MITSUBISHI ELECTRIC RESEARCH LABORATORIES
http://www.merl.com

Approximate Dynamic Programming For Linear Systems with State and Input Constraints

Chakrabarty, A.; Quirynen, R.; Danielson, C.; Gao, W.

TR2019-059 June 29, 2019

Abstract

Enforcing state and input constraints during reinforcement learning (RL) in continuous state spaces is an open but crucial problem which remains a roadblock to using RL in safetycritical applications. This paper leverages invariant sets to update control policies within an approximate dynamic programming (ADP) framework that guarantees constraint satisfaction for all time and converges to the optimal policy (in a linear quadratic regulator sense) asymptotically. An algorithm for implementing the proposed constrained ADP approach in a data-driven manner is provided. The potential of this formalism is demonstrated via numerical examples.

European Control Conference (ECC)

Approximate Dynamic Programming For Linear Systems with State and Input Constraints

Ankush Chakrabarty[1,†], Rien Quirynen[1], Claus Danielson[1], Weinan Gao[2]

***Abstract*—Enforcing state and input constraints during reinforcement learning (RL) in continuous state spaces is an open but crucial problem which remains a roadblock to using RL in safety-critical applications. This paper leverages invariant sets to update control policies within an approximate dynamic programming (ADP) framework that guarantees constraint satisfaction for all time and converges to the optimal policy (in a linear quadratic regulator sense) asymptotically. An algorithm for implementing the proposed constrained ADP approach in a data-driven manner is provided. The potential of this formalism is demonstrated via numerical examples.**

***Index Terms*—Reinforcement learning; safe learning; data-driven; linear quadratic regulator; policy iteration; invariant sets.**

I. Introduction

Combining optimal control theory and reinforcement learning (RL) has yielded many excellent algorithms for generating control policies that imbue the closed-loop system with a desired level of performance in spite of unmodeled dynamics or modeling uncertainties [1], [2]. Specifically, approximate dynamic programming (ADP) (sometimes also referred to as adaptive dynamic programming), a modern embodiment of RL [3], [4] applied to continuous state and action spaces has gained traction for its ability to provide tractable solutions (in spite of the curse of dimensionality) to optimal control problems via function approximation and iterative updates of control policies and value functions [5], [6].

There are two main classes of ADP algorithms: policy iteration and value iteration [7]. A policy iteration algorithm for discrete-time linear systems was formulated in [8] that leverages Q-functions proposed in [9], [10], enabling control policy design without a complete system description. This methodology has been extended to continuous-time systems [11], $\mathcal{H}_2$ and $\mathcal{H}_\infty$ formulations [12], [13], tracking [14], output regulation [15], and game-theoretic settings [16], [17]. To reiterate, a particularly beneficial feature of this class of iterative methods is that control policies generated by policy iteration converge to the optimal control policy with data obtained by exciting the system dynamics, in spite of incomplete model knowledge [18], [19]. While optimality is important for certifying performance in a control system, often times the more critical concern is safety. A key aspect of safe control design is the ability of the system to respect both state and input constraints. To the best of our knowledge, this critical problem remains an open challenge in the context of ADP (and RL at large) in continuous state and action spaces.

In this paper, we modify the classical policy iteration algorithm to incorporate safety through constraint satisfaction. The key idea is to compute control policies and associated constraint admissible invariant sets that ensure the system states and control inputs never violate design constraints. In the spirit of ADP, these policies and invariant sets are computed iteratively, and the sequence of policies are guaranteed to converge asymptotically to the optimal constraint-satisfying policy, provided that the system is sufficiently excited. The use of invariant sets to incorporate safety in learning/adaptive control algorithms via constraint handling has been done in model-based control design, such as model predictive control (MPC) [20]–[23], but its application to data-driven or model-free RL methods is relatively unexplored. A recent paper [24] is a noteworthy exception, although our method is distinct from this work in that we do not compute a model of the system using the data obtained during operation; that is, our method is a direct data-driven approach, as defined in [25].

The main **contributions** of this paper are: (i) we extend classical policy iteration in continuous state-action spaces to enforce state and input constraints; (ii) we provide a data-driven variant of this constrained policy iteration algorithm with unknown state matrix; and, (iii) we provide new sufficient conditions for safety (via constraint satisfaction), stability, and convergence of the policies generated by our proposed algorithm to the optimal constrained control policy.

II. Notation

We denote by $\mathbb{R}$ the set of real numbers, $\mathbb{R}_+$ as the set of positive reals, and $\mathbb{N}$ as the set of natural numbers. For every $v \in \mathbb{R}^n$, we denote $\|v\| = \sqrt{v^\top v}$, where $v^\top$ is the transpose of v. The sup-norm is defined as $\|v\|_\infty \triangleq \sup_{t\in\mathbb{R}} \|v(t)\|$. We denote by $\underline{\sigma}(P)$ and $\overline{\sigma}(P)$ as the smallest and largest singular value of a square, symmetric matrix P, respectively. The symbol $\succ$ ($\prec$) indicates positive (negative) definiteness and $A \succ B$ implies $A - B \succ 0$ for A, B of appropriate dimensions. Similarly, $\succeq$ ($\preceq$) implies positive (negative) semi-definiteness. The operator norm is denoted $\|P\|$ and is defined as the maximum singular value of P, $\mathrm{vec}(P)$ denotes the column-wise vectorization of P, and $\otimes$ denotes the Kronecker product.

