root Mean Square (RMS) body pointing error of 0.39 arcseconds. To better evaluate the robustness and performance of this estimator and controller combination, this same simulation was run for a variety of initial conditions with a range of attitude sensing errors. The RMS body pointing error as a function of this attitude sensing error is shown in Figure 6.9, where the mean body pointing performance as well as three-sigma bounds are shown. We also note that the filter performs well enough that body-pointing errors are able to decrease with attitude sensing error, despite using

the same gyroscope for all simulations.

6.7 Conclusions

This paper proposes a novel attitude actuation strategy for fine-pointing nanosatellites. The new approach abandons traditional high-frequency reaction wheels in favor of low-frequency actuated booms. As shown by a spectral analysis of the environmental disturbance torques, these low-frequency deployable booms are a much better fit for the slowly varying disturbance torques encountered in low-Earth orbit. By performing control in the same frequency range as these disturbances, many of the complications that reaction wheels introduce, including high-frequency jitter and excitement of flexible structural modes, can be avoided. This actuation methodology results in finer body-pointing performance, reducing the need for second-stage correction systems to point sensitive payloads.

Control of a spacecraft equipped with the proposed boom actuators was demonstrated via a convex-optimization-based motion planner paired with an MEKF estimator and LQR controller during sensitive payload operations. This planner was able to balance magnetorquer usage and boom actuation to combat environmental disturbances, reason about actuator limits and boom constraints, and avoid magnetorquer usage during periods of eclipse for improved payload operations. This optimization problem was formulated as a quadratic program and can be solved quickly and reliably onboard spacecraft with limited computing resources.

Chapter 7

Differentiable Collision Detection

Collision detection between objects is critical for simulation, control, and learning for robotic systems. However, existing collision detection routines are inherently non-differentiable, limiting their applications in gradient-based optimization tools. In this work, we propose DCOL: a fast and fully differentiable collision-detection framework that reasons about collisions between a set of composable and highly expressive convex primitive shapes. This is achieved by formulating the collision detection problem as a convex optimization problem that solves for the minimum uniform scaling applied to each primitive before they intersect. The optimization problem is fully differentiable with respect to the configurations of each primitive and is able to return a collision detection metric and contact points on each object, agnostic of interpenetration. We demonstrate the capabilities of DCOL on a range of robotics problems from trajectory optimization and contact physics, and have made an open-source implementation available.

The contents of this chapter have been previously published at ICRA 2023 in Tracy, Howell, and Manchester [170].

7.1 Introduction

Computing collisions is of great interest to the computer graphics, video game, and robotics communities. Popular algorithms for collision detection include the Gilbert, Johnson, and Keerthi (GJK) algorithm [60], its updated variant enhanced-GJK [31], and Minkowski Portal Refinement (MPR) [151, 129]. For objects that have interpenetration, the Expanding Polytope Algorithm (EPA) [180] is used to return a metric that describes the depth of penetration between two objects. These algorithms are implemented in the widely used Flexible Collision Library (FCL) [131], and are employed in most physics engines including Bullet [37], Drake [163], Dart [96], and MuJoCo [166]. While efficient and robust, all of these algorithms are inherently non-differentiable due to their logical control flow and pivoting.

Two methods have been proposed for calculating approximate gradients of a collision metric: The first is sample-based randomized smoothing of GJK for finite-differenced gradients [122], and the second formulates the closest distance between spheres, capsules,

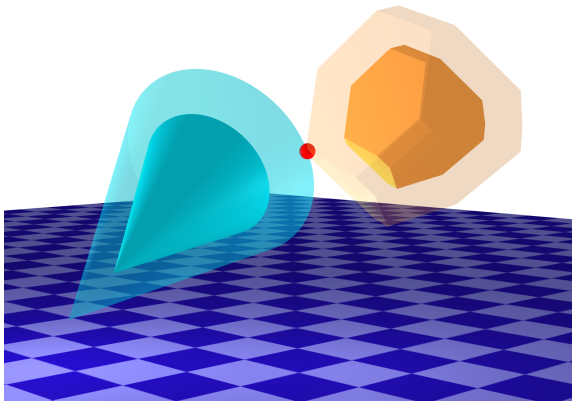


Figure 7.1: Collision detection between a cone and a polytope. DCOL works by solving an optimization problem for the minimum scaling of each object that produces an intersection which, in this example, is greater than one, meaning there is no collision. The scaled objects are translucent and the intersection point between these scaled objects is shown in red.

planes, and boxes, as a differentiable optimization problem [201], similar to [171]. The approximate gradients from the first method are expensive to compute and are unable to return useful information if penetration occurs, while the second method is only able to handle a limited selection of convex primitives.

This paper introduces DCOL, a differentiable collision-detection framework that computes closest points, minimum distance, and interpenetration depth between any pair of six convex primitive shapes: polytopes, capsules, cylinders, cones, ellipsoids, and padded polygons (Fig. 7.2). We do this by formulating a convex optimization problem that solves for the minimum uniform scaling that must be applied to the primitives for an intersection to occur, an idea first proposed for polytopes in [59]. When primitives are not in contact, the minimum scaling for an intersection is greater than one, and when there is interpenetration between objects, the minimum scaling is less than one. The ability to return an informative collision metric in the presence of interpenetration is a key distinction between DCOL and GJK variants. In addition to the scaling parameter detecting a collision, the contact points on each object can be calculated from this solution as well.

The optimization problems produced by our formulation are bounded, feasible, and well-defined for all configurations of the primitives. Differentiable convex optimization allows the sensitivities of the solution to be calculated with respect to problem parameters with minimal added computation. This allows informative and smooth derivatives of the minimum scaling as well as the contact points with respect to the configurations of the primitives to be computed efficiently.

The ability to differentiate through our collision detection algorithm enables the inclusion of accurate collision information into gradient-based robotic simulation, control, and learning frameworks. We demonstrate this on several relevant robotics problems from trajectory optimization and contact physics.

Our specific contributions in this paper are the following:

- An optimization-based collision detection formulation between convex primitives that returns an informative collision metric even in the case of interpenetration
- Efficient differentiation of this optimization problem with respect to the configurations of each primitive
- A fast and efficient open-source implementation of these algorithms built on a custom primal-dual interior-point solver

The paper proceeds by providing background on standard convex conic optimization and differentiation of conic optimization problems in Section 7.2, a derivation of DCOL in Section 7.3 with the corresponding constraints for each of the six convex primitives shown in Fig. 7.2, example use cases in trajectory optimization and contact physics in Section 7.4, and our conclusions in Section 7.5.

7.2 Background

The differentiable collision detection algorithm, DCOL, proposed in this paper is built on differentiable convex optimization. In this section, convex optimization with the relevant conic constraints is detailed, as well as a method for efficiently computing derivatives of these optimization problems with respect to problem parameters.

7.2.1 Conic Optimization

DCOL formulates collision detection problems as a convex optimization problem with conic constraints [24]. In standard form, these optimization problems have linear objectives and constraints of the following form:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & c^T x \\ \text{subject to} \quad & h - Gx \in \mathcal{K}, \end{aligned} \tag{7.1}$$

where $x \in \mathbf{R}^n$, $c \in \mathbf{R}^n$, $G \in \mathbf{R}^{m \times n}$, $h \in \mathbf{R}^m$, and $\mathcal{K} = \mathcal{K}_1 \times \cdots \times \mathcal{K}_N$ is a Cartesian product of N proper convex cones. The optimality conditions for problem 7.1 are as follows:

$$c + G^T z = 0, \tag{7.2}$$
$$h - Gx \in \mathcal{K}, \tag{7.3}$$
$$z \in \mathcal{K}^*, \tag{7.4}$$
$$(h - Gx) \circ z = 0, \tag{7.5}$$

where a dual variable $z \in \mathbf{R}^m$ is introduced, $\mathcal{K}^*$ is the dual cone, and $\circ$ is a cone product specific to each cone [183].

The cones required for DCOL include the nonnegative orthant, denoted as $\mathbf{R}^m_+$, and the second-order cone, denoted as $\mathcal{Q}_m$. The nonnegative orthant contains any vector $s \in \mathbf{R}^m_+$ where $s \geq 0$, and the a second-order cone contains any vector $s \in \mathcal{Q}_m$ such that $\|s_{2:m}\|_2 \leq s_1$.

We develop a custom primal-dual interior-point solver for DCOL, with support for both of the relevant cones, based on the *conelp* solver from [183], with features taken from [41, 127, 9, 126]. The memory for this custom solver is entirely stack-allocated and is optimized for the small problems that DCOL creates, dramatically outperforming off-the-shelf primal-dual interior-point conic solvers like ECOS [41] and Mosek [124].

7.2.2 Differentiating Through a Cone Program

Recent advances in differentiable convex optimization have enabled efficient differentiation through problems of the form (7.1) [5, 4, 8]. Solutions to (7.1) can be differentiated with respect to any parameters used in c, G, and h.

At the core of differentiable convex optimization is the implicit function theorem. An implicit function $g : \mathbf{R}^a \times \mathbf{R}^b \to \mathbf{R}^a$ is defined as:

$$g(y^*, \theta) = 0, \tag{7.6}$$

for an equilibrium point $y^* \in \mathbf{R}^a$, and problem parameters $\theta \in \mathbf{R}^b$. Approximating (7.6) with a first-order Taylor series results in:

$$\frac{\partial g}{\partial y}\delta y + \frac{\partial g}{\partial \theta}\delta\theta = 0, \tag{7.7}$$

which can be re-arranged to solve for the sensitivities of the solution with respect to the problem parameters:

$$\frac{\partial y}{\partial \theta} = -\left(\frac{\partial g}{\partial y}\right)^{-1}\frac{\partial g}{\partial \theta}. \tag{7.8}$$

By treating the optimality conditions in equations (7.2) and (7.5) as an implicit function at a primal-dual solution, the sensitivities of the solution with respect to the problem data can be computed. When the original optimization problem is solved using a primal-dual interior-point method as described in [183], these derivatives can be computed after the solve without any additional matrix factorizations [8]. This enables fast differentiation of conic programs that are fit for use in our differentiable collision detection algorithm.

In the case where only the gradient of the objective value J with respect to the problem parameters θ is needed, the implicit function theorem is unnecessary. Instead, we need only the gradient of the Lagrangian for (7.1):

$$\mathcal{L}(x, z, \theta) = c(\theta)^T x + z^T(G(\theta)x - h(\theta)), \tag{7.9}$$

where the problem matrices c, h, and G, are functions of the problem parameters θ. Given a primal-dual solution (x^*, z^*), the gradient of the objective value with respect to the problem parameters is simply the gradient of the Lagrangian with respect to these problem parameters, $\nabla_\theta J = \nabla_\theta \mathcal{L}(x^*, z^*, \theta)$. This allows for a faster computation of this specific gradient without using the implicit function theorem.

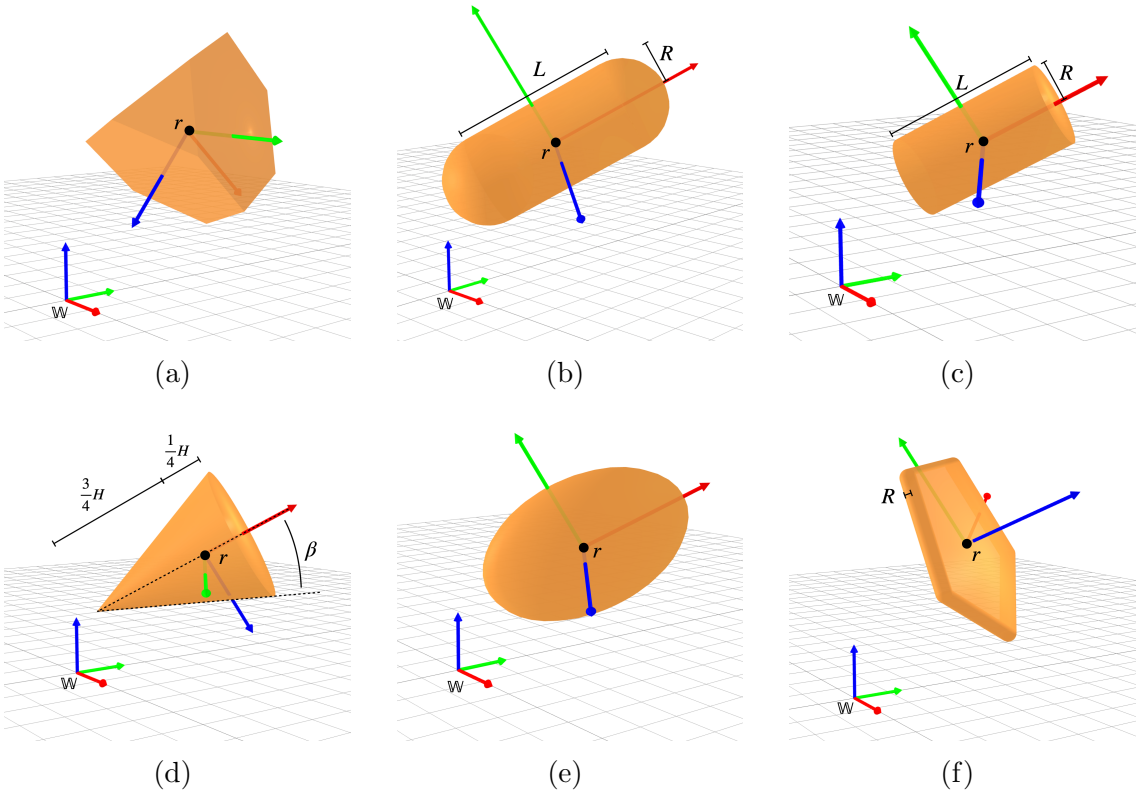


Figure 7.2: Geometric descriptions of the six primitive shapes that are compatible with this differentiable collision detection algorithm. These shapes include a polytope (a), capsule (b), cylinder (c), cone (d), ellipsoid (e), and padded polygon (f). Collision information including the collision status as well as the contact points can be computed between any of two of these primitives using DCOL.

7.3 The DCOL Algorithm

This section details how DCOL computes collision information between two convex primitives. The core part of this framework is an optimization problem that solves for a minimum uniform scaling $\alpha \in \mathbf{R}$ applied to both objects that result in an intersection. In the case where there is no collision between the two objects, the minimum scaling is greater than one, and when there is interpenetration, the minimum scaling is less than one. Because of this, we find the minimum scaling α is a better collision metric than the closest distance between the primitives, allowing for the straightforward description of collision constraints that are agnostic of interpenetration. All steps in the creation and solving of this optimization problem are fully differentiable, and average timing results for computing both solutions and derivatives are provided in Table 7.1 as an average over each primitive.

Table 7.1: Average DCOL Computation Times

	polyt.	**caps.**	**cyl.**	**cone**	**ellips.**	**polyg.**
evaluate	5.9 μs	8.5 μs	8.4 μs	5.0 μs	6.8 μs	9.4 μs
differentiate	1.4 μs	1.4 μs	1.6 μs	1.3 μs	1.3 μs	1.7 μs

7.3.1 Optimization Problem

Scaled convex primitives are described as a set $S(\alpha)$, which is a specific instance of the primitive scaled by some α. A point $x \in \mathbf{R}^3$ is said to be in the set $x \in S(\alpha)$ if x is within the scaled primitive. This notation allows for the following formulation of the optimization problem:

$$\begin{aligned} \underset{x,\,\alpha}{\text{minimize}} \quad & \alpha \\ \text{subject to} \quad & x \in \mathcal{S}_1(\alpha), \\ & x \in \mathcal{S}_2(\alpha), \\ & \alpha \geq 0, \end{aligned} \tag{7.10}$$

where the minimum scaling α is computed such that x is in the interior of both of the scaled primitives, making x an intersection point. This optimization problem is convex, bounded, and feasible for all of the primitives described in this paper. The boundedness comes from the constraint $\alpha \geq 0$, and the guarantee of feasibility comes from the fact that each object is uniformly scaled, so that in the limit $\alpha \to \infty$ each shape will encompass the entirety of $\mathbf{R}^3$, guaranteeing an intersection between objects. Another benefit to this problem formulation is that the only time the minimum scaling $\alpha = 0$ is when the origins of the two objects are coincident, in which the problem and its derivatives are still well defined.

7.3.2 Primitives

This section details the constraints that define set membership for each of the six scaled primitives. Each object is defined with an attached body reference frame $\mathbb{B}$ with an origin $r \in \mathbf{R}^3$ expressed in a world frame $\mathbb{W}$. The uniform scaling of these objects is always centered about this position r, and when the scaling parameter α is 0, the object is simply a point centered at r. The orientation of an object is defined by a rotation matrix ${}^{\mathbb{W}}Q^{\mathbb{B}} \in \mathbf{R}^{3\times3}$ relating the world frame to the object-fixed body frame, denoted as Q for shorthand. For each primitive, the constraints are also explicitly written in standard conic form for direct inclusion in our custom conic solver in the form of (7.1), where $h - Gx \in \mathcal{K}$.

Polytope

A polytope is a convex shape in $\mathbf{R}^3$ defined by a set of halfspace constraints, an example of which is shown in Fig. 7.2a. This polytope is described as the set of points $w \in \mathbf{R}^3$ such that $Aw \leq b$ for w expressed in $\mathbb{B}$, where $A \in \mathbf{R}^{m\times3}$ and $b \in \mathbf{R}^m$ represent the m halfspace

constraints comprising the polytope. This polytope can be scaled by α, resulting in the following constraint for x to be inside the polytope:

$$AQ^T(x - r) \leq \alpha b. \tag{7.11}$$

The scaling parameter α scales the vector b, resulting in uniform scaling of all halfspace constraints and subsequent uniform scaling of the polytope. This constraint in standard form is the following:

$$AQ^T r - \begin{bmatrix} AQ^T & -b \end{bmatrix} \begin{bmatrix} x \\ \alpha \end{bmatrix} \in \mathbf{R}_+. \tag{7.12}$$

Capsule

A capsule can be defined by the set of points within some radius R of a line segment, as shown in Fig. 7.2b. This internal line segment is along the x axis of the attached reference frame $\mathbb{B}$, and the end points of this line segment are some distance L apart. The scaled constraints for this primitive are that the point x must be within a scaled radius of the line segment, where the distance of the endpoints of the line segment from r is also scaled:

$$\|x - (r + \gamma \hat{b}_x)\|_2 \leq \alpha R, \tag{7.13}$$

$$-\alpha \frac{L}{2} \leq \gamma \leq \alpha \frac{L}{2}, \tag{7.14}$$

where $\hat{b}_x = Q[1, 0, 0]^T$, and $\gamma \in \mathbf{R}$ is a slack variable. These constraints contain a linear inequality and one second-order cone constraint, shown here in standard form:

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 0_{1\times 3} & -L/2 & 1 \\ 0_{1\times 3} & -L/2 & -1 \end{bmatrix} \begin{bmatrix} x \\ \alpha \\ \gamma \end{bmatrix} \in \mathbf{R}^2_+, \tag{7.15}$$

$$\begin{bmatrix} 0 \\ -r \end{bmatrix} - \begin{bmatrix} 0_{1\times 3} & -R & 0 \\ -I_3 & 0_{3\times 1} & \hat{b}_x \end{bmatrix} \begin{bmatrix} x \\ \alpha \\ \gamma \end{bmatrix} \in \mathcal{Q}_4. \tag{7.16}$$

Cylinder

The description of a cylinder is shown in Fig. 7.2c, with an orientation, a radius R, and a length L. The constraints for this primitive are the same as for the capsule in equations (7.13) and (7.14), with the introduction of two new scaled halfspace constraints that give the cylinder its flat ends:

$$[x - (r - \alpha \frac{L}{2} \hat{b}_x)]^T \hat{b}_x \geq 0, \tag{7.17}$$

$$[x - (r + \alpha \frac{L}{2} \hat{b}_x)]^T \hat{b}_x \leq 0. \tag{7.18}$$

These constraints in standard form include those shown in equations (7.15) and (7.16) with the following addition:

$$\begin{bmatrix} -\hat{b}_x^T r \\ \hat{b}_x^T r \end{bmatrix} - \begin{bmatrix} -\hat{b}_x^T & -L/2 & 0 \\ \hat{b}_x^T & -L/2 & 0 \end{bmatrix} \begin{bmatrix} x \\ \alpha \\ \gamma \end{bmatrix} \in \mathbf{R}_+^2. \tag{7.19}$$

Cone

As shown in Fig. 7.2d, a cone can be described with a height H, and a half angle β. The origin of the object-fixed frame r is one-quarter of the way from the flat face to the point of the cone, and α scales the distance of these two ends from the center point r:

$$\|\tilde{x}_{2:3}\|_2 \leq \tan(\beta)\tilde{x}_1, \tag{7.20}$$

$$(x - r - \alpha \frac{H}{4}\hat{b}_x)^T \hat{b}_x \leq 0, \tag{7.21}$$

where $\tilde{x} = Q^T(x - r + \alpha\frac{3H}{4}\hat{b}_x)$. These constraints in standard form are:

$$\hat{b}_x^T r - \begin{bmatrix} \hat{b}_x^T & -H/4 \end{bmatrix} \begin{bmatrix} x \\ \alpha \end{bmatrix} \in \mathbf{R}_+^m, \tag{7.22}$$

$$-EQ^T r - \begin{bmatrix} -EQ^T & v \end{bmatrix} \begin{bmatrix} x \\ \alpha \end{bmatrix} \in \mathcal{Q}_3, \tag{7.23}$$

where $E = \text{diag}(\tan\beta, 1, 1)$ and $v = (-\frac{3H}{4}\tan\beta, 0, 0)$.

Ellipsoid

An ellipsoid, shown in Fig. 7.2e, can be described by a quadratic inequality $x^T P x \leq 1$, where $P \in \mathbf{S}_{++}^n$ is strictly positive definite and has an upper-triangular Cholesky factor $U \in \mathbf{R}^{n\times n}$ [24]. From here, a scaled ellipsoid with arbitrary position and orientation can be expressed in the following way:

$$\|UQ^T(x - r)\|_2 \leq \alpha, \tag{7.24}$$

where a sphere of radius R is just a special case of an ellipsoid with $P = I/R^2$. These constraints can be written in standard form as:

$$\begin{bmatrix} 0 \\ -UQ^T r \end{bmatrix} - \begin{bmatrix} 0_{1\times 3} & -1 \\ -UQ^T & 0_{3\times 1} \end{bmatrix} \begin{bmatrix} x \\ \alpha \end{bmatrix} \in \mathcal{Q}_4. \tag{7.25}$$

Padded Polygon

A "padded" polygon is defined as the set of points within some radius R of a two-dimensional polygon. Shown in Fig. 7.2f, the first two basis vectors of $\mathbb{B}$ span the polygon, and the polygon itself is defined with a slack variable $y \in \mathbf{R}^2$, and $Cy \leq \alpha d$, where

$C \in \mathbf{R}^{m\times 2}$, and $d \in \mathbf{R}^m$, describe the m halfspace constraints for the polygon. This polygon is scaled in the same fashion as the polytope, and results in the following constraints:

$$\|x - (r + \tilde{Q}y)\|_2 \leq \alpha R, \tag{7.26}$$
$$Cy \leq \alpha d, \tag{7.27}$$

where $\tilde{Q} \in \mathbf{R}^{3\times 2}$ is the first two columns of Q. These constraints can be represented in standard form as the following:

$$0_m - \begin{bmatrix} 0_{m\times 3} & -d & C \end{bmatrix} \begin{bmatrix} x \\ \alpha \\ y \end{bmatrix} \in \mathbf{R}^m_+, \tag{7.28}$$

$$\begin{bmatrix} 0 \\ -r \end{bmatrix} - \begin{bmatrix} 0_{1\times 3} & -R & 0_{1\times 2} \\ -I_3 & 0_{3\times 1} & \tilde{Q} \end{bmatrix} \begin{bmatrix} x \\ \alpha \end{bmatrix} \in \mathcal{Q}_4. \tag{7.29}$$

7.3.3 Contact Points and Minimum Distance

While the computation of contact points and minimum distance between primitives is not needed for any of the examples in Section 7.4, they are easy to compute with DCOL if desired. The intersection point on the two scaled primitives is referred to as x^*, but unless $\alpha^* = 1$, this point does not exist on the surface of the primitives. The corresponding contact point for primitive i, $p_i \in \mathbf{R}^3$, is calculated using the optimal x^* and α^* from (7.10) as

$$p_i = r_i + \frac{x^* - r_i}{\alpha^*}, \tag{7.30}$$

where the intersection point between the scaled primitives is simply scaled back to each unscaled primitive. The distance between these points can also be calculated as follows:

$$\|d\|_2 = \|p_1 - p_2\|_2 = \|r_1 - r_2 + \frac{r_2 - r_1}{\alpha}\|_2. \tag{7.31}$$

Both of these operations are fully differentiable given the derivatives from DCOL, allowing for the calculation of the sensitivities of the contact points with respect to the configurations of the primitives.

7.4 Examples

In this section, we demonstrate the utility of differentiable collision detection in trajectory optimization problems where contact is to be avoided, and in physics simulation with contact where exact and differentiable collision information is required. In both of these applications, a collision constraint $\alpha \geq 1$ is used to enforce no interpenetration between each pair of primitives, where α is the minimum scaling from DCOL.

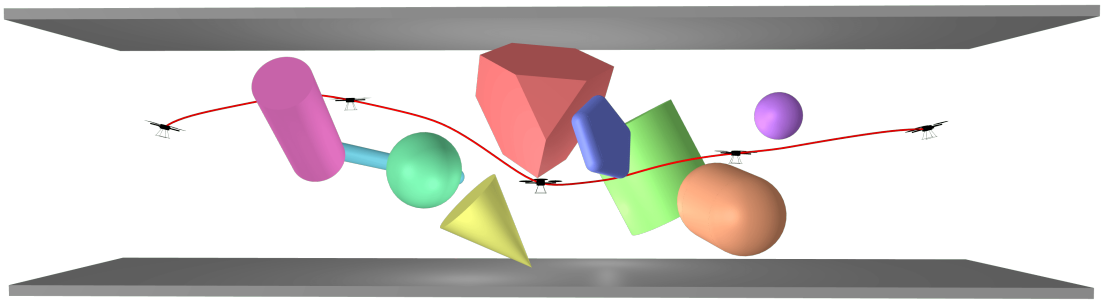


Figure 7.3: Trajectory optimization for a 6-DOF quadrotor as it moves from left to right through a cluttered hallway. The collision constraints were represented with DCOL, and the trajectory optimizer was initialized with a static hover at the initial condition.

