Online inverse reinforcement learning for nonlinear systems

Ryan Self, Michael Harlan, and Rushikesh Kamalapurkar

Abstract— **This paper focuses on the development of an online inverse reinforcement learning (IRL) technique for a class of nonlinear systems. The developed approach utilizes observed state and input trajectories, and determines the unknown reward function and the unknown value function online. A parameter estimation technique is utilized to facilitate estimation of the reward function in the presence of unknown dynamics. Theoretical guarantees for convergence of the reward function estimates are established, and simulation results are presented to demonstrate the performance of the developed technique.**

I. Introduction

Over the past decade, autonomous systems have been utilized to perform increasingly complex tasks due to their repeatability and increased precision, but their impact is ultimately limited by their inability to adapt to change. Humans can intuitively alter task objectives without being explicitly told, and can infer task objectives based on what other humans in their environment are trying to achieve. The ability of humans to detect the intent of others allows for better cooperation in teams. Similarly, the development of techniques to detect the intent of other entities will endow autonomous systems with the ability to detect subterfuge, to improve teamwork, and to learn appropriate responses to changing circumstances from demonstrations.

Based on the premise that the most succinct representation of the behavior of an entity is its reward structure [1], this paper aims to recover the reward (or cost) function of an agent by observing the agent performing a task and monitoring its state and control trajectories. Methods to estimate the reward function using state and control trajectories fall under the umbrella of inverse reinforcement learning (IRL) (see, for example, [1] and [2]). The IRL method developed in this paper learns the reward function and the value function of an agent under observation online, and in the presence of modeling uncertainties.

IRL methods were proposed in [1] and reward function estimation techniques using IRL were initially used for problems formulated as Markov Decision Processes (MDP) in [3], [4], and [5]. Since solutions to the IRL problems are generally not unique, the maximum entropy principle in [6] was developed to help differentiate between the various solutions. In [7], the authors developed a Maximum Casual Entropy IRL technique for infinite time horizon problems where a stationary soft Bellman policy which helps enable the learning of the transition models is utilized. Beyond this, many extensions of IRL have been developed, including formulation of feature construction [8], solving IRL using gradient based methods [9], and game theoretic approaches, as in [10], which suggest the possibility of finding solutions that outperform the expert demonstrator. IRL was further extended to nonlinear problems using deep neural networks [11] or Gaussian Processes [12] in which a nonlinear reward function estimate is found using a probabilistic approach.

The aforementioned IRL techniques and inverse optimal control, methods such as [13], have been extensively utilized to teach autonomous machines to perform specific tasks in an *offline* setting [14]. However, offline approaches to IRL cannot handle changes to task objectives and are ill-suited for adaptation in real-time. The development of online IRL techniques is motivated by the need for robustness to uncertainties in the system model and responsiveness to adapt to changing reward structures. Inspired by the success of model-based real-time reinforcement learning methods such as [15], [16] and the preliminary online IRL results for linear systems in [17], this paper presents an IRL technique for nonlinear systems.

The main contribution of this paper is the development of a novel method for reward function estimation in nonlinear systems using model-based IRL in an online setting. The developed technique is inspired by IRL results for linear systems presented in [17] and [18]. The challenges in extending the aforementioned methods to uncertain nonlinear systems are twofold: (a) value functions for nonlinear systems may not be linearly parameterizable, and (b) uncertainties in the nonlinear dynamics need to be estimated to achieve model-based IRL.

The paper is organized as follows: Section II details the mathematics notation used throughout the paper. Section III introduces the problem formulation. Section IV details an overview of the parameter estimation technique incorporated in the simulation. Section V shows stability analysis for the parameter estimator. Section VI explains the error metric used for the calculations. Section VII introduces the IRL algorithm. Section VIII details how inaccurate data from the parameter estimation is removed. Section IX is the analysis for convergence of the algorithm. Section X shows the simulation for a nonlinear system and Section XI is the conclusion of the paper.

II. Notation

The notation $\mathbb{R}^n$ represents the $n-$dimensional Euclidean space, and the elements of $\mathbb{R}^n$ are interpreted as column vectors, where $(\cdot)^T$ denotes the vector transpose operator. The set of positive integers excluding 0 is denoted by $\mathbb{N}$. For

The authors are with the School of Mechanical and Aerospace Engineering, Oklahoma State University, Stillwater, OK, USA. {rself, michael.c.harlan, rushikesh.kamalapurkar}@okstate.edu.

