Data-Driven Optimal Tracking with Constrained Approximate Dynamic Programming for Servomotor Systems

Chakrabarty, Ankush; Danielson, Claus; Wang, Yebin

TR2020-116 August 27, 2020

Abstract

We design real-time optimal tracking controllers for servomotor systems engaged in single-axis point-to-point positioning tasks. The design is challenging due to the presence of unmodeled dynamics, along with speed and acceleration constraints. As model-based optimal control design methods cannot be applied directly to this uncertain system, we propose a data-driven approximate dynamic programming approach to learn an optimal tracking controller that is constraint-enforcing. The potential of our proposed method is illustrated on a servomotor that positions the head of a laser drilling machine.

IEEE Conference on Control Technology and Applications

Data-Driven Optimal Tracking with Constrained Approximate Dynamic Programming for Servomotor Systems

Ankush Chakrabarty[†], Claus Danielson, and Yebin Wang

***Abstract*—We design real-time optimal tracking controllers for servomotor systems engaged in single-axis point-to-point positioning tasks. The design is challenging due to the presence of unmodeled dynamics, along with speed and acceleration constraints. As model-based optimal control design methods cannot be applied directly to this uncertain system, we propose a data-driven approximate dynamic programming approach to learn an optimal tracking controller that is constraint-enforcing. The potential of our proposed method is illustrated on a servomotor that positions the head of a laser drilling machine.**

***Index Terms*—Safe reinforcement learning, data-driven methods, output tracking, constrained control, convex programming, invariant sets.**

I. Introduction

Servomotors are highly prevalent in industrial applications for positioning due to their low cost, simple structure, ease of maintenance, and high power-mass ratio [1]. For applications like robotics and laser drilling, specifications generally include high-precision position control and reference tracking [2]. However, due to modeling uncertainty caused by compressibility of fluids, or friction forces, optimally tracking references with high precision in spite of operational constraints becomes a very challenging problem [3].

Rather than decomposing the motion planner and tracking controller as in [4], we adopt a method that concurrently solves these sub-problems, and this is expected to result in improved performance, albeit at higher computational expense. The approximate dynamic programming (ADP) approach provides a computationally tractable method for learning optimal policies despite unmodeled dynamics [5], [6], has been particularly successful for optimal tracking of dynamical systems [7], [8]. An issue that remains unexplored in ADP methods is constraint satisfaction during tracking: for example, velocity and acceleration constraints on the servomotor.

Enforcing constraints during learning for control has been investigated recently in a variety of domains such as model predictive control (MPC) for repetitive tasks [9], MPC with model adaptation using Gaussian processes [10], explicit MPC [11], and safe reinforcement learning with [12], [13] and without complete state-space model knowledge [14]–[17]: although these methods are for regulation, not tracking.

In this paper, we propose a constrained approximate/adaptive dynamic programming (ADP) algorithm for optimal reference tracking for piece-wise constant reference, despite hard constraints on the motor velocity and acceleration. The method relies on generating an augmented incremental system, whose equilibrium is agnostic to the system's steady-state: this enables the tracking problem to be cast as a regulation problem and allows the use of the constrained ADP formalism of [15] with slight modifications. Concretely, we propose both a model-based and data-driven version of updating constraint enforcing sets, value functions, and corresponding control policies when there is no reference change. In the event of a change in the reference, we adopt a reference scaling method akin to reference governors [18] to adapt the reference so that the updated state lies within the constraint admissible invariant set associated with the current control policy. This enables constraint satisfaction during reference signal jumps.

II. Servomotor Control Problem

We consider a servomotor system for performing point-to-point positioning tasks in a single axis. The servomotor dynamics are modeled by

$$J\ddot{\theta} = -d_0\dot{\theta} - c_0\,\mathrm{sgn}(\dot{\theta}) + K_t u \tag{1}$$

where θ is the angular position of the servomotor, $\dot{\theta}$ is the angular velocity, u is the controlled current, J is the lumped inertia of the servomotor and a load, c_0 is the amplitude of the Coulomb friction force, d_0 is the viscous friction coefficient, and K_t is the torque constant. We enforce that the servomotor always rotates in the same direction, thus $\dot{\theta} \geqslant 0$ and $c_0\,\mathrm{sgn}(\dot{\theta}) = c_0$.

Selecting the states to be the position θ and velocity $\omega := \dot{\theta}$ and discretizing the dynamics (1) with a sampling time τ yields the discrete-time state-space representation

$$\theta(t+1) = \theta(t) + \tau\omega(t) \tag{2a}$$

$$\omega(t+1) = \qquad d\omega(t) + bu(t) - c, \tag{2b}$$

where $x(t) = \begin{bmatrix}\theta(t) & \omega(t)\end{bmatrix}^\top$ is the state with initial conditions $x(t_0) = \begin{bmatrix}\theta(t_0) & \omega(t_0)\end{bmatrix}^\top$. The parameters of (2) are $d = 1 - \tau d_0/J$, $c = \tau c_0/J$, and $b = \tau K_t/J$.

