Linear inverse reinforcement learning in continuous time and space

Rushikesh Kamalapurkar

Abstract— **This paper develops a data-driven inverse reinforcement learning technique for a class of linear systems to estimate the cost function of an agent online, using input-output measurements. A simultaneous state and parameter estimator is utilized to facilitate output-feedback inverse reinforcement learning, and cost function estimation is achieved up to multiplication by a constant.**

I. Introduction

Seamless cooperation between humans and autonomous agents is a vital yet challenging aspect of modern robotic systems. Effective cooperation between humans and autonomous systems can be achieved if the autonomous systems are capable of learning to act by observing other cognitive entities. Based on the premise that a cost (or reward) function fully characterizes the intent of the demonstrator, a method to learn the cost function from observations is developed in this paper. The cost-estimation problem first appears in [1] in a linear-quadratic regulation (LQR) setting, and a solution is provided in [2] via linear matrix inequalities. For nonlinear systems and cost functions, computation of closed-form solutions is generally intractable, and hence, approximate solutions are sought.

In [3]–[6], the cost function of a Markov decision process (MDP) is learned using inverse reinforcement learning (IRL). It is demonstrated that the IRL problem is inherently ill-posed in the sense that it has multiple possible solutions, including the trivial ones. To overcome the degeneracy, the cost function that differentiates the optimal behavior from the suboptimal behaviors *by a margin* is sought. In [7] the maximum entropy principle (cf. [8]) is utilized to solve the ill-posed IRL problem for deterministic MDPs. In [9] a causal version of the maximum entropy principle is developed and utilized to solve IRL problems in a fully stochastic setting. An IRL algorithm based on minimization of the Kullback-Leibler divergence between the empirical distribution of trajectories obtained from a baseline policy and the trajectories obtained from the cost-based policy is developed in [10].

In the past two decades, Bayesian [11], natural gradient [12], game theoretic [13], and feature construction based methods [14] have also been developed for IRL. IRL is extended to problems with locally optimal demonstrations in [15] using likelihood optimization and to problems with nonlinear cost functions in [16] using Gaussian processes (GP). Another GP-based IRL algorithm that increases the efficiency and applicability of IRL techniques by autonomously segmenting the overall task into sub-goals is developed in [17]. Over the years, intent-based approaches such as IRL have been successfully utilized to teach UxVs and humanoid robots to perform specific maneuvers in an *offline* setting [4], [18], [19].

Offline approaches are ill suited for applications where teams of autonomous agents with varying levels of autonomy work together to achieve evolving tasks. For example, consider a fleet of unmanned air vehicles where only a few of the vehicles are remotely controlled by human operators and the rest are fully autonomous and capable of synthesizing their own control policies based on the task. If the tasks are subject to change and are known only to the human operators, the autonomous agents need the ability to identify the changing objectives from observations in real-time.

Motivated by recent progress in real-time reinforcement learning (see, e.g., [20]–[24]), this paper develops an output-feedback IRL technique for a class of linear systems to estimate the cost function online using input-output measurements. The paper is organized as follows. Section II details the notation used throughout the paper. Section III formulates the problem. Section IV details the development of a simultaneous state and parameter estimator that facilitates output-feedback cost estimation. Section V formulates the error signal that is utilized in Section VI to achieve online IRL. Section VII details the purging algorithm used to facilitate IRL in conjunction with the state and parameter estimator developed in Section IV. Section VIII analyzes the convergence of the developed algorithm and Section X concludes the paper.

II. Notation

The $n-$dimensional Euclidean space is denoted by $\mathbb{R}^n$. Elements of $\mathbb{R}^n$ are interpreted as column vectors and $(\cdot)^T$ denotes the vector transpose operator. The set of positive integers excluding 0 is denoted by $\mathbb{N}$. For $a \in \mathbb{R}$, $\mathbb{R}_{\geq a}$ denotes the interval $[a, \infty)$ and $\mathbb{R}_{>a}$ denotes the interval (a, ∞). Unless otherwise specified, an interval is assumed to be right-open. If $a \in \mathbb{R}^m$ and $b \in \mathbb{R}^n$ then $[a; b]$ denotes the concatenated vector $\begin{bmatrix} a \\ b \end{bmatrix} \in \mathbb{R}^{m+n}$. The notations and I_n and 0_n denote $n \times n$ identity matrix and the zero element of $\mathbb{R}^n$, respectively. Whenever clear from the context, the subscript n is suppressed.

III. Problem formulation

Consider an agent under observation with linear dynamics of the form

$$\dot{p} = q, \quad \dot{q} = Ax + Bu, \tag{1}$$

Rushikesh Kamalapurkar is with the School of Mechanical and Aerospace Engineering at the Oklahoma State University. Email: rushikesh.kamalapurkar@okstate.edu

where $p : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ denotes the generalized position, $q : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ denotes the generalized velocity, $u : \mathbb{R}_{\geq 0} \to \mathbb{R}^m$ denotes the control input, $x := [p^T, q^T]^T$ denotes the state, and $A \in \mathbb{R}^{n\times 2n}$ and $B \in \mathbb{R}^{n\times m}$ denote the *unknown* constant system matrices. Assume that the pair (A', B') is controllable, where $A' := \begin{bmatrix} 0_{n\times n} & \mathrm{I}_{n\times n} \\ & A \end{bmatrix}$, $B' := \begin{bmatrix} 0_{n\times m} \\ B \end{bmatrix}$. The agent under observation executes a policy that minimizes the infinite horizon cost

$$J\left(x_0, u\left(\cdot\right)\right) \triangleq \int\limits_0^\infty r\left(x\left(\tau; x_0, u\left(\cdot\right)\right), u\left(\tau\right)\right) \mathrm{d}\tau \tag{2}$$