7.4.1 Trajectory Optimization

Trajectory optimization is a powerful tool in motion planning and control, where a numerical optimization problem is formulated to solve for a constrained trajectory that minimizes a cost function. A generic trajectory optimization problem with collision avoidance constraints from DCOL is as follows:

$$\begin{array}{ll} \underset{x_{1:N},u_{1:N-1}}{\text{minimize}} & \ell_N(x_N) + \sum_{k=1}^{N-1} \ell_k(x_k, u_k) \\ \text{subject to} & x_{k+1} = f_k(x_k, u_k), \\ & h_k(x_k, u_k) \leq 0, \\ & g_k(x_k, u_k) = 0, \\ & \alpha_k(x_k) \geq 1, \end{array} \tag{7.32}$$

where k is the time step, x_k and u_k are the state and control inputs, ℓ_k and ℓ_N are the stage and terminal costs, $f(x_k, u_k)$ is the discrete dynamics function, $h_k(x_k, u_k)$ and $g_k(x_k, u_k)$ are inequality and equality constraints, and $\alpha_k(x_k)$ are the collision avoidance constraints from DCOL. Problems of this form can be solved with general purpose nonlinear program solvers like SNOPT [61], and Ipopt [187], or more specialized solvers like ALTRO [77, 81].

A key requirement for any gradient-based solver used to solve (7.32) is the ability to differentiate all of the cost and constraint functions with respect to the state and control inputs. This requirement has made collision-avoidance constraints difficult to incorporate into trajectory optimization frameworks because traditional collision detection methods are non-differentiable. In this section, DCOL is used to formulate collision-avoidance constraints in trajectory optimization problems to solve for collision-free trajectories.

"Piano Mover" Problem

The first problem we will look at is a variant of the *"Piano Mover"* problem, where a piano must maneuver around a 90-degree turn in a hallway [196, 146]. The walls are 1 meter apart, and the "piano" (a line segment) is 2.6 meters long, making the path around the corner nontrivial. This problem is solved with trajectory optimization and collision constraints, where the piano is parameterized as a cylindrical rigid body in two dimensions,

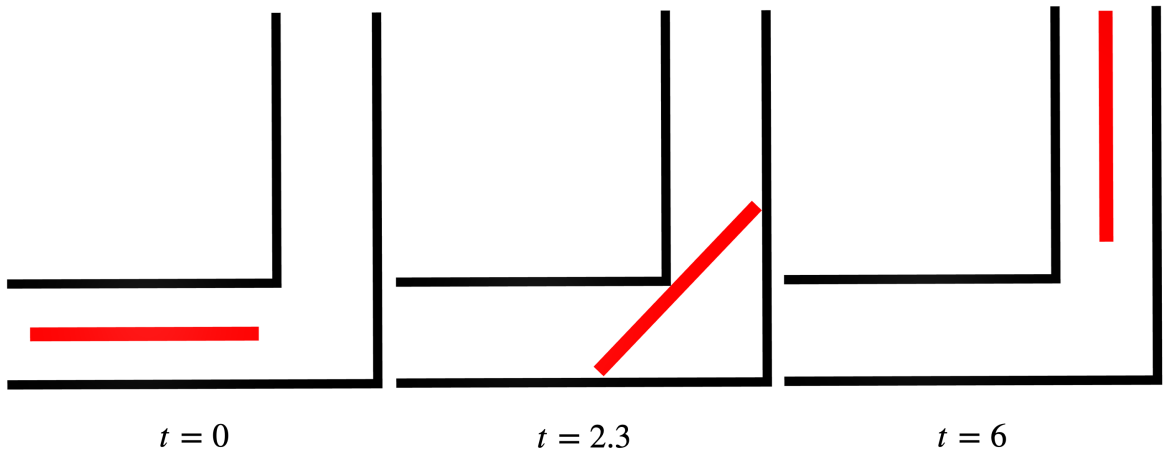


Figure 7.4: The *"Piano Movers"* problem, where a "piano" (red rectangle) has to make a turn down a hallway, is solved with trajectory optimization. The piano and the walls are modeled as rectangular prisms. DCOL was used to represent all of the collision avoidance constraints that ensure the piano cannot travel through the wall, and the trajectory optimizer was able to converge on a feasible trajectory to deliver the piano to the goal state.

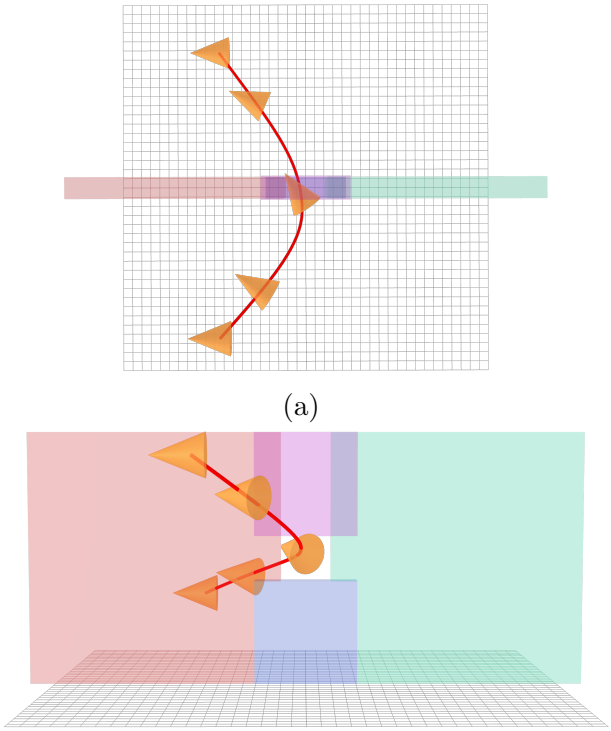
with a position and orientation, and the hallway is modeled with polytopes. The solution to this problem is shown in Fig. 7.4, where the piano successfully maneuvers around the tight corner and reaches the goal state without traveling through any of the walls. The trajectory optimizer was initialized with the piano in a static pose at the initial condition. [196] [146]

Quadrotor

Motion planning for quadrotors has received significant attention in recent years [117, 118, 157], with collision avoidance featured in many of these works [49, 137, 148]. With DCOL, we are able to directly and exactly incorporate collision avoidance constraints into a quadrotor motion planner to solve for trajectories through cluttered environments. In this example, we use trajectory optimization for a classic 6-DOF quadrotor model from [117, 80] to solve for a trajectory that traverses a cluttered hallway with 12 objects in it, shown in Fig. 7.3. The solver was initialized with the quadrotor hovering at the initial condition, and a spherical outer approximation of the quadrotor geometry was used to compute collisions. Despite this naive guess, the solver was able to quickly converge on a collision-free trajectory through the obstacles.

Cone Through an Opening

This example demonstrates how trajectory optimization with DCOL can route a cone through a square hole in a wall, as shown in Fig. 7.5. The dynamics of the cone are modeled as a rigid body with full translational and rotational control, and the wall is comprised of four rectangular prisms, making a rectangular opening in the wall. The



(a)

(b)

Figure 7.5: Trajectory optimization for a cone (orange) with translation and attitude control as it travels through a square opening in a wall. Top-down and side views are shown in (a) and (b), respectively. The cone is forced to slew to an attitude that allows for the passing of the cone through the opening before returning to the initial attitude. The trajectory optimizer was simply initialized with the static initial condition.

trajectory optimizer converged on a solution where the cone successfully passes through the opening in the wall, requiring that the cone slewed its orientation and "squeezed through" the opening. This example demonstrates the importance of the differentiability of the collision avoidance constraints, as the optimizer was forced to leverage both translational and rotational manipulation of the cone in order to successfully pass through the opening. As with the previous two examples, there was no expert initial guess provided to the trajectory optimizer, just a static initial condition.

7.4.2 Contact Physics

Another application of differentiable collision detection in robotics is contact physics for simulation. Rigid-body mechanics with inelastic collisions can be simulated using complementarity-based time-stepping schemes [78], where stationary points of a discretized action integral are solved for subject to contact constraints [112]. Normally these constraints are limited to traditionally differentiable ones like those between fixed contact points and a floor. The differentiability of DCOL enables these same methods to be extended for simulating con-

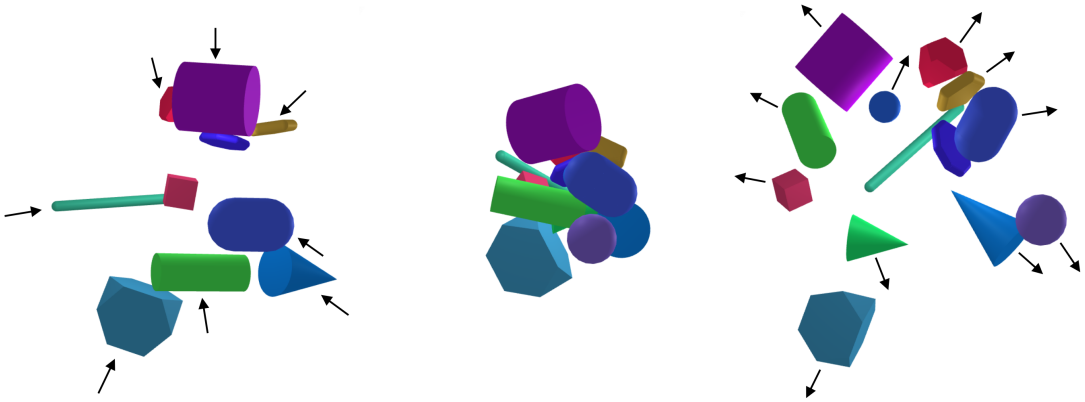


Figure 7.6: Contact physics with differentiable collision constraints embedded in a complementarity-based time-stepping scheme, simulated at 100 Hz. Twelve convex objects are started at random positions with velocities pointing towards the origin at $t = 0$. The objects impact each other at $t = 2$ and spread out again by $t = 3.5$. Despite the complexity of the simulation, the collision constraints can be enforced to machine precision and the integration is stable.

tact between convex primitives, as shown in Fig. (7.6) where twelve primitives collide. In terms of computation times, using DCOL for contact physics is reasonable given each constraint evaluation and differentiation are usually less than 10 μs as shown in Table 7.1.

7.5 Conclusion

We have presented DCOL, a fast differentiable collision detection algorithm capable of computing useful collision information and derivatives for pairs of any of six convex primitives. By formulating the collision-detection problem as an optimization problem that solves for the minimum uniform scaling that must be applied to each primitive before an intersection occurs, a surrogate proximity value is returned that is informative for primitives with or without a collision. Using differentiable convex optimization and a primal-dual interior-point conic solver, smooth derivatives of this optimization problem are returned after convergence with very little additional computation. The utility of DCOL is demonstrated in a wide variety of robotics applications, including motion planning and contact physics, where collision derivatives are required. Future work includes methods for convex decompositions of complex shapes as well as the incorporation of DCOL into existing physics engines. Our open-source Julia implementation of DCOL is available at https://github.com/kevin-tracy/DifferentiableCollisions.jl.

Chapter 8

Differentiable Continuous Collision Detection

Typical discrete collision detection can only consider robots and environments in their static configurations at discrete time steps. For sequences in which either the robot or the environment moves, discrete collision detection is unable to reason about collisions that occur *between* time steps. This can result in accidental collisions, as well as "tunneling," where two objects pass directly through one another. By contrast, continuous collision detection directly models and evaluates motion between two adjacent time steps. These continuous methods are often computationally expensive and nondifferentiable, restricting them to simple primitives for practical use in robotics. In this work, we present a method for performing continuous collision detection on arbitrary convex sets that reduces to a small convex optimization problem. The proposed formulation is indifferent to penetration, has no degenerate cases, and is fully differentiable and highly parallelizable. Videos, code, and more examples are available at https://continuous-collisions.github.io/.

8.1 Introduction

Given two objects in fixed configurations, Discrete Collision Detection (DCD) is used to determine if and where a collision exists. Modern robotic motion planning tools rely on collision detection between convex objects to synthesize collision-free trajectories. When DCD is used to identify collisions in a trajectory, it runs the risk of missing unintentional collisions between the discrete time steps. These collisions come from an effect called "tunneling," where an object speeds through an obstacle without violating collision avoidance constraints. This happens when collision checking is performed just before and after the object passes through the obstacle. In early video game engines, this would often result in players speeding through walls or getting stuck underneath floors [48].

To ensure that DCD is adequate for collision avoidance planning, a myriad of algorithmic enhancements have been proposed to combat these shortcomings. First, the time between discrete collision checks can be decreased, resulting in finer-grained collision checking at the expense of compute and problem complexity. Another common strategy is to add a collision margin that is conservative enough to combat any tunneling by making ob-

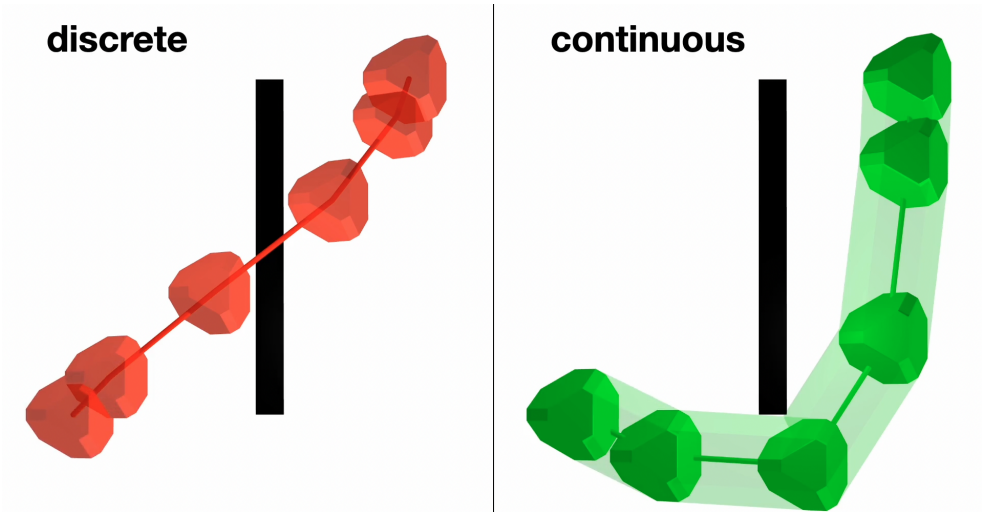


Figure 8.1: Example trajectory optimization solutions for a polytope passing near a wall where collision avoidance constraints are specified with discrete collision detection (red) and the proposed continuous collision detection (green). In the discrete case, the solver only has to satisfy the collision avoidance constraints at each of the discrete knot points, allowing the polytope to "speed" through the wall while appearing collision-free to the solver. Using the proposed continuous collision detection, the solver is able to directly reason about collision that happen at and between the time steps to ensure that no collision exists.

stacles appear thicker than they are. In cuRobo [158], this collision margin is augmented with a penalty that effectively increases the margin when the robot is moving faster, thus discouraging any "speeding through" obstacles. In TrajOpt [145], collisions between pairs of convex objects at adjacent time steps are avoided by forming a convex hull for each object and checking these convex hulls for any intersections. This approach successfully captures continuous collisions but is a potentially limiting over approximation of the collision environment since the objects are not actually encompassing the convex hulls for the duration of the time step.

Based on the complexity of the primitive, there are many available methods for DCD that each carry their own tradeoffs. For simple sphere-capsule-box interactions, analytical collision information can be solved for in closed form [48]. For generic convex shapes, iterative methods such as Gilbert-Johnson-Keerthi (GJK) [60, 31] and Minkowski Portal Refinement (MPR) [151, 129] are used. Both GJK and MPR only return useful collision information when objects are not in contact; as soon as penetration occurs, the Expanding Polytope Algorithm (EPA) [180] must be used to determine a penetration depth. Highly mature implementations of these algorithms are available in the Flexible Collision Library (FCL) [131], which many modern physics simulators rely on [37, 163, 96, 166]. While these DCD algorithms are fast and reliable, they involve significant logic and branching, making them inherently nondifferentiable and challenging to put on a GPU.

While DCD examines pairs of objects in static configurations, Continuous Collision

Detection (CCD) reasons about objects as they move between sequential configurations. CCD computes the same information that DCD does, but in addition, also calculates the time of impact, indicating when a collision takes place between two configurations. This new information comes at an expense, as CCD is significantly more difficult and limited in the primitives it can handle compared to DCD. For simple sphere-capsule-box primitives, a mix of analytical and iterative methods are available for computing the time of impact [48]. In [181], a GJK-based ray-casting method was proposed to compute the continuous collision information between a moving convex object and a static configuration-space obstacle. A method for direct handling of moving nonconvex meshes was introduced in [198], and was updated to include angular motion in [38]. While these existing CCD methods are readily available, they suffer from the same branching and nondifferentiability issues as DCD methods.

This work focuses on convex-convex CCD as it relates to gradient-based motion planning, where cost and constraint functions are required to be differentiable. There has been growing interest in differentiable DCD with sampling-based methods [122], and differentiable optimization-based methods [170, 201, 171]. The work proposed in this paper serves more closely as a continuous follow-on to the differentiable DCD solution introduced in [170], where a framework for computing smooth gradients through collision detection was enabled by scale-based collision detection and differentiable convex optimization.

We introduce a true CCD routine for pairs of arbitrary convex sets that is fully differentiable with respect to each configuration. By solving for collision information with a geometric scale factor, the collision check reduces to a small convex optimization problem that is guaranteed to be globally solvable, bounded, and feasible for any configuration of the objects. The solution of this optimization problem is then used to form smooth gradients of the collision information with respect to any problem parameters with very limited computational expense (significantly less than solving the problem). Together, this enables gradient-based motion planners to solve for collision-free trajectories with a coarse and efficient discretization of the trajectories.

Our specific contributions in this paper are the following:

- An optimization-based continuous collision-detection method between pairs of moving convex sets without any branching or degenerate cases
- A fast and efficient method for differentiating the method in automatic-differentiation frameworks
- A convex interior-point algorithm for solving the collision problem that can be arbitrarily parallelized on a GPUs or TPUs for problems with varying numbers of constraints

This paper proceeds by providing background information in Section 8.2, introducing our CCD algorithm in Section 8.3, detailing a parallelizable convex solver in Section 8.4, examples of these methods used for collision avoidance motion planning in Section 8.5, and our concluding remarks in Section 8.6.

8.2 Background

The continuous collision-detection algorithm introduced in this paper involves forming, solving, and differentiating a convex optimization problem. In this section, the necessary background to understand this formulation is presented.

8.2.1 Perspective Operators

In order to simplify the notation around scale-based collision detection, the idea of a perspective operator for a convex set is introduced. For collision purposes, each rigid body in a robot is decomposed into a collection of convex sets [145]. Given a closed convex set $\mathcal{X} \subseteq \mathbf{R}^N$, a perspective operator is used to map this n-dimensional set to a cone in $n+1$ dimensions [109, 108, 143, 70]. The new dimension comes from a scale factor $\alpha \in \mathbf{R}_+$ that geometrically scales up the set when $\alpha > 1$ and scales down the set when $\alpha \leq 1$ until it is just a point at $\alpha = 0$.

For practical purposes, the simplest way to exploit the perspective operator is to define the original convex set in standard conic form, $\mathcal{X} = \{x : Ax + b \in \mathcal{K}\}$, where $x \in \mathbf{R}^3$ is a point in some world frame $\mathbb{W}$, and $\mathcal{K}$ is a proper convex cone [24, 100]. Common convex sets such as polytopes, cylinders, cones, capsules, and ellipsoids, can easily be represented in this form [170]. From here, the perspective operator defines a new scaled convex set $\bar{\mathcal{X}}$ as the following:

$$\bar{\mathcal{X}} = \{(x, \alpha) : \alpha \geq 0, Ax + \alpha b \in \mathcal{K}\}. \tag{8.1}$$

For example, for a polytope described by $Bx + c \geq 0$ (where the cone $\mathcal{K}$ is the nonnegative orthant $\mathbf{R}_+$), the scaled polytope becomes $Bx + \alpha c \geq 0$.

It is often most convenient to express a convex set with an attached body frame $\mathbb{B}$, with an origin $\mathbb{B}_0$. Instead of redefining the convex set every time it moves in the world frame, we can instead simply define it once in its own body frame and parameterize any SE(3) rigid transformation with translation $r = {}^{\mathbb{W}}r_{\mathbb{B}}^{\mathbb{B}_0} \in \mathbf{R}^3$ and rotation $Q = {}^{\mathbb{W}}Q^{\mathbb{B}} \in \mathbf{R}^{3\times 3}$. This allows us to write a general constraint for a convex set given a specific position and attitude as:

$$AQ^T(x - r) + \alpha b \in \mathcal{K}. \tag{8.2}$$

This method for defining scaled convex sets in a world frame enables a straightforward extension to optimization-based collision detection built on this scaling.

8.2.2 Scale-Based Collision Detection

Traditionally, collision detection between two convex sets is done by searching for the closest point between these two sets. If the closest points are not coincident, the objects are not in collision. This is the backing for the popular collision detection algorithms GJK [60, 31] and MPR [151, 129], enabling fast and efficient collision detection between common convex primitives. The two main drawbacks of this approach come from nondifferentiability

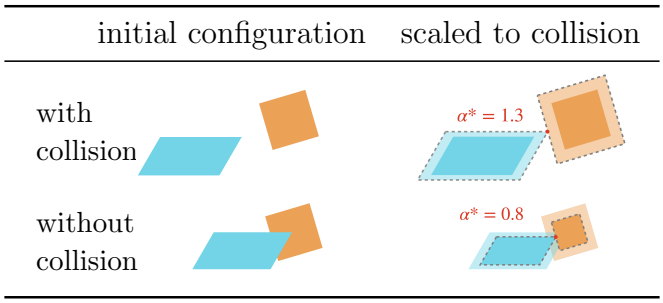


Figure 8.2: Scale-based collision detection for objects that are in and out of collision. In the upper row, the two objects are not in collision, so the minimum scaling that results in an intersection is able to "grow" the objects to $\alpha^* = 1.3$. In the bottom row, the two objects are already in collision, so the minimum uniform scaling shrinks the objects down to $\alpha^* = 0.8$.

of these algorithms and the degenerate case when the two sets are in collision. In the latter case, GJK and MPR cannot return useful information and instead must default to an alternative algorithm such as EPA to establish a "penetration depth."

As shown in [170], scale-based collision detection avoids both of these shortcomings by solving a different optimization problem. Instead of searching for the closest points between two fixed sets, the perspective operator is used to scale the two sets by a common α and solve for the minimum shared scaling that results in an intersection.

With a point in the world frame $x \in \mathbf{R}^3$ and two convex sets defined with (8.2) as $A_i Q_i^T (x - r_i) + \alpha b_i \in \mathcal{K}$, the optimization problem that finds the minimum scaling α that results in an intersection between two scaled convex sets is:

$$\begin{aligned} \underset{x,\,\alpha}{\text{minimize}} \quad & \alpha \\ \text{subject to} \quad & A_i Q_i^T (x - r_i) + \alpha b_i \in \mathcal{K}_i \quad i = 1, 2, \\ & \alpha \geq 0. \end{aligned} \tag{8.3}$$

This is a convex optimization problem that is guaranteed to be both feasible and bounded. In the event that the two convex sets are in collision, the sets are scaled down to $\alpha < 1$ to the smallest possible scale factor that has an intersection. If the two sets are not in collision, they are scaled up to $\alpha > 1$ until an intersection is reached. In both cases, the fundamental algorithm remains unchanged, as the optimization formulation in (8.3) handles both cases. Furthermore, there are no degenerate cases based on the geometry or configurations of the sets, since there always exists a minimum scaling for an intersection, even when the intersection is another set (a line or a plane).

8.2.3 Differentiable Optimization

Collision detection as a solution to an optimization problem in (8.3) makes taking derivatives with automatic differentiation tools challenging. Both forward and reverse-mode automatic differentiation are unable to accurately compute derivatives through an iterative

numerical solver [7]. To avoid this, differentiable optimization tools are used to compute these derivatives in a fast and accurate way without propagating derivatives through iterative schemes.

Given a generic optimization problem in the following form:

$$\begin{array}{ll} \underset{y}{\text{minimize}} & f_\theta(y) \\ \text{subject to} & c_\theta(y) \in \mathcal{K}, \end{array} \tag{8.4}$$

with the cost and constraints as functions of some problem parameters θ, a dual variable $\lambda \in \mathbf{R}^m$ is used to enforce the constraint in the following Lagrangian:

$$\mathcal{L}_\theta(y, \lambda) = f_\theta(y) + \lambda^T c_\theta(y). \tag{8.5}$$

Using this, the KKT conditions necessary for optimality are,

$$\nabla_y f_\theta(y) + \left[\frac{\partial c}{\partial y}\right]^T \lambda = 0 \tag{8.6}$$

$$c_\theta(y) \in \mathcal{K} \tag{8.7}$$

$$\lambda \in \mathcal{K}^* \tag{8.8}$$

$$c_\theta(y) \circ \lambda = 0, \tag{8.9}$$

where $\mathcal{K}$ is a proper cone and $\mathcal{K}^*$ and $\circ$ are the corresponding dual cone and cone product, respectively [183]. Any primal-dual values (y, λ) that satisfy equations (8.6)-(8.9) are a locally optimal solution to 8.4.

By viewing equations (8.6) and (8.9) as the following implicit function of $w = (y, \lambda)$,

$$g_\theta(w) = \begin{bmatrix} \nabla_y f_\theta(y) + \left[\frac{\partial c}{\partial y}\right]^T \lambda \\ c_\theta(y) \circ \lambda \end{bmatrix}, \tag{8.10}$$

an approximate primal-dual solution $w^* = (y^*, \lambda^*)$ is an equilibrium point where $g_\theta(w^*) \approx 0$. At this equilibrium, the implicit function theorem can be used to calculate the Jacobian of the primal-dual solution with respect to the to the problem parameters [78]:

$$\frac{\partial y}{\partial \theta} = -\left(\frac{\partial g}{\partial y}\right)^{-1} \frac{\partial g}{\partial \theta}. \tag{8.11}$$

Practically, this works by solving the problem with a standard numerical solver, after which the approximate solution is used with the implicit function theorem to provide the necessary derivatives.

Alternatively, if the only derivative we need is that of the optimal objective value $f(x^*)$ with respect to the problem parameters, an even simpler approach can be used: At a primal-dual solution, the gradient of the objective value with respect to the problem parameters is simply the gradient of the Lagrangian with respect to these same parameters [33],

$$\nabla_\theta f_\theta(y^*) = \nabla_\theta \mathcal{L}_\theta(y^*, \lambda^*), \tag{8.12}$$

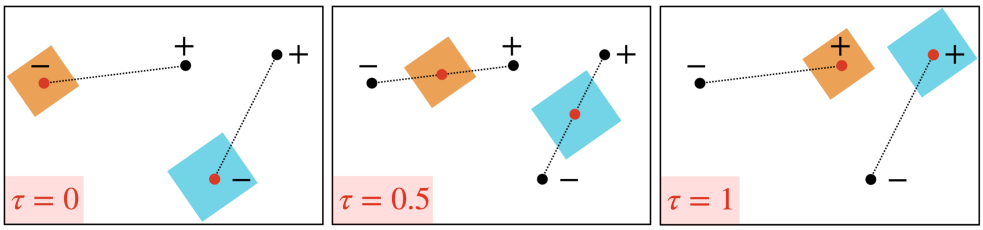


Figure 8.3: Graphical description of how the $\tau \in [0, 1]$ parameter linearly interpolates the origin of the objects from their initial positions at $r_i^{(-)}$ at $\tau = 0$, to $r_i^{(+)}$ at $\tau = 1$. Discrete collision detection can only check objects for collisions at $\tau = 0$ and $\tau = 1$, while the proposed continuous collision detection method can determine whether there is a collision as each object moves between the two positions. It is important to note that this sweeping motion is *not* simply checking if two convex hulls are in collision, but rather checks for contact as both objects move simultaneously on the path.

allowing us to calculate the derivative we need without forming or solving a linear system.