Note that the model (2) can be written compactly as the affine system

$$x(t+1) = Ax(t) + Bu(t) + W, \tag{3a}$$

$$e(t) = Cx(t) - r(t). \tag{3b}$$

[†]Corresponding author. All authors {`chakrabarty`, `danielson`, `yebinwang`}`@merl.com` are affiliated with Mitsubishi Electric Research Labs, Cambridge, MA, USA.

where

$$A = \begin{bmatrix} 1 & \tau \\ 0 & d \end{bmatrix},\ B = \begin{bmatrix} 0 \\ b \end{bmatrix},\ C = \begin{bmatrix} 1 \\ 0 \end{bmatrix}^\top,\ \text{and}\ W = \begin{bmatrix} 0 \\ -c \end{bmatrix}.$$

Here $r(t)$ denotes the desired servomotor position. We make the following assumptions on our knowledge of the system (3).

Assumption 1. The matrices B and C are known, while the state-transition A is unknown. The pair (A, B) is stabilizable. The state $x(t)$ is available at each time t.

Assumption 1 is mild, because one can typically find the torque constant K_t and the inertia of the motor J based on the motor specifications. This assumption particularly holds for drilling applications, which is the motivation of this work. We do not make any assumption on the knowledge of the viscous or Coulomb frictions, which is practically relevant, since these parameters are more difficult to ascertain with high precision. Additionally, the assumption on the availability of both position and speed is not strong: for instance, one could use encoders to obtain $\theta(t)$ and estimate the speed from consecutive measurements or via estimators.

Given a target output position $r(t)$, we will design a control policy that drives the system state $x(t)$ as close as possible to the desired state $\begin{bmatrix} r(t) & 0 \end{bmatrix}^\top$ while minimizing an infinite-horizon cost functional

$$V_\infty = \sum_{t=0}^{\infty} V(t) \tag{4}$$

where

$$V(t) = \|Cx(t) - r(t)\|_S^2 + \|\Delta x(t)\|_Q^2 + \|\Delta u(t)\|_R^2, \tag{5}$$

while enforcing the constraints

$$0 \leqslant \theta(t+1) - \theta(t) \leqslant \tau\omega_{\max}, \tag{6a}$$

$$\tau\alpha_{\min} \leqslant \omega(t+1) - \omega(t) \leqslant \tau\alpha_{\max}. \tag{6b}$$

In the above, $S \geq 0$ and $Q \geq 0$ are weighting matrices on the tracking error and state rate-of-change, $R > 0$ penalizes the actuator rate, $\omega_{\max}$ denotes the maximum allowable angular speed, and $\alpha_{\min} < 0 < \alpha_{\max}$ are limits on angular acceleration.

Assumption 2. The reference r is piecewise constant and $r(t+1)$ is available at time t for all $t \geqslant t_0$.

III. Tracking ADP with Constraints

A. Model-based ADP for Reference Tracking

ADP is a framework most commonly used for synthesizing regulators when complete model information is unavailable or when obtaining exact solutions to an optimal control problem is computationally intractable [5], making it an attractive method for servomotor control. However, since we are trying to control the servomotor for reference tracking, we need to first transform our tracking problem into a regulation problem. This is accomplished using the incremental form of the dynamics

$$\xi(t+1) = \underbrace{\begin{bmatrix} A & 0 \\ C & I \end{bmatrix}}_{\mathcal{A}} \xi(t) + \underbrace{\begin{bmatrix} B \\ 0 \end{bmatrix}}_{\mathcal{B}} \Delta u(t) + \underbrace{\begin{bmatrix} 0 \\ -I \end{bmatrix}}_{\mathcal{G}} \Delta r(t) \tag{7}$$

where

$$\xi(t) = \begin{bmatrix} \Delta x(t) \\ e(t) \end{bmatrix}$$

is an augmented state vector, $\Delta x(t) = x(t+1) - x(t)$ is the incremental state, $\Delta u(t) = u(t+1) - u(t)$ is the incremental control input, and $\Delta r(t) = r(t+1) - r(t)$ is the change in the piecewise constant reference $r(t)$. The Coulomb friction $c(t)$ does not appear in the incremental dynamics (7) since it is constant. The origin $[\Delta x, e] = [0, 0]$ of the incremental dynamics (7) corresponds to the original servo-system (3) being at equilibrium $x(t+1) = x(t)$ with zero tracking error $e(t) = 0$.

For constant references $r(t+1) = r(t)$, the incremental dynamics (7) are linear, rather than the affine dynamics we see in the original servo-dynamics (3). Thus, asymptotic reference tracking $e \to 0$ as $t \to \infty$ for the original servo-system (3) corresponds to asymptotically stabilizing the origin of the incremental system (7). Using full-state feedback

$$\Delta u(t) = \underbrace{\begin{bmatrix} F_P(t) & F_I(t) \end{bmatrix}}_{\mathcal{F}_t} \xi(t), \tag{8}$$

to stabilize the incremental dynamics (7) produces a proportional-integral control policy

$$u(t) = F_P(t)x(t) + \sum_{k=0}^{t-1} F_I(k)e(k)\,, \tag{9}$$

for the original servo-system (3) with $F_P : \mathbb{N} \to \mathbb{R}^{1\times 2}$ and $F_I : \mathbb{N} \to \mathbb{R}$. The integral-action of the controller (9) automatically compensates for the constant disturbance W caused by the Coulomb friction forces $c(t)$.

