Learning Convex Optimization Control Policies

Akshay Agrawal AKSHAYKA@CS.STANFORD.EDU
Shane Barratt SBARRATT@STANFORD.EDU
Stephen Boyd BOYD@STANFORD.EDU
450 Serra Mall, Stanford, CA, 94305

Bartolomeo Stellato* STELLATO@MIT.EDU
77 Massachusetts Ave, Cambridge, MA, 02139

Editors: A. Bayen, A. Jadbabaie, G. J. Pappas, P. Parrilo, B. Recht, C. Tomlin, M. Zeilinger

Abstract

Many control policies used in applications compute the input or action by solving a convex optimization problem that depends on the current state and some parameters. Common examples of such convex optimization control policies (COCPs) include the linear quadratic regulator (LQR), convex model predictive control (MPC), and convex approximate dynamic programming (ADP) policies. These types of control policies are tuned by varying the parameters in the optimization problem, such as the LQR weights, to obtain good performance, judged by application-specific metrics. Tuning is often done by hand, or by simple methods such as a grid search. In this paper we propose a method to automate this process, by adjusting the parameters using an approximate gradient of the performance metric with respect to the parameters. Our method relies on recently developed methods that can efficiently evaluate the derivative of the solution of a convex program with respect to its parameters. A longer version of this paper, which illustrates our method on many examples, is available at https://web.stanford.edu/~boyd/papers/learning_cocps.html.
Keywords: Stochastic control, convex optimization, approximate dynamic programming

1. Introduction

We consider the control of a stochastic dynamical system with known dynamics, using a control policy that determines the input or action by solving a convex optimization problem. We call such policies *convex optimization control policies* (COCPs). Many practical policies have this form, including the first modern control policy, the linear quadratic regulator (LQR) (Kalman, 1960). In LQR, the optimization problem has quadratic objective and linear equality constraints, and so can be solved explicitly, yielding a linear policy. More modern examples, relying on more general convex optimization problems such as quadratic programs (QPs), include convex model predictive control (MPC) (Borrelli et al., 2017) and convex approximate dynamic programming (ADP) (Bertsekas, 2017). These policies are used in many applications, including robotics (Kuindersma et al., 2014), vehicle and spacecraft control (Stewart and Borrelli, 2008; Blackmore, 2016), supply chains (Powell et al., 2012), and finance (Markowitz, 1952; Cornuejols and Tütüncü, 2006; Boyd et al., 2017).

Control policies are judged by application-specific metrics, evaluated in simulation with historical or synthetic values of the unknown quantities. In some but not all cases, the metrics have the traditional form of the average value of a given stage cost. In a few cases, the optimal policy for a

* Authors listed in alphabetical order.

traditional stochastic control problem has COCP form. A well-known case is LQR; another is when the dynamics are affine and the stage cost is convex, in which case the Bellman value function is convex, and evaluating the optimal policy reduces to solving a convex optimization problem (Keshavarz, 2012, §3.3.1). Unfortunately, we cannot in general express the value function in a tractable form. In a far wider set of cases, a COCP policy is not optimal, but only a good, practical heuristic.

COCPs have some attractive properties compared to other parametrized control policies. For one, COCPs can be evaluated quickly enough for real-time applications, thanks to fast embedded solvers (Domahidi et al., 2013; Stellato et al., 2020; Wang and Boyd, 2010) and code generators (Mattingley and Boyd, 2012; Chu et al., 2013; Banjac et al., 2017). Additionally, when the convex problem to be solved is well chosen, the policy is reasonable for any choice of the parameter values over the allowed set. As a specific example, consider a linear control policy parametrized by the gain matrix, seemingly the most natural parametrization of a linear policy. The set of gain matrices that lead to a stable closed-loop system (a minimal requirement) can be very complex, even disconnected. In contrast, consider an LQR control policy parametrized by a positive definite control cost matrix. In this case any choice of policy yields a stable closed-loop system. It is far easier and safer to tune parameters when any feasible choice leads to a reasonable policy.

All control policies are tuned by adjusting parameters that appear in them. In the case of COCPs, the parameters are in the optimization problem that is solved to evaluate the policy. The tuning is often done by hand, or by a grid search. In this paper we present an automated method for tuning parameters in COCPs to achieve good values of a performance metric. Our method simulates the system with the policy in the loop, and computes a stochastic gradient of the expected performance with respect to the parameters. It then updates the parameters via a projected stochastic gradient method. Central to our method is the fact that the solution map for convex programs is often differentiable, and its derivative can be efficiently computed (Amos, 2019; Agrawal et al., 2019a).

Our method is not guaranteed to find the best parameter values, since the performance metric is not a convex function of the COCP parameter, and we use a local search method. This is not a problem in practice, since in a typical use case, the COCP is initialized with reasonable parameters, and our method is used to tune these parameters to improve the performance (sometimes considerably).