where $\tau \mapsto x\left(\tau; x_0, u\left(\cdot\right)\right)$ denotes the trajectory of (1) under the control signal $u\left(\cdot\right)$ starting from the initial condition x_0 and $r : \mathbb{R}^{2n} \times \mathbb{R}^m \to \mathbb{R}$ denotes the *unknown* instantaneous cost function defined as $r\left(x, u\right) = Q\left(x\right) + u^T R u$, where $R \in \mathbb{R}^{m\times m}$ is a constant positive definite matrix and $Q : \mathbb{R}^{2n} \to \mathbb{R}^{2n}$ is a positive definite function such that an optimal policy $u^* : \mathbb{R}^{2n} \to \mathbb{R}^m$ exists. For ease of exposition, it is further assumed that $R = \mathrm{diag}\left\{r_1, \cdots, r_m\right\}$, and a basis $\sigma : \mathbb{R}^{2n} \to \mathbb{R}^L$ is known for Q such that for some $W_Q^* \in \mathbb{R}^L$,

$$Q\left(x\right) = \left(W_Q^*\right)^T \sigma_Q\left(x\right), \forall x \in \mathbb{R}^{2n}. \tag{3}$$

The objective of the observer is to estimate the cost function, r, using measurements of the generalized position and the control input. In the following, a model-based approximate dynamic programming approach is developed to achieve the stated objective. To facilitate model-based approximate dynamic programming, the state, x, and the parameters, A and B, of the agent are estimated from the input-output measurements using a simultaneous state and parameter estimator. The state and the parameters are then utilized in an approximate dynamic programming scheme to estimate the cost.

IV. Simultaneous state and parameter estimator

The simultaneous state and parameter estimator developed by the authors in [25] is utilized in this result. This section provides a brief overview of the same for completeness. For further details, the readers are directed to [25].

To facilitate parameter estimation, let $A_1,\ A_2 \in \mathbb{R}^{n\times n}$ be matrices such that $A =: [A_1,\ A_2]$. The dynamics in (1) can be rearranged to form the linear error system

$$\mathcal{F}\left(t\right) = \mathcal{G}\left(t\right)\theta,\ \forall t \in \mathbb{R}_{\geq 0}. \tag{4}$$

In (4), θ is a vector of unknown parameters, defined as $\theta \triangleq \begin{bmatrix}\mathrm{vec}\left(A_1\right)^T & \mathrm{vec}\left(A_2\right)^T & \mathrm{vec}\left(B\right)^T\end{bmatrix}^T \in \mathbb{R}^{2n^2+mn}$, where $\mathrm{vec}\left(\cdot\right)$ denotes the vectorization operator and the matrices $\mathcal{F} : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ and $\mathcal{G} : \mathbb{R}_{\geq 0} \to \mathbb{R}^{n\times\left(2n^2+mn\right)}$ are defined as

$$\mathcal{F}\left(t\right) \triangleq \begin{cases} p\left(t - T_2 - T_1\right) - p\left(t - T_1\right) \\ \quad + p\left(t\right) - p\left(t - T_2\right), & t \in \left[T_1 + T_2, \infty\right), \\ 0 & t < T_1 + T_2. \end{cases}$$

$$\mathcal{G}\left(t\right) \triangleq \begin{bmatrix}\left(F\left(t\right)\otimes \mathrm{I}_n\right)^T & \left(G\left(t\right)\otimes \mathrm{I}_n\right)^T & \left(U\left(t\right)\otimes \mathrm{I}_n\right)^T\end{bmatrix},$$

where $\otimes$ denotes the Kronecker product. The matrices F, G, and U are defined as $F\left(t\right) := \mathcal{I}p\left(t\right)$, $\quad G\left(t\right) := \mathcal{J}p\left(t\right) - \mathcal{J}p\left(t - T_1\right)$, $\quad U\left(t\right) := \mathcal{I}u\left(t\right)$, for $t \in \left[T_1 + T_2, \infty\right)$, and $F\left(t\right) = G\left(t\right) = U\left(t\right) = 0$, for $t < T_1 + T_2$, where $\mathcal{I} := p \mapsto \int_{t-T_2}^t \int_{\sigma - T_1}^\sigma p\left(\tau\right) \mathrm{d}\tau\, \mathrm{d}\sigma$, and $\mathcal{J} := p \mapsto \int_{t-T_2}^t \left(p\left(\sigma\right)\right) \mathrm{d}\sigma$.

For ease of exposition, it is assumed that a history stack, i.e., a set of ordered pairs $\left\{\left(\mathcal{F}_i, \mathcal{G}_i\right)\right\}_{i=1}^M$ such that

$$\mathcal{F}_i = \mathcal{G}_i\theta,\ \forall i \in \left\{1, \cdots, M\right\}, \tag{5}$$

is available a priori. A history stack $\left\{\left(\mathcal{F}_i, \mathcal{G}_i\right)\right\}_{i=1}^M$ is called *full rank* if there exists a constant $\underline{g} \in \mathbb{R}$ such that

$$0 < \underline{g} < \lambda_{\min}\left\{\mathscr{G}\right\}, \tag{6}$$

where the matrix $\mathscr{G} \in \mathbb{R}^{\left(2n^2+mn\right)\times\left(2n^2+mn\right)}$ is defined as $\mathscr{G} := \sum_{i=1}^M \mathcal{G}_i^T \mathcal{G}_i$. The concurrent learning update law to estimate the unknown parameters is then given by

$$\dot{\hat{\theta}}\left(t\right) = k_\theta \Gamma\left(t\right) \sum_{i=1}^M \mathcal{G}_i^T \left(\mathcal{F}_i - \mathcal{G}_i \hat{\theta}\left(t\right)\right), \tag{7}$$

where $k_\theta \in \mathbb{R}_{>0}$ is a constant adaptation gain and $\Gamma : \mathbb{R}_{\geq 0} \to \mathbb{R}^{\left(2n^2+mn\right)\times\left(2n^2+mn\right)}$ is the least-squares gain updated using the update law

$$\dot{\Gamma}\left(t\right) = \beta_1 \Gamma\left(t\right) - k_\theta \Gamma\left(t\right) \sum_{i=1}^M \mathcal{G}_i^T \mathcal{G}_i \Gamma\left(t\right). \tag{8}$$