These differentiable optimization methods are used to convert the always-feasible convex optimization problem in (8.3) into a fully differentiable algorithm, as shown in [170]. The optimal objective value (α) or the primal and dual variables can be differentiated with respect to both the configurations of the objects and the geometry of the objects themselves. In [170] this was used in trajectory optimization to solve for collision-free trajectories simply by enforcing $\alpha > 1$ as a differentiable constraint in a motion planner.

8.3 Continuous Collision Detection

With the necessary background on perspective operators, scale-based collision detection, and differentiable optimization, the novel CCD algorithm can now be introduced that checks for collisions between two convex sets as they move between configurations.

The core assumption in this CCD framework is that the two objects move linearly between time steps with a fixed attitude. This motion can be seen in Fig. 8.3, where two shapes "sweep" between the time steps. Given the two convex sets with positions $r_i \in \mathbf{R}^3$ and attitudes $Q_i \in \mathbf{R}^{3\times3}$, as described in Section 8.2.1, we can introduce the notation for time. Positions and attitudes from the first time step will be denoted with $\square^{(-)}$, and those from the second time step with $\square^{(+)}$. A new normalized time parameter $\tau \in [0, 1]$ is introduced where $\tau = 0$ denotes the first time step and $\tau = 1$ denotes the second. Together, we can express this notation as $r_i(\tau = 0) = r_i^{(-)}$, and $r_i(\tau = 1) = r_i^{(+)}$. A line segment between these two positions can be calculated with the following,

$$r_i(\tau) = \tau r_i^{(-)} + (1 - \tau) r_i^{(+)}, \tag{8.13}$$

where we simply linearly interpolate from the position at one time step to the position at the next with a constant linear velocity. The attitudes between the time steps are simply averaged as Q_i, which is straightforward depending on the parameterization of the

attitude [111, 110]. Now, a scale-based collision-detection problem, similar to (8.3), is introduced as follows,

$$\begin{aligned} \underset{x,\,\alpha,\,\tau}{\text{minimize}} \quad & \alpha \\ \text{subject to} \quad & A_i Q_i^T (x - \tilde{r}_i) + \alpha b_i \in \mathcal{K}_i, \quad i = 1, 2, \\ & \tau r_i^{(-)} + (1 - \tau) r_i^{(+)} = \tilde{r}_i, \quad i = 1, 2, \\ & 0 \leq \tau \leq 1, \\ & \alpha \geq 0, \end{aligned} \tag{8.14}$$

where we are searching for the minimum positive scale factor α such that an intersection exists between the two scaled convex sets as they sweep between the time steps. Similar to (8.3), a collision exists if and only if $\alpha^* \leq 1$. This is a small convex optimization problem that is guaranteed to have a solution no matter the configuration of the objects. It makes no difference to the algorithm if the objects are in or out of collision for any amount of the sweep, or if they don't collide at all. The constraint $\alpha \geq 0$ is written here as a formality, since it is implicitly satisfied by the perspective operator.

Once this problem is formed and solved, the primal and dual variables can be used to calculate the gradients of the optimal α^* with respect to the poses or geometries of the objects using (8.12). If Jacobians of the primal or dual variables with respect to the problem parameters are needed, the implicit function theorem can be used to form these derivatives with (8.11).

For numerical stability, it is often advantageous to add a very small regularization term to (8.14),

$$J_{reg} = \epsilon[\|x - r_{avg}\|_2^2 + (\alpha - 1)^2 + (\tau - 0.5)^2], \tag{8.15}$$

where $\epsilon << 1$ and $r_{avg} \in \mathbf{R}^3$ is the average of the four positions (two objects at two time steps). This regularizer serves to guarantee that (8.14) is *strongly* convex, and can sometimes help the solver converge in fewer iterations [24, 130]. With a small regularizer in the range $\epsilon \in [10^{-8}, 10^{-5}]$, the regularized optimal objective value does not differ meaningfully from α^*.

8.4 Parallelizable QP Solver

In this section, a fast and efficient Primal-Dual Interior-Point (PDIP) solver will be described that is capable of running in parallel on accelerator units (GPU/TPU). Since the optimization problem in (8.14) is small (5 primal variables), state-of-the-art PDIP methods can solve this problem quickly and reliably. We focus on polytopes as our convex primitive, as they are expressive and result in (8.14) being a convex Quadratic Program (QP).

Writing convex optimization solvers on a GPU has become significantly easier in recent years, with the advent of domain-specific languages like PyTorch [136] and JAX [25]. High-level Python code can be compiled to run directly on GPUs/TPUs allowing for massively parallel computation with minimal implementation effort.

The challenge in scaling up these computations for our proposed collision detection comes from the varying sizes of the convex sets in the scene. Since we are focusing on polytopes in this section, this translates to dealing with polytopes with different numbers of faces. In JAX, the best way to parallelize a computation is to call the same function over a list of equal sized arrays. In the case of polytopes with varying numbers of faces, this results in QPs with different numbers of constraints, violating our assumption of equal-sized arrays.

In order to ensure that the arrays being parallelized over are of equal size, many algorithms simply choose the largest array in the set and pad all of the smaller arrays with zeros. In the case of constraints for a QP solver, this results in redundant constraints, violating the Linear Independence Constraint Qualification (LICQ) assumption that many solvers rely on to function properly [130, 78]. Violations of LICQ result in solvers failing due to rank-deficient linear systems caused by a nonuniquness of the dual variables. We propose a custom PDIP algorithm that solves the necessary linear systems with a block reduction that is capable of handling redundant constraints.

The solver presented in Alg. 11 handles convex QPs in the following form:

$$\begin{aligned} \underset{x,\, s}{\text{minimize}} \quad & \frac{1}{2} x^T P x + q^T x \\ \text{subject to} \quad & Gx + s = h, \\ & s \geq 0, \end{aligned} \tag{8.16}$$

with a primal and slack variable $x \in \mathbf{R}^n$ and $s \in \mathbf{R}^m$, a cost described with $P \in \mathbb{S}^n_+$ and $q \in \mathbf{R}^n$, and an inequality constraint described with $G \in \mathbf{R}^{m \times n}$ and $h \in \mathbf{R}^m$. The constraint is enforced with a dual variable $z \in \mathbf{R}^m$, and the only requirement for the initialization of this solver is $(s, z) \geq 0$. An advanced initialization from [115] can also be used for added robustness.

To ensure that all the problem instances have the same number of constraints, the inequality constraint parameters G and h are padded with zeros. To avoid numerical problems in the solver, care is taken to use a binary mask vector $m \in \mathbf{R}^m$ that identifies zero-padded constraints, and a specialized block reduction method is used to calculate the step directions in Alg. 12. Once a step direction has been computed, an analytical linesearch is used to ensure non-negativity of the slack and dual variables:

$$\gamma = \min\left(1, \min_{i:\Delta s_i < 0} \frac{-s_i}{\Delta s_i}, \min_{i:\Delta z_i < 0} \frac{-z_i}{\Delta z_i}\right). \tag{8.17}$$

8.4.1 Differentiation of the QP

Using the differentiable optimization technique described in (8.12), the gradients of the optimal objective value J^* at the primal-dual solution (x^*, z^*) can be formed by simply

Algorithm 11 Primal-Dual Interior-Point Method

function solve_qp(P, q, G, h)
$m \leftarrow$ get_mask(G, h) ▷ identify empty rows
$x, s, z \leftarrow$ initialize(P, q, G, h)
for $i \leftarrow 1 :$ max_iters **do**
    // evaluate KKT and check convergence
    $v_1 \leftarrow G^T z + Px + q$ ▷ stationarity
    $v_2 \leftarrow s \circ z$ ▷ complimentarity
    $v_3 \leftarrow Gx + s - h$ ▷ primal feasibility
    if $\min(\|v_1\|_\infty, \|m \circ v_2\|_\infty, \|m \circ v_3\|_\infty) <$ tol **then**
        return: $x, m \circ s, m \circ z$
    end if

    // calculate affine step direction
    $\Delta x_a, \Delta s_a, \Delta z_a \leftarrow$ solve_step$(P, q, z, s, v_1, v_2, v_3)$

    // centering + step correction
    $\mu \leftarrow s^T z / \text{len}(s)$
    $\gamma_a \leftarrow$ linesearch$(s, \Delta s_a, z, \Delta z_a)$ ▷ (8.17)
    $\sigma \leftarrow [(s + \gamma_a \Delta s_a)^T (z + \gamma_a \Delta z_a)/(s^T z)]^3$
    $v_2 \leftarrow v_2 - \sigma\mu + \Delta s_a \circ \Delta z_a$
    $\Delta x, \Delta s, \Delta z \leftarrow$ solve_step$(P, q, z, s, v_1, v_2, v_3)$

    // final linesearch to ensure $s, z > 0$
    $\gamma \leftarrow \min(1, 0.99\,\text{linesearch}(s, \Delta s), z, \Delta z))$ ▷ (8.17)
    $(x, s, z) \leftarrow (x, s, z) + \gamma(\Delta x, m \circ \Delta s, m \circ \Delta z)$
end for

Algorithm 12 Interior-Point Linear System Solver

function solve_step$(P, G, s, z, v_1, v_2, v_3)$
$W \leftarrow D(z/s)$
$L \leftarrow$ cholesky$(P + G^T W G)$ ▷ cache and re-use
$\Delta x \leftarrow L^{-T} L^{-1}(-v_1 + G^T W(v_2/\lambda - v_3))$
$\Delta s \leftarrow -G\Delta x - v_3$
$\Delta z \leftarrow -(v_2 + (z \circ \Delta s))/s$
return: $\Delta x, \Delta s, \Delta z$

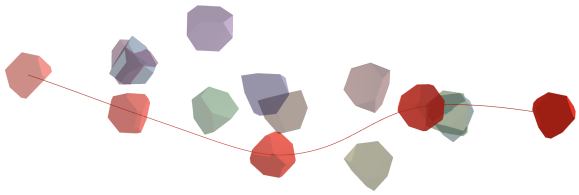


Figure 8.4: A polytope with position and attitude control navigates a field of moving obstacles. Differentiable continuous collision detection is used in a trajectory optimization to solve for the collision-free sequence.

taking the gradients of the Lagrangian as follows:

$$\nabla_P J^* = \frac{1}{2}(x^*)(x^*)^T, \qquad \nabla_q J^* = x^*, \tag{8.18}$$

$$\nabla_G J^* = z(x^*)^T, \qquad \nabla_q J^* = -z^*. \tag{8.19}$$

From here we can simply write a custom gradient rule in an automatic differentiation framework like JAX to seamlessly integrate CCD into existing differentiable pipelines.

8.5 Examples

The utility of flexible and differentiable CCD is demonstrated in motion-planning examples in which gradient-based trajectory optimization is used to solve for collision-free trajectories. To do this, we consider discrete-time dynamical systems of the form $x_{k+1} = f(x_k, u_k)$, and solve the motion planning problem as a numerical optimization problem:

$$\begin{aligned} \underset{x_{1:N},\, u_{1:N-1}}{\text{minimize}} \quad & \sum_{i=1}^{N-1} \ell(x_i, u_i) + \ell_N(x_N) \\ \text{subject to} \quad & x_1 = x_{init}, \\ & x_{k+1} = f(x_k, u_k) \quad i = 1 : N-1, \\ & \alpha(x_k, x_{k+1}) \geq 1 + d \quad i = 1 : N-1, \\ & x_N = x_{goal}, \end{aligned} \tag{8.20}$$

where $x_{1:N}$ and $u_{1:N-1}$ are the states and controls for an N-step trajectory, x_{init} and x_{goal} are the initial and final conditions, $\ell(x, u)$ and $\ell_N(x)$ make up the cost function, and $\alpha(x_k, x_{k+1})$ represent the continuous collision constraints that α from (8.14) must be greater than $1+d$, where $d \in \mathbb{R}_+$ is some margin. From here, this problem can be solved with generic nonlinear programming (NLP) solvers like SNOPT [61] or Ipopt [187].

8.5.1 Tunneling

The first example, shown in Fig. 8.1, showcases a classic failure mode of discrete collision checking — tunneling. Trajectory optimization is used to find a trajectory that guides a

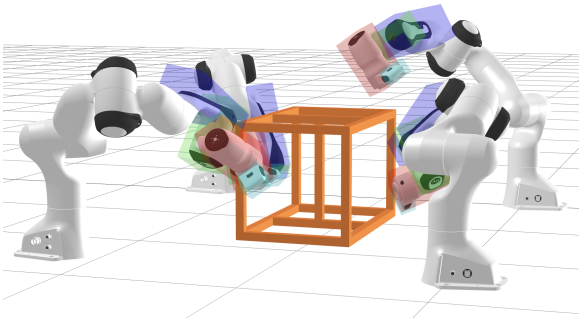


Figure 8.5: A multi-robot assembly task where four robotic arms interact with a structure in a common workspace. The proposed continuous collision detection is used to certify collision-free trajectories and is fully differentiable with respect to the configurations of the robots.

polytope around a wall, and we directly compare the optimal trajectories when DCD and CCD are used. With DCD (shown in red), the polytope passes directly through the wall in a straight line towards the goal. On inspection, this is clearly an infeasible trajectory despite each the configuration at each time step being out of collision. Since the solver can only reason about discrete collision checks, it is a feasible and optimal trajectory in the context of the solver. With our differentiable CCD, even with a very coarse trajectory discretization, the solver is able to avoid tunneling and navigate around the wall instead of through it.

8.5.2 Moving Obstacles

In this example, shown in Fig. 8.4, trajectory optimization with CCD constraints is used to navigate a polytope through a field of moving obstacles. Every object in this scene is moving, and the CCD is able to reason directly about this motion to ensure that there are no accidental collisions. The resulting collision-free trajectory successfully takes the polytope from the start to the goal without hitting any of the ten obstacles.

8.5.3 Multi-Robot Assembly

As shown in Fig. 8.5, this scenario involes four Franka Panda robots completing a mock assembly task in unison in a shared workspace. The robots have had their end effectors discretized into convex sets and are interacting with a structure made up of polytopes. CCD ensures the whole operation is collision-free, even between timesteps.

8.6 Conclusion

In this paper we present a fully differentiable continuous collision-detection framework that is fast, robust, and entirely branch free. This algorithm is readily parallelized in modern

compute frameworks such as JAX, and directly integrates with automatic differentiation tools for seamless differentiation through the routine. The resulting method is used in gradient-based trajectory optimization problems as collision avoidance constraints and is demonstrated on a range of practical examples.

Chapter 9

Efficient Online Learning of Contact Force Models

Contact-rich manipulation tasks with stiff frictional elements like connector insertion are difficult to model with rigid-body simulators. In this work, we propose a new approach for modeling these environments by learning a quasi-static contact force model instead of a full simulator. Using a feature vector that contains information about the configuration and control, we find a linear mapping adequately captures the relationship between this feature vector and the sensed contact forces. A novel Linear Model Learning (LML) algorithm is used to solve for the globally optimal mapping in real time without any matrix inversions, resulting in an algorithm that runs in nearly constant time on a GPU as the model size increases. We validate the proposed approach for connector insertion both in simulation and hardware experiments, where the learned model is combined with an optimization-based controller to achieve smooth insertions in the presence of misalignments and uncertainty. Our website featuring videos, code, and more materials is available at https://model-based-plugging.github.io/.

9.1 Introduction

Model-based control for robotic systems is incredibly effective when the model of the system is accurate [162]. In many instances, the model can be made accurate through system identification, where experimental data is used with a parametrized physics model to estimate geometric, mass, contact, and friction properties [54, 73]. In scenarios where contact-free rigid-body dynamics accurately capture the behavior of the robot, system identification can enable highly performant model-based control. For robots in contact-rich settings where they make and break contact with the environment, system identification and simulation become much more challenging due to discontinuities and inevitable approximations in the physics modeling [200].

Modern robotics simulators like MuJoCo [165], Bullet [37], Dart [96], Dojo [78], and Isaac Gym [105], approximate all bodies as rigid, where friction is either the idealized

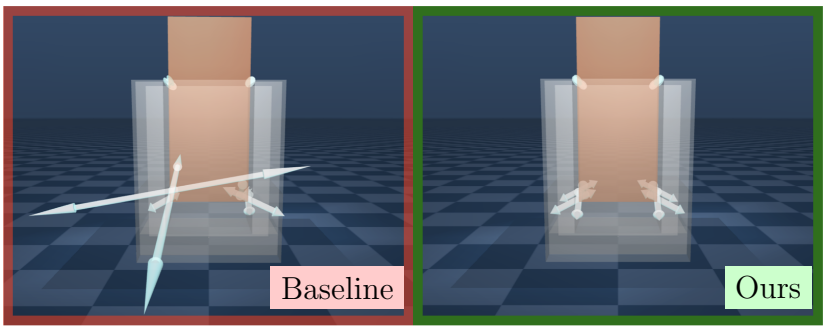


Figure 9.1: Inserting a significantly misaligned connector in MuJoCo, where a scripted insertion (left) results in large contact forces shown in white. During a short calibration sequence, a model is learned that predicts these contact forces as a function of the configuration of the connector and applied control wrench. A model-based controller is then able to solve for actions that minimize these forces and guide the connector to a smooth insertion (right). Our approach is fast, data efficient, robust to misalignments and uncertainty, and does not require any a priori knowledge of the plug.

Coulomb friction or a closely related approximation. This results in a significant gap between simulation and reality, even with the use of system identification [1, 76, 34, 47]. Accurate simulation of contact-rich manipulation tasks is challenging due to deformability in the bodies, approximate geometries, and complex friction behavior not captured by Coulomb friction [44].

This paper focuses on the problem of robotic connector insertion, which displays rich contact interactions between a plug and its corresponding socket. This task is difficult to simulate accurately because of the precision required (being off by a millimeter results in jamming), and the contact/friction interactions between the two slightly deformable objects. The point-contact assumption that rigid-body simulators make begins to break down as the contact area dynamically changes. As a result, accurately simulating the insertion is challenging, and a policy that relies on the accuracy of this simulation is poorly suited for the diverse range of connectors available. Furthermore, even if a single connector style is modeled in great detail, there exists a significant amount of variability amongst connectors of the exact same specification.

The difficulty of the connector-insertion problem has motivated multiple approaches, some model-based and others model-free. The authors in [199] propose an offline model-free meta-reinforcement-learning algorithm to pre-train policies for insertion tasks using offline demonstration data and then adapt them to new tasks using online fine tuning. In [52], the insertion is divided into an alignment phase and an insertion phase. During the alignment phase, a policy that relies on tactile feedback is used to line up the connector, and insertion is achieved with a separate RGB-image-based policy. In [125], a simulation is used with domain randomization to train a robust policy over a wide variety of connectors, enabling generalization; and in [161] a simulation-aware policy training style is used to prioritize experiences that align with the physics model. Alternatively, in [90], a policy is used solely to generate a search pattern that systematically inserts connectors in the

presence of model and state uncertainty. In all of these approaches, significant time and effort must be devoted to offline training from experiments, making rapid adaptation to new plugs challenging.

In this work, we present a different approach to model-based connector insertion that relies on a learned quasi-static *contact-force model* instead of a full simulator. Many contact-rich robotic manipulation tasks can be assumed to be quasi-static, meaning the system moves slowly enough that the Coriolis forces and accelerations in the dynamics are small enough to be ignored [113, 65]. This approximation allows us to describe the state of the robot solely by its configuration (without velocity). When compared to full second-order dynamics simulation, quasi-static simulators are more stable, simpler, and have shown great promise for use in contact-rich manipulation planning [132, 156].

When analyzing experimental data, a linear model with an engineered feature vector was found to be expressive enough to accurately predict contact forces on the connector during an insertion task. The linearity (in the features) of this model allows for an entirely decoupled model-learning process, where a novel Linear Model Learning (LML) filter is developed for learning the maximum a posteriori estimate to global optimality in a recursive fashion. The resulting algorithm is entirely free of matrix inversions, and the run time is nearly constant on a GPU as the size of the model increases. This model is then used in a convex-optimization-based control policy where the connector is smoothly inserted. The contributions of this paper include:

- A validated contact-force model that linearly maps a feature vector with information about the configuration and control to predicted force torque sensor values during a connector insertion.
- The Linear Model Learning (LML) algorithm that recursively updates the optimal estimated contact force model without any matrix inversions.
- A model-based controller that uses this contact force model to achieve smooth connector insertions.

9.2 Feature-Based Contact Force Model

Conventionally, physics simulations used in model-based control take the form of a discrete dynamics function $x_{k+1} = f(x_k, u_k)$, where the current state, x_k, and control, u_k, are used to calculate the state at the next time step, x_{k+1}. The state of the system, x, is normally comprised of a configuration, $q \in \mathbf{R}^{n_q}$, and velocity, $v \in \mathbf{R}^{n_v}$ [162]. For systems modeled with standard rigid-body dynamics, the full second-order dynamics depend on both the configuration of the system as well as the velocity. However, for the damped, slow-moving systems common in contact-rich manipulation, velocity-dependent terms like Coriolis forces can be small enough to be effectively ignored [113]. The quasi-static assumption also simplifies the actuator dynamics of impedance-controlled robots, where the behavior of the controller is directly incorporated as part of the model [134].

For quasi-static contact-rich tasks where the end effector of the robot is in constant contact with the environment, we can learn a contact-force model and still maintain all of the predictive power of a full simulator. To accurately represent the dynamics in this

regime, all we need is a model that takes in the current configuration of the robot and applied control, and outputs the predicted force-torque sensor reading. With the ability to predict how the force-torque sensor responds to control inputs, a model-based controller is able to fully reason about the contact environment.

The model we seek to learn takes the configuration of the system, the control, $u \in \mathbf{R}^{n_u}$, and the contact force torque measurement, $\tilde{y} \in \mathbf{R}^{n_y}$ as inputs. Specifically, we are going to map the configuration and control into a feature vector, $w \in \mathbf{R}^{n_w}$, using an arbitrary mapping function,

$$w = \Omega(q, u), \tag{9.1}$$

after which we learn a *linear* mapping that transforms the feature vector into the predicted force, $f \in \mathbf{R}^3$, and torque, $\tau \in \mathbf{R}^3$, on the connector, $\tilde{y} = [f^T, \tau^T]^T \in \mathbf{R}^6$:

$$\tilde{y} \approx \tilde{G} w, \tag{9.2}$$

where $\tilde{G} \in \mathbf{R}^{n_y \times n_w}$. There are multiple benefits to representing our force model as a linear function of the feature vector, both in terms of system identification and control. For system identification, the resulting optimization problem is convex and can be solved to global optimality in real time with minimal computational cost. For control, as long as the feature vector in (9.1) is linear in u, the predicted force and torque can be minimized with convex optimization. This is true even in the presence of a feature vector that is highly nonlinear in q.

9.3 Linear Model Learning

Given data from a trajectory, a maximum-likelihood estimate (MLE) of the linear mapping $\tilde{G}$ can be computed with numerical optimization by penalizing differences between the predicted and observed sensor measurements [197]. To express this, we consider our contact force model from (9.2) with a force-torque sensor noise covariance $R \in \mathbf{S}_+^{n_y}$, and introduce the following optimization problem:

$$\underset{\tilde{G}}{\text{minimize}} \quad \sum_{k=1}^{m} \|\tilde{G} w_k - \tilde{y}_k\|_{R^{-1}}^2, \tag{9.3}$$

where the subscript k denotes the quantity from the k^{th} timestep, m is the length of the trajectory, and $\|\nu\|_{R^{-1}}^2 = \nu^T R^{-1} \nu$. It is worth pointing out that the primal variable in (9.3) is a matrix and not a vector, so it is not a canonical least squares problem. That being said, the linearity of the model still makes this an unconstrained convex optimization problem, where a closed-form solution exists [24]. However, for systems where the dimension of the feature vector is large, this problem can be both expensive to solve and numerically ill-conditioned when the trajectory is long [16].

There is a long history of solving variants of (9.3) from data, starting with the Ho-Kalman algorithm [72] to more modern versions as shown in [54, 58, 68, 66]. Instead of

solving this problem directly, the following subsections walk through a method for decomposing (9.3) into a set of smaller decoupled problems that can be solved in parallel, as well as a recursive method for solving this problem online in a scalable and numerically robust way that completely avoids matrix inversions.

9.3.1 Decoupled Model Learning

The first step in decoupling (9.3) into a set of n_y smaller problems is to take a Cholesky decomposition of the inverse sensor covariance, R^{-1}, such that $R^{-1} = LL^T$, where $L \in \mathbf{R}^{n_y \times n_y}$ is the lower-triangular Cholesky factor. From here, (9.3) can be re-written as the following:

$$\underset{\tilde{G}}{\text{minimize}} \quad \sum_{k=1}^{m} \|L^T \tilde{G} w_k - L^T \tilde{y}_k\|_2^2 \tag{9.4}$$

Next, we introduce new variables $G = L^T \tilde{G}$ and $y = L^T \tilde{y}$ that decouple each row of G and y from the others:

$$\underset{G}{\text{minimize}} \quad \sum_{k=1}^{m} \|G w_k - y_k\|_2^2, \tag{9.5}$$

where the solution to (9.4) can be recovered from the solution to (9.5) by simply transforming our matrix back with $\tilde{G} = L^{-T} G$. The inverse L^{-T} is computed once offline for a specific sensor and stored.

Next, we add regularization to the problem to improve conditioning and ensure that values in G are of reasonable magnitudes. To do this, a quadratic regularizer is added in the following manner:

$$\underset{G}{\text{minimize}} \quad \sum_{k=1}^{m} \|G w_k - y_k\|_2^2 + \|G b\|_2^2, \tag{9.6}$$

where $b \in \mathbf{R}^{n_w}$ contains the regularization weights. This allows for (9.6) to be decoupled into a set of n_y problems where each row of G is solved for entirely independently. The resulting optimization problem over the j^{th} row of G, and y can now be formulated:

$$\underset{g^{(j)}}{\text{minimize}} \quad \sum_{k=1}^{m} \|(g^{(j)})^T w_k - y_k^{(j)}\|_2^2 + (b^T g^{(j)})^2, \tag{9.7}$$

where the $g^{(j)}$ corresponds to the j^{th} row of G. This allows us to solve n_y parallel instances of (9.7) with n_w decision variables each, instead of one large problem with $n_y \cdot n_w$ variables. It is important to note that there are no approximations made between (9.6) and (9.7), and that they will always recover the same solution.