1.1. Related work

Dynamic programming. The Markov decision process is a stochastic control problem that can be solved in principle using dynamic programming (DP) (Bellman, 1957a,b; Bertsekas, 2017). The optimal policy is evaluated by solving an optimization problem, one that includes a current stage cost and the expected value of cost-to-go at the next state. This optimization problem corresponds to a COCP when the system dynamics are affine and the stage cost is convex (Bertsekas, 2017). Unfortunately, the value function can be found in a tractable form in only a few cases, *e.g.*, when the cost is a convex extended quadratic and the dynamics are affine (Barratt and Boyd, 2018).

Approximate dynamic programming. ADP (Bertsekas and Tsitsiklis, 1996; Powell, 2007) refers to heuristic solution methods for stochastic control problems that replace the value function in DP with an approximation, or search over a parametric family of policies (Bertsekas, 2019, §2.1).

In many ADP methods, an optimization problem is solved offline to approximate the value function. When there is a finite number of states and inputs, the problem can be expressed as a linear program (De Farias and Van Roy, 2003). When the dynamics are linear, the cost is quadratic, and the input lies in a convex set, an approximate convex quadratic value function can be found by solv-

ing a particular semidefinite program (Wang and Boyd, 2009), and optionally iterating the Bellman inequality (Wang et al., 2014; Stellato et al., 2017). The resulting policy is a COCP. Other methods iteratively adjust the approximate value function to satisfy the Bellman equation. Examples include projected value iteration or fitted Q-iteration (Gordon, 1995), temporal difference learning (Sutton, 1988; Bertsekas et al., 2004), and approximate policy iteration (Nedić and Bertsekas, 2003). Examples of COCPs here include the use of quadratic approximate cost-to-go functions for input-affine systems with convex cost (Keshavarz and Boyd, 2014), and modeling the state-action cost-to-go function as an input-convex neural network (Amos et al., 2017, §6.4).

Still other ADP methods parametrize the policy and tune the parameters directly to improve performance (Bertsekas, 2019, §5.7). The most common method is stochastic gradient search (Powell, 2007, §7.2); this is the method we employ in this paper, using a parametrized COCP as the policy.

Reinforcement learning. Reinforcement learning (Sutton and Barto, 2018) is similar to ADP (Bertsekas, 2019, §1.4), but emphasizes problems in which mathematical models of the dynamics or cost are absent. Instead, one has a computational simulator for both. Our method might be used in this setting after learning suitable models. COCPs could also be used as part of the policy in policy gradient or actor-critic methods (Williams, 1987; Lillicrap et al., 2015; Schulman et al., 2017; Fazel et al., 2018).

Learning optimization-based policies. Other work has considered tuning optimization-based control policies. For example, there is prior work on learning for MPC, including nonconvex MPC (Amos et al., 2018), cost shaping (Tamar et al., 2017), differentiable path integral control (Okada et al., 2017), and identification of terminal constraint sets and costs (Rosolia and Borrelli, 2017). To our knowledge, our work is the first to consider the specific class of parametrized convex programs.

1.2. Outline

In §2, we introduce the controller tuning problem that we wish to solve. In §3, we describe some common forms of COCPs. In §4, we propose a heuristic for the controller tuning problem. In §5, we apply our methodology to LQR problems and a problem in vehicle control. A longer version of this paper, with additional discussion and examples, is available online (Agrawal et al., 2019c).

2. Controller tuning problem

We consider a dynamical system with dynamics given by

$$x_{t+1} = f(x_t, u_t, w_t), \quad t = 0, 1, \ldots. \tag{1}$$

At time period t, $x_t \in \mathbf{R}^n$ is the state, $u_t \in \mathbf{R}^m$ is the input or action, $w_t \in \mathcal{W}$ is the disturbance, and $f : \mathbf{R}^n \times \mathbf{R}^m \times \mathcal{W} \to \mathbf{R}^n$ is the state transition function. The initial state x_0 and the disturbances w_t are random variables.

The inputs are given by a state feedback control policy $\phi : \mathbf{R}^n \to \mathbf{R}^m$, *i.e.*,

$$u_t = \phi(x_t), \quad t = 0, 1, \ldots. \tag{2}$$

We specifically consider COCPs, which have the form

$$\begin{array}{rcll} \phi(x) & = & \underset{u}{\text{argmin}} & f_0(x, u; \theta) \\ & & \text{subject to} & f_i(x, u; \theta) \leq 0, \quad i = 1, \ldots, k, \\ & & & g_i(x, u; \theta) = 0, \quad i = 1, \ldots, \ell, \end{array} \tag{3}$$

where f_i are convex in u and g_i are affine in u. To evaluate a COCP we must solve a convex optimization problem, which we assume has a unique solution. The convex optimization problem (3) is given by a *parametrized problem description* (Boyd and Vandenberghe, 2004, §4.1.4), in which the vector $\theta \in \Theta \subseteq \mathbf{R}^p$ is the parameter (Θ is the set of allowable parameter values). The problem we address in this paper is the choice of the parameter θ.