Using arguments similar to [26, Corollary 4.3.2], it can be shown that provided $\lambda_{\min}\left\{\Gamma^{-1}\left(0\right)\right\} > 0$, the least squares gain matrix satisfies

$$\underline{\Gamma}\, \mathrm{I}_{\left(2n^2+mn\right)} \leq \Gamma\left(t\right) \leq \overline{\Gamma}\, \mathrm{I}_{\left(2n^2+mn\right)}, \tag{9}$$

where $\underline{\Gamma}$ and $\overline{\Gamma}$ are positive constants.

To facilitate parameter estimation based on a prediction error, a state observer is developed in the following. To facilitate the design, the dynamics in (1) are expressed in the form $\dot{p}\left(t\right) = q\left(t\right)$, $\dot{q}\left(t\right) = Y\left(x\left(t\right), u\left(t\right)\right)\theta$, where $Y : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^{n\times\left(2n^2+mn\right)}$ is defined as

$$Y\left(x, u\right) = \begin{bmatrix}\left(p \otimes \mathrm{I}_n\right)^T & \left(q \otimes \mathrm{I}_n\right)^T & \left(u \otimes \mathrm{I}_n\right)^T\end{bmatrix}.$$

The adaptive state observer is then designed as

$$\begin{aligned} &\dot{\hat{p}}\left(t\right) = \hat{q}\left(t\right), \quad \hat{p}\left(0\right) = p\left(0\right), \\ &\dot{\hat{q}}\left(t\right) = Y\left(x\left(t\right), u\left(t\right)\right)\hat{\theta}\left(t\right) + \nu\left(t\right), \quad \hat{q}\left(0\right) = 0, \end{aligned} \tag{10}$$

where $\hat{p} : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$, $\hat{q} : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$, $\hat{x} : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$, and $\hat{\theta} : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ are estimates of p, q, x, and θ, respectively, ν is the feedback component of the identifier, to be designed later, and the prediction error $\tilde{p} : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ is defined as $\tilde{p}\left(t\right) = p\left(t\right) - \hat{p}\left(t\right)$.

The update law for the generalized velocity estimate depends on the entire state x. However, using the structure of the matrix Y and integrating by parts, the observer can be implemented without using generalized velocity measurements. Using an integral form of (10), the update law in (10) can be implemented without generalized velocity measurements as

$$\hat{q}(t) = \int_0^t (u(\tau) \otimes \mathrm{I}_n)^T \operatorname{vec}\left(\hat{B}(\tau)\right) \mathrm{d}\tau + \int_0^t \nu(\tau) \, \mathrm{d}\tau$$
$$+ \hat{q}(0) + \int_0^t (p(\tau) \otimes \mathrm{I}_n)^T \left(\operatorname{vec}\left(\hat{A}_1(\tau)\right) - \operatorname{vec}\left(\dot{\hat{A}}_2(\tau)\right)\right) \mathrm{d}\tau$$
$$+ (p(t) \otimes \mathrm{I}_n)^T \operatorname{vec}\left(\hat{A}_2(t)\right) - (p(0) \otimes \mathrm{I}_n)^T \operatorname{vec}\left(\hat{A}_2(0)\right) \quad (11)$$

To facilitate the design of the feedback component ν, let

$$r(t) = \tilde{q}(t) + \alpha \tilde{p}(t) + \eta(t), \quad (12)$$

where $\alpha > 0$ is a constant observer gain and the signal η is added to compensate for the fact that the generalized velocity state, q, is not measurable. Based on the subsequent stability analysis, the signal η is designed as the output of the dynamic filter

$$\dot{\eta}(t) = -\beta \eta(t) - k r(t) - \alpha \tilde{q}(t), \quad \eta(0) = 0, \quad (13)$$

and the feedback component ν is designed as

$$\nu(t) = \tilde{p}(t) - (k + \alpha + \beta) \eta(t), \quad (14)$$

where $\beta > 0$ and $k > 0$ are constant observer gains. The design of the signals η and ν to estimate the state from output measurements is inspired by the $p-$filter (cf. [27]). Similar to the update law for the generalized velocity, using the the fact that $\tilde{p}(0) = 0$, the signal η can be implemented using the integral form

$$\eta(t) = -\int_0^t (\beta + k) \eta(\tau) \, \mathrm{d}\tau - \int_0^t k\alpha \tilde{p}(\tau) \, \mathrm{d}\tau - (k + \alpha) \tilde{p}(t). \quad (15)$$

Using a Lyapunov-based analysis, it can be shown that the developed parameter and state estimation results in exponential convergence of the state and parameter estimation errors to zero. For a detailed analysis of the developed state and parameter estimator, see [25].