[1]A. Chakrabarty, R. Quirynen, and C. Danielson are affiliated with the Control and Dynamical (CD) Systems Group, Mitsubishi Electric Research Laboratories, Cambridge, MA, USA.

[2]W. Gao is with the Department of Electrical and Computer Engineering, Allen E. Paulson College of Engineering and Computing, Georgia Southern University, Statesboro, GA 30460, USA.

[†]Corresponding author. Email: chakrabarty@merl.com. Phone: +1 (617) 758-6175.

We parameterize an ellipsoid $\mathcal{E}_{P_0}^{\rho} = \{x : x^\top P_0 x \leq \rho\}$ using a scalar $\rho > 0$ and a matrix $P_0 \succ 0$.

III. Motivation

In this section, we describe a general approximate dynamic programming formulation for solving the unconstrained discrete-time LQR problem.

A. Problem Statement

We consider discrete-time linear systems of the form

$$x_{t+1} = Ax_t + Bu_t, \tag{1}$$

where $t \in \mathbb{R}$ is the time index, $x \in \mathbb{X} \subset \mathbb{R}^n$ is the state of the system, $u \in \mathbb{U} \subset \mathbb{R}^m$ is the control input, and x_{t_0} is a known initial state of the system. We assume the admissible state and input constraints sets $\mathbb{X}$ and $\mathbb{U}$ are polytopic, and therefore, can be represented as

$$\mathcal{X}' = \left\{ \begin{bmatrix} x \\ u \end{bmatrix} \in \mathbb{R}^{n+m} : c_i^\top x + d_i^\top u \leq 1 \right\}, \tag{2}$$

for $i = 1, \ldots, r$, where r is the total number of state and input constraints and $c_i \in \mathbb{R}^n$ and $d_i \in \mathbb{R}^m$. The sets $\mathbb{X} \subset \mathbb{R}^n$ and $\mathbb{U} \subset \mathbb{R}^m$ are known, compact, convex, and contain the origin in their interiors. Note that with any fixed control policy K, the constraint set described in (2) is equivalent to the set

$$\mathbb{X}' = \left\{ x \in \mathbb{R}^n : (c_i^\top + d_i^\top K)x \leq 1 \right\}, \tag{3}$$

for $i = 1, \ldots, r$.

Remark 1. The inequalities (2) define a polytopic admissible state and input constraint set. Note that $c_i = 0$ implies that the ith constraint is an input constraint, and $d_i = 0$ implies that it is a state constraint. ■

The system matrix A and input matrix B have appropriate dimensions. We make the following assumption on our knowledge of the system; these are standard assumptions in policy iteration methods.

Assumption 1. *The matrix A is unknown, the matrix B is known, and the pair (A, B) is stabilizable. Furthermore, there exists a known control gain K_0 such that $u = K_0 x$ is a stabilizing control policy for the system* (1).

While the knowledge of the input matrix B is not needed in approaches like Q-learning [13], it is fairly standard for policy improvement in policy iteration methods, even with function approximators [2]. From a practical perspective, it is not uncommon for a designer to have knowledge of input channels and channel gains that represent the elements of the B matrix.

Our **objective** is to design an optimal control policy K_∞ such that the state-feedback controller $u = K_\infty x$ stabilizes the partially known system (1) while minimizing a cost functional

$$V := \sum_{t=0}^{\infty} x_t^\top Q x_t + u_t^\top R u_t \tag{4}$$

where $Q \succeq 0$ and $R \succ 0$ are user-defined symmetric matrices, with the pair $(A, Q^{1/2})$ being observable. The main contribution of this paper is to derive controller gains that stabilize the system (1) while strictly enforcing state and input constraints.

B. Overview of optimal control for discrete-time LQR

Let the value function be defined as

$$V_t(x_t, u_t) := \sum_{k=t}^{\infty} x_k^\top Q x_k + u_k^\top R u_k.$$

Clearly, V_t satisfies the recurrence relation

$$V_t(x_t, u_t) = x_t^\top Q x_t + u_t^\top R u_t + V_{t+1}(x_{t+1}, u_{t+1}). \tag{5}$$