B. Model-based ADP for Constraint Enforcement

In this section, we describe a model-based algorithm for synthesizing a tracking controller that enforces state constraints. The state constraints (6) for the servomotor can be written as constraints on the state $\xi(t) = [\Delta\theta - \omega_{\max}/2, \Delta\omega, e]^\top$ of the incremental dynamics (7)

$$\mathcal{X} = \left\{ \xi : \underbrace{\begin{bmatrix} \frac{2}{\omega_{\max}} & 0 & 0 \\ -\frac{2}{\omega_{\max}} & 0 & 0 \\ 0 & \frac{1}{\tau\alpha_{\max}} & 0 \\ 0 & -\frac{1}{\tau\alpha_{\max}} & 0 \end{bmatrix}}_{H} \xi(t) \leqslant \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \right\} \tag{10}$$

where the incremental state $\Delta\theta$ has been offset by $\omega_{\max}/2$ so that the origin is in contained in the interior of the constraints (10).

By the definition of stability, a stabilizing controller (8) will keep the state $\xi(t)$ in a neighborhood of the origin defined by a level-set of its Lyapunov function, called an invariant-set,

$$\mathcal{E}(\mathcal{P},\rho)=\{\xi:\xi^\top\mathcal{P}\xi\leqslant\rho\},\tag{11}$$

where $\rho\geqslant\xi(0)^\top P\xi(0)$ is the initial value of the quadratic Lyapunov function $V(\xi)=\xi^\top\mathcal{P}\xi$. Thus, the controller (8) will enforce constraints if the invariant-set (11) is contained $\mathcal{E}(\mathcal{P},\rho)\subseteq\mathcal{X}$ in the constraint set $\{H\xi(t)\leqslant 1\}$ described in (10).

We synthesize a constraint enforcing controller (8) in two steps; policy evaluation and policy improvement. In the policy evaluation step, we compute the optimal invariant-set (11) for the current controller (8) that contains the current state $\xi(t)$ and satisfies constraints (10) . This is accomplished by solving the following semi-definite program

$$\mathcal{P}_{t+1},\rho_{t+1}=\underset{\mathcal{P}>0,\rho>0}{\arg\min}\|\mathcal{J}(t)\|\tag{12a}$$

subject to:

$$(\mathcal{A}+\mathcal{B}\mathcal{F}_t)^\top\mathcal{P}(\mathcal{A}+\mathcal{B}\mathcal{F}_t)-\lambda\mathcal{P}\preceq 0\tag{12b}$$

$$\xi(t)^\top\mathcal{P}\xi(t)\leqslant\rho\tag{12c}$$

$$\rho HH^\top\preceq\mathcal{P}\tag{12d}$$

for some tuning parameter $\lambda\in(0,1)$. Here,

$$\mathcal{J}(t)=(\mathcal{A}+\mathcal{B}\mathcal{F}_t)^\top\mathcal{P}(\mathcal{A}+\mathcal{B}\mathcal{F}_t)-\mathcal{P}+\mathcal{Q}+\mathcal{F}_t^\top\mathcal{R}\mathcal{F}_t.$$

Minimizing the norm of $\mathcal{J}$ promotes a solution close to the optimal LQR control policy for the incremental system (7), since $\mathcal{J}(t)$ should be equal to zero if the discrete-time algebraic Riccati equation admits a control policy that is feasible, that is, that enforces state constraints. The LMI (12b) ensures that the invariant-set (11) parameterized by $\mathcal{P}_{t+1}$ and ρ_{t+1} is invariant under the current controller gain $\mathcal{F}_t$, that is, the state $\xi(t)$ will not leave the set $\mathcal{E}(\mathcal{P}_{t+1},\rho_{t+1})$ after it enters. The LMI (12c) ensures that the new invariant-set $\mathcal{E}(\mathcal{P}_{t+1},\rho_{t+1})$ contains the current state. Finally, the LMI (12d) ensures that the invariant-set satisfies the constraints (10).

The second step of the controller synthesis is policy improvement. In this step, we search among the controller gains $\mathcal{F}_{t+1}$ that render the set $\mathcal{E}(\mathcal{P}_{t+1},\rho_{t+1})$ invariant for the optimal gain. Since we do not consider input constraints, the policy improvement step merely updates the policy using

$$\mathcal{F}_{t+1}=-\left(\mathcal{R}+\mathcal{B}^\top\mathcal{P}_{t+1}\mathcal{B}\right)^{-1}\mathcal{B}^\top\mathcal{P}_{t+1}\mathcal{A}.\tag{13}$$