$a \in \mathbb{R}$, $\mathbb{R}_{\geq a}$ 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 I_n and 0_n denote the $n \times n$ identity matrix and the zero element of $\mathbb{R}^n$, respectively. Whenever it is clear from the context, the subscript n is suppressed.

III. Problem formulation

Consider an agent with the following nonlinear dynamics

$$\dot{x} = f(x, u) \tag{1}$$

where $x : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ is the state and $u : \mathbb{R}_{\geq 0} \to \mathbb{R}^m$ is the control. If a nominal dynamic model of the agent is available, then the dynamics in (1) can then be separated into

$$\dot{x} = f^o(x, u) + g(x, u) \tag{2}$$

where f^o represents the known terms and g represents the unknown terms.[1]

The agent being observed is using the policy which minimizes the following performance index

$$J(x_0, u(\cdot)) = \int_0^\infty r(x(t; x_0, u(\cdot)), u(t)) dt \tag{3}$$

where $x(t; x_0, u(\cdot))$ is the trajectory of the agent generated by the optimal control signal $u(\cdot)$ starting from the initial condition x_0, evaluated at time t. To aid in reward function estimation, the reward function $r : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ can be parameterized as $r = W^T \sigma$, to be made precise in Section VI, where W represents the weights of the reward function to be approximated and σ represents known continuously differentiable features. The objective of the proposed technique is for the system to identify the unknown weights of the reward function W and the unknown dynamics, g.

It is assumed that the optimal control problem is well-posed and that it has a unique solution. It is further assumed that the system is affine in control and that the reward function is quadratic in control. Extension of the developed method to non-affine, non-quadratic systems is a topic for further research.

IV. Parameter estimator

The parameter estimator is motivated by the authors' previous work in [19]. This section provides a brief overview of the parameter estimation design.

To facilitate parameter estimation, let $g(x, u) = \theta^T \sigma(x, u)$, where $\sigma : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^p$ denotes the basis vector and $\theta \in \mathbb{R}^{p \times n}$ is a constant matrix of unknown parameters. The system in (2) is expressed in the form

$$\dot{x} = f^o(x, u) + \theta^T \sigma(x, u) . \tag{4}$$

[1]If a nominal model is not available, $f^o(x, u) := 0 \; \forall \, (x, u) \in \mathbb{R}^n \times \mathbb{R}^m$.

Integrating (4) over the interval $[t - \tau_1, t]$ for some constant $\tau_1 \in \mathbb{R}_{>0}$,[2]

$$x(t) - x(t - \tau_1) = \int_{t-\tau_1}^{t} f^o(\gamma) \, \mathrm{d}\gamma + \theta^T \int_{t-\tau_1}^{t} \sigma(\gamma) \, \mathrm{d}\gamma, \tag{5}$$

where $f^o(\gamma)$ and $\sigma(\gamma)$ are used to denote $f^o(x(\gamma), u(\gamma))$ and $\sigma(x(\gamma), u(\gamma))$, respectively.

The expression in (5) can be rearranged to form the affine system

$$P(t) = F(t) + \theta^T G(t), \; \forall t \in \mathbb{R}_{\geq T_0} \tag{6}$$

where

$$P(t) := \begin{cases} x(t) - x(t - \tau_1), & t \in [T_0 + \tau_1, \infty), \\ 0 & t < T_0 + \tau_1 \end{cases} \tag{7}$$

$$F(t) := \begin{cases} \mathcal{I} f^o(t), & t \in [T_0 + \tau_1, \infty), \\ 0, & t < T_0 + \tau_1, \end{cases} \tag{8}$$

and

$$G(t) := \begin{cases} \mathcal{I} \sigma(t), & t \in [T_0 + \tau_1, \infty), \\ 0 & t < T_0 + \tau_1, \end{cases} \tag{9}$$

where $\mathcal{I}$ denotes the integral operator $f \mapsto \int_{t-\tau_1}^{t} f(x(\tau), u(\tau)) \, \mathrm{d}\tau$.

The affine error system in (6) motivates the adaptive estimation scheme that follows. The design is inspired by the *concurrent learning* technique [20]. Concurrent learning enables parameter convergence in adaptive control by using stored data to update the parameter estimates. Traditionally, adaptive control methods guarantee parameter convergence only if the appropriate persistence of excitation (PE) conditions are met [21, Chapter 4]. Concurrent learning uses stored data to soften the PE condition to an excitation condition over a finite time-interval.