V. Inverse Bellman Error

Since the agent under observation makes optimal decisions, and since the Hamiltonian $H : \mathbb{R}^{2n} \times \mathbb{R}^{2n} \times \mathbb{R}^m \to \mathbb{R}$, defined as $H(x, y, u) \triangleq y^T (A'x + B'u) + r(x, u)$, is convex in u, the control signal, $u(\cdot)$, and the state, $x(\cdot)$, satisfy the Hamilton-Jacobi-Bellman equation

$$H\left(x(t), \nabla_x (V^*(x(t)))^T, u(t)\right) = 0, \forall t \in \mathbb{R}_{\geq 0}, \quad (16)$$

where $V^* : \mathbb{R}^{2n} \to \mathbb{R}$ denotes the unknown optimal value function. The objective of inverse reinforcement learning is to generate an estimate of the unknown cost function, r. To facilitate estimation of the cost function, let $\hat{V} : \mathbb{R}^{2n} \times \mathbb{R}^P \to \mathbb{R}$, $\left(x, \hat{W}_V\right) \mapsto \hat{W}_V^T \sigma_V(x)$ be a parametric estimate of the optimal value function, where $\hat{W}_V \in \mathbb{R}^P$ are unknown parameters, and $\sigma_V : \mathbb{R}^{2n} \to \mathbb{R}^P$ are known continuously differentiable features. Assume that given any compact set $\chi \subset \mathbb{R}^{2n}$ and a constant $\bar{\epsilon} > 0$, sufficiently many features can be selected to ensure the existence of ideal parameters $W_V^* \in \mathbb{R}^P$ such that the error $\epsilon : \mathbb{R}^{2n} \to \mathbb{R}$, defined as $\epsilon(x) := V(x) - \hat{V}(x, W_V^*)$, satisfies $\sup_{x \in \chi} |\epsilon(x)| < \bar{\epsilon}$ and $\sup_{x \in \chi} |\nabla_x \epsilon(x)| < \bar{\epsilon}$. Using the estimates $\hat{A}_1$, $\hat{A}_2$, $\hat{B}$, $\hat{W}_V$, $\hat{W}_Q$, and $\hat{W}_R$ of the parameters A_1, A_2, B, W_V^*, W_Q^*, and $W_R := [r_1, \cdots, r_m]^T$, respectively, and the estimate $\hat{x}$ of the state, x, in (16), the inverse Bellman error $\delta' : \mathbb{R}^{2n} \times \mathbb{R}^m \times \mathbb{R}^{L+P+m} \times \mathbb{R}^{2n^2+mn} \to \mathbb{R}$ is obtained as

$$\delta'\left(\hat{x}, u, \hat{W}, \hat{\theta}\right) = \hat{W}_V^T \nabla_x \sigma_V(\hat{x}) \left(\hat{A}'\hat{x} + \hat{B}'u\right) + \hat{W}_Q^T \sigma_Q(\hat{x}) + \hat{W}_R^T \sigma_u(u), \quad (17)$$

where $\sigma_u(u) := \left[u_1^2, \cdots, u_m^2\right]$, $\hat{A}' := \begin{bmatrix} 0_{n\times n} & \mathrm{I}_{n\times n} \\ \hat{A}_1 & \hat{A}_2 \end{bmatrix}$, and $\hat{B}' := \begin{bmatrix} 0_{n\times m} \\ \hat{B} \end{bmatrix}$. Rearranging,

$$\delta'\left(\hat{x}, u, \hat{W}', \hat{\theta}\right) = \left(\hat{W}'\right)^T \sigma'\left(\hat{x}, u, \hat{\theta}\right), \quad (18)$$

where $\hat{W}' := \left[\hat{W}_V; \hat{W}_Q; \hat{W}_R\right]$, $\sigma'\left(\hat{x}, u, \hat{\theta}\right) := \left[\nabla_x \sigma_V(\hat{x}) \left(\hat{A}'\hat{x} + \hat{B}'u\right); \sigma_Q(\hat{x}); \sigma_u(u)\right]$. The following section details the developed model-based inverse reinforcement learning algorithm.

VI. Inverse Reinforcement Learning

The IRL problem can be solved by computing the estimates $\hat{W}$ that minimize the inverse Bellman error in (18). To facilitate the computation, the values of $\hat{x}$, u, and $\hat{\theta}$ are recorded at time instances $\{t_i < t\}_{i=1}^N$ to generate the values $\{\hat{\sigma}_t'(t_i)\}_{i=1}^N$, where $N \in \mathbb{N}$, $N >> L + P + m$, and $\hat{\sigma}_t'(t) := \sigma'\left(\hat{x}(t), u(t), \hat{\theta}(t)\right)$. The data in the history stack can be collected in a matrix form to yield

$$\Delta' = \hat{\Sigma}' \hat{W}', \quad (19)$$

where $\Delta' := [\delta_t'(t_1); \cdots; \delta_t'(t_N)]$, $\delta_t'(t) := \delta'\left(\hat{x}(t), u(t), \hat{W}', \hat{\theta}(t)\right)$, and $\hat{\Sigma}' := \left[(\hat{\sigma}_t')^T(t_1); \cdots; (\hat{\sigma}_t')^T(t_N)\right]$. Note that the solution $\hat{W}' = 0$ trivially minimizes Δ', which is to say that if the cost function is identically zero then every policy is optimal. Hence, as stated, the IRL problem is clearly ill-posed. In fact, the cost functions $r(x, u)$ and $Kr(x, u)$, where K is a positive constant, result in identical optimal policies and state trajectories. Hence, even if the trivial solution is discarded, the cost function can only be identified up to multiplication by a positive constant using the trajectories $x(\cdot)$ and $u(\cdot)$.

