Data-driven output feedback optimal control for a class of nonlinear systems via adaptive dynamic programming approach: Part I: Algorithms

Wang, Y.

TR2018-107 July 28, 2018

Abstract

Approximate/adaptive dynamic programming (ADP) has demonstrated great successes in the construction of datadriven output feedback optimal control for linear time-invariant systems and data-driven state feedback optimal control for nonlinear systems. This work investigates data-driven output feedback optimal control design for a class of nonlinear systems. It proposes to parameterize all admissible output feedback optimal control policies over accessible signals (system output and its time derivatives). In the case that system state can be parameterized as functions of accessible signals, then the value function and control policy can be parameterized over accessible signals, which allow ADP to be driven by accessible data. For a special case,where system state, value function and control policy can be linearly parameterized over a finite functional space over accessiblesignals, the policy iteration algorithm (PI) of ADP is reduced to solve a system of linear equations. Two data-driven PIs are developed to accomplish data-driven output feedback optimal control design. Simulation validates the proposed methodology.

Chinese Control Conference (CCC)

Data-driven output feedback optimal control for a class of nonlinear systems via adaptive dynamic programming approach: Part I–Algorithms

Yebin Wang
Mitsubishi Electric Research Laboratories, Cambridge, MA 02139, USA (e-mail: yebinwang@ieee.org).

Abstract: Approximate/adaptive dynamic programming (ADP) has demonstrated great successes in the construction of data-driven output feedback optimal control for linear time-invariant systems and data-driven state feedback optimal control for nonlinear systems. This work investigates data-driven output feedback optimal control design for a class of nonlinear systems. It proposes to parameterize all admissible output feedback optimal control policies over accessible signals (system output and its time derivatives). In the case that system state can be parameterized as functions of accessible signals, then the value function and control policy can be parameterized over accessible signals, which allow ADP to be driven by accessible data. For a special case, where system state, value function and control policy can be linearly parameterized over a finite functional space over accessible signals, the policy iteration algorithm (PI) of ADP is reduced to solve a system of linear equations. Two data-driven PIs are developed to accomplish data-driven output feedback optimal control design. Simulation validates the proposed methodology.

1 Introduction

Longstanding research on optimal control theory has resulted in encouraging contributions, to name a few, dynamic programming to determine a feedback control policy [1, 2], minimum principle to derive and solve necessary optimality conditions for an open-loop optimal control trajectory [3, 4], numerical optimization to compute an open-loop optimal control trajectory [5, 6], etc. This work follows the approximate/adaptive dynamic programming (ADP) approach [7] to synthesize output feedback optimal control policy without knowing the system dynamics. Consider a nonlinear system

$$\begin{aligned} \dot{x} &= f(x) + g(x)u, \quad x(0) = x_0 \in \Omega_x \\ y &= h(x), \end{aligned} \tag{1}$$

where $x \in \Omega_x \subset \mathbb{R}^n$ the system state vector, Ω_x a compact set containing the origin in its interior, $u \in \mathbb{R}^m$ the control input, $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is a vector field, $g : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ consists of m smooth vector fields, and $h : \mathbb{R}^n \to \mathbb{R}^p$ is a vector of p smooth functions. Both f and g are unknown. This work studies the data-driven output feedback optimal control design to minimize the following cost functional

$$J(u) = \int_0^\infty \left[y^\top Q y + u^\top R u \right] dt, \tag{2}$$

where Q and R are positive definite matrices. If system (1) is uniformly state observable through a virtual output $Q^{1/2}y$, finiteness of the cost function (2) implies the system stability.

The ADP, narrowly speaking standard policy iteration algorithm (PI) or value iteration algorithm (VI) [7,8], has been widely accepted as a powerful tool to construct data-driven feedback optimal control policies for a plethora of scenarios. Its success has been particularly acclaimed when the system is linear time-invariant (LTI), for example, state feedback optimal stabilization [9], state feedback optimal output regulation [10], output feedback optimal stabilizing control [11], output feedback optimal output regulation [12], etc. When the system is nonlinear, its applications have been limited to the state feedback case, for instance, state feedback optimal stabilization [8, 13–15], and state feedback optimal stabilization of descriptor systems [16].

To the best knowledge of authors, the endeavor in resolving data-driven output feedback optimal control for nonlinear systems turns out to be vain. This paper tackles data-driven output feedback optimal control design by making the following contributions

(i) proposes parameterizations of admissible output feedback optimal control policies which permits the exploitation of ADP: the control policy u is parameterized over y and its time derivatives up to a certain order. Such parameterizations are supported by established work on high gain and sliding mode observers;
(ii) reveals that as long as system state x can be parameterized as functions of y and its time derivatives, then the ADP can be applied without knowing x;
(iii) offers exemplary parameterizations of the value function V and control u over y and its time derivatives;
(iv) reduces the differential equations in PI into a system of linear equations to solve for V and u; and develops two algorithms to implement data-driven PI;
(v) performs simulation validation.