We know from optimal control theory that the optimization problem

$$V_\infty(x_t) := \min_u V_t(x_t, u_t) \tag{6}$$

is solved in order to obtain the optimal control action

$$u_\infty := \arg\min_u V_t(x_t, u_t) \tag{7}$$

for each time instant $t \geq t_0$. For discrete-time linear time-invariant systems of the form (1), we know that the value function V_t is quadratic in the state [2]. Therefore, solving (6) is equivalent to finding a symmetric matrix $P_\infty \succ 0$ that satisfies the equation

$$\begin{aligned} &A^\top P_\infty A - P_\infty + Q \\ &\quad - A^\top P_\infty B \left(R + B^\top P_\infty B\right)^{-1} B^\top P_\infty A = 0. \end{aligned} \tag{8}$$

Upon solving for P_∞, the optimal unconstrained discrete-time LQR policy generated by solving (7) is given by

$$K_\infty = -(R + B^\top P_\infty B)^{-1} B^\top P_\infty A. \tag{9}$$

Since by assumption, the model A is unknown, one cannot directly compute P_∞ from (8) or K_∞ from (9). Instead, we resort to ADP, an iterative method for 'learning' the optimal control policy (9) by using on-line data without knowing a full model of the system (1). A popular embodiment of ADP is *policy iteration*, wherein an initial stabilizing control policy K_0 is iteratively improved using operational data, that is, without full model information. The sequence of control policies converges asymptotically to the optimal control policy K_∞ defined in (9). The key steps of policy iteration without constraints are described next.

C. Unconstrained policy iteration

Let K_t be the t-th policy iterate, where $t \in \mathbb{N}$. Policy iteration has two key steps: policy evaluation and policy improvement. We begin by describing the steps in model-based policy iteration and subsequently demonstrate how to perform the same steps in a data-driven manner.

1) Model-based policy evaluation: In the policy evaluation step, the value function parameter $P_{t+1} \succ 0$ is estimated with the control gain K_t using the relation

$$(A+BK_t)^\top P_{t+1}(A+BK_t) - P_{t+1} + Q + K_t^\top R K_t = 0. \tag{10}$$

Note that (10) can be derived from (5) when $V_t = x_t^\top P_t x_t$ and replacing $u_t = K_t x_t$ and $x_{t+1} = (A + BK_t)x_t$.

2) Model-based policy improvement: Upon updating the value function via (10), one needs to update the corresponding control policy. This is done by computing the new controller gain via

$$K_{t+1} = -\left(R + B^\top P_{t+1} B\right)^{-1} B^\top P_{t+1} A. \tag{11}$$

This equation is reminiscent of the optimal control policy equation (9); in fact, the unique stationary point of the system of equations (10)–(11) is at $P_t = P_\infty$ and $K_t = K_\infty$ as demonstrated in [6].

This model-based implementation can be performed in a data-driven manner, described next.

3) Data-driven policy evaluation: We assume that policy iteration is performed a discrete-time instances t_i where

$$\mathcal{T} = \{t_i\}_{i=1}^{\infty} \tag{12}$$

denotes the set of all policy iteration times. The minimum number of data-points obtain between policy iterations $[t_i, t_{i+1}]$ is given by

$$N = \inf_{i \in \mathbb{N}} \{t_{i+1} - t_i | t_i, t_{i+1} \in \mathcal{T}\}, \tag{13}$$

that is, N denotes the minimum number of data points contained within any learning cycle. In a model based implementation, $\mathcal{T} = \mathbb{N}$.

At each learning time instant $t_i \in \mathcal{T}$, one can rewrite (10) as

$$x_t^\top P^+ x_t = x_t^\top Q x_t + u_t^\top R u_t + x_{t+1}^\top P^+ x_{t+1}, \tag{14}$$

for every $t \in \{t_i + 1, t_i + 2, \ldots, t_{i+1}\}$, with P^+ representing the updated value function matrix. Assuming that the state and input data is available to us, and that Q and R are known, we can rewrite (14) as

$$\Delta_{xx} \mathrm{vec}(P^+) = \begin{bmatrix} x_{t_i+1}^\top Q x_{t_i+1} + u_{t_i+1}^\top R u_{t_i+1} \\ x_{t_i+2}^\top Q x_{t_i+2} + u_{t_i+2}^\top R u_{t_i+2} \\ \vdots \\ x_{t_{i+1}}^\top Q x_{t_{i+1}} + u_{t_{i+1}}^\top R u_{t_{i+1}} \end{bmatrix}, \tag{15}$$

where

$$\Delta_{xx} = \begin{bmatrix} x_{t_i} \otimes x_{t_i} - x_{t_i+1} \otimes x_{t_i+1} \\ \vdots \\ x_{t_{i+1}} \otimes x_{t_{i+1}} - x_{t_{i+1}+1} \otimes x_{t_{i+1}+1} \end{bmatrix}. \tag{16}$$

Under well-known persistence of excitation conditions [2], one can solve (15) as a (regularized) least squares problem subject to the constraint that $P^+ \succ 0$ to obtain P^+ without knowing A or B. For the time instants $t \in \mathcal{T}$ when the learning occurs, the new value function matrix P_{t+1} is set to P^+ obtained by solving (15). For other time instants between learning time instants, that is $t \notin \mathcal{T}$, the value function matrix obtained in the previous learning cycle is utilized, that is, $P_{t+1} := P_t$.