For ease of exposition, it is assumed that a history stack, i.e., a set of ordered pairs $\{(P_i, F_i, G_i)\}_{i=1}^M$ such that

$$P_i = F_i + \theta^T G_i, \; \forall i \in \{1, \cdots, M\}, \tag{10}$$

is available a priori. A history stack $\{(P_i, F_i, G_i)\}_{i=1}^M$ is called *full rank* if there exists a constant $\underline{c} \in \mathbb{R}$ such that

$$0 < \underline{c} < \lambda_{\min} \{\mathscr{G}\}, \tag{11}$$

where the matrix $\mathscr{G} \in \mathbb{R}^{p \times p}$ is defined as $\mathscr{G} := \sum_{i=1}^M G_i G_i^T$. The concurrent learning update law to estimate the unknown parameters is then given by

$$\dot{\hat{\theta}} = k_\theta \Gamma \sum_{i=1}^{M} G_i \left(P_i - F_i - \hat{\theta}^T G_i \right)^T, \tag{12}$$

[2]If the integration interval is selected to be too short, there may not be enough information in the vector P_i to achieve accurate parameter estimation. If the integration interval is selected too long, parameter estimates may not be available during transients where they are needed the most. The authors have as yet been unable to develop reasonable heuristics to select the interval and the development of such a heuristic is a topic for future research.

where $k_\theta \in \mathbb{R}_{>0}$ is a constant adaptation gain, and $\Gamma : \mathbb{R}_{\geq 0} \to \mathbb{R}^{p\times p}$ is the least-squares gain updated using the update law

$$\dot{\Gamma} = \beta_1\Gamma - k_\theta\Gamma\mathscr{G}\Gamma. \tag{13}$$

where $\beta_1 \in \mathbb{R}_{>0}$ is a constant gain. Using arguments similar to [21, Corollary 4.3.2], it can be shown that provided $\lambda_{\min}\left\{\Gamma^{-1}(0)\right\} > 0$, the least squares gain matrix satisfies

$$\underline{\Gamma}\mathrm{I}_p \leq \Gamma(t) \leq \overline{\Gamma}\mathrm{I}_p, \tag{14}$$

where $\underline{\Gamma}$ and $\overline{\Gamma}$ are positive constants, and I_n denotes an $n\times n$ identity matrix.

V. Stability Analysis

Consider the following positive definite candidate Lyapunov function:

$$V(t,\theta) = \tilde{\theta}^T\Gamma^{-1}(t)\tilde{\theta} \tag{15}$$

Taking the time derivative of (15), and using (12) and (13), yields

$$\begin{aligned}\dot{V} =& \, 2\tilde{\theta}^T\Gamma^{-1}(t)\dot{\tilde{\theta}} + \tilde{\theta}^T\dot{\Gamma}^{-1}(t)\tilde{\theta}\\
=& -2\tilde{\theta}^T\Gamma^{-1}(t)\Big(k_\theta\Gamma(t)\sum_{i=1}^{M}G_i\left(P_i - F_i - \hat{\theta}^TG_i\right)^T\Big)\\
&- \tilde{\theta}^T\Big(\Gamma^{-1}(t)\Big[\beta_1\Gamma(t) - k_\theta\Gamma(t)\mathscr{G}\Gamma(t)\Big]\Gamma^{-1}(t)\Big)\tilde{\theta}\\
=& -2k_\theta\tilde{\theta}^T\Big(\sum_{i=1}^{M}G_i\left(\theta^TG_i - \hat{\theta}^TG_i\right)^T\Big) - \beta_1\tilde{\theta}^T\Gamma^{-1}(t)\tilde{\theta}\\
&+ k_\theta\tilde{\theta}^T\mathscr{G}\tilde{\theta}\\
=& -2k_\theta\tilde{\theta}^T\Big(\sum_{i=1}^{M}G_i\left((\theta^T - \hat{\theta}^T)G_i\right)^T\Big) - \beta_1\tilde{\theta}^T\Gamma^{-1}(t)\tilde{\theta}\\
&+ k_\theta\tilde{\theta}^T\mathscr{G}\tilde{\theta}\\
=& -2k_\theta\tilde{\theta}^T\mathscr{G}\tilde{\theta} - \beta_1\tilde{\theta}^T\Gamma^{-1}(t)\tilde{\theta}\\
&+ k_\theta\tilde{\theta}^T\mathscr{G}\tilde{\theta}\\
=& -2k_\theta\tilde{\theta}^T\mathscr{G}\tilde{\theta} - \beta_1\tilde{\theta}^T\Gamma^{-1}(t)\tilde{\theta} + k_\theta\tilde{\theta}^T\mathscr{G}\tilde{\theta}\\
=& -k_\theta\tilde{\theta}^T\mathscr{G}\tilde{\theta} - \beta_1\tilde{\theta}^T\Gamma^{-1}(t)\tilde{\theta}\end{aligned} \tag{16}$$