Performance metric. We judge the performance of a control policy, or choice of parameter θ, by the average value of a cost over trajectories of length T. The horizon T is chosen large enough so that the average over T time steps is close enough to the long term average. We denote the trajectories over $t = 0, \ldots, T$ as

$$\begin{array}{rcl} X & = & (x_0, x_1, \ldots, x_T) \in \mathbf{R}^N, \\ U & = & (u_0, u_1, \ldots, u_T) \in \mathbf{R}^M, \\ W & = & (w_0, w_1, \ldots, w_T) \in \mathcal{W}^{T+1}, \end{array}$$

where $N = (T+1)n$ and $M = (T+1)m$. These state, input, and disturbance trajectories are random variables, with distributions that depend on the parameter θ.

The cost is provided by a function $\psi : \mathbf{R}^N \times \mathbf{R}^M \times \mathcal{W}^{T+1} \to \mathbf{R} \cup \{+\infty\}$. Infinite values of ψ place constraints on the trajectories. Traditionally, the cost function is separable, with the form

$$\psi(X, U, W) = \frac{1}{T+1} \sum_{t=0}^{T} g(x_t, u_t, w_t), \tag{4}$$

where $g : \mathbf{R}^n \times \mathbf{R}^m \times \mathcal{W} \to \mathbf{R} \cup \{\infty\}$ is a stage cost function. We do not require this.

A policy is judged by the expected value of the cost,

$$J(\theta) = \mathbf{E}\, \psi(X, U, W).$$

Note that J depends on θ, since $x_1, \ldots, x_T$ and $u_0, \ldots, u_T$ depend on θ.

We generally cannot evaluate $J(\theta)$ exactly. Instead, if we can sample the initial state and the disturbances, we can compute an unbiased Monte Carlo approximation of it. In the simplest version, we generate K independent trajectories $(X^1, U^1, W^1), \ldots, (X^K, U^K, W^K)$ and compute

$$\hat{J}(\theta) = \frac{1}{K} \sum_{i=1}^{K} \psi(X^i, U^i, W^i).$$

This computation requires solving $K(T+1)$ convex programs.

Controller tuning problem. The controller tuning problem has the form

$$\begin{array}{ll} \text{minimize} & J(\theta) \\ \text{subject to} & \theta \in \Theta, \end{array} \tag{5}$$

with variable θ. This is the problem we seek to solve in this paper.

3. Examples of COCPs

In this section we describe some paradigmatic COCPs.

Optimal (DP) policy. In the traditional stochastic control setting, the cost resembles (4) and $x_0, w_0, w_1, \ldots$ are independent. Under some technical conditions, the optimal policy for $T \to \infty$, *i.e.*, the policy minimizing J over all possible state feedback policies, not just COCPs, has the form

$$\phi(x) = \operatorname*{argmin}_u \mathbf{E}\left(g(x,u,w) + V(f(x,u,w))\right), \tag{6}$$

where $V : \mathbf{R}^n \to \mathbf{R}$ is the Bellman value function. When f is affine in (x,u) and g is convex in (x,u), V is convex and the optimal policy is a COCP (but usually cannot be evaluated tractably).

Approximate dynamic programming policy. An ADP (Powell, 2007) or control-Lyapunov (Corless and Leitmann, 1988) policy has the form

$$\phi(x) = \operatorname*{argmin}_u \mathbf{E}(g(x,u,w) + \hat{V}(f(x,u,w))), \tag{7}$$

where $\hat{V}$ is an approximate value function for which the minimization over u is tractable. When g is convex in u, f is affine in u, and $\hat{V}$ is convex, the minimization is a convex optimization problem (Boyd and Vandenberghe, 2004), and this policy has COCP form (Keshavarz, 2012).

Model predictive control policy. In MPC, the cost has the form (4), and the policy solves an approximation to the control problem over a short horizon H (Rawlings and Mayne, 2009), applying only the first input. A terminal cost function g_H is often included. The policy has the form

$$\begin{array}{lll} \phi(x) = & \operatorname*{argmin}_{u_0} & \sum_{t=0}^{H-1} g(x_t, u_t, \hat{w}_t) + g_H(x_H) \\ & \text{subject to} & x_{t+1} = f(x_t, u_t, \hat{w}_t), \quad t = 0, \ldots, H-1, \\ & & x_0 = x, \end{array}$$

where $\hat{w}_0, \ldots, \hat{w}_{H-1}$ are predictions of the disturbances and the variables are $u_0, \ldots, u_{H-1}$ and $x_0, \ldots, x_H$. When f is affine in (x,u), g is convex in (x,u), and g_H is convex, this is a COCP.