4) Data-driven policy improvement: Since the control policy is restricted to be linear in this paper, finding an optimal policy is tantamount to finding the minimizer K_{t+1} of the optimization problem

$$\min_K \sum_{t=t_i+1}^{t_{i+1}} \left(x_t^\top K^\top R K x_t + x_t^\top Q x_t + x_t^\top (A+BK)^\top P_{t+1} (A+BK) x_t\right), \tag{17}$$

where $t_i, t_{i+1} \in \mathcal{T}$. This is a quadratic optimization problem in K because $\{x_t\}$, Q, R, and P_{t+1} are all known quantities in the window $\{t_i + 1, t_i + 2, \ldots, t_{i+1}\}$. Note that K_{t+1} can be updated recursively within each learning window $t_i \leq t \leq t_{i+1}$ using P_{t+1} for these time instants. Since (17) is a quadratic problem, using Newton-type iterative solvers are expected to yield quick convergence; in this case, in one step.

IV. Constrained ADP

In this section, we elucidate upon how to use invariant sets to generate new control policies that are both stabilizing and constraint satisfying. We also propose an algorithm for implementing a constrained ADP in a data-driven manner.

We begin with the following definition.

Definition 1 (CAIS)**.** *A non-empty set $\mathcal{E}$ within the admissible state space $\mathbb{X}$ is a constraint admissible invariant set (CAIS) for the closed-loop system* (1) *under a control law $u = Kx$ if, for every initial condition $x_{t_0} \in \mathcal{E}$, all subsequent states $x_t \in \mathcal{E}$ and inputs $Kx_t \in \mathbb{U}$ for all $t \geq t_0$.*

According to Assumption 1, the ADP iteration is initialized with a stabilizing linear controller K_0. This stabilizing controller renders a subset of the state-space invariant while satisfying state and input constraints. In particular, there exists an ellipsoidal region

$$\mathcal{E}_{P_0}^\rho = \{x : x^\top P_0 x \leq \rho\},$$

such that $\mathcal{E}_{P_0}^\rho \subset \mathbb{X}$ and $K_0 \mathcal{E}_{P_0}^\rho \subset \mathbb{U}$. We assume that the value function matrix P_0 defining the initial CAIS ellipsoid $\mathcal{E}_{P_0}^\rho$ is known. This is encapsulated formally herein.

Assumption 2 (Constrained ADP)**.** *There exists a symmetric positive definite matrix P_0 such that $\mathcal{E}_{P_0}^\rho \subset \mathbb{X}$ is a CAIS for the closed-loop system (1) under the initial control policy $u = K_0 x$, and $K_0 x \in \mathbb{U}$ for all $x \in \mathcal{E}_{P_0}^\rho$.*

A. Model-based constrained policy iteration

1) Model-based constrained policy evaluation: Let

$$\mathcal{J}_t(P) := (A + BK_t)^\top P (A + BK_t) - P + Q + K_t^\top R K_t.$$

In order to implement constrained model-based policy evaluation (that is, obtain P_{t+1} from K_t and P_t), we need to solve

the following semi-definite programming problem:

$$P_{t+1}, \rho_{t+1} = \arg\min_{P,\rho>0} \|\mathcal{J}_t(P)\| \tag{18a}$$

s.t.

$$(A + BK_t)^\top P(A + BK_t) - \lambda P \preceq 0 \tag{18b}$$

$$x_t^\top P x_t \leq \rho \tag{18c}$$

$$(c_k^\top + d_k^\top K_t)^\top \rho\, (c_k^\top + d_k^\top K_t) \preceq P \tag{18d}$$

$$\alpha_1 \mathbf{I} \preceq P \preceq \alpha_2 \mathbf{I} \tag{18e}$$

for some $\alpha_1, \alpha_2 > 0$ and $\lambda < (\alpha_1/\alpha_2)^{2/N}$. Here, $k \in \{1, \ldots, r\}$. Note that ensuring this problem is convex involves fixing the scalars α_1 and α_2, and pre-computing λ.

The rationale behind (18) can be explained as follows. Since (18e) ensures that $P \succ 0$, this constraint, along with the objective (18a), is equivalent to (10), which is identical to the unconstrained policy evaluation step. Therefore, constraint satisfaction is made possible by equipping the constraints (18b)–(18d) and $\rho > 0$.

The inequality (18b) ensures that the value function is contractive, and therefore, non-increasing for every $t \geq t_0$. To see this, we multiply (18b) by $x^\top$ and x from the left and right, respectively, which yields

$$x_{t+1}^\top P x_{t+1} - x_t^\top P x_t \leq -(1-\lambda)\, x_t^\top P x_t < 0,$$

for any t, since $0 < \lambda < 1$. This is a key ingredient to ensure that the updated control policies will provide stability certificates for the closed-loop system. The two inequalities (18c) and (18d) enforce that the state and input constraints with the current policy are satisfied in spite of the value function update, given the current state x_t. The condition (18e) ensures that the value function matrix P is positive definite, and the positive scalar ρ allows the selection of sub- and super-level sets of the Lyapunov function.