Using (15) and (16), along with the bounds in (11) and (14), [22, Theorem 4.10] can be invoked to conclude that $\tilde{\theta}$ converges exponentially to 0.

VI. Inverse Bellman Error

In this section, the parameter estimates from Section IV are utilized to formulate an indirect error metric that facilitates IRL.

Under the premise that the observed agent makes optimal decisions, the state and control trajectories, $x(\cdot)$ and $u(\cdot)$, satisfy the Hamilton-Jacobi-Bellman equation

$$H\left(x(t), \nabla_x\left(V^*(x(t))\right)^T, u(t)\right) = 0, \forall t \in \mathbb{R}_{\geq 0}, \tag{17}$$

where the unknown optimal value function is $V^* : \mathbb{R}^n \to \mathbb{R}$ and $H : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ is the Hamiltonian, defined as $H(x,p,u) := p^Tf(x,u) + r(x,u)$. The goal of IRL is to accurately estimate the reward function, r. To aid in the estimation of the reward function, let $\hat{V} : \mathbb{R}^n \times \mathbb{R}^P \to \mathbb{R}$, $\left(x, \hat{W}_V\right) \mapsto \hat{W}_V^T\sigma_V(x) + \epsilon_V(x)$ be a parameterized estimate of the optimal value function V^*, where $\hat{W}_V \in \mathbb{R}^P$ are the estimates of the ideal value function weights W_V^*, $\sigma_V : \mathbb{R}^n \to \mathbb{R}^P$ are known continuously differentiable features, and $\epsilon_V : \mathbb{R}^n \to \mathbb{R}$ is the resulting approximation error. Additionally, since the reward function r is quadratic in the control, it can be expressed as $r(x,u) = Q(x) + u^TRu$. In this formulation, $R \in \mathbb{R}^{m\times m}$ is a positive definite matrix, such that $R = \mathrm{diag}([r_1, \cdots, r_m])$, and $Q(x) = (W_Q^*)^T\sigma_Q(x) + \epsilon_Q(x)$ is a positive definite function, where $W_Q^* := [q_1, \cdots, q_L]^T$ are ideal reward function weights, $\sigma_Q : \mathbb{R}^n \to \mathbb{R}^L$ are known continuously differentiable features, and $\epsilon_Q : \mathbb{R}^n \to \mathbb{R}$ is the approximation error. Using $\hat{\theta}$, $\hat{W}_V$, $\hat{W}_Q$, and $\hat{W}_R$, which are the estimates of θ, W_V^*, W_Q^*, and $W_R^* := [r_1, \cdots, r_m]^T$, respectively, in (17), a parametric estimate of the Hamiltonian called the inverse Bellman error $\delta' : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^{L+P+m} \times \mathbb{R}^p \to \mathbb{R}$ is obtained as

$$\begin{aligned}\delta'\left(x, u, \hat{W}, \hat{\theta}\right) =& \hat{W}_V^T\nabla_x\sigma_V(x)\ \hat{Y}(x,u,\hat{\theta}) + \hat{W}_Q^T\sigma_Q(x)\\
&+ \hat{W}_R^T\sigma_u(u),\end{aligned} \tag{18}$$

where $\sigma_u(u) := \left[u_1^2, \cdots, u_m^2\right]$ and $\hat{Y}(x,u,\hat{\theta}) = \left[f^o(x,u) + \hat{g}(x,u,\hat{\theta})\right]$ where $\hat{g}(x,u,\hat{\theta}) := \hat{\theta}^T\sigma(x,u)$ from (4). Rearranging, we get

$$\delta'\left(x, u, \hat{W}', \hat{\theta}\right) = \left(\hat{W}'\right)^T\sigma'\left(x, u, \hat{\theta}\right), \tag{19}$$

where $\hat{W}' := \left[\hat{W}_V; \hat{W}_Q; \hat{W}_R\right]$ and $\sigma'\left(x, u, \hat{\theta}\right) := \left[\nabla_x\sigma_V(x)\hat{Y}(x,u,\hat{\theta}); \sigma_Q(x); \sigma_u(u)\right]$.