The remainder of this paper is structured as follows. Section 2 formulates feedback optimal control problems. Reparameterizations and data-driven PI are presented in Section 3. Simulation validation is performed in Section 4. Section 5 offers future research directions and conclusion.

2 Preliminaries

Definition 2.1 (Admissible state feedback control) *A state feedback control policy $u(x) \in U_x \subset \mathbb{C}^1[0, T]$ is admissible if, for any initial condition $x_0 \in \Omega_x$, the resultant closed-loop system is stable. Correspondingly, U_x is called the admissible state feedback control set.*

Definition 2.1 assumes state feedback. It makes the state feedback optimal control problem exposed to well-established theories, e.g. dynamic programming. Defining U_x as the set of all admissible state feedback control policies, we assume that U_x is not empty, i.e., $U_x \neq \emptyset$.

It is without loss of generality to take the cost function (2) with $T = \infty$. For such a case, an admissible state feedback control policy should yield a finite value of the cost function, and a stable closed-loop system. The state feedback optimal control problem for system (1) can be formulated as follows.

Problem 2.2 (State feedback optimal control problem) *Given system* (1), *find* $u^*(x^*) \in U_x$ *which minimizes the cost function* (2), *i.e.* $u^*(x^*) = \arg\min_{u(x)\in U_x} J(u(x))$.

According to dynamic programming, the optimal control solution $u^*(x)$ to Problem 2.2 can be obtained by solving the Hamilton-Jacobi-Bellman (HJB) equations

$$0 = \min_{u\in U_x} \{\nabla V(f + gu) + (h(x))^\top Qh(x) + u^\top Ru\}, \quad (3)$$

with $V(x(\infty)) = 0$ and $\nabla V = \partial V/\partial x$. A closed-form solution of HJB is notoriously difficult to establish. Instead, ADP techniques, e.g. PI and VI, are exploited to acquire an approximate solution [7,8,17]. Due to the similarity between PI and VI, this work concerns itself with PI.

PI for system (1) with state measurements is summarized in the following two iterated steps, with $i = 0, 1, \cdots$. Assume that an admissible control policy $u_0(x)$ is known.

(i) Policy evaluation: Solve for the positive definite function $V_i(x)$ satisfying

$$\begin{aligned}\nabla V_i(f(x) + g(x)u_i(x)) + y^\top Qy \\ + u_i^\top(x)Ru_i(x) = 0, \quad \forall x \in \Omega_x,\end{aligned} \quad (4)$$

where $\nabla V_i = \partial V_i(x)/\partial x$ is a row vector.

(ii) Policy improvement: Update the control policy

$$u_{i+1}(x) = -\frac{1}{2}R^{-1}(\nabla V_i g)^\top. \quad (5)$$

As a system of first order linear partial differential equations (PDEs), the closed-form solution of (4) remains difficult to establish. Instead, an approximate solution is practically of interest. Given parameterizations of u_i and V_i, (4) can be casted into algebraic equations, and the approximate solution can be computed. The two steps (4)-(5) shall be repeated until the convergence is attained.

Given an initial admissible state feedback control policy $u_0(x)$, PI generates a sequence of control $\{u_i\}$ which possesses the following properties

(i) u_i leads to a stable closed-loop system;

(ii) the closed-loop system performance measured by the cost function improves, i.e., $J(u_{i+1}) \leq J(u_i)$;

(iii) it is guaranteed to converge as $i \to \infty$.

Turning to the output feedback case, we restrict ourselves by assuming the following controller parameterizations.

Definition 2.3 (Admissible output feedback control) *A dynamic output feedback control policy* $u(z) \in U_z \subset \mathbb{C}^1[0,T]$ *with* $z = [y, \dot{y}, \ldots, y^{(m_y)}] \in \mathbb{R}^{n_z}$ *is admissible if, for any initial condition* $x_0 \in \Omega_x$, *the resultant closed-loop system is stable. Correspondingly,* U_z *is called the admissible output feedback control set.*

Remark 2.4 *Definition 2.3 assumes dynamic output feedback. Particularly, the controller is parameterized by output and its time derivatives, which can be estimated through dynamic estimators including high gain observer [18] and sliding mode-based exact differentiators [19,20]. With the dynamic estimator, an admissible output feedback control policy is implemented as* $u(\hat{z})$. *It is apparent that* $u(\hat{z})$ *does not necessarily stabilize the closed-loop system, even if* $u(z)$ *is admissible [21]. This topic however falls outside of the scope of this work, and is left for future research. Another interesting topic is to analyze the robustness of the resultant output feedback optimal control policy* $u^*(z^*)$

Defining U_z as the set of all admissible output feedback control policies, we assume that U_z is non-empty. The output feedback optimal control problem for system (1) can be formulated as follows.