2) Model-based constrained policy improvement: Unlike unconstrained policy iteration, adding state and input constraints could result in nonlinear optimal control policies. In this paper, we restrict ourselves to design linear control policies of the form $u = Kx$, and hence, our optimal policy improvement step is analogous to the unconstrained case (11), that is,

$$K_{t+1}^\star = -\left(R + B^\top P_{t+1} B\right)^{-1} B^\top P_{t+1} A. \tag{19}$$

Remark 2. In spite of parameterizing via linear control policies, our controller is actually nonlinear since K_t depends on P_t which depends on the states through (18). ■

We adopt a backtracking strategy in order to update the current constrained policy K_t to a new constrained policy K_{t+1} that is as close as possible to the unconstrained policy $K_{t+1}^\star$ in (19) that enforces state and input constraints.

A simplified version of this backtracking strategy is outlined in Algorithm 1.

A particular benefit of our proposed method is that it enables both expansion, contraction, and rotation of the constraint admissible invariant sets. This is important in reference tracking for instance where a more aggressive controller is required when the state is near the boundary of the state constraints. This could also be useful for applying this approach to nonlinear systems where (A, B) is a local linear approximation of the globally nonlinear dynamics. Our approach allows the ellipsoidal invariant sets to adapt its size and shape based on the local vector field. For example, suppose $\mathcal{E}_{P_\infty}$ denote the CAIS that is associated with the constrained optimal control policy K_∞ and optimal value function defined by P_∞. Also suppose that we have an initial admissible policy K_0 whose associated CAIS $\mathcal{E}_{P_0}^\rho$ is contained within $\mathcal{E}_{P_\infty}$. Then our proposed method will generate a sequence of $\mathcal{E}_{P_t}$ such that these invariant sets will expand, contract, and rotate as necessary until the sequence of invariant sets $\{\mathcal{E}_{P_t}\}$ converges to the optimal $\mathcal{E}_{P_\infty}$.

Algorithm 1 Constrained Policy Improvement: Backtracking

Input: Desired policy $K_{t+1}^\star$ and current constrained policy K_t.

1: $K_{t+1} \leftarrow K_{t+1}^\star, \quad \alpha \leftarrow 1.$
2: **while** $(c_i^\top + d_i^\top K_{t+1})^\top \rho\, (c_i^\top + d_i^\top K_{t+1}) \not\preceq P_{t+1}$ **do**
3: $\quad \alpha \leftarrow \beta\alpha$, where $0 < \beta < 1$.
4: $\quad K_{t+1} \leftarrow K_t + \alpha\,(K_{t+1}^\star - K_t)$.

B. Data-driven constrained policy iteration

In order to obtain a data-driven implementation of the constrained ADP method, one needs to gather a sequence of state-input data points $\{\bar{x}_t, \bar{u}_t, \bar{x}_{t+1}\}$ and control policies $\{\bar{K}_t\}$ which will be used to update the value function matrix and control policies at the learning time instants defined by $\mathcal{T}$ in (12). Given the discrete-time system dynamics in (1), the relation between these data points is given by

$$\bar{x}_{t+1} = A\bar{x}_t + B\bar{u}_t = A\bar{x}_t + B\left(\bar{K}_t \bar{x}_t + \nu_t\right), \tag{20}$$

where ν_t represents a known exploration noise signal that ensures the system (20) is persistently excited; see [2]. To arrive at a more compact notation, let us define $\tilde{x}_{t+1} := \bar{x}_{t+1} - B\nu_t$ and $\tilde{u}_t := \bar{K}_t\, \bar{x}_t$ such that

$$\tilde{x}_{t+1} = A\bar{x}_t + B\tilde{u}_t = (A + B\bar{K}_t)\bar{x}_t. \tag{21}$$

1) Data-driven constrained policy evaluation: Consider the i-th learning cycle, occuring at the time instant $t_i \in \mathcal{T}$. Let

$$\bar{\mathcal{J}}_t(P) := \tilde{x}_{t+1}^\top P \tilde{x}_{t+1} - \bar{x}_t^\top P \bar{x}_t + \bar{x}_t Q \bar{x}_t + \tilde{u}_t^\top R\, \tilde{u}_t.$$

The data-driven analogue of the constrained policy evaluation step discussed in the previous section is given by the following semi-definite program (SDP) with α_1 and α_2 fixed:

$$\bar{P}_{t+1}, \rho_{t+1} := \arg\min_{\rho>0, P} \ \frac{1}{2} \sum_{t=0}^{t_{i+1}-1} \left(\bar{\mathcal{J}}_t(P)\right)^2 - \lambda_\rho \rho \tag{22a}$$

s.t.