To remove the aforementioned ambiguity without loss of generality, the first element of $\hat{W}_R$ is assumed to be known. The inverse BE in (18) can then be expressed as

$$\delta'\left(\hat{x}, u, \hat{W}, \hat{\theta}\right) = \hat{W}^T \sigma''\left(\hat{x}, u, \hat{\theta}\right) + r_1 \sigma_{u1}(u), \quad (20)$$

where $\sigma_{ui}(u)$ denotes the i_{th} element of the vector $\sigma_u(u)$, the vector σ_u^- denotes σ_u, with the first element removed, and $\sigma''\left(\hat{x},u,\hat{\theta}\right) := \left[\nabla_x\sigma_V\left(\hat{x}\right)\left(\hat{A}'\hat{x}+\hat{B}'u\right);\sigma_Q\left(\hat{x}\right);\sigma_u^-\left(u\right)\right]$.

The closed-form optimal controller corresponding to (2) provides the relationship

$$-2Ru\left(t\right)=\left(B'\right)^T\nabla_x\sigma_V\left(x\left(t\right)\right)W_V^*+\left(B'\right)^T\nabla_x\epsilon\left(x\left(t\right)\right), \tag{21}$$

which can be expressed as

$$\begin{aligned}-2r_1u_1\left(t\right)+\Delta_{u1}&=\sigma_{B1}\hat{W}_V\\ \Delta_{u^-}&=\sigma_B^-\hat{W}_V+2\operatorname{diag}\left(u_2,\cdots,u_m\right)\hat{W}_R^-,\end{aligned}$$

where σ_{B1} and u_1 denote the first rows and σ_B^- and u^- denote all but the first rows of $\sigma_B := \left(B'\right)^T\nabla_x\sigma_V\left(x\right)$ and u, respectively, and $R^- := \operatorname{diag}\left(\left[r_2,\cdots,r_m\right]\right)$. For notational brevity let $\sigma := \left[\sigma'', \begin{bmatrix}\sigma_B^T\\ \Theta\end{bmatrix}\right]$, where

$$\Theta := \left[0_{m\times 2n},\ \begin{bmatrix}0_{1\times m-1}\\ 2\operatorname{diag}\left(\left[u_2,\cdots,u_m\right]\right)\end{bmatrix}\right]^T$$

The history stack can then be utilized to generate the linear system

$$-\Sigma_{u1}=\hat{\Sigma}\hat{W}-\Delta', \tag{22}$$

where $\hat{W} := \left[\hat{W}_V;\hat{W}_Q;\hat{W}_R^-\right]$, $\hat{\Sigma} := \left[\hat{\sigma}_t^T\left(t_1\right);\cdots;\hat{\sigma}_t^T\left(t_N\right)\right]$, and $\Sigma_{u1} := \left[\sigma_{u1}'\left(u\left(t_1\right)\right);\cdots;\sigma_{u1}'\left(u\left(t_N\right)\right)\right]$, where $\hat{\sigma}_t\left(\tau\right) := \sigma\left(\hat{x}\left(\tau\right),u\left(\tau\right),\hat{\theta}\left(\tau\right)\right)$, $\sigma_{u1}' := \left[\sigma_{u1};2r_1u_1;0_{(m-1)\times 1}\right]$, and the vector $\hat{W}_R^-$ denotes $\hat{W}_R$ with the first element removed.

At any time instant t, provided the history stack $\mathcal{G}\left(t\right)$ satisfies

$$\operatorname{rank}\left(\hat{\Sigma}\right)=L+P+m-1, \tag{23}$$

then a least-squares estimate of the weights can be obtained as

$$\hat{W}\left(t\right)=-\left(\hat{\Sigma}^T\hat{\Sigma}\right)^{-1}\hat{\Sigma}^T\Sigma_{u1}. \tag{24}$$

To improve numerical stability of the least-squares solution, the data recoded in the history stack is selected to maximize the condition number of $\hat{\Sigma}$ while ensuring that the vector Σ_{u1} remains nonzero. The data selection algorithm is detailed in Fig. 1.

VII. Purging to Exploit Improved State and Parameter Estimates

Since the matrices $\hat{\Sigma}$ and Δ' are functions of the state and parameter estimates, the accuracy of the least-squares solution in (24) depends on the accuracy of the state and parameter estimates recoded in $\mathcal{G}$. The state and parameter estimates are likely to be poor during the transient phase of the estimator dynamics. As a result, a least-squares solution computed using data recorded during the transient phase of the estimator may be inaccurate. Based on the observation that the state and the parameter estimates exponentially decay to the origin, a purging algorithm is developed in the following to remove erroneous state and parameter estimates from the history stack.

```
1: if an observed, estimated or queried data point $(x^*, u^*)$ is available at $t = t^*$ then
2:     if the history stack is not full then
3:         add the data point to the history stack
4:     else if $\kappa\left(\left(\hat{\Sigma}\left(i\leftarrow *\right)\right)^T\left(\hat{\Sigma}\left(i\leftarrow *\right)\right)\right)<\xi_1\kappa\left(\hat{\Sigma}^T\hat{\Sigma}\right)$,
       for some $i$, and $\left\|\Sigma_{u1}\left(i\leftarrow *\right)\right\|\geq\xi_2$ then
5:         add the data point to the history stack
6:         $\varpi\leftarrow 1$
7:     else
8:         discard the data point
9:         $\varpi\leftarrow 0$
10:    end if
11: end if
```

Fig. 1. Algorithm for selecting data for the history stack. The constants $\xi_1 \geq 0$ and $\xi_2 > 0$ are tunable thresholds. The operator $\kappa\left(\cdot\right)$ denotes the condition number of a matrix. For the matrix $\hat{\Sigma} = \left[\hat{\sigma}_t^T\left(t_1\right);\cdots;\hat{\sigma}_t^T\left(t_i\right);\cdots;\hat{\sigma}_t^T\left(t_N\right)\right]$, $\Sigma\left(i\leftarrow *\right) := \left[\hat{\sigma}_t^T\left(t_1\right);\cdots;\hat{\sigma}_t^T\left(t^*\right);\cdots;\hat{\sigma}_t^T\left(t_N\right)\right]$ and for the vector $\Sigma_{u1} = \left[\sigma_{u1}\left(u\left(t_1\right)\right);\cdots;\sigma_{u1}\left(u\left(t_i\right)\right);\cdots;\sigma_{u1}\left(u\left(t_N\right)\right)\right]$, $\Sigma_{u1}\left(i\leftarrow *\right) := \left[\sigma_{u1}\left(u\left(t_1\right)\right);\cdots;\sigma_{u1}\left(u\left(t^*\right)\right);\cdots;\sigma_{u1}\left(u\left(t_N\right)\right)\right]$.