9.4 Linear Model Learning with a Kalman Filter

The problem posed in (9.7) is an unconstrained convex optimization problem whose solution can be computed by solving a single linear system, with a solution complexity that is

cubic in the length of the feature vector, n_w. In scenarios where online, real-time system identification is desired, a recursive method can be used that updates the solution for each new sensor measurement. One option for solving (9.7) in a recursive fashion would be to use a standard Recursive Least Squares (RLS) algorithm. While this method can work, it has three major downsides: Long trajectories present numerical instability, adaptively updating the regularizer is expensive and approximate, and most importantly, there is no way to directly reason about process noise on the parameters [98, 99, 16].

To tackle these issues, we formulate our model-learning problem as a dynamical system and use a Kalman Filter to estimate each row of G independently. First, the connection between the canonical Kalman Filter and its corresponding optimization problem must be made [87]. For a canonical dynamical system with a state $x \in \mathbf{R}^{n_x}$ and measurement $z \in \mathbf{R}^{n_z}$, the dynamics and measurement functions are described by,

$$x_{k+1} = Ax_k + q_k, \qquad q \sim \mathcal{N}(0, V), \tag{9.8}$$

$$z_k = Cx_k + r_k, \qquad r \sim \mathcal{N}(0, W), \tag{9.9}$$

where q and r are the (unknown) process and sensor noises, and x is the (unknown) state being estimated. The Kalman Filter is then initialized with a Gaussian belief over the initial condition, $x_0 \sim \mathcal{N}(\mu_{ic}, \Sigma_{ic})$. Since the initial belief, the process noise, and the sensor noise are all assumed to be Gaussian, the maximum a posteriori (MAP) problem takes the form of an unconstrained convex quadratic program:

$$\underset{\mu_{0:m}}{\text{minimize}} \quad \|\mu_{ic} - \mu_0\|^2_{\Sigma_{ic}^{-1}} + \sum_{k=0}^{m-1} \|\mu_{k+1} - A\mu_k\|^2_{Q^{-1}} + \|z_{k+1} - C\mu_{k+1}\|^2_{R^{-1}}, \tag{9.10}$$

where the trajectory $\mu_{0:m}$ that minimizes the covariance-weighted errors in the dynamics, the sensor, and the initial belief is solved for. This is the optimization problem that the Kalman Filter solves (to global optimality) in a recursive fashion.

9.4.1 Parallelizing a Kalman Filter over Sensor Dimensions

We now convert our decoupled optimization problems from (9.7) into a form such that a Kalman Filter can be used to solve for the estimates of each of the n_y rows of G in parallel. From here, a common structure amongst these n_y filters allows us to handle the estimated G directly in a single, parallelizable LML algorithm in which the most expensive operations are matrix multiplications.

To start, the estimate of each row of G has its own Gaussian belief. Each of these n_y beliefs are represented with a mean estimate $\hat{g}^{(j)}$ and covariance $\Sigma^{(j)}$. From here, the discrete-time dynamics for the filter are:

$$g^{(j)}_{k+1} = g^{(j)}_k + q^{(j)}_k, \qquad q^{(j)} \sim \mathcal{N}(0, Q), \tag{9.11}$$

where $q^{(j)} \in \mathbf{R}^{n_w}$ is the (unknown) additive Gaussian process noise, with a covariance of $Q \in \mathbf{S}^{n_w}_+$. This can also be interpreted as a more expressive version of a "forgetting factor", where a non-zero process-noise covariance quantifies the expected change in parameters

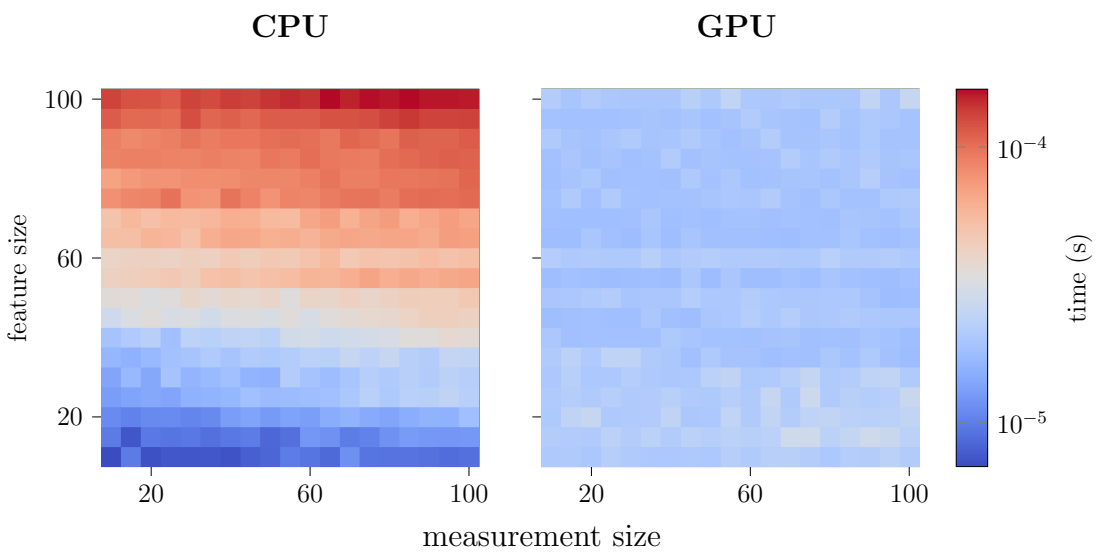


Figure 9.2: A comparison between run times of the LML algorithm on a CPU and an NVIDIA V100 GPU. The LML algorithm is comprised entirely of highly parallelizable operations (no matrix inversions), so the GPU is able to run the algorithm in constant time as the number of parameters in the model grows by a factor of 100.

over the course of the experiment, therefore putting more emphasis on newer measurements than old.

The measurement function in the filter for row j is where we use the row parameters $g^{(j)}$ with our feature vector w to predict the j^{th} measurement:

$$y_k^{(j)} = w_k^T g_k^{(j)} + v_k, \qquad v \sim \mathcal{N}(0, 1.0), \tag{9.12}$$

where $v \in \mathbf{R}$ is the (unknown) sensor noise with a variance of one, a result that comes from our change of variables in (9.5). Importantly, the measurement in (9.12) is a scalar, which means the computation of the Kalman Gain only requires the inversion of a scalar instead of a matrix:

$$\ell = (\Sigma w)(w^T \Sigma w + 1)^{-1} = \frac{\Sigma w}{w^T \Sigma w + 1}, \tag{9.13}$$

eliminating the only matrix inversion present in the Kalman Filter.

The dynamics and measurement equations in (9.11-9.12) are in the canonical form shown in (9.8-9.9), where $A = I$ and $C = w_k^T$. Critically missing from these matrices is any reference to row j, meaning the filters for all n_y rows have the same process and measurement functions. This enables two important commonalities amongst the n_y filters: They all share the same covariance and Kalman Gain. Therefore, we only need to maintain a single "global" covariance Σ that represents each of the n_y rows, and use the shared Kalman Gain, $\ell \in \mathbf{R}^{n_y}$ to update everything.

Using the Kalman Gain, the updates to the mean estimate of each row are:

$$\hat{g}_{k+1|k+1}^{(j)} = \hat{g}_{k+1|k}^{(j)} + \ell(y_{k+1}^{(j)} - \hat{g}_{k+1|k}^{(j)} w_{k+1}) \qquad \forall j \in [1,\ n_y], \tag{9.14}$$

Algorithm 13 Linear Model Learning

input $\hat{G}_{k|k}, \Sigma_{k|k}, w_{k+1}, y_{k+1}$ ▷ belief at time t
`/* predict belief updates */`
$\hat{G}_{k+1|k} \leftarrow \hat{G}_{k|k}$ ▷ no change to mean
$\Sigma_{k+1|k} \leftarrow \Sigma_{k|k} + Q$ ▷ covariance propagation
`/* innovation and Kalman Gain */`
$z \leftarrow y_{k+1} - \hat{G}_{k+1|k} w_{k+1}$ ▷ innovation
$s \leftarrow w_{k+1}^T \Sigma_{k+1|k} w_{k+1} + 1$ ▷ vector of innovation variances
$\ell \leftarrow \Sigma_{k+1|i} w_{k+1} / s$ ▷ Kalman Gain via element-wise division
`/* update mean and covariance */`
$\hat{G}_{k+1|k+1} \leftarrow \hat{G}_{k+1|k} + z\ell^T$ ▷ parallelized mean update with outer product
$\Sigma_{k+1|k+1} \leftarrow (I - \ell w_{k+1}^T)\Sigma_{k+1|k}(I - \ell w_{k+1}^T) + \ell\ell^T$ ▷ Joseph form covariance update
return $\hat{G}_{k+1|k+1}, \Sigma_{k+1|k+1}$ ▷ belief at time $t+1$

Algorithm 14 Adaptive Regularization of Belief State

input $\hat{G}, \Sigma, \rho$ ▷ belief and regularizer
for $i = 1$ **to** n_w **do** ▷ this loop assumes 1-based indexing
    $c \leftarrow 0_{n_w}$
    $c[i] \leftarrow 1$
    $r \leftarrow (1/\rho[i])^2$
    $s \leftarrow \Sigma[i,i] + r$
    $\ell \leftarrow \Sigma[:,i]/s$
    $\Sigma \leftarrow (I - \ell c^T)\Sigma(I - \ell c^T) + r\ell\ell^T$
    $\hat{G} \leftarrow \hat{G} - \hat{G}[:,i]\ell^T$
end for
return $\hat{G}, \Sigma$

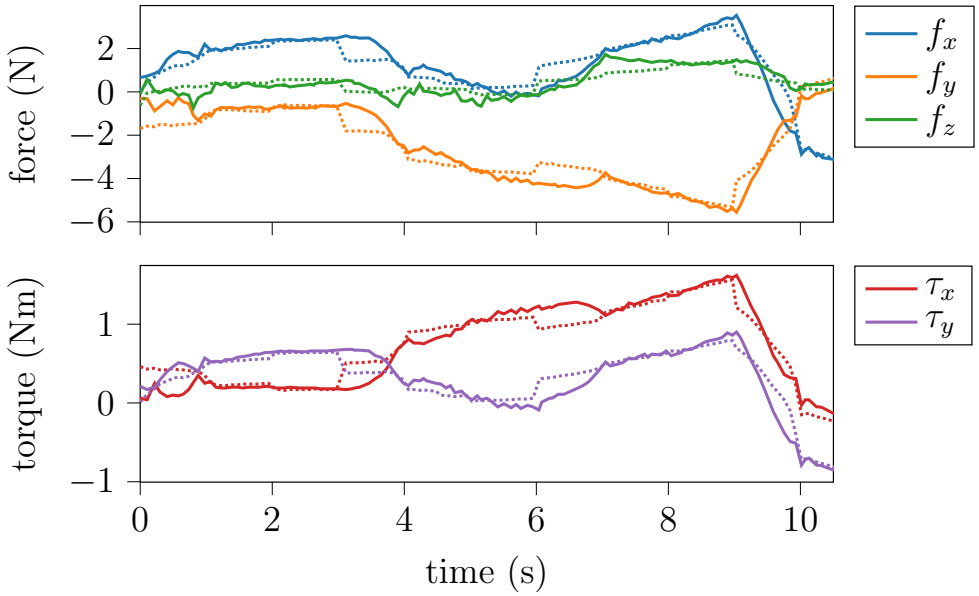


Figure 9.3: The sim-to-real gap for a connector insertion calibration sequence, where the real force torque sensor measurements are solid and the predicted measurements from our learned model are dotted. The learned model is able to accurately capture the contact force behavior of the connector inside the socket.

and since the Kalman Gain is shared, these individual row updates can be broadcast to an estimate of the full matrix $\hat{G}$:

$$\hat{G}_{k+1|k+1} = \hat{G}_{k+1|k} + (y_{k+1} - \hat{G}_{k+1|k} w_{k+1})\ell^T, \tag{9.15}$$

where the parallelism is simply handled by a vector outer product and matrix addition. Using this technique on the parallel filters results in the LML algorithm, which is detailed in Algorithm 13.

Lastly, the regularization term from (9.7) must be included in the LML algorithm as an initialization. To do this, the cost terms associated with the initial Gaussian belief in the MAP problem (9.10) are adapted to replicate the regularization in (9.7) in the form of an initial Gaussian belief over row j:

$$\hat{g}_{0|0}^{(j)} = 0_{n_w}, \qquad \Sigma_{0|0} = \text{diag}(1/b^2). \tag{9.16}$$

With this initialization, the LML algorithm is called recursively every time a new measurement is available. The entire algorithm is matrix-inversion free, requiring only simple matrix products and additions that are trivial to parallelize on a GPU. To demonstrate the importance of this, timing results of the LML algorithm run on a GPU and CPU are shown in Fig. 9.2, where the run time on a GPU stays constant as the number of parameters in $\hat{G}$ grows by a factor of 100.

9.4.2 Adaptive Regularization

In the original cost function of (9.3), the relative importance of the regularization term is dependent on the trajectory length. As the trajectory gets longer, the impact the regularization has on the estimate decreases. In cases where the LML algorithm is being run for an unknown amount of time, choosing the appropriate regularizer offline is difficult. To deal with this, we present a method for updating the regularization term online that can handle arbitrary quadratic regularization and does not require any matrix inversions.

We specify this additional regularizer in a general form $\|G\rho\|_2^2$ that will be included in the cost function of (9.3). To add this to the LML algorithm, the connection between the MAP problem in (9.10) and the Kalman Filter is used once again. As before, we formulate a fictitious measurement equation for each $\hat{g}^{(j)}$ such that we recover the quadratic regularizer in the MAP cost, then use the same parallelization strategy to generalize it to $\hat{G}$.

Our estimated mean $\hat{g}^{(j)}$ will have stationary dynamics, but we use a measurement function where each row has the values of the row j returned,

$$z_k = \hat{g}_k^{(j)} + r_k. \tag{9.17}$$

By expecting this "measurement" to be zero, we use our regularizer as a covariance for the fictitious sensor, $r \sim \mathcal{N}(0, \text{diag}(1/\rho^2))$, reconstructing our desired regularization cost in the MAP problem. Normally the Kalman Filter requires a matrix the size of the measurement to be inverted, however, since this fictitious sensor noise is diagonal, we can use the method of sequential scalar measurement updates to once again avoid any matrix inversions [86]. Similar to the LML algorithm, these operations can be broadcast to each row of G using matrix multiplication, as shown in Algorithm 14.

9.5 Experiments

We demonstrate LML algorithm on a contact-rich connector-insertion task, where the resulting contact-force model is used in a convex-optimization control policy. This approach is first shown to work in simulation with MuJoCo, then on hardware with a Franka robot arm with a parallel gripper holding a standard C13 power plug.

To learn the relationship between the configuration, control, and contact wrench on the plug, the LML algorithm will be used with the following feature vector:

$$w = \text{vcat}(r,\, \text{vec}(R),\, r_{des},\, \phi,\, 1), \tag{9.18}$$

where $r \in \mathbf{R}^3$ is the cartesian position of the end effector, $R \in \mathbf{SO(3)}$ is the rotation matrix describing the attitude of the end effector, $r_{des} \in \mathbf{R}^3$ is the desired position sent to an impedance controller, $\phi \in \mathbf{R}^3$ is the axis-angle rotation between current attitude and desired attitude, and the 1 is included to account for a constant bias. The control inputs in this feature vector are r_{des} and ϕ, where the axis-angle parameterization is used to avoid any attitude-related constraints [83].

The LML algorithm learns a linear relationship between this feature vector and the force torque sensor, $y \approx Gw$. To do this, a calibration sequence is run where both the position

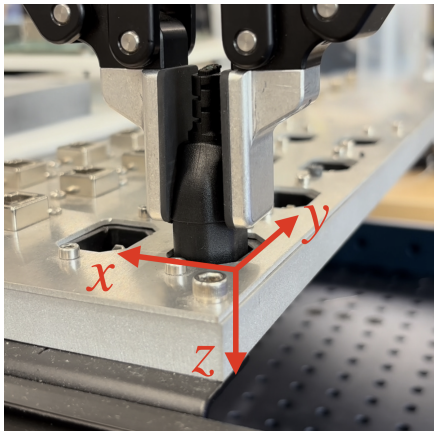


Figure 9.4: Calibration and insertion of a power plug into a socket with a Franka robot arm and parallel gripper.

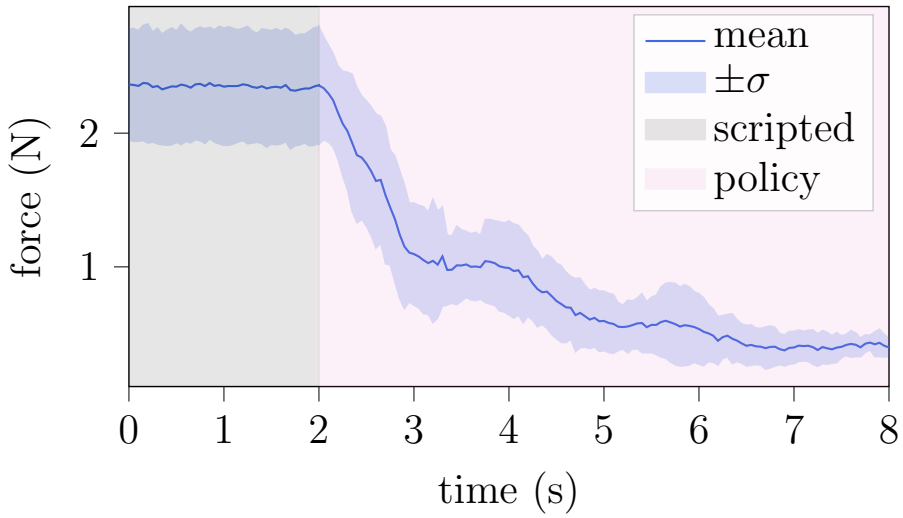


Figure 9.5: Magnitude of the force on the plug in the XY axis during insertion of eight different plugs. The scripted insertion leaves a constant force on the plug, after which the policy is used to reduce the force on the connector.

and attitude are varied. The force-torque sensor measurements and feature vectors are input to the LML algorithm during this calibration, and the resulting estimated sensor measurements are able to closely align with the true sensor measurements as shown in Fig. 9.3. To align the connector for insertion, a model-based controller uses this linear relationship to solve for controls that minimize the resistance measured by the force-torque sensor. The controller is posed as the solution to the following optimization problem:

$$\begin{array}{ll} \underset{r_{des},\phi}{\text{minimize}} & \|S\hat{y}\|_2^2 + \lambda\|r - r_{des}\|_2^2 + \mu\|\phi\|_2^2 \\ \text{subject to} & \hat{y} = G[r^T,\, \text{vec}(R)^T,\, r_{des}^T,\, \phi^T,\, 1]^T, \\ & \|r - r_{des}\|_\infty \leq 1 \text{ cm}, \\ & \|\phi\|_\infty \leq 3^\circ, \end{array} \tag{9.19}$$

where $S = \text{diag}([1, 1, 0, 1, 1, 1])$ is used to penalize all forces and torques except for force in the Z (plug insertion) direction, $\lambda \in \mathbf{R}^+$ and $\mu \in \mathbf{R}^+$ are regularization weights, and the primal constraints ensure the commanded end-effector pose doesn't deviate too far from the current pose. Due to the linearity of our model, this problem is a small convex quadratic program, for which there are many available solvers for fast online execution [115]. We have found that a solver is not necessary since the controls can be sub-optimally clipped in the rare case the constraints are violated.

This system-identification and control strategy was first demonstrated in MuJoCo, as seen in Fig. 9.1, where the controller was successfully able to align the plug with initial misalignments. Second, a Franka robot arm with a parallel gripper was able to insert the C13 power plug into eight different sockets in the presence of misalignment, sensor noise, and actuator uncertainty. In both of these scenarios, LML was run on the force-torque sensor data during a quick 10-second calibration sequence, after which the controller was turned on and smooth insertion was achieved. On the Franka, the forces on the plug during these eight alignments are shown in Fig. 9.5, where the forces from the initial misalignment are dramatically reduced by over 80% once the controller takes over.

9.6 Conclusion

This work proposes a new approach for the modeling and control of connector-insertion tasks, where an estimated quasi-static contact model is used to predict the contact forces on the connector as a function of the configuration and control. This contact force model is learned online to global optimality with a novel Linear Model Learning algorithm that is free of matrix inversions. The runtime of this algorithm is shown to be constant on a GPU as the model size increases, since it is entirely comprised of parallelizable operations. The effectiveness of this modeling strategy is demonstrated both in simulation and in hardware experiments, where the contact-force model is used to solve for an optimal alignment policy for C13 power connectors.

Chapter 10

Convex Quasi-Dynamic Simulation of Rigid Point Clouds with Torsional Friction

Many common manipulation tasks move slowly enough that velocities and Coriolis forces do not contribute meaningfully to the dynamics of the robot. As a result, quasi-dynamic simulation of these systems, in which all forces are assumed to be in equilibrium, can produce physically accurate results for use in motion-planning and control. Recent work has combined the quasi-dynamic model with a relaxed formulation of Coulomb friction, where the resulting dynamics are the solution to a convex quadratic program. We extend this recent work by directly manipulating rigid point clouds, without the need for meshing or decomposition into convex primitives. We also introduce a novel torsional friction model to mimic the frictional behaviors of the sorts of patch contacts that exist on real systems. These ideas are demonstrated in a grasping example, where a dense point cloud is manipulated with and without torsional friction, clearly showing the utility of the torsional friction model.

The contents of this chapter have been previously published at the ICRA Leveraging Models in Contact-Rich Manipulation Workshop in Tracy and Manchester [172]

10.1 Introduction

Accurate and computationally efficient simulation for contact-rich robotic manipulation tasks is crucial for developing model-based control policies. Existing simulators like Bullet [37], Drake [163], Dart [96], and MuJoCo [166], are able to model complex contact and friction interactions between a wide variety of geometries. While these simulators evaluate the full second-order dynamics of a robot, for many manipulation tasks, these second-order dynamics are not always necessary, and substantial computational savings are possible.

For manipulation systems where objects are generally moving slowly, the velocities of the bodies are small enough that accelerations and Coriolis forces do not impact the dynamics in a meaningful way. As a result, quasi-dynamic [114] simulation methods, in which all forces are assumed to be in equilibrium, can be employed to solve for displacements in

the configurations at each time step.

This work is inspired by [134], where a relaxed friction model from [10] is incorporated into a quasi-dynamic simulation method that treats actuators as impedances. Each simulation step consists of solving a small, well-defined convex Quadratic Program (QP), making the method fast and efficient, with virtually none of the stability problems encountered with full second-order dynamics formulations. This quasi-dynamic model was used successfully in a global planner in [132] where it was leveraged to build sophisticated motion plans for common manipulation tasks. In both of these works, only relatively simple shapes like spheres, capsules, and boxes were considered, due to the complications involved in collision checking.

In this paper, we make two contributions to the state of the art for quasi-dynamic simulation: the first is a computationally efficient framework for computing the dynamics of rigid objects whose complex geometry is represented by point clouds, and the second is a convex torsional friction model to effectively model patch contacts and avoid some of the non-physical behaviors seen when manipulating a rigid point cloud with rigid manipulators. Since the new torsional friction model is also represented as a convex constraint, each simulation step can still be computed efficiently by solving a QP. Together, these contributions enable fast and robust simulation of manipulation tasks involving complex non-convex geometries that would otherwise be challenging to decompose into simpler primitives.

10.2 Convex Quasi-Dynamic Models

This section describes the general quasi-dynamic formulation with actuators modeled by impedances, closely following [134, 133]. The configuration of the whole system, $q \in \mathbf{R}^{n_u+n_a}$, can be partitioned into an actuated part, $q_a \in \mathbf{R}^{n_a}$, and an un-actuated part $q_u \in \mathbf{R}^{n_u}$, stacked as $q = [q_a^T, q_u^T]^T$. In a quasi-dynamic model, there is no accumulation of velocity between time steps. As a result, a new velocity or a configuration displacement is computed at each time step.

The actuated degrees of freedom are treated as if they were controlled with impedance control. This means that we have some diagonal joint stiffness matrix $K_q \in \mathbb{S}^{n_a \times n_a}$, with the generalized force τ_k modeled as:

$$K_q \delta \bar{q}_a = \tau_k, \tag{10.1}$$

where $\delta \bar{q}_a$ is the desired change in actuated configuration. The un-actuated degrees of freedom are simply acted upon by an external generalized force τ_a, and contact forces.

10.2.1 Contact and Friction Model

Rigid contact and Coulomb friction can be incorporated into a quasi-dynamic model in the same fashion as a second-order dynamic model. First, collision information between pairs of convex objects can be computed with any number of methods, given that they return the closest points between objects as well as a contact normal vector [60, 151, 131, 169].

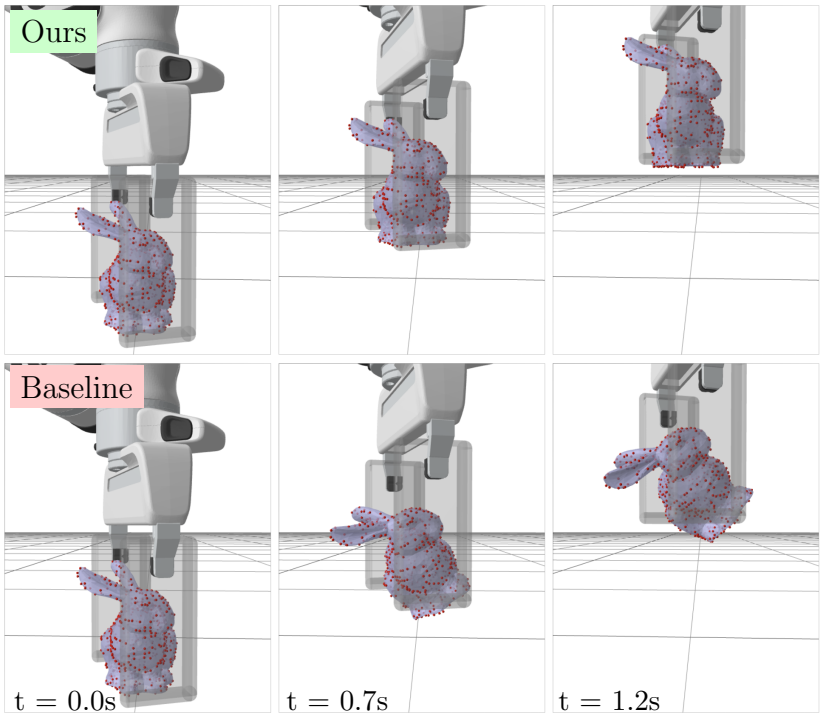


Figure 10.1: Quasi-dynamic simulation of grasping a point cloud with and without torsional friction on the points. In the bottom sequence of frames, a simulation without torsional friction results in non-physical rotation of the bunny about single-point contacts with each gripper. Our method (top) includes torsional Coulomb friction at each contact point to avoid this behavior.