4. Solution method

Solving the controller tuning problem (5) exactly is in general hard, so we will solve it approximately. Historically, many practitioners have used derivative-free methods to tune parameters in control policies. Some of these methods include evolutionary strategies (Hansen and Ostermeier, 2001; Salimans et al., 2017), Bayesian optimization (Močkus, 1975), grid search, and random search (Anderson, 1953; Solis and Wets, 1981; Bergstra and Bengio, 2012). These methods can certainly yield improvements over an initialization, but they often converge very slowly.

It is well-known that first-order optimization methods, which make use of derivatives, can outperform derivative-free methods. In this paper, we apply the projected stochastic (sub)gradient method (Robbins and Monro, 1951) to approximately solve (5). That is, starting with initial parameters θ^0, at iteration k, we simulate the system and compute $\hat{J}(\theta^k)$. We then compute an unbiased stochastic gradient of J, $g^k = \nabla \hat{J}(\theta^k)$, by the chain rule, and update $\theta^{k+1} = \Pi_\Theta(\theta^k - \alpha^k g^k)$, where $\Pi_\Theta(\theta)$ denotes the Euclidean projection of θ onto Θ and $\alpha^k > 0$ is a step size.

Computing g^k requires differentiating through the dynamics f, the cost ψ, and the solution map ϕ of a convex program. Methods for differentiating through special subclasses of convex optimization have existed for many decades; for example, literature on differentiating through QPs

dates back to at least the 1960s (Boot, 1963). Similarly, it is well known that if the objective and constraints of a convex optimization problem are all smooth, and some regularity conditions are satisfied, then its derivative can be computed by differentiating the KKT optimality conditions (Kuhn and Tucker, 1951; Barratt, 2018). Until very recently, however, it was not generically possible to differentiate through a convex optimization problem with nondifferentiable objective or constraints; recent work (Busseti et al., 2019; Agrawal et al., 2019b,a; Amos, 2019) has shown how to efficiently compute this derivative, by implicitly differentiating optimality conditions for a cone program.

Until this point, we have assumed the differentiability of all of the functions involved (f, ψ, and ϕ). In real applications, this assumption might not hold. So long as the functions are differentiable almost everywhere, however, it is reasonable to speak of applying a projected stochastic gradient method to (5). At non-differentiable points, we compute a heuristic quantity. For example, at some non-differentiable points of ϕ, a certain matrix fails to be invertible, and we compute a least-squares approximation of the derivative instead, as in (Agrawal et al., 2019b). In this sense, we overload the notation $\nabla f(x)$ to denote a gradient when f is differentiable at x, or some heuristic quantity (a "gradient") when f is not differentiable at x. In practice, as our examples in §5 demonstrate, we find that this method works well. Indeed, most modern neural networks are not differentiable, but it is possible to successfully train them using stochastic "gradient" descent (Goodfellow et al., 2016).

5. Examples

Here, we illustrate our method with examples. Our COCPs were implemented using CVXPY (Diamond and Boyd, 2016; Agrawal et al., 2018), and we use SCS (O'Donoghue et al., 2016) to solve the convex programs and cvxpylayers (Agrawal et al., 2019a) and PyTorch (Paszke et al., 2019) to differentiate through them. Evaluating each COCP takes about 3 ms, which could be made smaller using a more specialized solver. Our code is available at `https://github.com/cvxgrp/cocp`.

5.1. LQR

We first apply our method to the classical LQR problem, with dynamics and cost

$$f(x,u,w) = Ax + Bu + w, \quad \psi(X,U,W) = \frac{1}{T+1}\sum_{t=0}^{T} x_t^T Q x_t + u_t^T R u_t,$$

where $A \in \mathbf{R}^{n\times n}$, $B \in \mathbf{R}^{n\times m}$, $Q \in \mathbf{S}_+^n$, $R \in \mathbf{S}_{++}^m$, and $w \sim \mathcal{N}(0,\Sigma)$.

We use the COCP

$$\phi(x) = \operatorname*{argmin}_u \; u^T R u + \|\theta(Ax + Bu)\|_2^2, \tag{8}$$

with parameter $\theta \in \mathbf{R}^{n\times n}$. This policy has an analytical solution, which is linear. (This COCP is clearly over-parametrized.) If the matrix $\theta^T\theta$ satisfies a particular algebraic Riccati equation, then (8) is optimal (over all control policies) when $T \to \infty$ (Bertsekas, 2017).