An indicator, η, that quantifies the quality of the current state and parameter estimates using a guess-and-check method is used to purge and update the history stack and to update the estimate $\hat{W}$ according to the algorithm detailed in Fig. 2. The algorithm begins with an empty history stack and an initial estimate of the weights W_0. Values of $\hat{x}$, u, $\hat{\theta}$, and η are recorded in the history stack using the algorithm detailed in Fig. 1, where $\eta\left(t\right)$ is assumed to be infinite for $t<T$. The estimate $\hat{W}$ is held at the initial guess until the history stack is full. Then, it is updated using (24) every time a new data point is added to the history stack.

Communication with the entity under observation, wherever possible can be easily incorporated in the developed framework. In the query-based implementation of the developed algorithm, the observed input-output trajectories are utilized to learn the dynamics of the UxV. Instead of using the estimated state and control trajectories for cost estimation, control actions, u_i of the entity under observation in response to randomly selected states, x_i, are queried. If the queried state-input pair improves the condition number of the history stack then it is stored in the history stack and utilized for cost estimation.

VIII. Analysis

A detailed analysis of the simultaneous state and parameter estimator is excluded for brevity, and is available in [25]. To facilitate the analysis of the IRL algorithm, let $\Sigma := \left[\sigma\left(x\left(t_1\right),u\left(t_1\right),\theta\right);\cdots;\sigma\left(x\left(t_M\right),u\left(t_M\right),\theta\right)\right]$ and let $\hat{W}^*$ denote the least-squares solution of $\Sigma\hat{W}=-\Sigma_{u1}$. Furthermore, let W denote an appropriately scaled version of the ideal weights, i.e, $W := {}^{W}\!/_{r_1}$. Provided the rank condition in (23) is satisfied, the inverse HJB equation in 16 implies that $\Sigma W = -\Sigma_{u1}-E$, where $E := \left[\nabla_x\epsilon\left(x\left(t_1\right)\right)\left(Ax\left(t_1\right)+Bu\left(t_1\right)\right);\right.$

```
1:  $\hat{W}(0) \leftarrow W_0$, $s \leftarrow 0$
2:  **if** $\kappa\left(\hat{\Sigma}^T\hat{\Sigma}\right) < \underline{\kappa_1}$ and $\varpi = 1$ **then**
3:      $\hat{W}(t) \leftarrow -\left(\hat{\Sigma}^T\hat{\Sigma}\right)^{-1}\hat{\Sigma}^T\Sigma_{u1}$
4:  **else**
5:      Hold $\hat{W}$ at the previous value
6:  **end if**
7:  **if** $\kappa\left(\hat{\Sigma}^T\hat{\Sigma}\right) < \underline{\kappa_2}$ and $\eta(t) < \overline{\eta}(t)$ **then**
8:      empty the history stack
9:      $s \leftarrow s+1$
10: **end if**
```

Fig. 2. Algorithm for updating the weights and the history stack. The constants $\underline{\kappa_1} > 0$ and $\underline{\kappa_2} > 0$ are tunable thresholds, the index s denotes the number of times the history stack was purged, and $\overline{\eta}(t) := \min\{\eta(t_1), \cdots, \eta(t_M)\}$.

$\cdots; \quad \nabla_x \epsilon(x(t_M))(Ax(t_M) + Bu(t_M))]$. That is, $\left\|W + \left(\Sigma^T\Sigma\right)^{-1}\Sigma^T\Sigma_{u1}\right\| \leq \left\|\left(\Sigma^T\Sigma\right)^{-1}\Sigma^T E\right\|$. Since $\hat{W}^*$ is a least squares solution, $\left\|W - \hat{W}^*\right\| \leq \left\|\left(\Sigma^T\Sigma\right)^{-1}\Sigma^T E\right\|$.

Let $\hat{\Sigma}_s$, Σ_{u1_s}, and $\hat{W}_s$ denote the regression matrices and the weight estimates corresponding to the s$^{\text{th}}$ history stack, respectively, and let Σ_s denote the ideal regression matrix where $\hat{x}(t_i)$ and $\hat{\theta}(t_i)$ in $\hat{\Sigma}_s$ are replaced with the corresponding ideal values $x(t_i)$ and θ. Let $\hat{W}_s^*$ denote the least-squares solution of $\Sigma_s\hat{W} = -\Sigma_{u1_s}$. Provided $\hat{\Sigma}_s$ satisfies the rank condition in (23), then $\left\|W - \hat{W}_s^*\right\| \leq \left\|\left(\Sigma_s^T\Sigma_s\right)^{-1}\Sigma_s^T E\right\|$. Furthermore, $\hat{W}_s - \hat{W}_s^* = \left(\left(\left(\hat{\Sigma}_s^T\hat{\Sigma}_s\right)^{-1}\hat{\Sigma}_s^T\right) - \left(\left(\Sigma_s^T\Sigma_s\right)^{-1}\Sigma_s^T\right)\right)\Sigma_{u1_s}$. Since the estimates $\hat{x}$ and $\hat{\theta}$ exponentially converge to x and θ, respectively, the function $(x,\theta) \mapsto \sigma(x,u,\theta)$ is continuous for all u, and under the rank condition in (23), the function $\Sigma \mapsto \left(\Sigma^T\Sigma\right)^{-1}\Sigma^T$ is continuous, it can be concluded that $\hat{W}_s \to \hat{W}_s^*$ as $s \to \infty$, and hence, the error between the estimates $\hat{W}_s$ and the ideal weights W is $O(\overline{\epsilon})$ as $s \to \infty$.