$$\tilde{x}_{t+1}^\top P \tilde{x}_{t+1} - \lambda \bar{x}_t^\top P \bar{x}_t \leq 0 \tag{22b}$$

$$x_{t_{i+1}}^\top P x_{t_{i+1}} \leq \rho \tag{22c}$$

$$(c_k^\top + d_k^\top \bar{K}_t)^\top \rho\, (c_k^\top + d_k^\top \bar{K}_t) \preceq P \tag{22d}$$

$$\alpha_1 \mathbf{I} \preceq P \preceq \alpha_2 \mathbf{I} \tag{22e}$$

for $t \in \{t_i+1, t_i+2, \ldots, t_{i+1}\}$ and $k \in \{1, \ldots, r\}$. Note that the final four inequalities in (22) are exactly the set of inequalities presented in (18) with the model information replaced by state and input data. Also, replacing $\tilde{x}_{t+1}$ in (22b) with $(A + B\bar{K}_t)\bar{x}_t$ using equality (21) shows that it is equivalent to the inequality (18b).

2) Data-driven constrained policy improvement: Once a value function is found whose sub-level set is constraint admissible, the corresponding policy K_{t+1} is to be computed. If A and B are known, this step would be easy: indeed, one could utilize Eq. (19) to this end. However, since only B is known (by assumption), we resort to a data-driven iterative update methodology for generating the new policy.

Given the current policy K_t, we gather another batch of measurements $\{\bar{x}_t, \bar{u}_t, \bar{K}_t, \bar{x}_{t+1}\}_{t=t_i+1,\ldots,t_{i+1}}$ where a new policy $\bar{K}_t$ is the optimizer of the least squares problem

$$\min_{K} \ \frac{1}{2} \sum_{t=t_i+1}^{t_{i+1}} \bar{x}_t^\top \left(K^\top R K + (A+BK)^\top \bar{P}_{t+1}(A+BK)\right) \bar{x}_t. \tag{23}$$

The problem (23) can be solved in a data-driven manner efficiently using a real-time recursive least squares (RLS) implementation [26]

$$H_{t+1} = H_t + \bar{x}_t \bar{x}_t^\top \otimes (R + B^\top \bar{P}_{t+1} B), \tag{24a}$$

$$g_{t+1} = \bar{x}_t \otimes (R\bar{K}_t \bar{x}_t + B^\top \bar{P}_{t+1} \tilde{x}_{t+1}), \tag{24b}$$

$$\mathrm{vec}(\bar{K}_{t+1}) = \mathrm{vec}(\bar{K}_t) - \beta_t \, H_{t+1}^{-1} \, g_{t+1}, \tag{24c}$$

for $t = t_i+1, \ldots, t_{i+1}-1$. Note that (24) is solved without knowledge of A using the updates. Also, the starting value for the Hessian matrix is chosen as the identity matrix $\rho \mathbf{I}$ and $\rho > 0$ to ensure non-singularity. The step size β_t is typically equal to one, even though a smaller step $\beta_t \leq 1$ can be chosen, e.g., based on the backtracking procedure in order to impose the affine state and input constraints in (22d) for each updated control policy $\bar{K}_{t+1}$. The Hessian matrix in (24) can be reset to $H = q\mathbf{I} \succ 0$ whenever a new value function is obtained from solving the SDP in (22). Note that (24a) corresponds to a rank-m matrix update, where m denotes the number of control inputs. Therefore, its matrix inverse H_{t+1}^{-1} can be updated efficiently using the Sherman-Morrison formula, for example, in the form of m rank-one updates.

3) Algorithm Implementation: Pseudocode: The general procedure corresponds to the sequence of high-level steps:

(i) We require an initial stabilizing policy $\bar{K}_0$ and a corresponding constraint admissible invariant set (CAIS) $\mathcal{E}_{P_0}^{\rho}$; see Assumptions 1 and 2.

(ii) Obtain a sequence of at least $t_i + 1$ data points $\{\bar{x}_t, \bar{u}_t, \bar{K}_t, \bar{x}_{t+1}\}$ while the system is persistently excited and compute a new ellipsoidal set defined by the matrix $\bar{P}_{t+1}$ by solving the least squares SDP in (22).

(iii) At each time step, perform the policy improvement step to compute $\bar{K}_{t+1}$ based on the real-time recursive least squares method as described in (24), in combination with the backtracking procedure as in Algorithm 1 to enforce state and input constraints.

(iv) If the policy improvement has converged based on the condition $\|g_t\| \leq \epsilon$, return to step (ii).

V. Constraint Satisfaction, Stability, and Algorithm Convergence

We present theoretical guarantees for our proposed constrained policy iteration. For the data-driven case, we adhere to the standard assumption that the system is persistently excited. The following theorem demonstrates constraint enforcement and stability guarantees of the closed-loop system.

Theorem 1. *Suppose Assumptions 1 and 2 hold. Then the system* (1) *in closed-loop with the time-varying controller* $u_t = K_t x_t$ *has the following properties:*

(i) The constraints $x_t \in \mathbb{X}$ *and* $u_t \in \mathbb{U}$ *are satisfied for all* $t \in \mathbb{N}$.