Numerical example. We consider a numerical example with $n = 4$ states, $m = 2$ inputs, and $T = 100$. The entries of A and B were sampled from $\mathcal{N}(0,1)$, with A scaled such that its spectral radius is one. The cost matrices are $Q = I$ and $R = I$, and the noise covariance is $W = (0.25)I$. We initialize θ with the identity. We trained our policy for 50 iterations, using $K = 6$ simulations per step, with an initial step size of 0.5 that was decreased to 0.1 after 25 iterations. Figure 1 plots

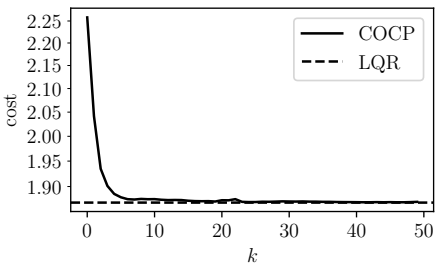

Figure 1: LQR.

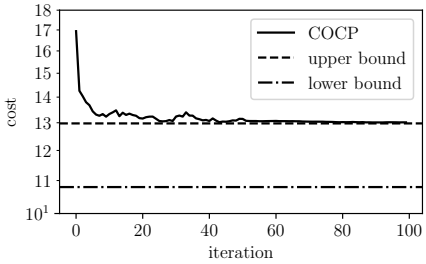

Figure 2: Box-constrained LQR.

the cost of the COCP during learning on a held-out random seed and the cost of the optimal LQR policy (for $T \to \infty$). Our method appears to converge to near the optimal cost in just 10 iterations.

5.2. Box-constrained LQR

A box-constrained LQR problem has the same dynamics as LQR, with cost

$$\psi(X,U,W) = \frac{1}{T+1}\sum_{t=0}^{T} g(x_t,u_t,w_t), \quad g(x_t,u_t,w_t) = \begin{cases} x_t^T Q x_t + u_t^T R u_t, & \|u_t\|_\infty \le u_{\max}, \\ +\infty & \text{otherwise.} \end{cases}$$

Unlike LQR, in general, there is no known solution to this problem, analytical or otherwise.-

Our COCP is an ADP policy (7) with a quadratic value function and parameter $\theta \in \mathbf{R}^{n\times n}$:

$$\begin{array}{lcll} \phi(x) & = & \underset{u}{\text{argmin}} & u^T R u + \|\theta(Ax+Bu)\|_2^2 \\ & & \text{subject to} & \|u\|_\infty \le u_{\max}. \end{array}$$

A non-trivial lower bound on the optimal cost yields a policy of this form (Wang and Boyd, 2009).

Numerical example. We use $n = 8$, $m = 2$, $T = 100$, $u_{\max} = 0.1$, and data generated as in the LQR example above. The technique from (Wang and Boyd, 2009) yields a lower bound on the optimal cost of around 11. It also suggests a value of θ that gives average cost around 13, an upper bound on the optimal cost. We initialize our COCP with $\theta = P^{1/2}$, where P comes from the cost-to-go function for the unconstrained problem. Figure 2 plots the expected cost of our COCP, and the upper and lower bounds. Our method converges to roughly the same cost as the upper bound.

5.3. Tuning a vehicle controller to track curved paths

We consider a vehicle moving relative to a smooth path, with state and input

$$x_t = (e_t, \Delta\psi_t, v_t, v_t^{\text{des}}, \kappa_t), \quad u_t = (a_t, z_t).$$

At time t, e_t is the lateral path deviation (m), $\Delta\psi_t$ is the heading deviation from the path (rad), v_t is the velocity (m/s), v_t^{des} is the desired velocity (m/s), κ_t is the current curvature (*i.e.*, inverse radius) of the path (1/m), a_t is the acceleration (m/s^2), and $z_t := \tan(\delta_t) - L\kappa_t$, where δ_t is the wheel angle (rad) and L is the vehicle's wheelbase (m).

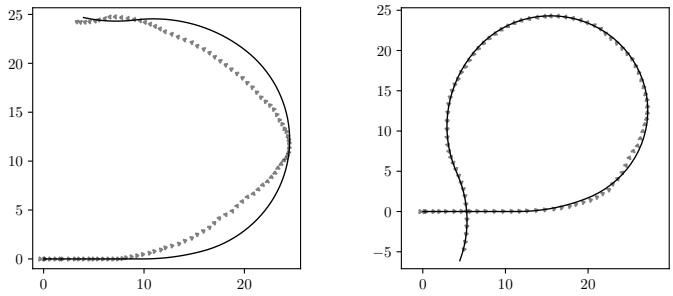

Figure 3: Tracking curved paths. Left: untuned policy; right: tuned policy. Black line is the path, and triangles are the vehicle position and orientation.