IX. Simulation

To verify the performance of the developed method, a linear quadratic optimal control problem is selected where

$$A = \begin{bmatrix} 1 & 1 & -1 & 1 \\ 5 & 1 & 1 & 1 \end{bmatrix}, \qquad B = \begin{bmatrix} 1 & 3 \\ 0 & 1 \end{bmatrix}.$$

The weighing matrices in the cost function are selected as $Q = \text{diag}([1,\ 2,\ 3,\ 6])$ and $R = [20,\ 10]$, where $R(1,1)$ is assumed to be known. The observed input-output trajectories, along with a prerecorded history stack are used to implement the simultaneous state and parameter estimation algorithm in Section IV. The design parameters in the system identification algorithm are selected using trial and error as $M = 150$, $T_1 = 1s$, $T_2 = 0.8sk = 100$, $\alpha = 20$, $\beta = 10$, $\beta_1 = 5$, $k_\theta = 0.3/M$, and $\Gamma(0) = 0.1 * \mathrm{I}_{L+P+m-1}$.

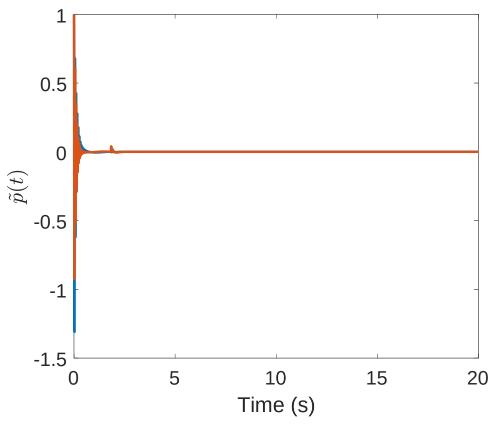

Fig. 3. Generalized position estimation error.

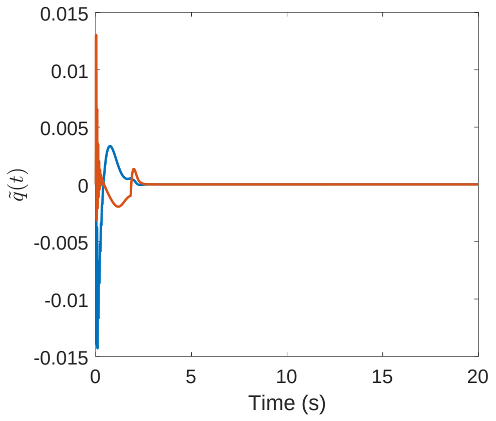

Fig. 4. Generalized velocity estimation error.

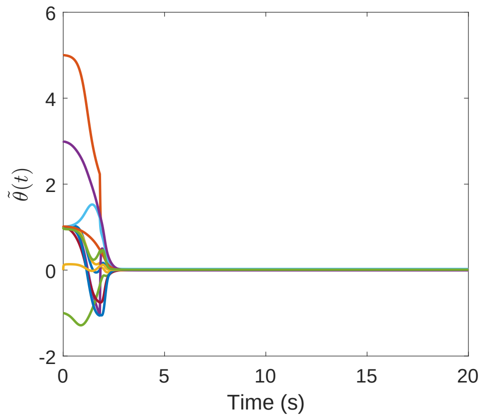

Fig. 5. Estimation error for the unknown parameters in the system dynamics.

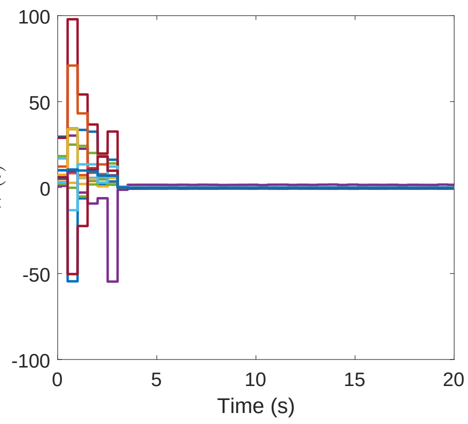

Fig. 6. Estimation error for the unknown parameters in the cost function.

The behavior of the system under the optimal controller $u(t) = R^{-1}(B')^T Px(t)$ is observed, where $P \in \mathbb{R}^{2n \times 2n}$ is the solution to the algebraic Riccati equation corresponding to (2). At each time step, a random state vector x^* is selected and the optimal action u^* corresponding to the random state vector is queried from the entity under observation. The queried state-action pairs (x^*, u^*) are utilized in conjunction with the estimated state-action pairs $(\hat{x}(t), u(t))$ to implement the IRL algorithm developed in Section VI.

Figs. 3 and 4 demonstrate the performance of the developed state estimator and Fig. 5 illustrates the performance of the developed parameter estimator. The estimation errors in the generalized position, the generalized velocity, and the unknown plant parameters exponentially decay to the origin. Fig. 6 indicates that the developed IRL technique can be successfully utilized to estimate the cost function of an entity under observation.

X. Conclusion

A data-driven inverse reinforcement learning technique is developed for a class of linear systems to estimate the cost function of an agent online, using input-output measurements. A simultaneous state and parameter estimator is utilized to facilitate output-feedback inverse reinforcement learning, and cost function estimation is achieved up to multiplication by a constant. A purging algorithm is utilized to update the stored state and parameter estimates and bounds on the cost estimation error are obtained.