C. Enforcing Constraints during Reference Changes

The controller (9) designed in the previous section ensures constraint satisfaction as long as the reference is constant $r(t+1)=r(t)$. However, large reference changes Δr can cause constraint violation. To overcome this issue, rather than performing policy improvement, we use the previously estimated policy $\mathcal{F}_t$ and scale the implemented reference to obtain

$$\hat{r}(t+1)=r(t)+\frac{1}{\mu_{t+1}}\frac{\Delta r}{\|\Delta r\|}\tag{14}$$

where $\Delta r=r(t+1)-r(t)$ and the scaling factor μ is obtained by solving the following semi-definite program

$$\mu_{t+1}=\arg\min\ \mu^2\tag{15a}$$

subject to:

$$\begin{bmatrix}\mathcal{L}(\mathcal{F}_t,\mathcal{P}_{t+1}) & (\mathcal{A}+\mathcal{B}\mathcal{F}_t)^\top\mathcal{P}_{t+1}\mathcal{G}\\ \star & (1-\lambda)\rho_{t+1}\mu^2 I-\mathcal{G}^\top\mathcal{P}_{t+1}\mathcal{G}\end{bmatrix}\succeq 0\tag{15b}$$

$$\mu^2\|\Delta r(t)\|^2\geqslant 1\tag{15c}$$

where the Lyapunov equation

$$\mathcal{L}(\mathcal{F}_t,\mathcal{P}_{t+1})=(\mathcal{A}+\mathcal{B}\mathcal{F}_t)^\top\mathcal{P}_{t+1}(\mathcal{A}+\mathcal{B}\mathcal{F}_t)-\lambda\mathcal{P}_{t+1}.$$

Note that the condition (15c) ensures that the scaled reference $\hat{r}(t+1)$ is a convex combination of $r(t+1)$ and $r(t)$.

The following proposition shows that the scaled reference (14) ensures constraint satisfaction.

Proposition 1. *If Assumption 2 holds, then the reference* (14) *with μ_{t+1} obtained by solving* (15) *ensures that closed-loop system* (7) *and* (8) *satisfies the constraints* (10).

Proof. We drop the time dependence on $\mathcal{P}_{t+1}$ and ρ_{t+1} for brevity. For the scaled reference $\hat{r}(t+1)$ obtained using (14), the incremental model dynamics in (7) become

$$\xi(t+1)=(\mathcal{A}+\mathcal{B}\mathcal{F}_t)\xi(t)+\mathcal{G}w(t),$$

where $w(t)=\Delta r(t)/(\mu\|\Delta r(t)\|)$.

With this insight, we take a congruence transformation of (15b) with the vector $\begin{bmatrix}\xi(t) & w(t)\end{bmatrix}^\top$, which yields

$$\begin{aligned}\xi(t+1)^\top\mathcal{P}\xi(t+1)-\lambda\xi(t)^\top\mathcal{P}\xi(t)&\\ +(1-\lambda)\rho\mu^2 w(t)^\top w(t)&\leqslant 0.\end{aligned}\tag{16}$$

Since $\mu^2 w^\top w=1$, which implies that

$$\xi(t+1)^\top\mathcal{P}\xi(t+1)\leqslant\lambda\xi(t)^\top\mathcal{P}\xi(t)+(1-\lambda)\rho.\tag{17}$$

Since $\xi(t)\in\mathcal{E}(\mathcal{P},\rho)$, we know that $\xi(t)^\top\mathcal{P}\xi(t)\leqslant\rho$. Thus, the inequality (17) can be written as

$$\xi(t+1)^\top\mathcal{P}\xi(t+1)\leqslant\lambda\rho+(1-\lambda)\rho=\rho.$$

Therefore, $\xi(t+1)\in\mathcal{E}(\mathcal{P},\rho)\subseteq\mathcal{X}$. $\square$

D. Data-driven/model-free implementation

Since complete model knowledge is not available to us, by assumption, we will now provide a 'data-driven' or 'model-free' algorithm to compute the constrained ADP algorithm with the reference scaling mechanism.

1) Data-driven constrained ADP: Recall that the state $x(t)$, control $u(t)$, the reference $r(t)$, and all their past values from $t_0,\dots,t-1$ are available to us at time t. This implies that we can compute the augmented state $\xi(t)$ (that is, Δx and e), the incremental control action $\Delta u(t)$, and therefore the quadratic cost $V(t)$ at every t in order to obtain $\mathcal{P}_{t+1}$ and $\mathcal{F}_{t+1}$. With the history $\{\xi_k\}_{k=t_0}^t$ and $\{\Delta u_k\}_{k=t_0}^t$, one can pose a semi-definite programming problem to obtain $\mathcal{P}_{t+1}$ given $\mathcal{P}_t$ and $\mathcal{K}_t$. The constrained policy evaluation step yields $\mathcal{P}_{t+1}$

by solving

$$\mathcal{P}_{t+1}, \rho_{t+1} = \underset{\mathcal{P}>0,\rho>0}{\arg\min} \|\tilde{\mathcal{J}}(t)\| + \gamma_\rho \, \rho \tag{18a}$$

subject to:

$$\xi(t)^\top \mathcal{P}\xi(t) - \lambda\xi(t-1)^\top \mathcal{P}\xi(t-1) \leqslant 0 \tag{18b}$$

$$\xi(t-1)^\top \mathcal{P}\, \xi(t-1) \leqslant \rho \tag{18c}$$

$$\rho\, HH^\top \preceq \mathcal{P}. \tag{18d}$$

Here, γ_ρ is a regularization term that promotes the uniqueness of the solution, and is typically a small positive scalar. The norm of

$$\tilde{\mathcal{J}}(t) = \xi(t)^\top \mathcal{P}\xi(t) - \xi(t-1)^\top \mathcal{P}\xi(t-1) + V(t)$$

is analogous to the norm of $\mathcal{J}(t)$ in (12a), and can be evaluated because $V(t)$ can be computed using (5). Similarly, (18b) is analogous to (12b) with the model replaced with data. For a unique solution to be admitted, one cannot solve (18) with a single data point, and typically, a train of historical data is first collected and then the problem is solved with the LMIs (18b)–(18c) stacked for each instance of the collected data. More details are given in [15].