For the dynamics, we use a kinematic bicycle model in path coordinates (Gerdes, 2019), discretized at $h = 0.2\,\mathrm{s}$, with random processes for v_t^{des} and κ_t. The dynamics are

$$\begin{aligned}
e_{t+1} &= e_t + hv_t \sin(\Delta\psi_t) + w_1, \quad w_1 \sim \mathcal{N}(0, .01), \\
\Delta\psi_{t+1} &= \Delta\psi_t + hv_t \left(\kappa_t + \frac{z_t}{L} - \frac{\kappa_t}{1 - e_t\kappa_t}\cos(\Delta\psi_t)\right) + w_2, \quad w_2 \sim \mathcal{N}(0, .0001), \\
v_{t+1} &= v_t + ha_t + w_3, \quad w_3 \sim \mathcal{N}(0, .01), \\
v_{t+1}^{\mathrm{des}} &= v_t^{\mathrm{des}} w_4 + w_5(1 - w_4), \quad w_4 \sim \mathrm{Bernoulli}(0.98), \quad w_5 \sim \mathcal{U}(3, 6), \\
\kappa_{t+1} &= \kappa_t w_6 + w_7(1 - w_6), \quad w_6 \sim \mathrm{Bernoulli}(0.95), \quad w_7 \sim \mathcal{N}(0, .01).
\end{aligned}$$

Our goal is to travel the desired speed ($v_t \approx v_t^{\mathrm{des}}$), while tracking the path ($e_t \approx 0$, $\Delta\psi \approx 0$) and expending minimal control effort ($a_t \approx 0$, $z_t \approx 0$). We consider the cost

$$\psi(X, U, W) = \frac{1}{T+1}\sum_{t=0}^{T}(v_t - v_t^{\mathrm{des}})^2 + \lambda_1 e_t^2 + \lambda_2 \Delta\psi_t^2 + \lambda_3|a_t| + \lambda_4 z_t^2 + I(a_t, z_t, \kappa_t),$$

for positive $\lambda_1, \ldots, \lambda_4$. Here, $I(a, z, \kappa)$ is 0 when $|a| \leq a_{\max}$ and $|z + L\kappa| \leq \tan(\delta_{\max})$, for a maximum absolute acceleration $a_{\max}$ and maximum absolute wheel angle $\delta_{\max}$, and $+\infty$ otherwise.

We consider a COCP that computes (a_t, z_t) as

$$\begin{aligned}
\phi(x_t) = \underset{a,z}{\mathrm{argmin}} \quad & \lambda_3|a| + \lambda_4 z^2 + \|Sy\|_2^2 + q^T y \\
\text{subject to} \quad & y = \begin{bmatrix} e_t + hv_t \sin(\Delta\psi_t) \\ \Delta\psi_t + hv_t\left(\kappa_t + \frac{z}{L} - \frac{\kappa_t}{1 - e_t\kappa_t}\cos(\Delta\psi_t)\right) \\ v_t + ha - (0.98)v_t^{\mathrm{des}} - (0.02)4.5 \\ y_1 + hv_t \sin(y_2 - hv_t\frac{z}{L}) + \frac{h^2 v_t^2}{L} z \end{bmatrix} \\
& |a| \leq a_{\max} \\
& |z + L\kappa_t| \leq \tan(\delta_{\max}),
\end{aligned}$$

with parameters $\theta = (S, q)$, where $S \in \mathbf{R}^{4\times 4}$ and $q \in \mathbf{R}^4$. The variable $y \in \mathbf{R}^4$ represents portions of the next state, since $y_1 = e_{t+1}$, $y_2 = \Delta\psi_{t+1}$, $y_3 = v_{t+1} - \mathbf{E}[v_{t+1}^{\mathrm{des}}]$, and $y_4 \approx e_{t+2}$ (since it assumes $a_t = 0$). This is an ADP policy, with approximate value function $\hat{V}(x) = \|Sx\|_2^2 + q^T x$.

Numerical example. We apply our method to a numerical example with $L = 2.8\,\mathrm{m}$, $\lambda_1 = \lambda_2 = 1$, $\lambda_3 = \lambda_4 = 10$, $a_{\max} = 2\,\mathrm{m/s^2}$, $\delta_{\max} = 0.6\,\mathrm{rad}$, and $T = 100$, with initial state $x_0 = (.5, .1, 3, 4.5, 0)$. We run the stochastic gradient method for 100 iterations using $K = 6$ simulations and a step size of 0.1. We initialize the parameters with $S = I$ and $q = 0$. Tuning decreased the cost from 3.978 to 0.971. Figure 3 plots untuned and tuned sample paths on a held-out instance.

Acknowledgements

Shane Barratt is supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-1656518. Akshay Agrawal is supported by a Stanford Graduate Fellowship.