VII. Inverse Reinforcement Learning

Using the formulation of the inverse Bellman error in Section VI, a history stack of the data can be formulated into matrix form, resulting in

$$\Delta' = \hat{\Sigma}'\hat{W}', \tag{20}$$

where $\Delta' := [\delta_t'(t_1); \cdots; \delta_t'(t_N)]$, $\delta_t'(t) := \delta'\left(x(t), u(t), \hat{W}', \hat{\theta}(t)\right)$, and $\hat{\Sigma}' := \left[\left(\hat{\sigma}_t'\right)^T(t_1); \cdots; \left(\hat{\sigma}_t'\right)^T(t_N)\right]$. Using (20), the IRL problem can be solved by finding the solution of the linear system for $\hat{W}'$ that minimize the inverse Bellman error in (19). To facilitate the computation, the values of 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 $\hat{\sigma}_t'(t) := \sigma'\left(x(t), u(t), \hat{\theta}(t)\right)$.

Since the IRL problem is ill-posed, the solution to (20) is not unique. The trivial solution $\hat{W}' = 0$ is not desirable because it does not provide any useful information and should be discarded. In addition to this trivial solution, due to the uniqueness assumption, there are an infinite number of other solutions to the IRL problem. But, these solutions are constant multiples of the optimal solution, meaning the solutions to $r(x,u)$ and $ar(x,u)$, where a is a positive constant, result in the exact same policy.

Without loss of generality, a known value is assigned to one of the reward function weights, and as a result, the trivial solution is no longer viable and the scaling ambiguity is removed. Taking the first element of $\hat{W}_R$ to be known, the inverse BE in (19) can then be expressed as

$$\delta'\left(x, u, \hat{W}, \hat{\theta}\right) = \hat{W}^T \sigma''\left(x, u, \hat{\theta}\right) + r_1 \sigma_{u1}(u), \quad (21)$$

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

The closed-form nonlinear optimal controller corresponding to the reward structure in (3) provides the relationship

$$\begin{aligned} -2Ru(t) = &\left(g'(x(t))\right)^T \left(\nabla_x \sigma_V(x(t))\right)^T W_V^* \\ &+ \left(g'(x(t))\right)^T \nabla_x \epsilon(x(t)), \end{aligned} \quad (22)$$

which can be expressed as

$$\begin{aligned} -2r_1 u_1(t) + \Delta_{u1} &= \sigma_{g1} \hat{W}_V \\ \Delta_{u^-} &= \sigma_g^- \hat{W}_V + 2\text{diag}\left(u_2, \cdots, u_m\right) \hat{W}_R^-, \end{aligned}$$

where $g'(x) := \nabla_u f(x,u)$, σ_{g1} and Δ_{u_1} denote the first rows and σ_g^- and Δ_{u^-} denote all but the first rows of $\sigma_g(x) := \left(g'(x)\right)^T \left(\nabla_x \sigma_V(x)\right)^T$ and $\Delta_u(x) := \left(g'(x)\right)^T \nabla_x \epsilon(x)$, respectively, and $R^- := \text{diag}\left([r_2, \cdots, r_m]\right)$. For simplification, let $\sigma := \left[\sigma'', \begin{bmatrix} \sigma_g^T \\ \Theta \end{bmatrix}\right]$, where

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

Updating the history stack in (20) and removing the known reward weight element will result in the linear system

$$-\Sigma_{u1} = \hat{\Sigma} \hat{W} - \Delta', \quad (23)$$

where $\hat{\Sigma} := \left[\hat{\sigma}_t^T(t_1); \cdots; \hat{\sigma}_t^T(t_N)\right]$, and $\Sigma_{u1} := \left[\sigma'_{u1}(u(t_1)); \cdots; \sigma'_{u1}(u(t_N))\right]$, where $\hat{\sigma}_t(\tau) := \sigma\left(x(\tau), u(\tau), \hat{\theta}(\tau)\right)$, $\sigma'_{u1}(\tau) := \left[r_1 \sigma_{u1}(\tau); 2r_1 u_1(\tau); 0_{(m-1)\times 1}\right]$.