The constrained policy improvement step can also be performed by collecting data on-line and setting up a least squares problem, described herein. In the presence of a model, one could obtain the updated policy by solving

$$\begin{aligned}\mathcal{F}_{t+1} = \arg\min_{\mathcal{F}} &\frac{1}{2}\xi(t)^\top \left(\mathcal{F}^\top \mathcal{R}\mathcal{F}\right)\xi(t) \\ &+ \xi(t)^\top \left(\mathcal{A} + \mathcal{B}\mathcal{F}\right)^\top \mathcal{P}_{t+1}\left(\mathcal{A} + \mathcal{B}\mathcal{F}\right)\xi(t).\end{aligned} \tag{19}$$

However, it is possible to solve this problem without knowing $\mathcal{A}$ by adopting the procedure described in [15], with the recursion

$$\mathcal{H}_{t+1} = \mathcal{H}_t + \xi(t)\xi(t)^\top \otimes (\mathcal{R} + \mathcal{B}^\top \mathcal{P}_{t+1}\mathcal{B}), \tag{20a}$$

$$\varphi_{t+1} = \xi(t) \otimes \left(\mathcal{R}\mathcal{F}_t\xi(t) + \mathcal{B}^\top \mathcal{P}_{t+1}\xi(t+1)\right), \tag{20b}$$

$$\mathrm{vec}(\mathcal{F}_{t+1}) = \mathrm{vec}(\mathcal{F}_t) - \beta_t\, \mathcal{H}_{t+1}^{-1}\, \varphi_{t+1}, \tag{20c}$$

where $\otimes$ denotes the Kronecker product, vec denotes the vectorization operator, and $\mathcal{H}$, φ are Hessian and gradient estimates initialized at the identity and zero, respectively. The step size β_t is a tuning parameter.

Remark 1. For existence of solutions, the system needs to be persistently exciting. It is standard practice, therefore, to add a small-magnitude, random exploratory noise to the control input for data collection.

Remark 2. We do not formally provide convergence guarantees of the value function and the control policy in this paper, and consign this to future work. However, the arguments of the proof will closely follow the arguments made in the proofs of [15], with the additional assumption that there is some finite time $T_0 \geqslant t_0$ after which $\Delta r \equiv 0$.

2) Data-driven reference scaling: Recall that we scale the reference only when $\Delta r \neq 0$, and that the reference change Δr (and hence, $\|\Delta r\|$) is known *a priori*, by assumption. One can then design the scaling factor by solving the following problem

$$\mu_{t+1}^{-1} = \arg\max\ \mu^{-1} \tag{21a}$$

subject to:

$$\xi_\mu^\top \mathcal{P}_{t+1}\xi_\mu - \lambda\xi(t)^\top \mathcal{P}_{t+1}\xi(t) + (1-\lambda)\rho \leqslant 0 \tag{21b}$$

$$0 \leqslant \mu^{-1} \leqslant \|\Delta r(t)\|, \tag{21c}$$

where

$$\begin{aligned}\xi_\mu &= (\mathcal{A} + \mathcal{B}\mathcal{F}_t)\xi(t) + \mathcal{G}\mu^{-1}\frac{\Delta r(t)}{\|\Delta r(t)\|} \\ &= \mathcal{A}\xi(t) + \mathcal{B}\Delta u(t) + \mathcal{G}\mu^{-1}\frac{\Delta r(t)}{\|\Delta r(t)\|}.\end{aligned} \tag{22}$$

Clearly, the component $\mathcal{A}\xi(t)$ is unknown, since $\mathcal{A}$ is unknown, but the other terms are known, or can easily be calculated for a given μ. However, one can estimate $\mathcal{A}\xi(t)$ if the state $\xi(t)$ has been attained before. This condition is formally stated next.

Condition 1. *For each $t \in \mathcal{T}' := \{t : \Delta r(t) \neq 0\}$, there exists some time $t_e(t) \in [t_0, t]$ such that $\xi(t_e(t)) = \xi(t)$.*

Condition 1 invokes a requirement similar to that used in experience replay: a mechanism deployed in reinforcement learning [19], [20] to improve learning quality by relying on past experiences (states, actions, and Q-function/value-function values). We rely on having some knowledge in the past about how the system behaved at a certain state $\xi(t_e(t))$, so that this knowledge can then be leveraged if the system returns to that same state at some future time.

If Condition 1 is satisfied, we can obtain the estimate

$$\mathcal{A}\xi(t) = \mathcal{A}\xi(t_e(t)) = \xi(t_e(t)+1) - \mathcal{B}\Delta u(t_e(t)) - \mathcal{G}\Delta r(t_e(t)).$$

Substituting this into (22), we get,

$$\begin{aligned}\xi_\mu = \xi(t_e(t)+1) - \mathcal{B}\Delta u(t_e(t)) - \mathcal{G}\Delta r(t_e(t)) \\ + \mathcal{B}\Delta u(t) + \mathcal{G}\frac{\Delta r(t)}{\mu\|\Delta r(t)\|}.\end{aligned}$$

With this form of ξ_μ, we can solve the data-driven version of the reference scaling problem (21) by employing a line search to maximize $\mu^{-1} \in [0, \|\Delta r\|]$ which satisfies (21b).