References

Akshay Agrawal, Robin Verschueren, Steven Diamond, and Stephen Boyd. A rewriting system for convex optimization problems. *Journal of Control and Decision*, 5(1):42–60, 2018.

Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J Zico Kolter. Differentiable convex optimization layers. In *Advances in Neural Information Processing Systems*, pages 9558–9570, 2019a.

Akshay Agrawal, Shane Barratt, Stephen Boyd, Enzo Busseti, and Walaa Moursi. Differentiating through a cone program. *Journal of Applied and Numerical Optimization*, 1(2):107–115, 2019b.

Akshay Agrawal, Shane Barratt, Boyd Stephen, and Stellato Bartolomeo. Learning convex optimization control policies. http://web.stanford.edu/~boyd/papers/learning_cocps.html, 2019c.

Brandon Amos. *Differentiable optimization-based modeling for machine learning*. PhD thesis, Carnegie Mellon University, 2019.

Brandon Amos, Lei Xu, and J Zico Kolter. Input convex neural networks. In *Proc. Intl. Conf. on Machine Learning*, pages 146–155, 2017.

Brandon Amos, Ivan Jimenez, Jacob Sacks, Byron Boots, and J Zico Kolter. Differentiable MPC for end-to-end planning and control. In *Advances in Neural Information Processing Systems*, pages 8299–8310, 2018.

Richard Anderson. Recent advances in finding best operating conditions. *Journal of the American Statistical Association*, 48(264):789–798, 1953.

Goran Banjac, Bartolomeo Stellato, Nicholas Moehle, Paul Goulart, Alberto Bemporad, and Stephen Boyd. Embedded code generation using the OSQP solver. In *IEEE Conference on Decision and Control*, 2017.

Shane Barratt. On the differentiability of the solution to convex optimization problems. *arXiv preprint arXiv:1804.05098*, 2018.

Shane Barratt and Stephen Boyd. Stochastic control with affine dynamics and extended quadratic costs. *arXiv preprint arXiv:1811.00168*, 2018.

Richard Bellman. *Dynamic Programming*. Princeton University Press, 1957a.

Richard Bellman. A Markovian decision process. *Journal of Mathematics and Mechanics*, pages 679–684, 1957b.

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012.

Dimitri Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, 4th edition, 2017.

Dimitri Bertsekas. *Reinforcement Learning and Optimal Control*. Athena Scientific, 1st edition, 2019.

Dimitri Bertsekas and John Tsitsiklis. *Neuro-dynamic Programming*, volume 5. Athena Scientific, 1996.

Dimitri Bertsekas, Vivek Borkar, and Angelia Nedić. Improved temporal difference methods with linear function approximation. *Learning and Approximate Dynamic Programming*, pages 231–255, 2004.

Lars Blackmore. Autonomous precision landing of space rockets. *The BRIDGE*, 26(4), 2016.

John Boot. On sensitivity analysis in convex quadratic programming problems. *Operations Research*, 11(5):771–786, 1963.

Francesco Borrelli, Alberto Bemporad, and Manfred Morari. *Predictive Control for Linear and Hybrid Systems*. Cambridge University Press, 2017.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

Stephen Boyd, Enzo Busseti, Steven Diamond, Ronald Kahn, Kwangmoo Koh, Peter Nystrup, and Jan Speth. Multi-period trading via convex optimization. *Foundations and Trends® in Optimization*, 3(1):1–76, 2017.

Enzo Busseti, Walaa Moursi, and Stephen Boyd. Solution refinement at regular points of conic problems. *Computational Optimization and Applications*, 74:627–643, 2019.

Eric Chu, Neal Parikh, Alexander Domahidi, and Stephen Boyd. Code generation for embedded second-order cone programming. In *European Control Conference*, pages 1547–1552. IEEE, 2013.

Martin Corless and George Leitmann. Controller design for uncertain systems via Lyapunov functions. In *American Control Conference*, pages 2019–2025. IEEE, 1988.

Gerard Cornuejols and Reha Tütüncü. *Optimization Methods in Finance*. Cambridge University Press, 2006.

Daniela De Farias and Benjamin Van Roy. The linear programming approach to approximate dynamic programming. *Operations Research*, 51(6):850–865, 2003.

Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

Alexander Domahidi, Eric Chu, and Stephen Boyd. ECOS: An SOCP solver for embedded systems. In *European Control Conference*, pages 3071–3076. IEEE, 2013.

Maryam Fazel, Rong Ge, Sham Kakade, and Merhan Mesbahi. Global convergence of policy gradient methods for the linear quadratic regulator. In *Proc. Intl. Conf. on Machine Learning*, 2018.