At any time instant t, provided the data stored in the history stack $G(t)$ satisfies

$$\text{rank}\left(\hat{\Sigma}\right) = L + P + m - 1, \quad (24)$$

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

$$\hat{W}(t) = -\left(\hat{\Sigma}^T \hat{\Sigma}\right)^{-1} \hat{\Sigma}^T \Sigma_{u1}. \quad (25)$$

To improve numerical stability of the least-squares solution, the data recorded 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.

```
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 $\kappa\left(\left(\hat{\Sigma}(i \leftarrow *)\right)^T \left(\hat{\Sigma}(i \leftarrow *)\right)\right) < \xi_1 \kappa\left(\hat{\Sigma}^T \hat{\Sigma}\right)$,
        for some $i$, and $\|\Sigma_{u1}(i \leftarrow *)\| \geq \xi_2$ 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 $\kappa(\cdot)$ denotes the condition number of a matrix. For the matrix $\hat{\Sigma} = \left[\hat{\sigma}_t^T(t_1); \cdots; \hat{\sigma}_t^T(t_i); \cdots; \hat{\sigma}_t^T(t_N)\right]$, $\Sigma(i \leftarrow *) := \left[\hat{\sigma}_t^T(t_1); \cdots; \hat{\sigma}_t^T(t^*); \cdots; \hat{\sigma}_t^T(t_N)\right]$ and for the vector $\Sigma_{u1} = \left[\sigma_{u1}(u(t_1)); \cdots; \sigma_{u1}(u(t_i)); \cdots; \sigma_{u1}(u(t_N))\right]$, $\Sigma_{u1}(i \leftarrow *) := \left[\sigma_{u1}(u(t_1)); \cdots; \sigma_{u1}(u(t^*)); \cdots; \sigma_{u1}(u(t_N))\right]$.

VIII. Purging to Exploit Improved Parameter Estimates

Due to the fact that $\hat{\Sigma}$ and Δ' depend on the quality of the parameter estimates, a purging technique was incorporated in an attempt to remove poor estimates of $\hat{W}$. During the transient phase of the signals, the estimates in $\hat{\theta}$ are likely to be less accurate and the solution to the least squares problem will likely be poor. Therefore, a purging algorithm was developed and is detailed in Fig. 2.

To determine the quality of the estimate, $\hat{\theta}$, a performance metric is sought. Using the dynamics in (4) and integrating over an interval $[t-\tau_1, t]$, a metric η can be determined using the known state variable, x, and the dynamics, $f^o(x,u) + \hat{\theta}^T \sigma(x,u)$. More specifically,

$$\begin{aligned} \Bigg| x(t) - x(t-\tau_1) - \Bigg(&\int_{t-\tau_1}^{t} f^o(x(\tau), u(\tau))\, \mathrm{d}\tau \\ &+ \int_{t-\tau_1}^{t} \hat{\theta}^T(\tau) \sigma(x(\tau), u(\tau))\, \mathrm{d}\tau \Bigg) \Bigg| = \eta(t). \end{aligned} \quad (26)$$

If this is less than a predetermined constant threshold, then the estimate of $\hat{\theta}$ has improved and the data in the history stack should be purged. Since the parameter error estimates, $\tilde{\theta}$, exponentially converge to zero, a simpler time-based purging technique can also be incorporated, such as $|t-\eta| > \epsilon$, for a predetermined constant $\epsilon > 0$, where η is the time instant of the last purged event.

An indicator variable, η, as defined above, quantifies the quality of the current parameter estimates using a guess-and-check method, which is used to purge and update the history stack. This updated history stack then updates the weight estimate $\hat{W}$ according Fig. 2. The algorithm is initialized with an empty history stack, and an estimate for W_0 is found. Using the algorithm in Fig. 1, estimated values for x, u, $\hat{\theta}$,

```
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 $\overline{\eta}(t) := \min\{\eta(t_1), \cdots, \eta(t_M)\}$.

and η are recorded in the history stack, where $t < T$ and $\eta(t)$ is assumed to be infinite. The initial estimate of $\hat{W}$ is kept constant until the history stack is full. Then, using (25), every time a new data point is added to the history stack, the weight estimate is updated.

In addition to the data recorded along the trajectories of the demonstrator, a query-based approach can also be incorporated in IRL. In the query-based approach, a randomly selected state, x_i, is sent to the demonstrator and the corresponding control value, u_i, is queried, and if these values are deemed to improve the estimate, then they are added to the history stack and used in the subsequent reward estimation calculations.