References

[1] R. E. Kalman, "When is a linear control system optimal?" *J. Basic Eng.*, vol. 86, no. 1, pp. 51–60, 1964.

[2] S. Boyd, L. E. Ghaoui, E. Feron, and V. Balakrishnan, *Linear matrix inequalities in system and control theory*. SIAM, 1994.

[3] A. Y. Ng and S. Russell, "Algorithms for inverse reinforcement learning," in *Proc. Int. Conf. Mach. Learn.* Morgan Kaufmann, 2000, pp. 663–670.

[4] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2004.

[5] P. Abbeel and Y. Ng, Andrew, "Exploration and apprenticeship learning in reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2005.

[6] N. D. Ratliff, J. A. Bagnell, and M. A. Zinkevich, "Maximum margin planning," in *Proc. Int. Conf. Mach. Learn.*, 2006.

[7] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning," in *Proc. AAAI Conf. Artif. Intel.*, 2008, pp. 1433–1438.

[8] E. T. Jaynes, "Information theory and statistical mechanics," *Phys. Rev.*, vol. 106, no. 4, pp. 620–630, May 1957.

[9] B. D. Ziebart, J. A. Bagnell, and A. K. Dey, "Modeling interaction via the principle of maximum causal entropy," in *Proc. Int. Conf. Mach. Learn.*, Sep. 2010, pp. 1255–1262.

[10] A. Boularias, J. Kober, and J. Peters, "Relative entropy inverse reinforcement learning," in *Proc. Int. Conf. Artif. Intell. Stat.*, G. Gordon, D. Dunson, and M. Dudík, Eds., vol. 15, 2011.

[11] D. Ramachandran and E. Amir, "Bayesian inverse reinforcement learning," in *Proc. Int. Joint Conf. Artif. Intell.* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007, pp. 2586–2591.

[12] G. Neu and C. Szepesvari, "Apprenticeship learning using inverse reinforcement learning and gradient methods," in *Proc. Anu. Conf. Uncertain. Artif. Intell.* Corvallis, Oregon: AUAI Press, 2007, pp. 295–302.

[13] U. Syed and R. E. Schapire, "A game-theoretic approach to apprenticeship learning," in *Advances in Neural Information Processing Systems 20*, J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, Eds. Curran Associates, Inc., 2008, pp. 1449–1456.

[14] S. Levine, Z. Popovic, and V. Koltun, "Feature construction for inverse reinforcement learning," in *Advances in Neural Information Processing Systems 23*, J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, Eds. Curran Associates, Inc., 2010, pp. 1342–1350.

[15] S. Levine and V. Koltun, "Continuous inverse optimal control with locally optimal examples," in *Proc. Int. Conf. Mach. Learn.*, J. Langford and J. Pineau, Eds. New York, NY, USA: ACM, 2012, pp. 41–48.

[16] S. Levine, Z. Popovic, and V. Koltun, "Nonlinear inverse reinforcement learning with gaussian processes," in *Advances in Neural Information Processing Systems 24*, J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2011, pp. 19–27.

[17] B. Michini and J. P. How, "Bayesian nonparametric inverse reinforcement learning," in *Machine Learning and Knowledge Discovery in Databases*, ser. Lecture Notes in Computer Science, P. A. Flach, T. D. Bie, and N. Cristianini, Eds. Springer Berlin Heidelberg, 2012, vol. 7524, pp. 148–163.

[18] K. Mombaur, A. Truong, and J.-P. Laumond, "From human to humanoid locomotion—an inverse optimal control approach," *Auton. Robot.*, vol. 28, no. 3, pp. 369–383, 2010.

[19] B. Michini, T. J. Walsh, A. A. Agha-Mohammadi, and J. P. How, "Bayesian nonparametric reward learning from demonstration," *IEEE Trans. Robot.*, vol. 31, no. 2, pp. 369–386, Apr. 2015.

[20] K. G. Vamvoudakis and F. L. Lewis, "Online actor-critic algorithm to solve the continuous-time infinite horizon optimal control problem," *Automatica*, vol. 46, no. 5, pp. 878–888, 2010.

[21] T. Bian, Y. Jiang, and Z.-P. Jiang, "Adaptive dynamic programming and optimal control of nonlinear nonaffine systems," *Automatica*, vol. 50, no. 10, pp. 2624–2632, 2014.

[22] H. Modares and F. L. Lewis, "Optimal tracking control of nonlinear partially-unknown constrained-input systems using integral reinforcement learning," *Automatica*, vol. 50, no. 7, pp. 1780–1792, 2014.

[23] R. Kamalapurkar, P. Walters, and W. E. Dixon, "Model-based reinforcement learning for approximate optimal regulation," *Automatica*, vol. 64, pp. 94–104, Feb. 2016.

[24] D. Wang, D. Liu, H. Li, B. Luo, and H. Ma, "An approximate optimal control approach for robust stabilization of a class of discrete-time nonlinear systems with uncertainties," *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 46, no. 5, pp. 713–717, 2016.

[25] R. Kamalapurkar, "Online output-feedback parameter and state estimation for second order linear systems," in *Proc. Am. Control Conf.*, Seattle, WA, USA, May 2017, pp. 5672–5677.

[26] P. Ioannou and J. Sun, *Robust adaptive control*. Prentice Hall, 1996.

[27] B. Xian, M. S. de Queiroz, D. M. Dawson, and M. McIntyre, "A discontinuous output feedback controller and velocity observer for nonlinear mechanical systems," *Automatica*, vol. 40, no. 4, pp. 695–700, 2004.