As shown in figure 10.2, these closest points are denoted α and β, with the signed-distance function (SDF) between these denoted by ϕ. Each of the n_c pairwise contact interactions has its own α, β, normal vector pointing from α to β, and a set of orthogonal tangent vectors d_1 and d_2. Together, the set $\{d_1, d_2, n\}$ describe a dextral triad of orthonormal vectors [120, 89].

In [10], Anitescu introduces a convex relaxation of tangential Coulomb friction that is exact when objects are sticking, and only introduces a slight "boundary layer" when objects are sliding. This relaxation allows for the inclusion of friction as a linear inequality in a quadratic program, enabling convex time-stepping methods [133]. In this work, we extend Anitescu's friction model to handle torsional friction about the contact normal. This section omits indices that specify which contact pair is being considered for clarity, but the method extends to an arbitrary number of contacts.

A Jacobian mapping forces $\lambda_i \in \mathbf{R}^3$ at the contact point into generalized coordinates is calculated by taking the Jacobian of the following function with respect to the generalized velocity $v \in \mathbf{R}^{n_v}$:

$$f(q, v) = \begin{bmatrix} \nu_\beta - \nu_\alpha \\ n^T(\omega_\beta - \omega_\alpha) \end{bmatrix}, \tag{10.2}$$

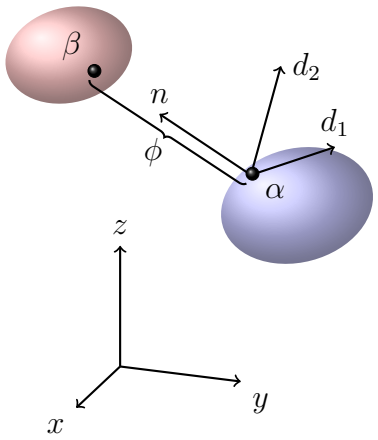


Figure 10.2: Description of contact geometry for a given signed distance function $\phi = \|\beta - \alpha\|$, where α and β are the closest points between two convex shapes. The unit contact normal vector n extends from α to β, with a set of orthogonal tangent-plane vectors d_1 and d_2.

where the relative velocity of the contact points is calculated by taking the difference between the velocity of β ($\nu_\beta \in \mathbf{R}^3$) and α ($\nu_\alpha \in \mathbf{R}^3$). This Jacobian will be referred to as J:

$$J = \frac{\partial f(q, v)}{\partial v}. \tag{10.3}$$

where each vector in (10.2) is resolved in the world frame. With this Jacobian, we can now write our friction constraints for a given point as the following:

$$\phi + [J^T(\tilde{n} + \tilde{d}_i)]^T \delta q \geq 0 \quad \forall i \tag{10.4}$$

where $\tilde{n} = [n^T, 0]^T$, and the the first four $\tilde{d}$'s describing the standard pyramidal tangential friction-cone approximation are the following:

$$\tilde{d}_1 = \begin{bmatrix} \mu_v d_1 \\ 0 \end{bmatrix}, \qquad \tilde{d}_2 = \begin{bmatrix} \mu_v d_2 \\ 0 \end{bmatrix}, \tag{10.5}$$

$$\tilde{d}_3 = \begin{bmatrix} -\mu_v d_1 \\ 0 \end{bmatrix}, \qquad \tilde{d}_4 = \begin{bmatrix} -\mu_v d_2 \\ 0 \end{bmatrix}. \tag{10.6}$$

We also introduce two more $\tilde{d}$'s describing torsional friction:

$$\tilde{d}_5 = \begin{bmatrix} 0_3 \\ \mu_\omega \end{bmatrix}, \qquad \tilde{d}_6 = \begin{bmatrix} 0_3 \\ -\mu_\omega \end{bmatrix}, \tag{10.7}$$

where $\mu_v \in \mathbf{R}_+$ is the tangential coefficient of friction and $\mu_\omega \in \mathbf{R}_+$ is the torsional coefficient of friction. These six linear inequalities describe the contact and friction for a given

pair and can be vertically concatenated into a single $A\delta q \geq b$ constraint for convenience:

$$A = \begin{bmatrix} J^T(\tilde{n} + \tilde{d}_1) \\ J^T(\tilde{n} + \tilde{d}_1) \\ \vdots \\ J^T(\tilde{n} + \tilde{d}_6) \end{bmatrix}, \qquad b = -\phi 1_6. \tag{10.8}$$

10.2.2 Optimization Formulation

The force balance for the actuated and un-actuated components of the configuration with contact forces can be posed as,

$$\begin{bmatrix} hK_q\delta q_a - hK_q\delta\bar{q}_a + \tau_a \\ \frac{1}{h}M_u\delta q_u - h\tau_u \end{bmatrix} + \sum_n^{n_c} A_j^T\lambda_j = 0, \tag{10.9}$$

where the contact/friction model is described as follows:

$$A_j\delta q \geq b_j, \tag{10.10}$$
$$\lambda_j \geq 0, \tag{10.11}$$
$$(A_j\delta q - b_j) \circ \lambda = 0. \tag{10.12}$$

These conditions are the KKT optimality conditions for a convex Quadratic Program (QP) where (10.9) is stationarity, (10.10) is primal feasibility, (10.11) is dual feasibility, and (10.12) is complementarity [24]. This QP is as follows:

$$\begin{aligned} \min_{\delta q} \quad & \frac{1}{2}\delta q^T \begin{bmatrix} hK_q & 0 \\ 0 & M_u/h \end{bmatrix} \delta q - h \begin{bmatrix} K_q\delta\bar{q}_a + \tau_a \\ \tau_u \end{bmatrix}^T \delta q \\ \text{s.t.} \quad & A_j\delta q \geq b_j, \qquad \forall j \in \mathcal{E} \end{aligned} \tag{10.13}$$

where $\mathcal{E}$ is the set of contact pairs being considered and $h \in \mathbf{R}_+$ is the time-step size. For large point clouds, this pruning of constraints is critical for keeping the resulting QP small. Another benefit to the QP formulation is differentiability: In recent years, differentiable convex optimization has enabled smooth automatic differentiation through QP solvers with good results [177, 78, 8, 156].

10.3 Numerical Examples

Two examples are shown to demonstrate the behavior and utility of our formulation: The first example is a simple spinning sphere that showcases the tightness of the relaxed friction constraints, and the second is a more practical example of gripping a dense point cloud with and without torsional friction.

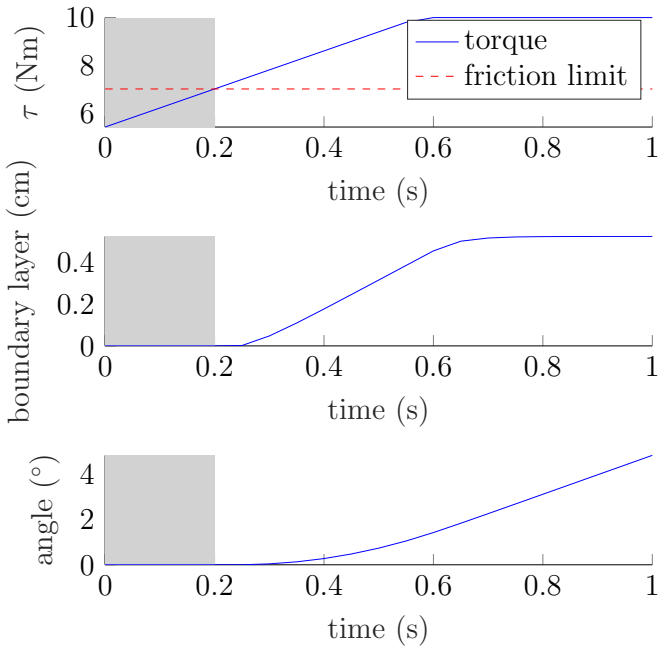


Figure 10.3: Visualization of the boundary layer that exists only when a point in contact begins rotating due to an external torque τ. In the exact same fashion as the tangential friction constraints, a boundary layer is introduced during sliding that is proportional to the time step h.

10.3.1 Torsional Boundary Layer

To demonstrate the boundary layer induced by the relaxed torsional friction constraints, figure 10.3 demonstrates what happens when a sphere in contact with the floor is spun. An increasing external torque is applied to the sphere, and up until the maximum allowable friction torque is saturated, the sphere does not move. Once the external torque exceeds this limit, the sphere begins to spin and lifts off from the surface by a fraction of a centimeter.

This boundary-layer behavior mirrors the tangential friction case, where this is the only deviation from Coulomb friction and the size of this boundary layer decreases with step size. For most manipulation tasks that aren't pushing or sliding, full sticking behavior is desirable, making this relaxed model appropriate.

10.3.2 Grasping a Point Cloud

In this example, shown in Fig. 10.1, parallel grippers are used to grasp a point-cloud bunny with and without torsional friction. The bunny was modeled as a rigid body with 992 points, each of which was constrained to consider the floor, the left gripper, and the right gripper. The simulator checks the signed distance function for each of these 992 points with respect to the three objects, and selects five points for each object to include in the quadratic program. Each of the selected points contributes the constraints shown

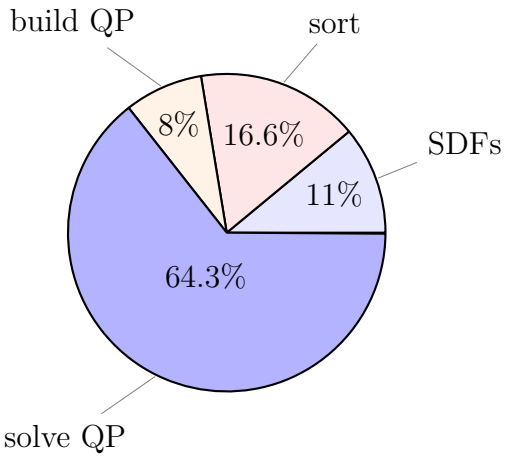


Figure 10.4: Average timing results for a step in the quasi-dynamic simulator with the bunny where the total time was <90 μs for a time step size of $h = 0.01s$. A majority of the time is spent solving the QP, despite the fact there are almost 3,000 signed distance functions evaluations.

in (10.8) to the optimization.

While these five closest points in question can change from time step to time step, the size of the QP remains the same. The resulting QP for this example has 13 primal variables with 80 inequality constraints, and is solved with a custom primal-dual interior-point algorithm [116, 130, 183] that leverages templating based on the number of contacts considered during a simulation.

When the grippers close on the bunny, it is possible for there to be single points of contact with each gripper. In this case, the lack of torsional friction results in the bunny rotating about these contact points. While this is technically correct behavior given point contacts, the true system does not exhibit this behavior due to patch-contact effects, resulting in a significant sim-to-real gap. On a real robotic system, there is always some compliance either on the gripper or the object that manifests in a patch contact instead of a point contact. The patch contact provides a lever arm for torsional friction, preventing the bunny from pivoting. Just like with tangential friction, the torsional friction is proportional to the normal force on the contact.

By including torsional friction for each point contact, we are able to replicate torsional friction behaviors while still treating the object as a rigid point cloud.

The full simulation step takes less than 90 μs for a time step size of $h = 0.01s$., with the profile of this function shown in Fig. 10.4.[1] Since a majority of the time is spent solving the QP, there is reason to believe this sort of simulation makes more sense on a CPU than a GPU, despite the possibility of a very dense point cloud. There may be cases where the point cloud is dense enough that computing and sorting all of the pairwise contacts is the limiting computational factor, in which case a GPU implementation would make more sense.

[1]Timing results are reflective of a Mac M1 Pro, running Julia 1.9.2

Chapter 11

The Trajectory Bundle Method

We present a derivative-free trajectory optimization technique that approximates functions with linear interpolation between samples instead of using a Taylor series. The resulting algorithm, the trajectory bundle method, is able to exploit accelerators to compute parallelized samples of the cost, dynamics, and constraint functions to form the linear interpolants. These so-called bundles are then used in a convex optimization problem that is solved for the next iterates within a trust region. This process is repeated until tight constraint satisfaction is achieved, and is a significant improvement over existing derivative-free optimization techniques when applied to trajectory optimization problems. The efficacy of this method is demonstrated on a variety of robotics platforms.

11.1 Introduction

Linear dynamical systems of the form $x_+ = Ax + Bu$ underpin many of the foundational methods in modern optimal control. Ideas like the Linear Quadratic Regulator (LQR) and convex trajectory optimization are able to reason about dynamical systems of this form in a way that is exact and globally optimal [87, 23, 24]. As a result, these techniques are often applied to nonlinear systems where a locally approximate linear dynamical system is used instead of the nonlinear model in a process called linearization [150]. In many cases, this approximation is appropriate given this approximation is not used too far outside of the linearization point where the approximation was formed. By being careful about where these approximations are initialized and how far from their linearization point they are being used, this method of linearizing nonlinear systems can be extremely effective in practice, even for highly nonlinear systems. The two caveats here are that the nonlinear system must be both smooth and differentiable.

For many systems such as robotic arms, quadrotors, and vehicles, this assumption of smooth differentiability is appropriate. For rigid-body dynamics, there are specialized methods for computing derivatives through the continuous-time dynamics in a fast and efficient way [50]. However, there are also scenarios where these derivatives are either unavailable, prohibitively expensive to compute, or are unreliable. If the dynamics model

is learned from data, the approximation of the dynamics function may be good while the approximation of the derivatives may be very poor. This scenario is often explored in the context of model-predictive path-integral control, where a learned simulator is only used to produce parallelized rollouts [195, 188]. Another scenario in which derivatives are unavailable or unusable is in the presence of systems that make or break contact. While there has been a lot of recent interest in making simulations with contact differentiable [51, 128, 132, 170, 155, 78], there remains a strong need for optimal control methods that do not rely on these derivatives at all.

A prevalent trend in robotic simulation is the introduction of simulators that can be run on accelerators for massively parallel simulation. Popular simulators like Isaac Sim [105, 121], Brax [51], and MuJoCo XLA (MJX) [165], are all capable of running thousands of simulations in parallel. This paper looks to leverage the innovation of parallel simulation to motivate a new optimal control paradigm where derivative-free simulations are used to describe the dynamics and cost landscapes present in the problem. To this effect, we present the trajectory bundle method for solving nonconvex trajectory optimization problems in an entirely derivative-free manner. This method uses interpolated short simulations instead of derivative-based linearization to approximate the cost, dynamics, and constraint functions in the problem. The result is a simple and efficient trajectory optimization solver that can fully utilize parallelized simulation without requiring any derivatives.

Derivative-Free Optimization (DFO) is a well-studied technique for solving optimization problems where derivatives are unavailable for whatever reason. John Dennis gives an appropriate description of motivation for DFO problem in [140] as to "find the deepest point of a muddy lake, given a boat and a plumb line, when there is a price to be paid for each sounding". With modern accelerators capable of massively parallel dynamics, cost, and constraint evaluation, there is still a price to take soundings, but no longer a price *for each* sounding. In a seminal 1965 paper, the Nelder-Mead method for derivative-free function minimization was proposed where a simplex of sample points is used to approximate the cost landscape instead of derivatives. Methods like Mesh Adaptive Direct Search (MADS) [11] and NOMAD [95, 12] followed, with improvements to the Nelder-Mead method.

Although there has been much work on general DFO [35, 93, 36], there are two notably similar approaches to the trajectory bundle method. The first is the Constrained Optimization by Linear Interpolation (COBYLA) solver, where the cost and constraint functions are approximated with linear interpolation [140], and [32], where a residual cost function structure is assumed and samples are used for linear interpolation. The trajectory bundle method builds on these two methods and specializes to trajectory optimization problems where the decision variables are only coupled via the dynamics constraints. By exploiting this problem-specific structure and massively parallelizable simulation, the trajectory bundle method is simple and robust method for solving trajectory optimization problems to tight constraint and optimality tolerances.

11.2 Background

Like many nonlinear optimization algorithms, the trajectory bundle method handles nonlinear cost and constraint functions by approximating them locally with affine functions. In this section, the standard method for approximating functions as affine with a Taylor series is detailed, followed by a derivative-free method using linear interpolation over sampled points. Using these linear interpolants, the process for locally approximating a generic constrained optimization problem as convex is shown.

11.2.1 Affine Function Approximation

An arbitrary function $p : \mathbf{R}^a \to \mathbf{R}^b$ is only affine if it can be represented in the following form: $p_{\text{aff}}(y) = d + Cy$, where $d \in \mathbf{R}^b$ and $C \in \mathbf{R}^{b \times a}$. The process of locally approximating a nonlinear function with an affine function around a point $\bar{y}$ is often referred to as linearization, with $\bar{y}$ denoted as the linearization point. In this section, the standard method of approximation by first-order Taylor series is presented, followed by a derivative-free method involving linear interpolation of sample points.

Taylor Series

An affine approximation of this function $p(y) \approx \hat{p}(y)$ can be formed in the vicinity of an input value $\bar{y}$ through the use of the first-order Taylor series,

$$p(y) \approx \hat{p}(y) = p(\bar{y}) + \frac{\partial p}{\partial y}(y - \bar{y}), \tag{11.1}$$

where both the value and the Jacobian of p are calculated at the point $\bar{y}$. This approximation is exact at $\bar{y}$, and, generally speaking, becomes less accurate the further the deviation from $\bar{y}$. For affine functions, the first-order Taylor series simply recovers the original function and the approximation is exact.

Linear Interpolation

Alternatively, nonlinear functions can be approximated in a derivative-free manner by linearly interpolating between sampled function values. This is useful for when the derivatives of a function are unavailable, challenging to compute, or unreliable.

For an affine function, any linear interpolation between two inputs is equal to the linear interpolation of the outputs. This means for an interpolation parameter $\theta \in [0, 1]$ and two inputs y_1 and y_2, the following holds:

$$p_{\text{aff}}(\underbrace{\theta y_1 + (1-\theta) y_2}_{\text{interpolated inputs}}) = \underbrace{\theta p_{\text{aff}}(y_1) + (1-\theta) p_{\text{aff}}(y_2)}_{\text{interpolated outputs}}. \tag{11.2}$$

This concept can be extended to m points with an interpolation vector $\alpha \in \mathbf{R}^m$ that belongs to a standard simplex:

$$\alpha \in \Delta^{m-1} = \left\{ \alpha \in \mathbf{R}^m \;\middle|\; \sum_{i=1}^{m} \alpha_i = 1,\ \alpha \geq 0 \right\}, \tag{11.3}$$

where again the convex combination of these m inputs is equal to the same convex combination of the m outputs,

$$p_{\text{aff}}\left(\sum_{i=1}^{m} \alpha_i y_i \right) = \sum_{i=1}^{m} \alpha_i p_{\text{aff}}(y_i). \tag{11.4}$$

This means that we can locally approximate the original nonlinear function p in the neighborhood of $\bar{y}$ by sampling m points from a distribution centered around $\bar{y}$ with $y_i \sim \mathcal{D}(\bar{y})$, and constraining the inputs to this approximation to be a linear combination of the sample points. For notational convenience, the lists of inputs and outputs are horizontally concatenated as columns of the matrices

$$W_y = \begin{bmatrix} y_1 & y_2 & \cdots & y_m \end{bmatrix} \in \mathbf{R}^{n_y \times m}, \tag{11.5}$$

$$W_p = \begin{bmatrix} p(z_1) & p(z_2) & \cdots & p(z_m) \end{bmatrix} \in \mathbf{R}^{n_r \times m}, \tag{11.6}$$

enabling the affine approximation $\hat{p}$ to be summarized as the following:

$$y = W_y \alpha \qquad \text{linear interpolation of inputs} \tag{11.7}$$

$$\hat{p} = W_p \alpha \qquad \text{linear interpolation of outputs} \tag{11.8}$$

11.2.2 Approximation for Optimization

We will now consider a general nonlinear optimization problem and examine how these linearization techniques can be utilized to form a convex approximation of the original problem. This approach is used in sequential convex programming (SCP) methods where nonconvex optimization problems are solved by iteratively approximating the problem as convex in the neighborhood of the local iterate and solving for a step direction [61, 130, 106, 135].

To demonstrate how an SCP method would work with the two approximation techniques outlined, we examine a generic constrained optimization of the following form:

$$\begin{aligned} \underset{z}{\text{minimize}} \quad & \|r(z)\|_2^2 \\ \text{subject to} \quad & c(z) = 0, \end{aligned} \tag{11.9}$$

with a decision variable $z \in \mathbf{R}^{n_z}$, residual cost function $r : \mathbf{R}^{n_z} \to \mathbf{R}^{n_r}$, and constraint function $c : \mathbf{R}^{n_z} \to \mathbf{R}^{n_c}$. By approximating both of these functions as affine with a

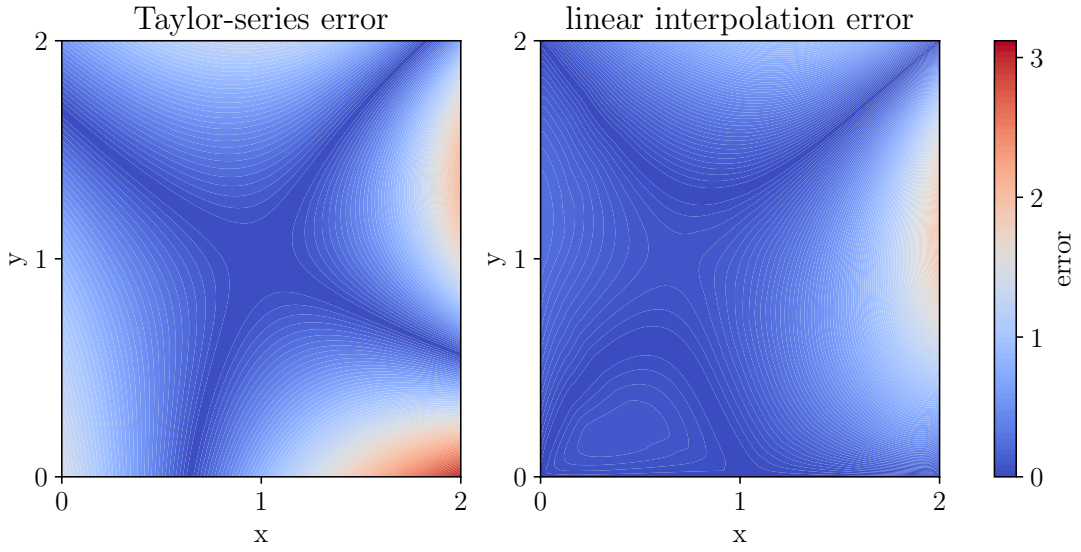


Figure 11.1: A comparison of the accuracy of a first-order Taylor series taken about $(x, y) = (1, 1)$ with linear interpolation of the four corner points on the function $f(x, y) = \sin(x)e^y$. While the errors are comparable between these two approximations, the patterns of these errors are notably different. This is because approximation by linear interpolation results in an affine model that can be different than the first-order Taylor series.

first-order Taylor series around a current iterate $\bar{z}$, we are left with the following convex optimization problem:

$$\begin{array}{ll} \underset{z}{\text{minimize}} & \|\hat{r}\|_2^2 \\ \text{subject to} & \hat{r} = r(\bar{z}) + \dfrac{\partial r}{\partial z}(z - \bar{z}), \\ & \hat{c} = c(\bar{z}) + \dfrac{\partial c}{\partial z}(z - \bar{z}) = 0 \end{array} \tag{11.10}$$

While this problem is convex and we are guaranteed to find a globally optimal solution if one exists, we do not have a guarantee of feasibility. There are circumstances in which the linearization of the constraint function results in infeasible problems [130]. In order to guarantee that this problem is always feasible, many sequential convex programming methods convert the constraint into a penalty and reformulate (11.10) with an always-feasible variant,

$$\begin{array}{ll} \underset{z}{\text{minimize}} & \|\hat{r}\|_2^2 + \phi(\hat{c}) \\ \text{subject to} & \hat{r} = r(\bar{z}) + \dfrac{\partial r}{\partial z}(z - \bar{z}), \\ & \hat{c} = c(\bar{z}) + \dfrac{\partial c}{\partial z}(z - \bar{z}), \end{array} \tag{11.11}$$

where $\phi : \mathbf{R}^{n_c} \to \mathbf{R}_+$ is a nonnegative penalty that discourages constraint violations. No matter the structure of the cost and constraint functions, the convex optimization problem in (11.11) is guaranteed to always have a solution.

Alternatively, a linear interpolant like that shown in (11.8) can be used to approximate the cost and constraint functions. To do this, m sample points centered around the current iterate $\bar{z}$ are used to evaluate the cost and constraint functions. These values are then horizontally concatenated into the following matrices:

$$W_z = \begin{bmatrix} z_1 & z_2 & \cdots & z_m \end{bmatrix} \in \mathbf{R}^{n_z \times m}, \tag{11.12}$$

$$W_r = \begin{bmatrix} r(z_1) & r(z_2) & \cdots & r(z_m) \end{bmatrix} \in \mathbf{R}^{n_r \times m}, \tag{11.13}$$

$$W_c = \begin{bmatrix} c(z_1) & c(z_2) & \cdots & c(z_m) \end{bmatrix} \in \mathbf{R}^{n_c \times m}. \tag{11.14}$$

The interpolation vector $\alpha \in \mathbf{R}^m$ is used to interpolate between these samples and their corresponding cost and constraint values. These approximations are used to recreate the relaxed problem shown in (11.11) as the following:

$$\begin{aligned} \underset{\alpha}{\text{minimize}} \quad & \|\hat{r}\|_2^2 + \phi(\hat{c}) \\ \text{subject to} \quad & \hat{r} = W_r \alpha, \\ & \hat{c} = W_c \alpha, \\ & \alpha \in \Delta^{m-1}, \end{aligned} \tag{11.15}$$

where the optimal solution is $z^* = W_z \alpha^*$. It is important to note the similarities between (11.11) and (11.15), where the only difference is the the method for approximating the cost residual and constraint functions. Another key difference between these methods is the implicit trust region present in the simplex constraint on α. Since $\alpha \in \Delta^{m-1}$, this means that the approximated functions are constrained to be within the convex hull of the sampled points. By utilizing a sampling scheme that only samples points within a set trust region, the solution to (11.15) is guaranteed to stay within the trust region.

11.3 The Trajectory Bundle Method

In this section we outline a canonical trajectory optimization problem specification and use linearly interpolated trajectory bundles to approximate the cost and constraint functions. This formulation is not only general enough to handle a variety of robotics platforms, it can also be extended to solve batch state estimation problems that share the trajectory optimization problem structure.

Trajectories will be represented in discrete-time as a list of vectors. For a dynamical system with a state $x \in \mathbf{R}^{n_x}$ and control $u \in \mathbf{R}^{n_u}$, the discrete-time dynamics function $x_{k+1} = f(x_k, u_k)$ maps the state and control at time-step k to the state at $k+1$. A trajectory comprised of N time-steps is represented with $(x_{1:N}, u_{1:N-1})$, such that numerical optimization can be used to solve for these values.