A full pseudocode is presented in Algorithm 1 to help facilitate implementation.

IV. Servomotor Performance Evaluation

In the drilling process, a drilling head is first moved atop of the location of the hole, stands still, and then the laser beam will be fired to melt material and drill a hole. Because the drilling head has to stop at the hole, the movement of the drilling head for drilling a series of holes can be decomposed into a number of point to point single-axis positioning tasks.

We illustrate this behavior in Fig. 1. The piece on which the holes are to be drilled, the servomotor and drill bit, the direction of motion of the drill to attain the correct sequence of cuts, and the corresponding reference to be tracked to make the four holes (without returning to the origin), are all shown in the figure. Although in practice, there are two servomotors, each of which are capable of single-axis

Algorithm 1 Constrained ADP for Tracking

```
Require: Initial feasible policy and value matrix: $\mathcal{F}_0$, $\mathcal{P}_0$
Require: System matrices $B$, $C$
Require: Initial reference $r_0$
 1: for $t \geqslant 0$ do
 2:     Store prior $\xi, u$ in history
 3:     Acquire reference update $r(t+1)$
 4:     Compute $\Delta r$
 5:     if $\Delta r \neq 0$ then
 6:         Compute $\hat{r}$ using (21)                ▷ reference scaling
 7:         $r(t+1) \leftarrow \hat{r}(t+1)$
 8:     else
 9:         if $t \in T_{\text{learn}}$ then             ▷ enough data collected
10:             Compute $\mathcal{P}_{t+1}$ using (18)
11:             Compute $\mathcal{F}_{t+1}$ using (20)
12:             Apply control using $\mathcal{F}_{t+1}$       ▷ use new policy
13:         else
14:             Apply control using $\mathcal{F}_t$           ▷ use old policy
15:         end if
16:     end if
17: end for
```

motion, since the two motors are temporally decoupled, we focus our results on one of the servomotors for simplicity. Consequently, we have one reference signal r to track. The stroke lengths required by each servomotor along the edges of the piece to attain the correct positioning for drilling are shown in radians: these stroke lengths correspond exactly to the magnitude of the reference signal. The servomotor

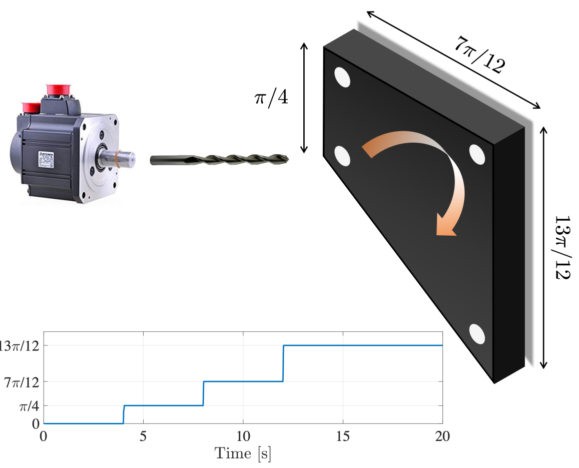

Fig. 1. Exemplar drilling task and stroke lengths required as reference input.

parameters and constraint values are provided in Table I. While most parameter values have been taken from [3], [21],

TABLE I
TRUE SERVOMETER PARAMETERS AND CONSTRAINTS.

J	7.2×10^{-5}	K_t	0.2723
c_0	6.37×10^{-2}	d_0	1.01×10^{-3}
$\omega^{\text{feas}}_{\text{min}}$	0	$\omega^{\text{feas}}_{\text{max}}$	100π
$\alpha^{\text{feas}}_{\text{min}}$	-1.326×10^4	$\alpha^{\text{feas}}_{\text{max}}$	1.326×10^4
$\omega^{\text{infeas}}_{\text{min}}$	0	$\omega^{\text{infeas}}_{\text{max}}$	10π
$\alpha^{\text{infeas}}_{\text{min}}$	-1.326×10^3	$\alpha^{\text{infeas}}_{\text{max}}$	1.326×10^3

we consider two sets of velocity and acceleration constraints in this paper to illustrate the effectiveness of our approach under much tighter constraints than those investigated before. The first set, labeled 'feas', denotes a scenario when the optimal LQR tracking control policy is feasible, and when the constrained ADP is expected to converge to the optimal policy: these are the constraint values considered in [3]. The second set of constraints, labeled 'infeas', considers a more challenging problem: namely, when the constraints are too tight to admit the optimal LQR policy. In this scenario, the constrained ADP is expected to converge to the best feasible controller rather than the optimal LQR policy.

Hyperparameters required for the constrained ADP algorithm are as follows. The cost function weight matrices are given by $S = 1$, $Q = 0$, and $R = 10$. The sampling time $\tau = 10$ ms, the initial state is the origin, and $\lambda = 0.999$. The initial feasible policy and initial CAIS ellipsoid is parametrized by

$$\mathcal{K}_0 = \begin{bmatrix} -0.3501 & -0.0325 & -0.0150 \end{bmatrix}$$

and

$$\mathcal{P}_0 = 10^3 \begin{bmatrix} 5.4859 & 0.0523 & 0.2568 \\ 0.0523 & 0.0055 & 0.0023 \\ 0.2568 & 0.0023 & 0.0233 \end{bmatrix}$$

which is obtained by model-free tuning of a proportional-integral control policy and estimating an ellipsoidal admissible domain of attraction from data that lies within the constraint set $\mathcal{X}$, for example, using methods described in [22].