IX. Analysis

To facilitate the analysis of the IRL algorithm, let $\Sigma := [\sigma(x(t_1), u(t_1), \theta); \cdots; \sigma(x(t_M), u(t_M), \theta)]$ 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 (24) is satisfied, the inverse HJB equation in (17) implies that $\Sigma W = -\Sigma_{u1} - E$, where $E := [\nabla_x\epsilon_V(x(t_1))(f(x(t_1), u(t_1))) + \epsilon_Q(x(t_1)); \cdots; \nabla_x\epsilon_V(x(t_M))(f(x(t_M), u(t_M))) + \epsilon_Q(x(t_M))]$, so $\left\|W + \left(\Sigma^T\Sigma\right)^{-1}\Sigma^T\Sigma_{u1}\right\| \leq \left\|\left(\Sigma^T\Sigma\right)^{-1}\Sigma^T E\right\|$. 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 $\hat{\theta}(t_i)$ in $\hat{\Sigma}_s$ is replaced with the corresponding ideal value 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 (24), $\left\|W - \hat{W}_s^*\right\| \leq \left\|\left(\Sigma_s^T\Sigma_s\right)^{-1}\Sigma_s^T E\right\|$. 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 parameter estimates, $\hat{\theta}$, exponentially converge to θ, the function $(x, \theta) \mapsto \sigma(x, u, \theta)$ is continuous for all u, and under the rank condition in (24), the function $\Sigma \mapsto \left(\Sigma^T\Sigma\right)^{-1}\Sigma^T$ is continuous, so it can be concluded that $\hat{W}_s \to \hat{W}_s^*$ as $s \to \infty$. This means the error between the estimates $\hat{W}_s$ and the ideal weights W is $O(\overline{\epsilon})$ as $s \to \infty$.

X. Simulation

To verify the performance of the developed method, a nonlinear optimal control problem is selected with a known value function. The agent under observation has the following nonlinear dynamics

$$\dot{x_1} = x_2$$
$$\dot{x_2} = f_1x_1\left(\frac{\pi}{2} + \tan^{-1}(5x_1)\right) + \frac{f_2x_1^2}{1 + 25x_1^2} + f_3x_2 + 3u \tag{27}$$

where the parameters f_1, f_2, and f_3 are unknown constants to be estimated. The exact values of these parameters are $f_1 = -1$, $f_2 = -\frac{5}{2}$, and $f_3 = 4$.

The performance index that the agent is trying to minimize is

$$J(x_0, u(\cdot)) = \int_0^\infty (x_2^2 + u^2)dt,$$

resulting in the reward function weights to be estimated as $Q = \text{diag}(0, 1)$ and $R = 1$. The observed state and control trajectories and a prerecorded history stack are used in the estimation of unknown parameters in the dynamics, along with the optimal value function parameters and the reward function weights.

The closed form optimal nonlinear controller is

$$u^* = -\frac{1}{2}R^{-1}g^T(x)(\nabla_x V)^T$$

resulting in the optimal controller

$$u = -3x_2$$

and the optimal value function

$$V^* = x_1^2(v_1 + v_2\tan^{-1}(5x_1)) + v_3x_2^2$$

resulting in the ideal function parameters $v_1 = \frac{\pi}{2}$, $v_2 = 1$, and $v_3 = 1$.

Figs. 3 - 4 show the performance of the proposed method. Fig. 3 shows the parameter convergence of the unknown part of the dynamics in (27). Fig. 4 shows the IRL method estimates the unknown weights in both the reward function and the optimal value function. The parameters used for the simulation were: $T_1 = 1$s, $T_2 = 0.6$s, $k_\theta = 0.5/N$, $N = 150$, $M = 100$, $\beta_1 = 1$, $\Gamma(0) = \text{I}_{3\times3}$, and simulation time step $T_s = 0.005$s.