11.3.1 Trajectory Optimization

Trajectory optimization is a powerful tool for reasoning about challenging control problems by solving for the optimal control sequence with numerical optimization. This technique

leverages advances in nonlinear programming to solve a variety of underactuated control tasks with nonlinear dynamics. A canonical trajectory optimization problem considering a trajectory with N time-steps is represented as the following:

$$\begin{aligned} \underset{x_{1:N},\, u_{1:N-1}}{\text{minimize}} \quad & \|r_N(x_N)\|_2^2 + \sum_{k=1}^{N-1} \|r_k(x_k, u_k)\|_2^2 \\ \text{subject to} \quad & x_{k+1} = f(x_k, u_k), \\ & c(x_k, u_k) \geq 0, \end{aligned} \tag{11.16}$$

where $c(x_k)$ is a generic constraint function. We will assume that all relevant constraints will be expressed in this form, including initial/goal constraints, state/control limits, and other arbitrary constraints present in the problem.

The problem format in (11.16) is often referred to as multiple shooting [67, 17], where both the state and control histories are optimized over, and the trajectory only becomes dynamically feasible at constraint convergence. This differs from single shooting, where only the controls are optimized over, and a rollout is performed to recover the states. One important distinction between these two methods is that in single shooting, the discrete-time dynamics must be evaluated sequentially $N-1$ times during the rollout, where in multiple shooting, the $N-1$ dynamics constraints can be evaluated entirely in parallel. This is especially relevant with GPU-based physics simulation, where the speed of a single simulation can be comparable to thousands of simulations run in parallel.

Solving Multiple Shooting with Trajectory Bundles

The trajectory bundle method is able to reason about the trajectory optimization problem in (11.16) without having to differentiate any of the cost, dynamics, or constraint functions. Instead of using derivatives to approximate these functions with their first-order Taylor series, sampled trajectories near the current iterate are used to evaluate these functions for approximation with linear interpolation. This idea shown in 11.2.1 Given an initial guess or current iterate $(\bar{x}_{1:N}, \bar{u}_{1:N-1})$, the costs, constraint, and dynamics functions are computed for each of the M samples surrounding each knot points. To demonstrate this, let us examine a single knot point, k, where the current iterate is $(\bar{x}_k, \bar{u}_k)$. From here, m points are sampled near the iterate, and these samples are horizontally concatenated into the following matrices:

$$W_x^{(k)} = \begin{bmatrix} x_{k,1} & x_{k,2} & \cdots & x_{k,m} \end{bmatrix} \in \mathbf{R}^{n_x \times m}, \tag{11.17}$$

$$W_u^{(k)} = \begin{bmatrix} u_{k,1} & u_{k,2} & \cdots & u_{k,m} \end{bmatrix} \in \mathbf{R}^{n_u \times m}, \tag{11.18}$$

after which, all of the cost, dynamics, and constraint functions are computed and stored in a similar fashion,

$$W_r^{(k)} = \begin{bmatrix} r(x_{k,1}, u_{k,1}) & r(x_{k,2}, u_{k,2}) & \cdots & r(x_{k,m}, u_{k,m}) \end{bmatrix} \in \mathbf{R}^{n_r \times m}, \tag{11.19}$$

$$W_f^{(k)} = \begin{bmatrix} f(x_{k,1}, u_{k,1}) & f(x_{k,2}, u_{k,2}) & \cdots & f(x_{k,m}, u_{k,m}) \end{bmatrix} \in \mathbf{R}^{n_x \times m}, \tag{11.20}$$

$$W_c^{(k)} = \begin{bmatrix} c(x_{k,1}, u_{k,1}) & c(x_{k,2}, u_{k,2}) & \cdots & c(x_{k,m}, u_{k,m}) \end{bmatrix} \in \mathbf{R}^{n_c \times m}. \tag{11.21}$$

These matrices are computed for time-steps $1 \to N$, with time-step N omitting the dynamics evaluation matrix W_f (since there is no need to propagate dynamics after the end of the trajectory). Together, these matrices can be used to locally approximate the potentially nonconvex optimization problem in (11.16) as the following convex optimization problem:

$$\begin{aligned}
\underset{\alpha_{1:N}}{\text{minimize}} \quad & \|W_r^{(N)}\alpha^{(N)}\|_2^2 + \sum_{k=1}^{N-1} \|W_r^{(k)}\alpha^{(k)}\|_2^2 + \phi(s) + \phi(w) \\
\text{subject to} \quad & x_k = W_x^{(k)}\alpha^{(k)}, \\
& u_k = W_u^{(k)}\alpha^{(k)}, \\
& c_k = W_c^{(k)}\alpha^{(k)}, \\
& x_{k+1} = W_f^{(k)}\alpha^{(k)} + s_{k+1}, \\
& c_k + w_k \geq 0, \\
& \alpha^{(k)} \in \Delta^{m-1}.
\end{aligned} \tag{11.22}$$

After each time this problem is solved, the new iterates $(x_{1:N}, u_{1:N-1})$ are used to generate m new samples around the iterate, and the problem is formed and solved again. This is a sequential convex programming algorithm, where the solution to the original nonconvex optimization problem is solved for with a sequence of convex ones [130]. In this implementation, the penalty function used is $\phi(\cdot) = \rho\|\cdot\|_1$ with a large ($\rho \in [10^5, 10^8]$). As soon as the solver is able to treat the constraints as real constraints instead of penalties, the penalties are removed and true constraint satisfaction is achieved.

11.3.2 Batch State Estimation

Another valuable extension of the trajectory bundle method is for batch state estimation, where the maximum a posteriori (MAP) can be solved for in a framework compatible with (11.16). Like the Unscented Kalman Filter (UKF), this approach does not require any derivatives from the dynamics or measurement functions.

We consider a dynamical system with the following process and measurement models:

$$x_{k+1} = f(x_k) + q_k, \qquad q \sim \mathcal{N}(0, Q), \tag{11.23}$$

$$v_k = g(x_k) + r_k, \qquad r \sim \mathcal{N}(0, R), \tag{11.24}$$

where $f : \mathbf{R}^{n_x} \to \mathbf{R}^{n_x}$ is the dynamics function, and $g : \mathbf{R}^{n_x} \to \mathbf{R}^{n_v}$ is the measurement function. Oftentimes there exists a Gaussian prior belief state, that is represented with $x_1 \sim \mathcal{N}(\mu_{ic}, \Sigma_{ic})$, representing the belief over the state estimate at the first time-step. From here, the MAP problem solving for the optimal mean $(\mu_{1:N})$ takes the form of the following optimization problem:

$$\begin{aligned}
\underset{\mu_{1:N}}{\text{minimize}} \quad & \|\mu_{ic} - \mu_1\|_{\Sigma^{-1}}^2 + \sum_{k=1}^{N-1} \|q_k\|_{Q^{-1}}^2 + \|r_{k+1}\|_{R^{-1}}^2 \\
\text{subject to} \quad & \mu_{k+1} = f(\mu_k) + q_k, \\
& y_k = g(\mu_k) + r_k,
\end{aligned} \tag{11.25}$$

which falls within the structure of the canonical trajectory optimization problem shown in (11.16). The trajectory bundle method is then used to approximate the dynamics and measurement functions with m samples at each time-step in the same way shown in (11.17)-(11.21)

$$\begin{aligned} \underset{\mu_{1:N}}{\text{minimize}} \quad & \|\mu_{ic} - \mu_1\|^2_{\Sigma^{-1}} + \sum_{k=1}^{N-1} \|q_k\|^2_{Q^{-1}} + \|r_{k+1}\|^2_{R^{-1}} \\ \text{subject to} \quad & \mu_k = W_\mu^{(k)} \alpha^{(k)}, \\ & \mu_{k+1} = W_f^{(k)} \alpha^{(k)} + q_k, \\ & v_k = W_g^{(k)} \alpha^{(k)} + r_k, \\ & \alpha^{(k)} \in \Delta^{m-1}, \end{aligned} \tag{11.26}$$

where v_k are the true measurements (not being optimized over). Since there are no hard constraints in this formulation, there is no need for a penalty ϕ to reduce the constraint violations. Instead, the process for solving (11.25) simply involves forming the bundles, solving (11.26), and repeating until convergence.

11.4 Examples

To demonstrate the efficacy of the trajectory bundle method for solving derivative-free nonconvex trajectory optimization problems, a variety of classic and challenging robotics problems are solved. In each of these examples, the discrete-time dynamics were evaluated with MuJoCo XLA (MJX) [165], such that the evaluation of the bundles is parallelized over both the samples for each time-step, as well as over all time-steps. The convex optimization problems are solved with Clarabel [62] through CVXPY [40].

11.5 Conclusion

In this work, we present the trajectory bundle method, a derivative-free trajectory optimization technique capable of solving nonconvex constrained optimization problems with strong constraint satisfaction. Instead of approximating the nonconvex functions with first-order Taylor series, the trajectory bundle method samples points locally and computes the cost, dynamics, and constraint functions for each of these samples in what we refer to as bundles. These bundles are used to linearly interpolate between these sampled values to approximate the cost, dynamics, and constraint functions. After the computation of these highly parallelizeable function calls, a convex optimization problem is solved where the nonconvex functions are replaced with linear interpolants, and the solution is used to generate new samples for the bundles. The effectiveness of this method is demonstrated on a variety of robotics platforms.

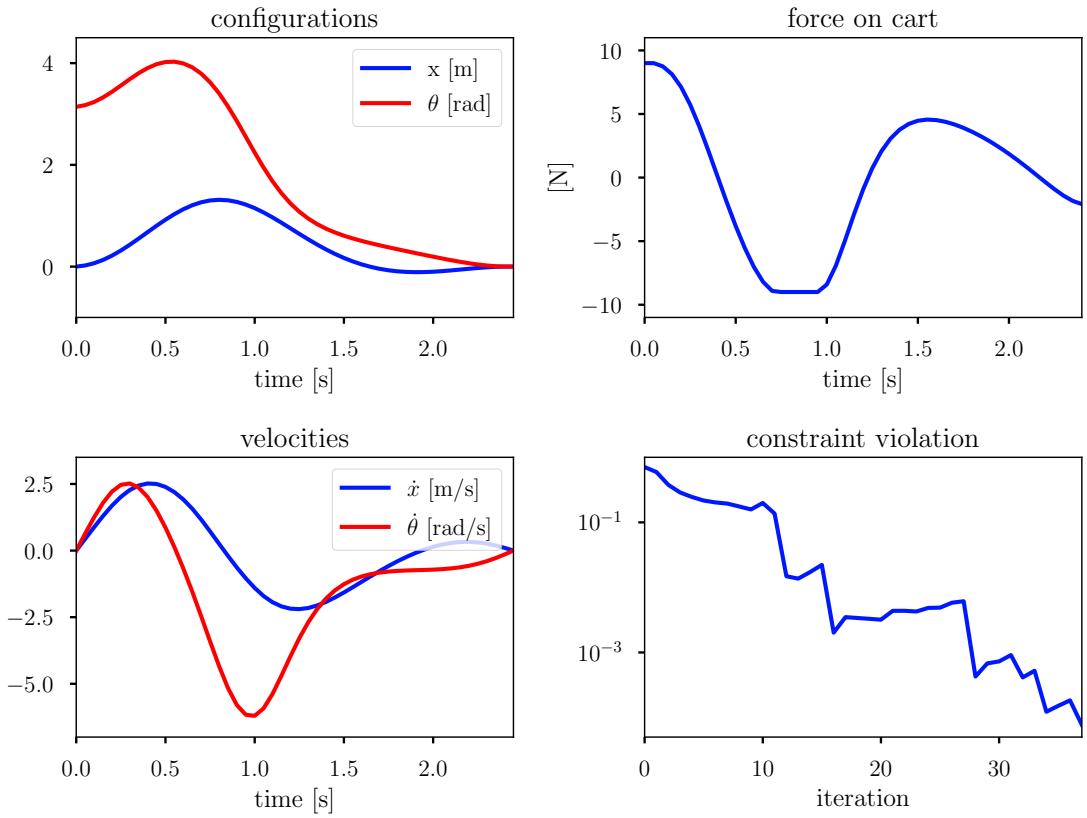


Figure 11.2: The classic cartpole swingup task solved with the trajectory bundle method. In 2.5 seconds, the pole must swing from the lowest position to the highest position, with control constraints on the force on the cart. Without the need for any derivatives, the proposed method is able to achieve tight constraint satisfaction in fewer than 40 iterations.

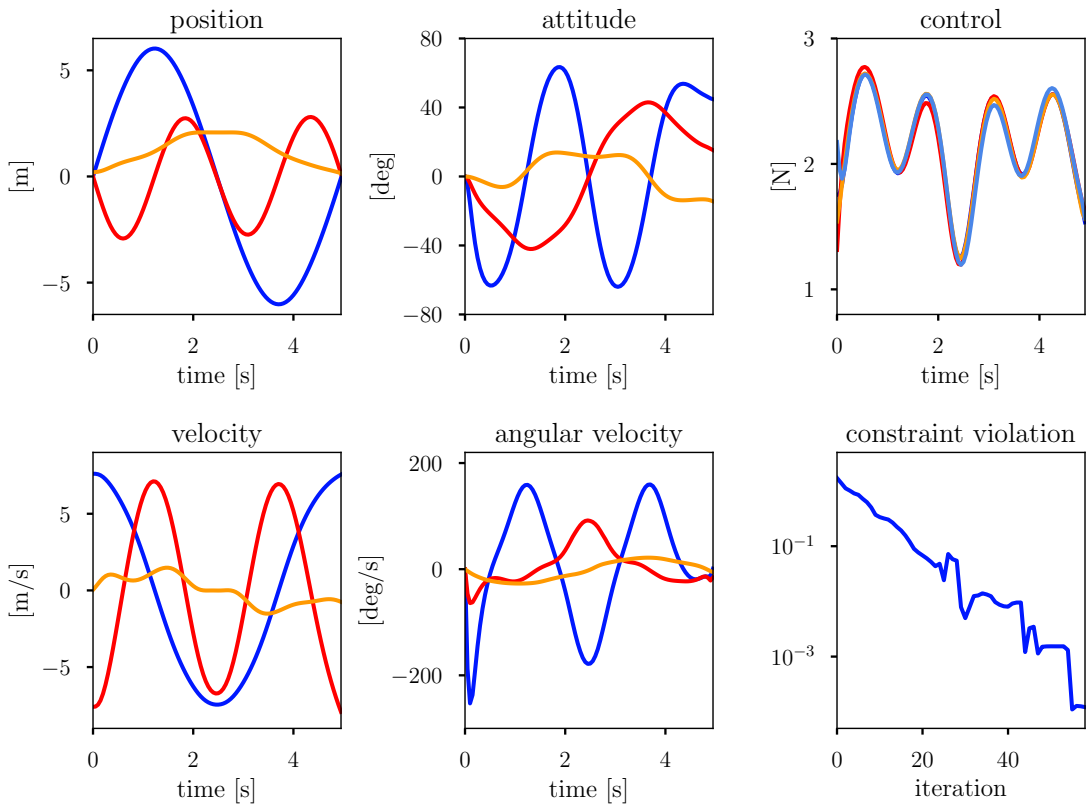


Figure 11.3: Given a figure eight reference trajectory, a quadrotor with rotor velocity control is tasked with tracking this trajectory as closely as possible. The trajectory bundle method is used to solve for the aggressive trajectory, with angular velocity in excess of 200 degrees per second at times. The trajectory is discretized into 100 time-steps and the method converges to a solution in fewer than 60 iterations.

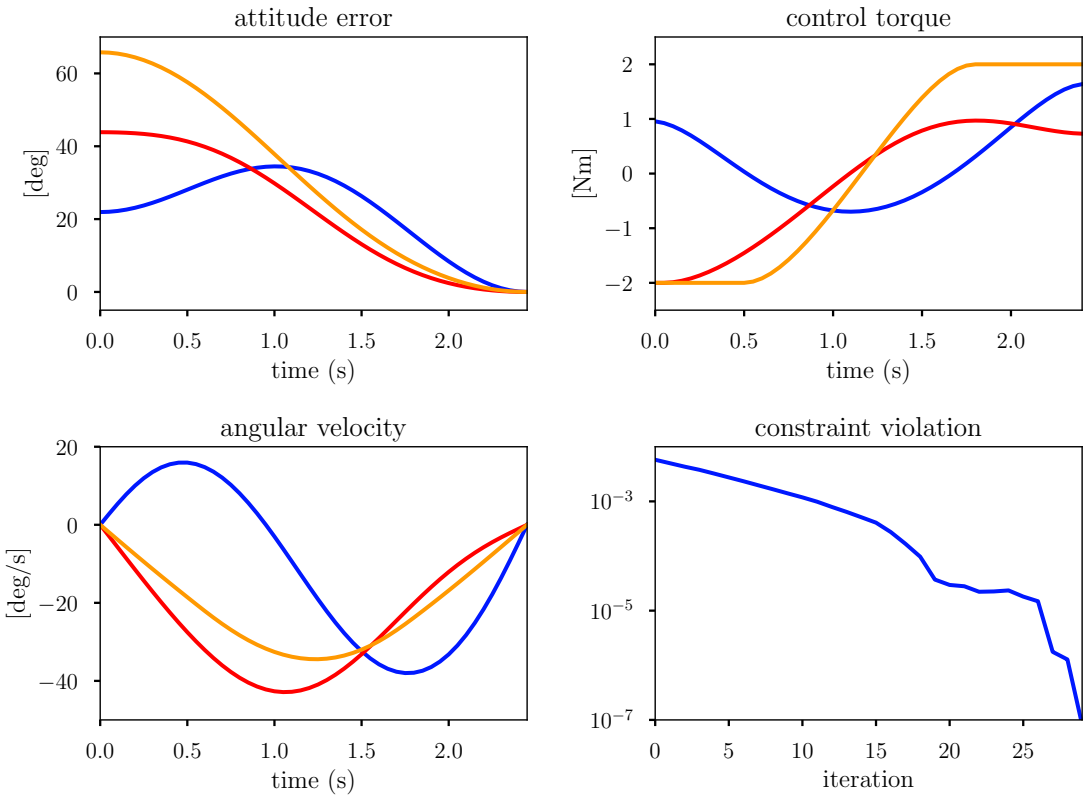


Figure 11.4: A highly nonlinear satellite rest-to-rest slew is solved for with the trajectory bundle method. The satellite has direct torque control on the body with limits of 2 Nm.

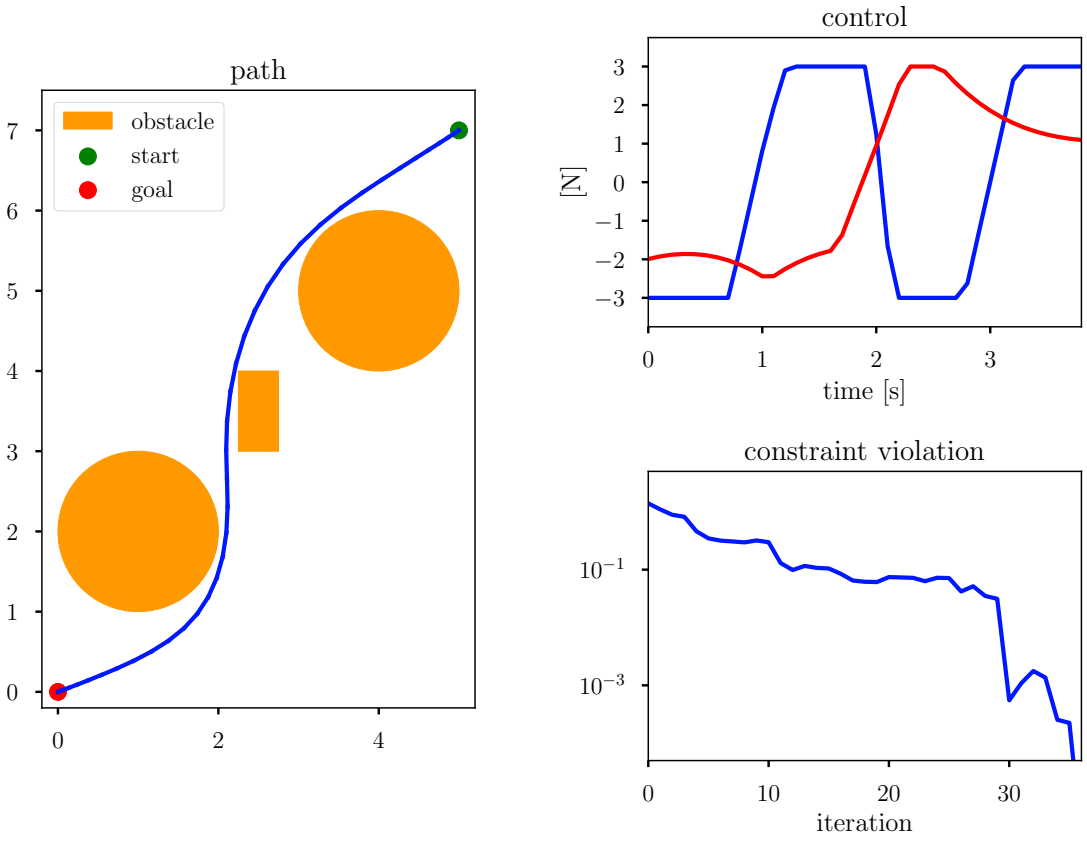


Figure 11.5: A double integrator with acceleration control is tasked with navigating around three obstacles to arrive at a goal position. The trajectory bundle method is able to directly reason about these arbitrary nonconvex constraints without the need for any derivatives, with strong constraint satisfaction and optimality achieved in fewer than 40 iterations.

Bibliography

This bibliography contains 201 references.

[1] Vincent Acary, Maurice Brémond, and Olivier Huber. "On Solving Contact Problems with Coulomb Friction: Formulations and Numerical Comparisons". In: *Advanced Topics in Nonsmooth Dynamics*. Ed. by Remco Leine, Vincent Acary, and Olivier Brüls. Cham: Springer International Publishing, 2018, pp. 375–457. ISBN: 978-3-319-75971-5. DOI: 10.1007/978-3-319-75972-2_10. (Visited on 12/29/2019).

[2] Behcet Acikmese, John M. Carson, and Lars Blackmore. "Lossless Convexification of Nonconvex Control Bound and Pointing Constraints of the Soft Landing Optimal Control Problem". In: *IEEE Transactions on Control Systems Technology* 21.6 (2013), pp. 2104–2113. DOI: 10.1109/TCST.2012.2237346.

[3] Behcet Acikmese and Scott R. Ploen. "Convex Programming Approach to Powered Descent Guidance for Mars Landing". In: *Journal of Guidance, Control, and Dynamics* 30.5 (Sept. 2007), pp. 1353–1366. DOI: 10.2514/1.27553. (Visited on 10/27/2020).

[4] Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and Zico Kolter. "Differentiable Convex Optimization Layers". In: *Advances in Neural Information Processing Systems* (2019), pp. 9558–9570. arXiv: 1910.12430. (Visited on 02/08/2021).

[5] Akshay Agrawal, Shane Barratt, Stephen Boyd, Walaa M Moursi, and Enzo Busseti. "Differentiating Through a Conic Program". In: *Journal of Applied and Numerical Optimization* 1.2 (2019), pp. 107–115.

[6] Gregory Allan et al. "The Deformable Mirror Demonstration Mission (DeMi) CubeSat: Optomechanical Design Validation and Laboratory Calibration". In: *arXiv:1807.02649 [astro-ph]* (July 2018). arXiv: 1807.02649 [astro-ph]. (Visited on 10/29/2020).

[7] Brandon Amos and J Zico Kolter. "Optnet: Differentiable Optimization as a Layer in Neural Networks". In: *International Conference on Machine Learning*. PMLR, 2017, pp. 136–145.

[8] Brandon Amos and J. Zico Kolter. "OptNet: Differentiable Optimization as a Layer in Neural Networks". In: *arXiv:1703.00443 [cs, math, stat]* (Oct. 2019). arXiv: 1703.00443 [cs, math, stat]. (Visited on 02/08/2021).

[9] E.D. Andersen, C. Roos, and T. Terlaky. "On Implementing a Primal-Dual Interior-Point Method for Conic Quadratic Optimization". In: *Mathematical Programming* 95.2 (Feb. 2003), pp. 249–277. ISSN: 0025-5610, 1436-4646. DOI: 10.1007/s10107-002-0349-3. (Visited on 09/06/2022).

[10] Mihai Anitescu. "Optimization-Based Simulation of Nonsmooth Rigid Multibody Dynamics". In: *Mathematical Programming* 105.1 (Jan. 2006), pp. 113–143. ISSN: 0025-5610, 1436-4646. DOI: 10.1007/s10107-005-0590-7. (Visited on 05/13/2018).

[11] Charles Audet and J. E. Dennis. "Mesh Adaptive Direct Search Algorithms for Constrained Optimization". In: *SIAM Journal on Optimization* 17.1 (Jan. 2006), pp. 188–217. ISSN: 1052-6234, 1095-7189. DOI: 10.1137/040603371. (Visited on 11/08/2024).

[12] Charles Audet, Sébastien Le Digabel, Viviane Rochon Montplaisir, and Christophe Tribes. *NOMAD Version 4: Nonlinear Optimization with the MADS Algorithm*. May 2021. arXiv: 2104.11627 [math]. (Visited on 11/08/2024).

[13] Goran Banjac, Bartolomeo Stellato, Nicholas Moehle, Paul Goulart, Alberto Bemporad, and Stephen Boyd. "Embedded Code Generation Using the OSQP Solver". In: *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. Melbourne, VIC: IEEE, Dec. 2017, pp. 1906–1911. ISBN: 978-1-5090-2873-3. DOI: 10.1109/CDC.2017.8263928. (Visited on 11/01/2020).

[14] Yaakov Bar-Shalom, Thiagalingam Kirubarajan, and X.-Rong Li. *Estimation with Applications to Tracking and Navigation*. USA: John Wiley & Sons, Inc., 2002. ISBN: 978-0-471-22127-2.

[15] G. A. Beals, R. C. Crum, H. J. Dougherty, D. K. Hegel, J. L. Kelley, and J. J. Rodden. "Hubble Space Telescope Precision Pointing Control System". In: *Journal of Guidance, Control, and Dynamics* 11.2 (Mar. 1988), pp. 119–123. ISSN: 0731-5090, 1533-3884. DOI: 10.2514/3.20280. (Visited on 08/23/2021).

[16] J. Benesty and T. Gansler. "A Recursive Estimation of the Condition Number in the RLS Algorithm [Adaptive Signal Processing Applications]". In: *Proceedings. (ICASSP '05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.* Vol. 4. Mar. 2005, iv/25–iv/28 Vol. 4. DOI: 10.1109/ICASSP.2005.1415936. (Visited on 11/23/2023).

[17] John T Betts. *Practical Methods for Optimal Control Using Nonlinear Programming*. Vol. 3. Advances in Design and Control. Philadelphia, PA: Society for Industrial and Applied Mathematics (SIAM), 2001.