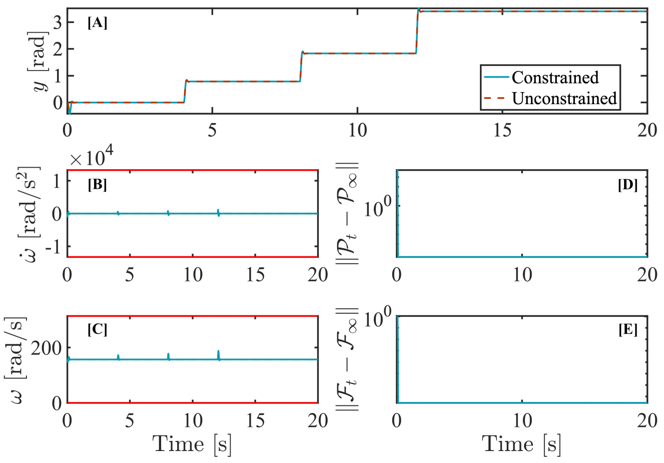

Fig. 2. [A] Position in radians for the unconstrained and constrained ADP for a problem where the optimal tracking controller is feasible. [B, C] Angular velocity and acceleration with hard constraints for the servomotor driven by unconstrained (dashed line) and constrained ADP (continuous line). [D] Evolution of normed error in the $\mathcal{P}$ matrix. [E] Evolution of normed error in the $\mathcal{F}$ matrix.

We begin with the case when the optimal LQR tracking controller $\mathcal{F}_\infty$ is feasible: that is, one can track the desired $r(t)$ signal without violating constraints using the control policy $\Delta u(t) = \mathcal{F}_\infty \xi(t)$. Fig. 2 illustrates the performance of constrained ADP tracking in such a scenario. From subplot [A], we observe that the output trajectories are identical for the optimal policy and the constrained ADP policy, which disparity seen only in the first few seconds when the ADP iterations are still reaching the fixed point; see [D] and [E] for convergence of the iterates $\mathcal{P}_t$ and $\mathcal{F}_t$ to their respective optima. The fact that the optimal policy can be achieved without violating the velocity and acceleration constraints is evident from subplots [B] and [C].

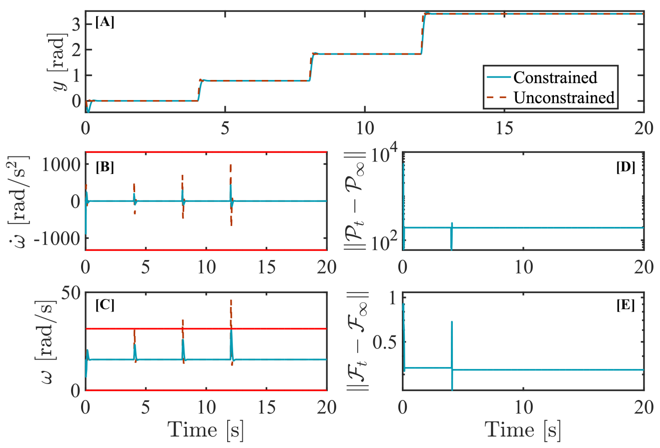

Fig. 3. [A] Position in radians for the unconstrained and constrained ADP for a problem where the optimal tracking controller is infeasible. [B, C] Angular velocity and acceleration with hard constraints for the servomotor driven by unconstrained (dashed line) and constrained ADP (continuous line). [D] Evolution of normed error in the $\mathcal{P}$ matrix. [E] Evolution of normed error in the $\mathcal{F}$ matrix.

A more interesting scenario is investigated next, wherein the optimal (unconstrained) policy $\mathcal{F}_\infty$ will result in constraint violation for the given reference signal. We ensured infeasibility of $\mathcal{F}_\infty$ by tightening to more restrictive constraints than considered in the prior experiment. The fact that employing $\mathcal{F}_\infty$ results in velocity constraints being violate is apparent from subplot [C] of Fig. 3: constraint violation occurs twice, during the drilling of the final two holes. Conversely, the constrained ADP formulation results in no constraint violation, despite the same reference being tracked: the effectiveness of the reference adaptation is especially noteworthy around 12 s, when the velocity grazes and recovers from the constraint. As expected, the output trajectories in subplot [A] are different at the reference jumps: the constrained ADP has a smoother, less aggressive halting motion compared with the optimal tracking controller. From the subplots [D] and [E], we notice another interesting phenomenon: that the sequence of $\mathcal{P}_t$ and $\mathcal{F}_t$ converges not to $\mathcal{P}_\infty$ and $\mathcal{F}_\infty$ but to the pair of value function and policy matrices that are *feasibly* optimal. In other words, the system learns the best tracking policy that is constraint-enforcing automatically and does not attain the optimal tracking controller that is infeasible.