(ii) The closed-loop system is asymptotically stable.

Previous stability results for approximate dynamic programming rely on the tacit assumption that the learning converges after a finite number of batch iterations (typically one). In other words, the adaptive controller only works because it stops adapting. In contrast, for constraint satisfaction, the controller may need to continually adapt since the set of active constraints will change as the state evolves. This necessitates the development of a more involved set of conditions to ensure that feedback control loop and the learning loop do not destabilize each other.

Theorem 2. *Suppose Assumptions 1 and 2 hold. Let* $\alpha_1 \leq \underline{\sigma}(P_\infty)$, $\alpha_2 \geq \underline{\sigma}(P_\infty)$, *and*

$$\lambda \geq \overline{\sigma}\left(I - P_\infty^{-1/2}(Q + K_\infty^\top R K_\infty) P_\infty^{-1/2}\right).$$

Under the iteration (18) *and* (19)*, the value* P_t *and policy* K_t *converge to the LQR cost-to-go* P_∞ *and controller gain* K_∞. *That is,*

$$\lim_{t\to\infty} P_t = P_\infty \quad \textit{and} \quad \lim_{t\to\infty} K_t = K_\infty. \tag{25}$$

VI. Numerical Example

A. Linear system with two states, one control input

We randomly generate controllable systems of the form (1) to test the proposed algorithm. A particular realization of these randomly generated systems, $A = \left[\begin{smallmatrix} 1.1387 & 0.0491 \\ -0.8680 & 0.9679 \end{smallmatrix}\right]$, $B = \left[\begin{smallmatrix} -0.5507 \\ 0.0758 \end{smallmatrix}\right]$ is investigated to illustrate constraint satisfaction and stability of the algorithm. Of course, A is unknown (and unstable), B is known, and it is verified that (A, B) is a controllable pair. The admissible state space is given by $\mathbb{X} = \{x \in \mathbb{R}^2 : \|x\|_\infty \leq 1\}$, and the operational cost is parameterized by $Q = I_2$ and $R = 0.5$. For learning, the window length is fixed at $N = 8$ samples ($\mathcal{T} = \{8, 16, 24, \ldots\}$), and the regularization parameter for policy updating is given by $\rho_K = 10^{-4}$. Persistence excitation is ensured by generating uniformly distributed noise bounded within $[-0.02, 0.02]$. An initial policy is generated that satisfies state constraints using the randomly chosen cost matrices that are distinct from Q

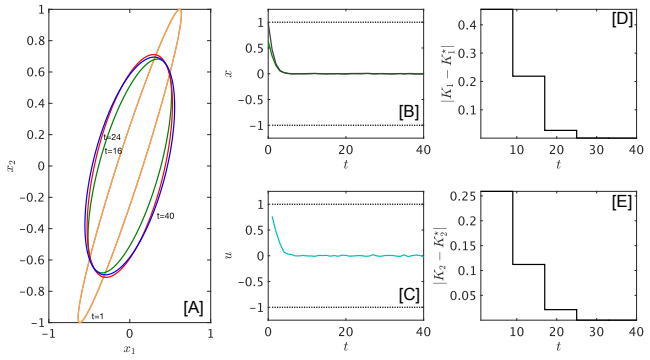

Fig. 1. Results of constrained ADP for 2-state dynamic system: [A] Sequence of invariant sets learned on-line. Each set is labeled with the time iteration t when it was learned. [B] State evolution (x_1: blue, x_2: red) with constraints (black, dashed). [C] Control input (blue) evolution with constraints (black, dashed). [D, E] Convergence of learned LQR policy to the true LQR policy.

and R, and an initial condition is generated randomly on the boundary of the initial domain of attraction. Therefore, the initial state is ensured to be within $\mathbb{X}$ but sufficiently far from the origin to require non-trivial control for stabilization.

The results of the constrained policy iteration algorithm are illustrated in Fig. 1. In Fig. 1[A], a sequence of ellipsoids generated by our proposed algorithm is presented. Note that the ellipsoids generated in subsequent learning cycles after the first (the orange elongated ellipsoid) are not mere sub- or super-level sets of the initial ellipsoid; instead, the policy iterator allows for contractions and expansions on both x_1 and x_2 axes until the true policy is learned. As evident from subplots [B] and [C], state constraints are not violated throughout the learning procedure. The subplots [D, E] demonstrate the convergence of a sub-optimal initial control policy at $t = 0$ to the true and optimal LQR policy at around $t = 24$, after three learning cycles.

VII. Concluding Remarks

In this paper, we provide a methodology for implementing constraint satisfying policy iteration for continuous-time, continuous-state systems via invariant sets. Benefits of our approach include computational tractability, and safety guarantees through constraint satisfaction. In future work, we will extend this framework to more general dynamical systems.

References

[1] 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.