[18] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B. Shah. "Julia: A Fresh Approach to Numerical Computing". In: *SIAM Review* 59.1 (Jan. 2017), pp. 65–98. ISSN: 0036-1445, 1095-7200. DOI: 10.1137/141000671. (Visited on 01/30/2020).

[19] Arun L. Bishop, John Z. Zhang, Swaminathan Gurumurthy, Kevin Tracy, and Zachary Manchester. "ReLU-QP: A GPU-Accelerated Quadratic Programming Solver for Model-Predictive Control". In: *International Conference on Robotics and Automation (ICRA)*. Yokohama, Japan, 2024. (Visited on 02/13/2024).

[20] L. Blackmore, B. Açikmeşe, and J. M. Carson. "Lossless Convexification of Control Constraints for a Class of Nonlinear Optimal Control Problems". In: *2012 American Control Conference (ACC)*. June 2012, pp. 5519–5525. DOI: 10.1109/ACC.2012.6314722.

[21] Lars Blackmore. "Autonomous Precision Landing of Space Rockets". In: *The Bridge* 4.46 (2016), pp. 15–20.

[22] Charles Blair. “Problem Complexity and Method Efficiency in Optimization (A. S. Nemirovsky and D. B. Yudin)”. In: *SIAM Review* 27.2 (June 1985), pp. 264–265. ISSN: 0036-1445. DOI: 10.1137/1027074. (Visited on 05/12/2024).

[23] Francesco Borrelli, Alberto Bemporad, and Manfred Morari. *Predictive Control for Linear and Hybrid Systems*. Cambridge University Press, 2017.

[24] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[25] James Bradbury et al. *JAX: Composable Transformations of Python+NumPy Programs*. 2018.

[26] R.D. Braun, R.M. Manning, R.D. Braun, and R.M. Manning. “Mars Exploration Entry, Descent and Landing Challenges”. In: *2006 IEEE Aerospace Conference*. Big Sky, MT, USA: IEEE, 2006, pp. 1–18. ISBN: 978-0-7803-9545-9. DOI: 10.1109/AERO.2006.1655790. (Visited on 11/02/2019).

[27] Christopher W. Brunner and Ping Lu. “Comparison of Fully Numerical Predictor-Corrector and Apollo Skip Entry Guidance Algorithms”. In: *The Journal of the Astronautical Sciences* 59.3 (Sept. 2012), pp. 517–540. ISSN: 0021-9142, 2195-0571. DOI: 10.1007/s40295-014-0005-1. (Visited on 03/18/2018).

[28] Christopher W. Brunner and Ping Lu. “Skip Entry Trajectory Planning and Guidance”. In: *Journal of Guidance, Control, and Dynamics* 31.5 (Sept. 2008), pp. 1210–1219. ISSN: 0731-5090, 1533-3884. DOI: 10.2514/1.35055. (Visited on 08/23/2021).

[29] Christopher W. Brunner and Ping Lu. “Skip Entry Trajectory Planning and Guidance”. In: *Journal of Guidance, Control, and Dynamics* (May 2012). DOI: 10.2514/1.35055. (Visited on 05/12/2021).

[30] Adolf Busemann, Nguyen Vinh, and Robert Culp. *Hypersonic Flight Mechanics*. Tech. rep. NASA-CR-149170. Sept. 1976, p. 440.

[31] S. Cameron. “Enhancing GJK: Computing Minimum and Penetration Distances between Convex Polyhedra”. In: *Proceedings of International Conference on Robotics and Automation*. Vol. 4. Albuquerque, NM, USA: IEEE, 1997, pp. 3112–3117. ISBN: 978-0-7803-3612-4. DOI: 10.1109/ROBOT.1997.606761. (Visited on 06/24/2022).

[32] Coralia Cartis and Lindon Roberts. “A Derivative-Free Gauss–Newton Method”. In: *Mathematical Programming Computation* 11.4 (Dec. 2019), pp. 631–674. ISSN: 1867-2949, 1867-2957. DOI: 10.1007/s12532-019-00161-7. (Visited on 11/11/2024).

[33] Enrique Castillo, Antonio J. Conejo, Roberto Mínguez, and Carmen Castillo. “A Closed Formula for Local Sensitivity Analysis in Mathematical Programming”. In: *Engineering Optimization* 38.1 (Jan. 2006), pp. 93–112. ISSN: 0305-215X, 1029-0273. DOI: 10.1080/03052150500229418. (Visited on 03/11/2024).

[34] Anindya Chatterjee. “On the Realism of Complementarity Conditions in Rigid Body Collisions”. In: (), p. 10.

[35] A. R. Conn, K. Scheinberg, and Ph. L. Toint. “Recent Progress in Unconstrained Nonlinear Optimization without Derivatives”. In: *Mathematical Programming* 79.1-3 (Oct. 1997), pp. 397–414. ISSN: 0025-5610, 1436-4646. DOI: 10.1007/BF02614326. (Visited on 11/08/2024).

[36] A. R. Conn, Katya Scheinberg, and Luís N. Vicente. *Introduction to Derivative-Free Optimization*. MPS-SIAM Series on Optimization 8. Philadelphia, Pa: SIAM, 2009. ISBN: 978-0-89871-668-9. DOI: 10.1137/1.9780898718768.

[37] Erwin Coumans. "Bullet Physics Simulation". In: *SIGGRAPH*. Los Angeles: ACM, 2015. ISBN: 978-1-4503-3634-5. DOI: 10.1145/2776880.2792704.

[38] Erwin Coumans. "Continuous Collision Detection and Physics". In: ().

[39] George B. Dantzig. "Origins of the Simplex Method". In: *A History of Scientific Computing*. Ed. by Stephen G. Nash. New York, NY, USA: ACM, June 1990, pp. 141–151. ISBN: 978-0-201-50814-7. DOI: 10.1145/87252.88081. (Visited on 05/12/2024).

[40] Steven Diamond and Stephen Boyd. "CVXPY: A Python-Embedded Modeling Language for Convex Optimization". In: ().

[41] Alexander Domahidi, Eric Chu, and Stephen Boyd. "ECOS: An SOCP Solver for Embedded Systems". In: *2013 European Control Conference (ECC)*. Zurich: IEEE, July 2013, pp. 3071–3076. ISBN: 978-3-033-03962-9. DOI: 10.23919/ECC.2013.6669541. (Visited on 10/19/2020).

[42] Alexander Domahidi, Manfred Morari, Manfred Morari, Stephen P Boyd, and Stephen P Boyd. "Methods and Tools for Embedded Optimization and Control". PhD thesis. Zürich: ETH, 2013. ISBN: 9783906031385.

[43] Ewan S. Douglas, Kevin Tracy, and Zachary Manchester. "Practical Limits on Nanosatellite Telescope Pointing: The Impact of Disturbances and Photon Noise". In: *Frontiers in Astronomy and Space Sciences* 8 (Aug. 2021), p. 676252. ISSN: 2296-987X. DOI: 10.3389/fspas.2021.676252. (Visited on 08/25/2021).

[44] Evan Drumwright, Dylan A. Shell, Evan Drumwright, and Dylan A. Shell. "Modeling Contact Friction and Joint Friction in Dynamic Robotic Simulation Using the Principle of Maximum Dissipation". In: *Wokrshop on the Algorithmic Foundations of Robotics (WAFR)*. Ed. by Frans Groen, David Hsu, Volkan Isler, Jean-Claude Latombe, and Ming C. Lin. Singapore, 2010. ISBN: 978-3-642-17451-3. DOI: 10.1007/978-3-642-17452-0_15. (Visited on 02/20/2019).

[45] Soumyo Dutta, Robert D. Braun, and Christopher D. Karlgaard. "Uncertainty Quantification for Mars Entry, Descent, and Landing Reconstruction Using Adaptive Filtering". In: *Journal of Spacecraft and Rockets* 51.3 (May 2014), pp. 967–977. ISSN: 0022-4650, 1533-6794. DOI: 10.2514/1.A32716. (Visited on 11/02/2019).

[46] Karl T. Edquist, Brian R. Hollis, Artem A. Dyakonov, Bernard Laub, Michael J. Wright, Tomasso P. Rivellini, Eric M. Slimko, and William H. Willcockson. "Mars Science Laboratory Entry Capsule Aerothermodynamics and Thermal Protection System". In: *2007 IEEE Aerospace Conference*. Big Sky, MT, USA: IEEE, 2007, pp. 1–13. ISBN: 978-1-4244-0524-4. DOI: 10.1109/AERO.2007.352823. (Visited on 04/02/2021).

[47] Ryan Elandt, Evan Drumwright, Michael Sherman, and Andy Ruina. "A Pressure Field Model for Fast, Robust Approximation of Net Contact Force and Moment between Nominally Rigid Objects". In: *arXiv:1904.11433 [cs]* (Apr. 2019). arXiv: 1904.11433 [cs]. (Visited on 09/25/2019).

[48] Christer Ericson. *Real-Time Collision Detection*. CRC Press, Dec. 2004. ISBN: 978-1-00-075055-3.

[49] Davide Falanga, Kevin Kleber, and Davide Scaramuzza. "Dynamic Obstacle Avoidance for Quadrotors with Event Cameras". In: *Science Robotics* 5.40 (Mar. 2020), eaaz9712. DOI: 10.1126/scirobotics.aaz9712. (Visited on 09/06/2022).

[50] Roy Featherstone. *Robot Dynamics Algorithms*. The Springer International Series in Engineering and Computer Science. Springer US, 1987. ISBN: 978-1-4757-6437-6. (Visited on 09/27/2019).

[51] C Daniel Freeman, Anton Raichuk, Sertan Girgin, Erik Frey, Igor Mordatch, and Olivier Bachem. "Brax - A Differentiable Physics Engine for Large Scale Rigid Body Simulation". In: *NeurIPS*. New Orleans, Dec. 2021, p. 13.

[52] Letian Fu, Huang Huang, Lars Berscheid, Hui Li, Ken Goldberg, and Sachin Chitta. *Safe Self-Supervised Learning in Real of Visuo-Tactile Feedback Policies for Industrial Insertion*. Mar. 2023. DOI: 10.48550/arXiv.2210.01340. arXiv: 2210.01340 [cs]. (Visited on 11/23/2023).

[53] Patrick Gallais. *Atmospheric Re-Entry Vehicle Mechanics*. Berlin ; New York: Springer, 2007. ISBN: 978-3-540-73646-2.

[54] Miguel Galrinho. "Least Squares Methods for System Identification of Structured Models". In: (2016). (Visited on 11/23/2023).

[55] Michael Garstka, Mark Cannon, and Paul Goulart. "COSMO: A Conic Operator Splitting Method for Convex Conic Problems". In: *arXiv:1901.10887 [math]* (Sept. 2020). arXiv: 1901.10887 [math]. (Visited on 10/19/2020).

[56] Andrew Gatherer and Zachary Manchester. "Magnetorquer-Only Attitude Control of Small Satellites Using Trajectory Optimization". In: *AAS/AIAA Astrodynamics Specialist Conference*. Portland, ME, Aug. 2019.

[57] Antonio Genova, Sander Goossens, Frank G. Lemoine, Erwan Mazarico, Gregory A. Neumann, David E. Smith, and Maria T. Zuber. "Seasonal and Static Gravity Field of Mars from MGS, Mars Odyssey and MRO Radio Science". In: *Icarus* 272 (July 2016), pp. 228–245. ISSN: 0019-1035. DOI: 10.1016/j.icarus.2016.02.050. (Visited on 11/22/2020).

[58] Zoubin Ghahramani. "Parameter Estimation for Linear Dynamical Systems". In: ().

[59] E.G. Gilbert and Chong Jin Ong. "New Distances for the Separation and Penetration of Objects". In: *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*. San Diego, CA, USA: IEEE Comput. Soc. Press, 1994, pp. 579–586. ISBN: 978-0-8186-5330-8. DOI: 10.1109/ROBOT.1994.351237. (Visited on 03/04/2023).

[60] E.G. Gilbert, D.W. Johnson, and S.S. Keerthi. "A Fast Procedure for Computing the Distance between Complex Objects in Three-Dimensional Space". In: *IEEE Journal on Robotics and Automation* 4.2 (Apr. 1988), pp. 193–203. ISSN: 08824967. DOI: 10.1109/56.2083. (Visited on 06/24/2022).

[61] Philip E. Gill, Walter Murray, and Michael A. Saunders. "SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization". In: *SIAM Review* 47.1 (Jan.

2005), pp. 99–131. ISSN: 0036-1445, 1095-7200. DOI: 10.1137/S0036144504446096. (Visited on 06/04/2020).

[62] Paul J. Goulart and Yuwen Chen. *Clarabel: An Interior-Point Solver for Conic Programs with Quadratic Objectives*. May 2024. arXiv: 2405.12762 [math]. (Visited on 11/11/2024).

[63] Michael Grant and Stephen Boyd. *CVX: Matlab Software for Disciplined Convex Programming — CVX Research, Inc.* https://cvxr.com/cvx/. (Visited on 05/12/2024).

[64] C. Graves and A. Harpold. *Apollo Experience Report: Mission Planning for Apollo Entry*. Technical Report NASA-TN-D-6725. Houston, TX: NASA Johnson Space Center, Mar. 1972. (Visited on 03/18/2018).

[65] Mathew Halm and Michael Posa. *A Quasi-static Model and Simulation Approach for Pushing, Grasping, and Jamming*. Feb. 2019. DOI: 10.48550/arXiv.1902.03487. arXiv: 1902.03487 [cs]. (Visited on 11/23/2023).

[66] Moritz Hardt, Tengyu Ma, and Benjamin Recht. *Gradient Descent Learns Linear Dynamical Systems*. Feb. 2019. DOI: 10.48550/arXiv.1609.05191. arXiv: 1609.05191 [cs, math, stat]. (Visited on 11/23/2023).

[67] C R Hargraves and S W Paris. "Direct Trajectory Optimization Using Nonlinear Programming and Collocation". In: *J. Guidance* 10.4 (1987), pp. 338–342. DOI: 10.2514/3.20223.

[68] Elad Hazan, Karan Singh, and Cyril Zhang. *Learning Linear Dynamical Systems via Spectral Filtering*. Nov. 2017. DOI: 10.48550/arXiv.1711.00946. arXiv: 1711.00946 [cs, math, stat]. (Visited on 11/23/2023).

[69] Casey R. Heidrich, Marcus J. Holzinger, and Robert D. Braun. "Optimal Information Filtering for Robust Aerocapture Trajectory Generation and Guidance". In: *Journal of Spacecraft and Rockets* (Oct. 2021). DOI: 10.2514/1.A35175. (Visited on 11/09/2021).

[70] Jean-Baptiste Hiriart-Urruty and Claude Lemaréchal. *Convex Analysis and Minimization Algorithms I*. Ed. by M. Artin et al. Vol. 305. Grundlehren Der Mathematischen Wissenschaften. Berlin, Heidelberg: Springer, 1993. ISBN: 978-3-642-08161-3. DOI: 10.1007/978-3-662-02796-7. (Visited on 03/07/2024).

[71] C. Hirt, S.J. Claessens, M. Kuhn, and W.E. Featherstone. "Kilometer-Resolution Gravity Field of Mars: MGM2011". In: *Planetary and Space Science* 67.1 (July 2012), pp. 147–154. ISSN: 00320633. DOI: 10.1016/j.pss.2012.02.006. (Visited on 12/02/2022).

[72] *B* L. Ho and R. E. Kalman. "Editorial: Effective construction of linear state-variable models from input/output functions: Die Konstruktion von linearen Modeilen in der Darstellung durch Zustandsvariable aus den Beziehungen für Ein- und Ausgangsgrößen". In: *at - Automatisierungstechnik* 14.1-12 (Dec. 1966), pp. 545–548. ISSN: 2196-677X. DOI: 10.1524/auto.1966.14.112.545. (Visited on 11/23/2023).

[73] Warren Hoburg and Russ Tedrake. "System Identification of Post Stall Aerodynamics for UAV Perching". In: *AIAA Infotech@Aerospace Conference*. American Institute of Aeronautics and Astronautics, 2009. (Visited on 12/04/2015).

[74] Max Holliday, Kevin Tracy, Zachary Manchester, and Anh Nguyen. "The V-R3x Mission: Towards Autonomous Networking and Navigation for CubeSat Swarms". In: *4S Symposium*. Vilamoura, Portugal, May 22.

[75] Honeywell. *GG1320AN Digital Laser Gyro*.

[76] Peter C. Horak and Jeff C. Trinkle. "On the Similarities and Differences Among Contact Models in Robot Simulation". In: *IEEE Robotics and Automation Letters* 4.2 (Apr. 2019), pp. 493–499. ISSN: 2377-3766, 2377-3774. DOI: 10.1109/LRA.2019.2891085. (Visited on 04/15/2020).

[77] Taylor A Howell, Brian E Jackson, and Zachary Manchester. "ALTRO: A Fast Solver for Constrained Trajectory Optimization". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Macau, China, Nov. 2019.

[78] Taylor A. Howell, Simon Le Cleac'h, J. Zico Kolter, Mac Schwager, and Zachary Manchester. "Dojo: A Differentiable Physics Engine for Robotics". In: *http://arxiv.org/abs/2203.00* (June 2022). DOI: 10.48550/arXiv.2203.00806. arXiv: 2203.00806. (Visited on 09/12/2022).

[79] Taylor A. Howell, Brian E. Jackson, and Zachary Manchester. "ALTRO: A Fast Solver for Constrained Trajectory Optimization". In: *IEEE International Conference on Intelligent Robots and Systems* (Nov. 2019), pp. 7674–7679. ISSN: 9781728140049. DOI: 10.1109/IROS40897.2019.8967788.

[80] Brian Jackson, Taylor Howell, Kunal Shah, Mac Schwager, and Zachary Manchester. "Scalable Cooperative Transport of Cable-Suspended Loads with UAVs Using Distributed Trajectory Optimization". In: *International Conference on Robotics and Automation*. Paris, France, June 2020, p. 8.

[81] Brian E Jackson, Tarun Punnoose, Daniel Neamati, Kevin Tracy, and Rianna Jitosho. "ALTRO-C: A Fast Solver for Conic Model-Predictive Control". In: *International Conference on Robotics and Automation (ICRA)*. Xi'an, China, 2021, p. 8.

[82] Brian E Jackson, Tarun Punnoose, Daniel Neamati, Kevin Tracy, Rianna Jitosho, and Zachary Manchester. "ALTRO-C: A Fast Solver for Conic Model-Predictive Control; ALTRO-C: A Fast Solver for Conic Model-Predictive Control". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)* (2021). ISSN: 9781728190778. DOI: 10.1109/ICRA48506.2021.9561438.

[83] Brian E Jackson, Kevin Tracy, and Zachary Manchester. "Planning with Attitude". In: *IEEE Robotics and Automation Letters* (Jan. 2021).

[84] Brian E. Jackson, Kevin Tracy, and Zachary Manchester. "Planning With Attitude". In: *IEEE Robotics and Automation Letters* (2021), pp. 1–1. ISSN: 2377-3766, 2377-3774. DOI: 10.1109/LRA.2021.3052431. (Visited on 08/25/2021).

[85] Hilary L. Justh and K. Lee Lee. "Mars-GRAM 2010: Additions and Resulting Improvements". In: *International Planetary Probe Workshop*. 17-21 Jun. 2013, United States, June 2013. (Visited on 03/18/2018).

[86] Thomas Kailath, Ali H. Sayed, and Babak Hassibi. *Linear Estimation*. Prentice Hall, 2000. ISBN: 978-0-13-022464-4.

[87] R. E. Kalman. “A New Approach to Linear Filtering and Prediction Problems”. In: *Journal of Basic Engineering* 82.1 (Mar. 1960), pp. 35–45. ISSN: 0021-9223. DOI: 10.1115/1.3662552. (Visited on 04/07/2021).

[88] P. Kaminski, A. Bryson, and S. Schmidt. “Discrete Square Root Filtering: A Survey of Current Techniques”. In: *IEEE Transactions on Automatic Control* 16.6 (Dec. 1971), pp. 727–736. ISSN: 0018-9286. DOI: 10.1109/TAC.1971.1099816. (Visited on 04/21/2021).

[89] Thomas R. Kane, Peter W. Likins, and David A. Levinson. *Spacecraft Dynamics*. New York: McGraw-Hill Book Co, 1983. ISBN: 978-0-07-037843-8.

[90] Hanwen Kang, Yaohua Zang, Xing Wang, and Yaohui Chen. “Uncertainty-Driven Spiral Trajectory for Robotic Peg-in-Hole Assembly”. In: *IEEE Robotics and Automation Letters* 7.3 (July 2022), pp. 6661–6668. ISSN: 2377-3766. DOI: 10.1109/LRA.2022.3176718. (Visited on 11/23/2023).

[91] Martin Kirchengast, Martin Steinberger, and Martin Horn. “A Fast Quadratic Program Solver for Constrained Control Allocation”. In: *2018 IEEE Conference on Decision and Control (CDC)*. Miami Beach, FL: IEEE, Dec. 2018, pp. 4065–4071. ISBN: 978-1-5386-1395-5. DOI: 10.1109/CDC.2018.8619828. (Visited on 06/12/2024).

[92] Mary Knapp et al. “Demonstrating High-Precision Photometry with a CubeSat: ASTERIA Observations of 55 Cancri e”. In: *The Astronomical Journal* 160.1 (June 2020), p. 23. ISSN: 1538-3881. DOI: 10.3847/1538-3881/ab8bcc. arXiv: 2005.14155. (Visited on 10/29/2020).

[93] Mykel J. Kochenderfer and Tim A. Wheeler. *Algorithms for Optimization*. Cambridge, Massachusetts: The MIT Press, 2019. ISBN: 978-0-262-03942-0.

[94] Scott Kuindersma, Frank Permenter, and Russ Tedrake. “An Efficiently Solvable Quadratic Program for Stabilizing Dynamic Locomotion”. In: *Proceedings of the International Conference on Robotics and Automation (ICRA)*. Hong Kong, China: IEEE, May 2014, pp. 2589–2594. ISBN: 978-1-4799-3685-4. DOI: 10.1109/ICRA.2014.6907230.

[95] Sébastien Le Digabel. “Algorithm 909: NOMAD: Nonlinear Optimization with the MADS Algorithm”. In: *ACM Trans. Math. Softw.* 37.4 (Feb. 2011), 44:1–44:15. ISSN: 0098-3500. DOI: 10.1145/1916461.1916468. (Visited on 11/08/2024).

[96] Jeongseok Lee, Michael X. Grey, Sehoon Ha, Tobias Kunz, Sumit Jain, Yuting Ye, Siddhartha S. Srinivasa, Mike Stilman, and C. Karen Liu. “DART: Dynamic Animation and Robotics Toolkit”. In: *The Journal of Open Source Software* 3.22 (Feb. 2018), p. 500. ISSN: 2475-9066. DOI: 10.21105/joss.00500. (Visited on 09/12/2018).

[97] Shuang Li and Xiuqiang Jiang. “Review and Prospect of Guidance and Control for Mars Atmospheric Entry”. In: *Progress in Aerospace Sciences* 69 (Aug. 2014), pp. 40–57. ISSN: 03760421. DOI: 10.1016/j.paerosci.2014.04.001. (Visited on 11/02/2019).

[98] A.P. Liavas and P.A. Regalia. “Numerical Stability Issues of the Conventional Recursive Least Squares Algorithm”. In: *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP ’98 (Cat.*

No.98CH36181). Vol. 3. May 1998, 1409–1412 vol.3. DOI: 10.1109/ICASSP.1998.681711. (Visited on 11/23/2023).

[99] A.P. Liavas and P.A. Regalia. "On the Numerical Stability and Accuracy of the Conventional Recursive Least Squares Algorithm". In: *IEEE Transactions on Signal Processing* 47.1 (Jan. 1999), pp. 88–96. ISSN: 1941-0476. DOI: 10.1109/78.738242. (Visited on 11/23/2023).

[100] Miguel Sousa Lobo, Lieven Vandenberghe, Stephen Boyd, and Hervé Lebret. "Applications of Second-Order Cone Programming". In: *Linear Algebra and its Applications* 284.1-3 (Nov. 1998), pp. 193–228. ISSN: 00243795. DOI: 10.1016/S0024-3795(98)10032-0. (Visited on 06/18/2018).

[101] Dongning Lu and Yiwu Liu. "Singular Formalism and Admissible Control of Spacecraft with Rotating Flexible Solar Array". In: *Chinese Journal of Aeronautics* 27.1 (Feb. 2014), pp. 136–144. ISSN: 10009361. DOI: 10.1016/j.cja.2013.12.010. (Visited on 05/29/2020).

[102] Ping Lu. "Entry Guidance: A Unified Method". In: *Journal of Guidance, Control, and Dynamics* 37.3 (May 2014), pp. 713–728. ISSN: 0731-5090, 1533-3884. DOI: 10.2514/1.62605. (Visited on 02/21/2018).

[103] Ping Lu. "Predictor-Corrector Entry Guidance for Low-Lifting Vehicles". In: *Journal of Guidance, Control, and Dynamics* 31.4 (July 2008), pp. 1067–1075. ISSN: 0731-5090, 1533-3884. DOI: 10.2514/1.32055. (Visited on 11/02/2019).

[104] Miles Lubin and Iain Dunning. "Computing in Operations Research Using Julia". In: *INFORMS Journal on Computing* 27.2 (Apr. 2015), pp. 238–248. ISSN: 1091-9856, 1526-5528. DOI: 10.1287/ijoc.2014.0623. arXiv: 1312.1431. (Visited on 12/16/2020).

[105] Viktor Makoviychuk et al. *Isaac Gym: High Performance GPU-Based Physics Simulation For Robot Learning*. Aug. 2021. DOI: 10.48550/arXiv.2108.10470. arXiv: 2108.10470 [cs]. (Visited on 04/23/2023).

[106] Danylo Malyuta, Taylor P. Reynolds, Michael Szmuk, Thomas Lew, Riccardo Bonalli, Marco Pavone, and Behcet Acikmese. "Convex Optimization for Trajectory Generation". In: *arXiv:2106.09125 [cs, eess, math]* (June 2021). DOI: 10.48550/arXiv.2106.09125. arXiv: 2106.09125 [cs, eess, math]. (Visited on 07/09/2021).

[107] Yuanqi Mao, Michael Szmuk, Xiangru Xu, and Behcet Acikmese. "Successive Convexification: A Superlinearly Convergent Algorithm for Non-convex Optimal Control Problems". In: *arXiv:1804.06539 [math]* (Feb. 2019). DOI: 10.48550/arXiv.1804.06539. arXiv: 1804.06539 [math]. (Visited on 02/09/2021).

[108] Tobia Marcucci, Parth Nobel, Russ Tedrake, and Stephen Boyd. "Fast Path Planning Through Large Collections of Safe Boxes". In: ().