V. Conclusions

In this paper, we propose a methodology for computing optimal constraint-enforcing tracking policies for servomotors under velocity and acceleration constraints. We learn the optimal feasible policies using a novel combination of constrained ADP and reference scaling for applications where the reference signal is known and piece-wise constant and illustrate the potential of the approach on a positioning task for laser drilling operations. The problem is posed as a semidefinite program that can be solved on-line using standard convex programming methods—both in a model-based and data-driven manner. In future work, we will provide formal guarantees on the convergence of the learned value function and control policies to their respective optima.

References

[1] S. Kim, “Moment of inertia and friction torque coefficient identification in a servo drive system,” *IEEE Trans. Ind. Elec.*, vol. 66, no. 1, pp. 60–70, 2018.

[2] G. Zhong, Z. Shao, H. Deng, and J. Ren, “Precise position synchronous control for multi-axis servo systems,” *IEEE Trans. Ind. Elec.*, vol. 64, no. 5, pp. 3707–3717, 2017.

[3] Y. Wang, K. Ueda, and S. A. Bortoff, “A Hamiltonian approach to compute an energy efficient trajectory for a servomotor system,” *Automatica*, vol. 49, no. 12, pp. 3550–3561, 2013.

[4] D. Verscheure *et al.*, “Time-optimal path tracking for robots: A convex optimization approach,” *IEEE Trans. Aut. Control*, vol. 54, no. 10, pp. 2318–2327, 2009.

[5] D. P. Bertsekas, *Reinforcement Learning and Optimal Control*. Athena Scientific, Belmont, MA, 2019.

[6] F. L. Lewis and D. Vrabie, “Reinforcement learning and adaptive dynamic programming for feedback control,” *IEEE Circuits and Systems Magazine*, vol. 9, no. 3, pp. 32–50, 2009.

[7] W. Gao and Z.-P. Jiang, “Adaptive dynamic programming and adaptive optimal output regulation of linear systems,” *IEEE Trans. Aut. Control*, vol. 61, no. 12, pp. 4164–4169, 2016.

[8] X. Li, L. Xue, and C. Sun, “Linear quadratic tracking control of unknown discrete-time systems using value iteration algorithm,” *Neurocomputing*, vol. 314, pp. 86–93, 2018.

[9] U. Rosolia, A. Carvalho, and F. Borrelli, “Autonomous racing using learning model predictive control,” in *Proc. of the American Control Conference*, 2017, pp. 5115–5120.

[10] L. Hewing, J. Kabzan, and M. N. Zeilinger, “Cautious model predictive control using gaussian process regression,” *IEEE Trans. on Control Sys. Tech.*, pp. 1–8, 2019.

[11] A. Chakrabarty *et al.*, “Support vector machine informed explicit nonlinear model predictive control using low-discrepancy sequences,” *IEEE Trans. Aut. Control*, vol. 62, no. 1, pp. 135–148, 2016.

[12] F. Berkenkamp *et al.*, “Safe model-based reinforcement learning with stability guarantees,” in *Adv. in Neural Information Processing Systems*, 2017, pp. 908–918.

[13] Z. Li, U. Kalabić, and T. Chu, “Safe reinforcement learning: Learning with supervision using a constraint-admissible set,” in *Proc. of the American Control Conference*, 2018, pp. 6390–6395.

[14] A. Chakrabarty *et al.*, “Safe approximate dynamic programming via kernelized Lipschitz estimation,” *IEEE Trans. on Neural Networks and Learning Systems*, pp. 1–15, 2020.

[15] A. Chakrabarty *et al.*, “Approximate dynamic programming for linear systems with state and input constraints,” in *Proc. of the European Control Conference*, 2019, pp. 524–529.

[16] Y. Yang *et al.*, “Safety-aware reinforcement learning framework with an actor-critic-barrier structure,” in *Proc. of the American Control Conference*, 2019, pp. 2352–2358.

[17] A. Chakrabarty, D. K. Jha, and Y. Wang, “Data-driven control policies for partially known systems via kernelized Lipschitz learning,” in *Proc. of the American Control Conference*, July 2019, pp. 4192–4197.

[18] U. Kalabić and S. Di Cairano, “Smooth transitions in shared control using constraint-admissible sets,” in *Proc. of the IEEE Conference on Control Technology and Applications*, 2019, pp. 21–26.

[19] H. Modares, F. L. Lewis, and M.-B. Naghibi-Sistani, “Integral reinforcement learning and experience replay for adaptive optimal control of partially-unknown constrained-input continuous-time systems,” *Automatica*, vol. 50, no. 1, pp. 193–202, 2014.

[20] D. Zhao *et al.*, “Experience replay for optimal control of nonzero-sum game systems with unknown dynamics,” *IEEE Trans. Cybernetics*, vol. 46, no. 3, pp. 854–865, 2015.

[21] Y. Wang *et al.*, “A real-time energy-optimal trajectory generation method for a servomotor system,” *IEEE Trans. Ind. Elec.*, vol. 62, no. 2, pp. 1175–1188, 2014.

[22] A. Chakrabarty *et al.*, “Data-driven estimation of backward reachable and invariant sets for unmodeled systems via active learning,” in *Proc. of the IEEE Conference on Decision and Control*, 2018, pp. 372–377.