XI. Conclusion

In this paper, an online nonlinear inverse reinforcement learning method is developed. In order to facilitate reward function estimation online, a parameter estimator is incorporated to allow reward function estimation in the presence of unknown dynamics. Due to the dependency of the reward estimation method on the estimated dynamics, a purging

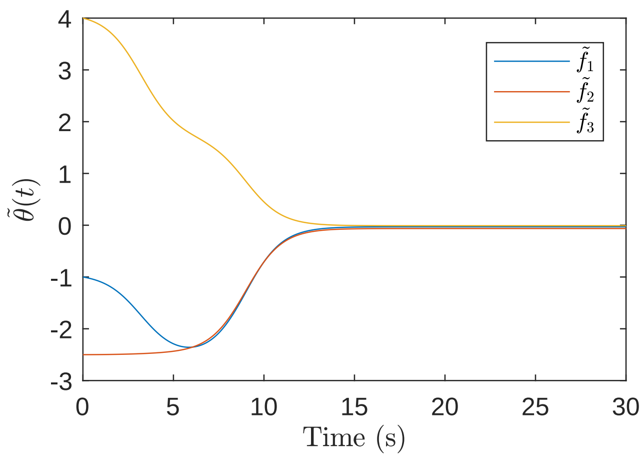

Fig. 3. Estimation error for the unknown parameters in the system dynamics.

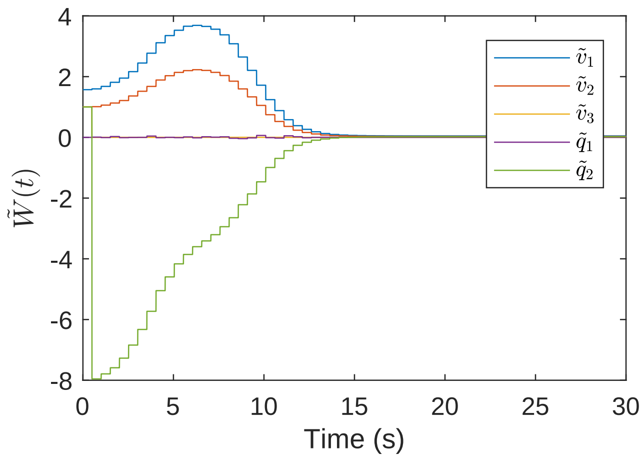

Fig. 4. Estimation error for the unknown parameters in the reward function.

technique was developed to ensure that reward function estimation always utilizes the best available estimates of the system model. The method was validated by simulating a nonlinear system to show convergence of the unknown parameters in the dynamics, the reward function, and the optimal value function.

Future work will focus on the development of output feedback IRL methods that utilize both state and parameter estimation methods, and extensions of the developed technique to handle suboptimal demonstrations.

REFERENCES

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

[2] S. Russell, "Learning agents for uncertain environments (extended abstract)," in *Proceedings of the eleventh annual conference on Computational learning theory*, 1998.

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

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

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

[6] 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.

[7] Z. Zhou, M. Bloem, and N. Bambos, "Infinite time horizon maximum causal entropy inverse reinforcement learning," *IEEE Transactions on Automatic Control*, vol. 63, no. 9, pp. 2787–2802, 2018.

[8] 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. [Online]. Available: http://papers.nips.cc/paper/3918-feature-construction-for-inverse-reinforcement-learning.pdf

[9] 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.

[10] 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. [Online]. Available: http://papers.nips.cc/paper/3293-a-game-theoretic-approach-to-apprenticeship-learning.pdf

[11] M. Wulfmeier, P. Ondruska, and I. Posner, "Maximum entropy deep inverse reinforcement learning," arXiv:1507.04888, 2015.

[12] 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. [Online]. Available: http://papers.nips.cc/paper/4420-nonlinear-inverse-reinforcement-learning-with-gaussian-processes.pdf

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

[14] 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.

[15] 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.

[16] 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.

[17] R. Kamalapurkar, "Linear inverse reinforcement learning in continuous time and space," in *Proc. Am. Control Conf.*, Jun. 2018, pp. 1683–1688. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8431430/

[18] T. Molloy, J. Ford, and T. Perez, "Online inverse optimal control on infinite horizons," in *IEEE Conf. Decis. Control.* IEEE, 2018, pp. 1663–1668.

[19] R. Kamalapurkar, "Simultaneous state and parameter estimation for second-order nonlinear systems," in *Proc. IEEE Conf. Decis. Control*, Melbourne, VIC, Australia, Dec. 2017, pp. 2164–2169. [Online]. Available: http://ieeexplore.ieee.org/document/8263965/

[20] G. Chowdhary, "Concurrent learning for convergence in adaptive control without persistency of excitation," Ph.D. dissertation, Georgia Institute of Technology, Dec. 2010.

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

[22] H. K. Khalil, *Nonlinear systems*, 3rd ed. Upper Saddle River, NJ: Prentice Hall, 2002.