[2] F. L. Lewis, D. Vrabie, and K. G. Vamvoudakis, "Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controllers," *IEEE Control Systems*, vol. 32, no. 6, pp. 76–105, 2012.

[3] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, 2nd ed. Athena Scientific, 2000.

[4] B. Kiumarsi, K. G. Vamvoudakis, H. Modares, and F. L. Lewis, "Optimal and Autonomous Control Using Reinforcement Learning: A Survey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 6, pp. 2042–2062, 2018.

[5] G. Hewer, "An iterative technique for the computation of the steady state gains for the discrete optimal regulator," *IEEE Transactions on Automatic Control*, vol. 16, no. 4, pp. 382–384, aug 1971. [Online]. Available: http://ieeexplore.ieee.org/document/1099755/

[6] D. L. Kleinman, "On an iterative technique for riccati equation computations," *IEEE Trans. on Automatic Control*, vol. 13, no. 1, pp. 114–115, 1968.

[7] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 2011.

[8] S. J. Bradtke, B. E. Ydstie, and A. G. Barto, "Adaptive linear quadratic control using policy iteration," in *Proc. of the American Control Conference*, vol. 3. Citeseer, 1994, pp. 3475–3475.

[9] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[10] P. J. Werbos, "Neural networks for control and system identification," in *Proc. of the 28th IEEE Conf. Dec. and Control*. IEEE, 1989, pp. 260–265.

[11] D. Vrabie, O. Pastravanu, M. Abu-Khalaf, and F. L. Lewis, "Adaptive optimal control for continuous-time linear systems based on policy iteration," *Automatica*, vol. 45, no. 2, pp. 477–484, 2009.

[12] T. Landelius, "Reinforcement learning and distributed local model synthesis," Ph.D. dissertation, Linköping University Electronic Press, 1997.

[13] A. Al-Tamimi, F. L. Lewis, and M. Abu-Khalaf, "Model-free Q-learning designs for linear discrete-time zero-sum games with application to $\mathcal{H}_\infty$ control," *Automatica*, vol. 43, no. 3, pp. 473–481, 2007.

[14] R. Kamalapurkar, H. Dinh, S. Bhasin, and W. E. Dixon, "Approximate optimal trajectory tracking for continuous-time nonlinear systems," *Automatica*, vol. 51, pp. 40–48, 2015.

[15] W. Gao and Z.-P. Jiang, "Adaptive dynamic programming and adaptive optimal output regulation of linear systems," *IEEE Transactions on Automatic Control*, vol. 61, no. 12, pp. 4164–4169, 2016.

[16] Q. Zhang, D. Zhao, and Y. Zhu, "Data-driven adaptive dynamic programming for continuous-time fully cooperative games with partially constrained inputs," *Neurocomputing*, vol. 238, pp. 377–386, 2017.

[17] H. Zhang, H. Jiang, C. Luo, and G. Xiao, "Discrete-time nonzero-sum games for multiplayer using policy-iteration-based adaptive dynamic programming algorithms," *IEEE transactions on cybernetics*, vol. 47, no. 10, pp. 3331–3340, 2017.

[18] A. Al-Tamimi, F. L. Lewis, and M. Abu-Khalaf, "Discrete-time nonlinear hjb solution using approximate dynamic programming: Convergence proof," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 38, no. 4, pp. 943–949, 2008.

[19] A. Heydari, "Revisiting Approximate Dynamic Programming and its Convergence," *IEEE Transactions on Cybernetics*, vol. 44, no. 12, pp. 2733–2743, 2014.

[20] M. V. Kothare, V. Balakrishnan, and M. Morari, "Robust constrained model predictive control using linear matrix inequalities," *Automatica*, vol. 32, no. 10, pp. 1361–1379, 1996.

[21] A. Chakrabarty, V. C. Dinh, M. J. Corless, A. E. Rundell, S. H. Zak, G. T. Buzzard *et al.*, "Support vector machine informed explicit nonlinear model predictive control using low-discrepancy sequences." *IEEE Trans. Automat. Contr.*, vol. 62, no. 1, pp. 135–148, 2017.

[22] K. Berntorp, A. Weiss, C. Danielson, I. V. Kolmanovsky, and S. Di Cairano, "Automated driving: Safe motion planning using positively invariant sets," in *Proc. of the 20th Int. Conf. on Intelligent Transportation Sys. (ITSC)*. IEEE, 2017, pp. 1–6.

[23] F. Berkenkamp, R. Moriconi, A. P. Schoellig, and A. Krause, "Safe learning of regions of attraction for uncertain, nonlinear systems with gaussian processes," *Proc. of the IEEE Conf. Decision and Control*, pp. 4661–4666, 2016.

[24] 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 (ACC)*. IEEE, 2018, pp. 6390–6395.

[25] D. Piga, S. Formentin, and A. Bemporad, "Direct data-driven control of constrained systems," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 4, pp. 1422–1429, 2018.

[26] L. Ljung, *System identification: Theory for the User*. Upper Saddle River, N.J.: Prentice Hall, 1999.