[109] Tobia Marcucci, Jack Umenberger, Pablo A. Parrilo, and Russ Tedrake. *Shortest Paths in Graphs of Convex Sets*. July 2023. arXiv: 2101.11565 [cs, math]. (Visited on 03/07/2024).

[110] F. Landis Markley, Yang Cheng, John Lucas Crassidis, and Yaakov Oshman. "Averaging Quaternions". In: *Journal of Guidance, Control, and Dynamics* 30.4 (July

2007), pp. 1193–1197. ISSN: 0731-5090, 1533-3884. DOI: 10.2514/1.28949. (Visited on 06/13/2018).

[111] F. Landis Markley and John L. Crassidis. *Fundamentals of Spacecraft Attitude Determination and Control*. New York, NY: Springer New York, 2014. ISBN: 978-1-4939-0801-1. DOI: 10.1007/978-1-4939-0802-8. (Visited on 05/29/2020).

[112] J. E. Marsden and M. West. "Discrete Mechanics and Variational Integrators". In: *Acta Numerica* 10 (2001), pp. 357–514.

[113] Matthew T. Mason. *Mechanics of Robotic Manipulation*. The MIT Press, June 2001. ISBN: 978-0-262-25662-9. DOI: 10.7551/mitpress/4527.001.0001. (Visited on 11/23/2023).

[114] Matthew T. Mason. *Mechanics of Robotic Manipulation*. The MIT Press, June 2001. ISBN: 978-0-262-25662-9. DOI: 10.7551/mitpress/4527.001.0001. (Visited on 08/23/2023).

[115] Jacob Mattingley and Stephen Boyd. "CVXGEN: A Code Generator for Embedded Convex Optimization". In: *Optimization Engineering*. 2012, pp. 1–27.

[116] Sanjay Mehrotra. "On the Implementation of a Primal-Dual Interior Point Method". In: *SIAM Journal on Optimization* 2.4 (Nov. 1992), pp. 575–601. ISSN: 1052-6234, 1095-7189. DOI: 10.1137/0802028. (Visited on 06/24/2022).

[117] D. Mellinger and V. Kumar. "Minimum Snap Trajectory Generation and Control for Quadrotors". In: *2011 IEEE International Conference on Robotics and Automation (ICRA)*. May 2011, pp. 2520–2525. DOI: 10.1109/ICRA.2011.5980409.

[118] Daniel Mellinger, Nathan Michael, and Vijay Kumar. "Trajectory Generation and Control for Precise Aggressive Maneuvers with Quadrotors". In: (), p. 13.

[119] Gavin F. Mendeck and Lynn Craig McGrew. "Entry Guidance Design and Postflight Performance for 2011 Mars Science Laboratory Mission". In: *Journal of Spacecraft and Rockets* 51.4 (July 2014), pp. 1094–1105. ISSN: 0022-4650, 1533-6794. DOI: 10.2514/1.A32737. (Visited on 02/21/2018).

[120] Paul Mitiguy. *Advanced Dynamics & Motion Simulation*. Motion Genesis, Aug. 2018.

[121] Mayank Mittal et al. *ORBIT: A Unified Simulation Framework for Interactive Robot Learning Environments*. Jan. 2023. DOI: 10.48550/arXiv.2301.04195. arXiv: 2301.04195 [cs]. (Visited on 04/23/2023).

[122] Louis Montaut, Quentin Le Lidec, Antoine Bambade, Vladimir Petrik, Josef Sivic, and Justin Carpentier. *Differentiable Collision Detection: A Randomized Smoothing Approach*. Sept. 2022. arXiv: 2209.09012 [cs]. (Visited on 03/04/2023).

[123] O Montenbruck, E Gill, and Fh Lutze. "Satellite Orbits: Models, Methods, and Applications". In: *Applied Mechanics Reviews* 55.2 (2002), B27. ISSN: 00036900. DOI: 10.1115/1.1451162. (Visited on 12/15/2020).

[124] Mosek ApS. *The MOSEK Optimization Software*. Tech. rep. 2014.

[125] Ashvin Nair, Brian Zhu, Gokul Narayanan, Eugen Solowjow, and Sergey Levine. *Learning on the Job: Self-Rewarding Offline-to-Online Finetuning for Industrial*

Insertion of Novel Connectors from Vision. Feb. 2023. DOI: 10.48550/arXiv.2210.15206. arXiv: 2210.15206 [cs]. (Visited on 11/23/2023).

[126] Yu. E. Nesterov and M. J. Todd. "Primal-Dual Interior-Point Methods for Self-Scaled Cones". In: *SIAM Journal on Optimization* 8.2 (May 1998), pp. 324–364. ISSN: 1052-6234, 1095-7189. DOI: 10.1137/S1052623495290209. (Visited on 09/06/2022).

[127] Yu. E. Nesterov and M. J. Todd. "Self-Scaled Barriers and Interior-Point Methods for Convex Programming". In: *Mathematics of Operations Research* 22.1 (Feb. 1997), pp. 1–42. ISSN: 0364-765X, 1526-5471. DOI: 10.1287/moor.22.1.1. (Visited on 09/06/2022).

[128] Rhys Newbury, Jack Collins, Kerry He, Jiahe Pan, Ingmar Posner, David Howard, and Akansel Cosgun. "A Review of Differentiable Simulators". In: *IEEE Access* 12 (2024), pp. 97581–97604. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2024.3425448. (Visited on 07/23/2024).

[129] Joshua Newth. "Minkowski Portal Refinement and Speculative Contacts in Box2D". Master of Science. San Jose, CA, USA: San Jose State University, Apr. 2013. DOI: 10.31979/etd.q6rm-ch9a. (Visited on 06/24/2022).

[130] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. 2nd. Springer, 2006.

[131] Jia Pan, Sachin Chitta, and Dinesh Manocha. "FCL: A General Purpose Library for Collision and Proximity Queries". In: *2012 IEEE International Conference on Robotics and Automation*. St Paul, MN, USA: IEEE, May 2012, pp. 3859–3866. ISBN: 978-1-4673-1405-3. DOI: 10.1109/ICRA.2012.6225337. (Visited on 09/06/2022).

[132] Tao Pang, H. J. Terry Suh, Lujie Yang, and Russ Tedrake. *Global Planning for Contact-Rich Manipulation via Local Smoothing of Quasi-dynamic Contact Models*. Feb. 2023. arXiv: 2206.10787 [cs]. (Visited on 08/23/2023).

[133] Tao Pang and Russ Tedrake. "A Convex Quasistatic Time-stepping Scheme for Rigid Multibody Systems with Contact and Friction". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. Xi'an, China: IEEE, May 2021, pp. 6614–6620. ISBN: 978-1-72819-077-8. DOI: 10.1109/ICRA48506.2021.9560941. (Visited on 08/23/2023).

[134] Tao Pang and Russ Tedrake. "A Robust Time-Stepping Scheme for Quasistatic Rigid Multibody Systems". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Madrid: IEEE, Oct. 2018, pp. 5640–5647. ISBN: 978-1-5386-8094-0. DOI: 10.1109/IROS.2018.8594378. (Visited on 08/23/2023).

[135] J. F. O. De O. Pantoja and D. Q. Mayne. "A Sequential Quadratic Programming Algorithm for Discrete Optimal Control Problems with Control Inequality Constraints". In: *Proceedings of the 28th IEEE Conference on Decision and Control,* Dec. 1989, 353–357 vol.1. DOI: 10.1109/CDC.1989.70136.

[136] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc., 2019. (Visited on 12/13/2021).

[137] Robert Penicka, Yunlong Song, Elia Kaufmann, and Davide Scaramuzza. "Learning Minimum-Time Flight in Cluttered Environments". In: *IEEE Robotics and Au-*

tomation Letters 7.3 (July 2022), pp. 7209–7216. ISSN: 2377-3766, 2377-3774. DOI: 10.1109/LRA.2022.3181755. (Visited on 09/06/2022).

[138] Christopher Pong. "On-Orbit Performance & Operation of the Attitude & Pointing Control Subsystems on ASTERIA". In: (), p. 20.

[139] Christopher M Pong, Matthew W Smith, Matthew W Knutson, Sungyung Lim, David W Miller, Sara Seager, Jesus S Villaseñor, and Shawn D Murphy. "ONE-ARCSECOND LINE-OF-SIGHT POINTING CONTROL ON EXOPLANETSAT, A THREE-UNIT CUBESAT". In: (), p. 21.

[140] M. J. D. Powell. "A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation". In: *Advances in Optimization and Numerical Analysis*. Ed. by Susana Gomez and Jean-Pierre Hennart. Dordrecht: Springer Netherlands, 1994, pp. 51–67. ISBN: 978-90-481-4358-0 978-94-015-8330-5. DOI: 10.1007/978-94-015-8330-5_4. (Visited on 11/08/2024).

[141] Zachary R. Putnam and Robert D. Braun. "Precision Landing at Mars Using Discrete-Event Drag Modulation". In: *Journal of Spacecraft and Rockets* 51.1 (Jan. 2014), pp. 128–138. ISSN: 0022-4650, 1533-6794. DOI: 10.2514/1.A32633. (Visited on 12/02/2022).

[142] Zachary R. Putnam, Matthew D. Neave, and Gregg H. Barton. "PredGuid Entry Guidance for Orion Return from Low Earth Orbit". In: *2010 IEEE Aerospace Conference*. Big Sky, MT, USA: IEEE, Mar. 2010, pp. 1–13. ISBN: 978-1-4244-3887-7. DOI: 10.1109/AERO.2010.5447010. (Visited on 12/02/2022).

[143] R. Tyrrell Rockafellar. *Convex Analysis*. Princeton University Press, Jan. 1997. ISBN: 978-0-691-01586-6.

[144] Evan Roelke, Phil D Hattis, and R D Braun. "IMPROVED ATMOSPHERIC ESTIMATION FOR AEROCAPTURE GUIDANCE". In: *Spaceflight Mechanics Meeting*. Savannah, Georgia: AAS/AIAA, Feb. 2009, p. 16.

[145] John Schulman, Jonathan Ho, Alex Lee, Ibrahim Awwal, Henry Bradlow, and Pieter Abbeel. "Finding Locally Optimal, Collision-Free Trajectories with Sequential Convex Optimization". In: *Robotics: Science and Systems IX*. Robotics: Science and Systems Foundation, June 2013. ISBN: 978-981-07-3937-9. DOI: 10.15607/RSS.2013.IX.031. (Visited on 03/07/2024).

[146] Jacob T. Schwartz and Micha Sharir. "On the "Piano Movers"' Problem I. The Case of a Two-Dimensional Rigid Polygonal Body Moving amidst Polygonal Barriers". In: *Communications on Pure and Applied Mathematics* 36.3 (May 1983), pp. 345–398. ISSN: 00103640. DOI: 10.1002/cpa.3160360305. (Visited on 09/06/2022).

[147] Paul Serra et al. "Optical Front-End for a Quantum Key Distribution Cubesat". In: *International Conference on Space Optics — ICSO 2020*. Ed. by Zoran Sodnik, Bruno Cugny, and Nikos Karafolas. Online Only, France: SPIE, June 2021, p. 118. ISBN: 978-1-5106-4548-6. DOI: 10.1117/12.2599542. (Visited on 10/15/2021).

[148] Hassan Shraim, Ali Awada, and Rafic Youness. "A Survey on Quadrotors: Configurations, Modeling and Identification, Control, Collision Avoidance, Fault Diagnosis and Tolerant Control". In: *IEEE Aerospace and Electronic Systems Magazine* 33.7

(July 2018), pp. 14–33. ISSN: 0885-8985, 1557-959X. DOI: 10.1109/MAES.2018.160246. (Visited on 09/06/2022).

[149] Dan Simon. *Optimal State Estimation: Kalman, H [Infinity] and Nonlinear Approaches*. Hoboken, N.J: Wiley-Interscience, 2006. ISBN: 978-0-471-70858-2.

[150] Jean-Jacques Slotine and Weiping Li. *Applied Nonlinear Control*. Engelwood Cliffs, NJ: Prentice Hall, 1991. ISBN: 0-13-040890-5.

[151] Gary Snethen. "XenoCollide: Complex Collision Made Simple". In: *undefined* (2008). (Visited on 06/24/2022).

[152] Bartolomeo Stellato, Goran Banjac, Paul Goulart, Alberto Bemporad, and Stephen Boyd. "OSQP: An Operator Splitting Solver for Quadratic Programs". In: (), p. 40.

[153] Robert F Stengel. *Optimal Control and Estimation*. Dover, 1994.

[154] Gilbert Strang. *Linear Algebra and Its Applications*. 4th ed. 1968.

[155] H. J. Terry Suh, Max Simchowitz, Kaiqing Zhang, and Russ Tedrake. *Do Differentiable Simulators Give Better Policy Gradients?* Aug. 2022. arXiv: 2202.00817 [cs]. (Visited on 04/02/2024).

[156] H. J.Terry Suh and Russ Tedrake. "The Surprising Effectiveness of Linear Models for Visual Foresight in Object Pile Manipulation". In: *Springer Proceedings in Advanced Robotics* 17 (Feb. 2020), pp. 347–363. DOI: 10.48550/arxiv.2002.09093.

[157] Sihao Sun, Angel Romero, Philipp Foehn, Elia Kaufmann, and Davide Scaramuzza. "A Comparative Study of Nonlinear MPC and Differential-Flatness-Based Control for Quadrotor Agile Flight". In: *IEEE Transactions on Robotics* (2022), pp. 1–17. ISSN: 1552-3098, 1941-0468. DOI: 10.1109/TRO.2022.3177279. (Visited on 09/06/2022).

[158] Balakumar Sundaralingam et al. *cuRobo: Parallelized Collision-Free Minimum-Jerk Robot Motion Generation*. Nov. 2023. arXiv: 2310.17274 [cs]. (Visited on 03/07/2024).

[159] Michael Szmuk, Taylor P. Reynolds, Behcet Acikmese, Mehran Mesbahi, and John M. Carson III. "Successive Convexification for 6-DoF Powered Descent Guidance with Compound State-Triggered Constraints". In: *arXiv:1901.02181 [math]* (Jan. 2019). arXiv: 1901.02181 [math]. (Visited on 04/22/2022).

[160] Michael Szmuk, Taylor P. Reynolds, and Behçet Açıkmeşe. "Successive Convexification for Real-Time Six-Degree-of-Freedom Powered Descent Guidance with State-Triggered Constraints". In: *Journal of Guidance, Control, and Dynamics* 43.8 (Aug. 2020), pp. 1399–1413. ISSN: 1533-3884. DOI: 10.2514/1.G004549. (Visited on 04/22/2022).

[161] Bingjie Tang, Michael A. Lin, Iretiayo Akinola, Ankur Handa, Gaurav S. Sukhatme, Fabio Ramos, Dieter Fox, and Yashraj Narang. *IndustReal: Transferring Contact-Rich Assembly Tasks from Simulation to Reality*. May 2023. DOI: 10.48550/arXiv.2305.17110. arXiv: 2305.17110 [cs]. (Visited on 11/23/2023).

[162] Russ Tedrake. *Underactuated Robotics: Algorithms for Walking, Running, Swimming, Flying, and Manipulation (Course Notes for MIT 6.832)*. Tech. rep. Downloaded in Fall, 2014 from http://people.csail.mit.edu/russt/underactuated/, 2014.

[163] Russ Tedrake and The Drake Development Team. "Drake: Model-based Design and Verification for Robotics". In: (2019).

[164] Erwan Thébault, Christopher C Finlay, and Ciarán D Beggan. "International Geomagnetic Reference Field: The 12th Generation". In: *Earth, Planets and Space* 67.1 (Dec. 2015). ISSN: 1880-5981. DOI: 10.1186/s40623-015-0228-9. (Visited on 07/31/2019).

[165] E. Todorov, T. Erez, and Y. Tassa. "MuJoCo: A Physics Engine for Model-Based Control". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Oct. 2012, pp. 5026–5033. DOI: 10.1109/IROS.2012.6386109.

[166] Emanuel Todorov, Tom Erez, and Yuval Tassa. "MuJoCo: A Physics Engine for Model-Based Control". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2012, pp. 5026–5033. DOI: 10.1109/IROS.2012.6386109.

[167] Kevin Tracy. *A Square-Root Kalman Filter Using Only QR Decompositions*. Aug. 2022. arXiv: 2208.06452 [cs, eess]. (Visited on 12/03/2022).

[168] Kevin Tracy, Giusy Falcone, and Zachary Manchester. "Robust Entry Guidance with Atmospheric Adaptation". In: *AIAA SciTech Forum and Exposition*. National Harbor, Maryland: AIAA, Jan. 2023.

[169] Kevin Tracy, Taylor A. Howell, and Zachary Manchester. *Differentiable Collision Detection for a Set of Convex Primitives*. July 2022. DOI: 10.48550/arXiv.2207.00669. arXiv: 2207.00669 [cs]. (Visited on 09/12/2022).

[170] Kevin Tracy, Taylor A. Howell, and Zachary Manchester. "Differentiable Collision Detection for a Set of Convex Primitives". In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. London, United Kingdom: IEEE, May 2023, pp. 3663–3670. DOI: 10.1109/ICRA48891.2023.10160716. (Visited on 03/08/2024).

[171] Kevin Tracy, Taylor A. Howell, and Zachary Manchester. "DiffPills: Differentiable Collision Detection for Capsules and Padded Polygons". In: *http://arxiv.org/abs/2207.00202* (July 2022). DOI: 10.48550/arXiv.2207.00202. arXiv: 2207.00202. (Visited on 09/12/2022).

[172] Kevin Tracy and Zachary Manchester. "Convex Quasi-Dynamic Simulation of Rigid Point Clouds with Torsional Friction". In: *IROS 2023 Workshop on Leveraging Models for Contact-Rich Manipulation*. Detroit, MI, USA: IEEE, Oct. 2023.

[173] Kevin Tracy and Zachary Manchester. "CPEG: A Convex Predictor-corrector Entry Guidance Algorithm". In: *2022 IEEE Aerospace Conference (AERO)*. Big Sky, MT, USA: IEEE, Mar. 2022, pp. 1–10. ISBN: 978-1-66543-760-8. DOI: 10.1109/AERO53065.2022.9843641. (Visited on 09/13/2022).

[174] Kevin Tracy and Zachary Manchester. "Low-Thrust Trajectory Optimization Using the Kustaanheimo-Stiefel Transformation". In: *AAS/AIAA Space Flight Mechanics Meeting*. Charlotte, NC, Feb. 2021.

[175] Kevin Tracy and Zachary Manchester. "Model-Predictive Attitude Control for Flexible Spacecraft During Thruster Firings". In: *AAS/AIAA Astrodynamics Specialist Conference*. Lake Tahoe, CA, Aug. 2020.

[176] Kevin Tracy and Zachary Manchester. *On the Differentiability of the Primal-Dual Interior-Point Method*. June 2024. arXiv: 2406.11749 [math]. (Visited on 11/11/2024).

[177] Kevin Tracy, Zachary Manchester, and Ewan Douglas. "Ultra-Fine Pointing for Nanosatellite Telescopes With Actuated Booms". In: *2022 IEEE Aerospace Conference (AERO)*. Big Sky, MT, USA: IEEE, Mar. 2022, pp. 1–8. ISBN: 978-1-66543-760-8. DOI: 10.1109/AERO53065.2022.9843485. (Visited on 09/13/2022).

[178] Kevin Tracy, Zachary Manchester, Ajinkya Jain, Keegan Go, Stefan Schaal, Tom Erez, and Yuval Tassa. *Efficient Online Learning of Contact Force Models for Connector Insertion*. Dec. 2023. arXiv: 2312.09190 [cs]. (Visited on 06/12/2024).

[179] Madeleine Udell, Karanveer Mohan, David Zeng, Jenny Hong, Steven Diamond, and Stephen Boyd. "Convex Optimization in Julia". In: *2014 First Workshop for High Performance Technical Computing in Dynamic Languages*. LA, USA: IEEE, Nov. 2014, pp. 18–28. ISBN: 978-1-4799-7020-9. DOI: 10.1109/HPTCDL.2014.5. (Visited on 10/15/2021).

[180] Gino Van Den Bergen. "Proximity Queries and Penetration Depth Computation on 3D GAme Objects". In: (2001).

[181] Gino Van Den Bergen. *Ray Casting against General Convex Objects with Application to Continuous Collision Detection*. Breda, Netherlands, June 2004.

[182] R. Van der Merwe and E.A. Wan. "The Square-Root Unscented Kalman Filter for State and Parameter-Estimation". In: vol. 6. IEEE, 2001, pp. 3461–3464. ISBN: 978-0-7803-7041-8. DOI: 10.1109/ICASSP.2001.940586. (Visited on 07/17/2018).

[183] L Vandenberghe. "The CVXOPT Linear and Quadratic Cone Program Solvers". In: (), p. 30.

[184] Patrick Varin and Scott Kuindersma. "A Constrained Kalman Filter for Rigid Body Systems with Frictional Contact". In: *Algorithmic Foundations of Robotics XIII*. Ed. by Marco Morales, Lydia Tapia, Gildardo Sánchez-Ante, and Seth Hutchinson. Vol. 14. Cham: Springer International Publishing, 2020, pp. 474–490. ISBN: 978-3-030-44050-3 978-3-030-44051-0. DOI: 10.1007/978-3-030-44051-0_28. (Visited on 11/28/2023).

[185] N. X. Vinh, A. Busemann, and R. D. Culp. "Hypersonic and Planetary Entry Flight Mechanics". In: *NASA STI/Recon Technical Report A* 81 (Jan. 1980), p. 16245. (Visited on 10/15/2021).

[186] Nguyen Vinh, Wyatt Johnson, and James Longuski. "Mars Aerocapture Using Bank Modulation". In: *Astrodynamics Specialist Conference*. Denver,CO,U.S.A.: American Institute of Aeronautics and Astronautics, Aug. 2000. DOI: 10.2514/6.2000-4424. (Visited on 09/07/2021).

[187] Andreas Wächter and Lorenz T. Biegler. "On the Implementation of an Interior-Point Filter Line-Search Algorithm for Large-Scale Nonlinear Programming". In: *Mathematical Programming* 106.1 (Mar. 2006), pp. 25–57. ISSN: 0025-5610, 1436-4646. DOI: 10.1007/s10107-004-0559-y. (Visited on 02/22/2019).

[188] Nolan Wagener, Ching-An Cheng, Jacob Sacks, and Byron Boots. *An Online Learning Approach to Model Predictive Control*. Oct. 2019. DOI: 10.48550/arXiv.1902.08967. arXiv: 1902.08967 [cs]. (Visited on 04/23/2023).

[189] Zhenbo Wang and Michael J. Grant. "Constrained Trajectory Optimization for Planetary Entry via Sequential Convex Programming". In: *AIAA Atmospheric*

Flight Mechanics Conference. Washington, D.C.: American Institute of Aeronautics and Astronautics, June 2016. ISBN: 978-1-62410-430-5. DOI: 10.2514/6.2016-3241. (Visited on 04/02/2021).

[190] Zhenbo Wang and Michael J. Grant. "Improved Sequential Convex Programming Algorithms for Entry Trajectory Optimization". In: *AIAA Scitech 2019 Forum*. San Diego, California: American Institute of Aeronautics and Astronautics, Jan. 2019. ISBN: 978-1-62410-578-4. DOI: 10.2514/6.2019-0667. (Visited on 04/21/2021).

[191] Zhenbo Wang and Michael J. Grant. "Near-Optimal Entry Guidance for Reference Trajectory Tracking via Convex Optimization". In: *2018 AIAA Atmospheric Flight Mechanics Conference*. Kissimmee, Florida: American Institute of Aeronautics and Astronautics, Jan. 2018. ISBN: 978-1-62410-525-8. DOI: 10.2514/6.2018-0013. (Visited on 04/02/2021).

[192] Zhenbo Wang and Michael J. Grant. "Near-Optimal Entry Guidance for Reference Trajectory Tracking via Convex Optimization". In: American Institute of Aeronautics and Astronautics, Jan. 2018. ISBN: 978-1-62410-525-8. DOI: 10.2514/6.2018-0013. (Visited on 02/21/2018).

[193] David W. Way, Richard W. Powell, Allen Chen, Adam D. Steltzner, A. Miguel San Martin, P. Daniel Burkhart, and Gavin F. Mendeck. "Mars Science Laboratory: Entry, Descent, and Landing System Performance". In: *2007 IEEE Aerospace Conference*. Big Sky, MT, USA: IEEE, 2007, pp. 1–19. ISBN: 978-1-4244-0524-4. DOI: 10.1109/AERO.2007.352821. (Visited on 05/09/2024).

[194] James R. Wertz, ed. *Spacecraft Attitude Determination and Control*. Vol. 73. Astrophysics and Space Science Library. Dordrecht: Springer Netherlands, 1978. ISBN: 978-90-277-1204-2. DOI: 10.1007/978-94-009-9907-7. (Visited on 05/29/2020).

[195] Grady Williams, Paul Drews, Brian Goldfain, James M. Rehg, and Evangelos A. Theodorou. "Aggressive Driving with Model Predictive Path Integral Control". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. May 2016, pp. 1433–1440. DOI: 10.1109/ICRA.2016.7487277.

[196] David Wilson, James H. Davenport, Matthew England, and Russell Bradford. "A "Piano Movers" Problem Reformulated". In: *2013 15th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*. Sept. 2013, pp. 53–60. DOI: 10.1109/SYNASC.2013.14. arXiv: 1309.1588 [cs]. (Visited on 09/06/2022).

[197] X Xinjilefu, Siyuan Feng, and Christopher G. Atkeson. "Dynamic State Estimation Using Quadratic Programming". In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Chicago, IL, USA: IEEE, Sept. 2014, pp. 989–994. ISBN: 978-1-4799-6934-0. DOI: 10.1109/IROS.2014.6942679. (Visited on 11/28/2023).

[198] Xinyu Zhang, Minkyoung Lee, and Young J. Kim. "Interactive Continuous Collision Detection for Non-Convex Polyhedra". In: *The Visual Computer* 22.9-11 (Sept. 2006), pp. 749–760. ISSN: 0178-2789, 1432-2315. DOI: 10.1007/s00371-006-0060-0. (Visited on 03/13/2024).

[199] Tony Z. Zhao, Jianlan Luo, Oleg Sushkov, Rugile Pevceviciute, Nicolas Heess, Jon Scholz, Stefan Schaal, and Sergey Levine. *Offline Meta-Reinforcement Learning for*

Industrial Insertion. Sept. 2022. DOI: 10.48550/arXiv.2110.04276. arXiv: 2110.04276 [cs]. (Visited on 11/23/2023).

[200] Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. "Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: A Survey". In: *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. Dec. 2020, pp. 737–744. DOI: 10.1109/SSCI47803.2020.9308468.

[201] Simon Zimmermann, Matthias Busenhart, Simon Huber, Roi Poranne, and Stelian Coros. *Differentiable Collision Avoidance Using Collision Primitives*. Apr. 2022. arXiv: 2204.09352 [cs]. (Visited on 03/04/2023).