Christian Gerdes. ME 227 vehicle dynamics and control course notes, 2019. Lectures 1 and 2.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

Geoffrey Gordon. Stable function approximation in dynamic programming. In *Machine Learning*, pages 261–268. Elsevier, 1995.

Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation*, 9(2):159–195, 2001.

Rudolf Kalman. Contributions to the theory of optimal control. *Boletin de la Sociedad Matematica Mexicana*, 5(2):102–119, 1960.

Arezou Keshavarz. *Convex methods for approximate dynamic programming*. PhD thesis, Stanford University, 2012.

Arezou Keshavarz and Stephen Boyd. Quadratic approximate dynamic programming for input-affine systems. *Intl. Journal of Robust and Nonlinear Control*, 24(3):432–449, 2014.

Harold Kuhn and Albert Tucker. Nonlinear programming. In *Berkeley Symposium on Mathematical Statistics and Probability, 1950*, pages 481–492. University of California Press, 1951.

Scott Kuindersma, Frank Permenter, and Russ Tedrake. An efficiently solvable quadratic program for stabilizing dynamic locomotion. In *Proc. Intl. on Robotics and Automation (ICRA)*, page 2589–2594. IEEE, 2014.

Timothy Lillicrap, Jonathan Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Harry Markowitz. Portfolio selection. *The Journal of Finance*, 7(1):77–91, 1952.

Jacob Mattingley and Stephen Boyd. CVXGEN: A code generator for embedded convex optimization. *Optimization and Engineering*, 13(1):1–27, 2012.

Jonas Močkus. On Bayesian methods for seeking the extremum. In *Optimization Techniques IFIP Technical Conference*, pages 400–404. Springer, 1975.

Angelina Nedić and Dimitri Bertsekas. Least squares policy evaluation algorithms with linear function approximation. *Discrete Event Dynamic Systems*, 13(1-2):79–110, 2003.

Brendan O'Donoghue, Eric Chu, Neal Parikh, and Stephen Boyd. Conic optimization via operator splitting and homogeneous self-dual embedding. *Journal of Optimization Theory and Applications*, 169(3):1042–1068, 2016.

Masashi Okada, Luca Rigazio, and Takenobu Aoshima. Path integral networks: End-to-end differentiable optimal control. *arXiv preprint arXiv:1706.09597*, 2017.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035, 2019.

Warren Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. John Wiley & Sons, 2007.

Warren Powell, Hugo Simao, and Belgacem Bouzaiene-Ayari. Approximate dynamic programming in transportation and logistics: a unified framework. *EURO Journal on Transportation and Logistics*, 1(3):237–284, 2012.

James Rawlings and David Mayne. *Model Predictive Control: Theory and Design*. Nob Hill Publishing, 2009.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, pages 400–407, 1951.

Ugo Rosolia and Francesco Borrelli. Learning model predictive control for iterative tasks: A data-driven control framework. *IEEE Transactions on Automatic Control*, 63(7):1883–1896, 2017.

Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Francisco Solis and Roger Wets. Minimization by random search techniques. *Mathematics of Operations Research*, 6(1):19–30, 1981.

Bartolomeo Stellato, Tobias Geyer, and Paul Goulart. High-speed finite control set model predictive control for power electronics. *IEEE Transactions on Power Electronics*, 32(5):4007–4020, 2017. ISSN 0885-8993.

Bartolomeo Stellato, Goran Banjac, Paul Goulart, Alberto Bemporad, and Stephen Boyd. OSQP: An operator splitting solver for quadratic programs. *Mathematical Programming Computation*, 2020.

G. Stewart and F. Borrelli. A model predictive control framework for industrial turbodiesel engine control. In *IEEE Conference on Decision and Control (CDC)*, pages 5704–5711, 2008.

Richard Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3 (1):9–44, 1988.

Richard Sutton and Andrew Barto. *Reinforcement Learning: An Introduction*. MIT press, 2018.

Aviv Tamar, Garrett Thomas, Tianhao Zhang, Sergey Levine, and Pieter Abbeel. Learning from the hindsight plan–episodic MPC improvement. In *IEEE Intl. on Robotics and Automation (ICRA)*, pages 336–343, 2017.

Yang Wang and Stephen Boyd. Performance bounds for linear stochastic control. *Systems & Control Letters*, 58(3):178–182, 2009.

Yang Wang and Stephen Boyd. Fast evaluation of quadratic control-Lyapunov policy. *IEEE Transactions on Control Systems Technology*, 19(4):939–946, 2010.

Yang Wang, Brendan O'Donoghue, and Stephen Boyd. Approximate dynamic programming via iterated Bellman inequalities. *Intl. Journal of Robust and Nonlinear Control*, 25(10):1472–1496, 2014.

Ronald Williams. *Reinforcement-Learning Connectionist Systems*. College of Computer Science, Northeastern University, 1987.