Problem 2.5 (Output feedback optimal control problem) *Given system* (1), *find* $u^*(z) \in U_z$ *which minimizes the cost function* (2), *i.e.* $u^*(z^*) = \arg\min_{u(z)\in U_z} J(u(z))$.

3 Main Results

This section is dedicated to solve Problem 2.5. Different from the state feedback case where (4)-(5) during PI are parameterized over x, we need to re-parameterize equations over z to perform the output feedback control synthesis.

3.1 PI Parameterizations

We briefly recall parameterizations of the control policy $u_i(x)$ and value function $V_i(x)$ employed in the standard PI (driven by state measurements), where i is the iteration index.

Let $\{\phi_j^V(x)\}_{j=1}^N$ with $\phi_j^V : \mathbb{R}^n \to \mathbb{R}$ and $\{\phi_j^{Vg}(x)\}_{j=1}^q$ with $\phi_j^{Vg} : \mathbb{R}^n \to \mathbb{R}^m$ be two sets of linearly independent, continuously differentiable functions and vector fields, respectively. In addition, we assume that $\phi_j^V(0) = 0$, $\forall\, 1 \leq j \leq N$ and $\phi_j^{Vg}(0) = 0, \forall\, 1 \leq j \leq q$.

Assumption 3.1 *Provided that* $u_i(x) \in U_x$, *and* $u_i(x) \in \text{span}\{\phi_1^{Vg}(x), \cdots, \phi_q^{Vg}(x)\}$, *then,*

$$\begin{aligned}V_i(x) &\in \text{span}\{\phi_1^V(x), \cdots, \phi_N^V(x)\}, \\ u_{i+1}(x) &\in \text{span}\{\phi_1^{Vg}(x), \cdots, \phi_q^{Vg}(x)\},\end{aligned}$$

where $V_i(x)$ *and* $u_{i+1}(x)$ *are obtained from* (4) *and* (5).

Thanks to (5), Assumption 3.1 implies

$$(\nabla V_i g)^\top \in \text{span}\{\phi_1^{Vg}(x), \cdots, \phi_q^{Vg}(x)\},$$

and the existence of sets of weights $\{\theta_{i,1}^V, \cdots, \theta_{i,N}^V\}$, $\{\theta_{i,1}^{Vg}, \cdots, \theta_{i,q}^{Vg}\}$, and $\{\theta_{i+1,1}^{Vg}, \cdots, \theta_{i+1,q}^{Vg}\}$, such that

$$\begin{aligned}V_i(x) &= \sum_{j=1}^N \theta_{i,j}^V \phi_j^V(x) \\ u_i(x) &= \sum_{j=1}^q \theta_{i,j}^{Vg} \phi_j^{Vg}(x) \\ u_{i+1}(x) &= \sum_{j=1}^q \theta_{i+1,j}^{Vg} \phi_j^{Vg}(x).\end{aligned}$$

Remark 3.2 *When Assumption 3.1 is not satisfied, these weights can still be numerically obtained based on neural* *network approximation methods, such as the off-line approximation using Galerkin's method [17]. In addition, for uncertain nonlinear systems, these weights can be trained using ADP-based online learning methods [8, 22]. When approximation methods are used, Ω_x has to be a compact set to guarantee the boundedness of the approximation error.*

3.2 PI Re-Parameterizations

We next re-parameterize $u_i(x)$ and $V_i(x)$ over z so that data-driven PI can be driven by accessible signals z instead of x. For simplicity, the following assumption is introduced.

Assumption 3.3 *The state x can be represented by functions of y and its up to m_yth time derivatives, i.e.,*

$$x = x(y, \ldots, y^{(m_y)}).$$

Assumption 3.3 seems restrictive, but might be relaxed by introducing alternative parameterizations of admissible output feedback controllers in Definition 2.3. We identify several classes of nonlinear systems which satisfy Assumption 3.3: input-output linearizable systems, and flat systems. Relaxation of Assumption 3.3 could be an interesting topic for future research.

Remark 3.4 *With Assumption 3.3, all admissible state feedback control policy can be re-parameterized as functions of z, i.e., $U_x \subset U_z$. A simple treatment of Problem 2.5 is to search $u(z)$ over U_x instead of U_z. In the case that the map between z and x is globally diffeomorphic, Problems 2.2 and 2.5 are equivalent, since $U_x = U_z$.*

The forthcoming discussion is contingent on linear parameterizations of x over z. Let $\{\phi_j^x(z)\}_{j=1}^S$ with $\phi_j^z : \mathbb{R}^{n_z} \to \mathbb{R}$ be a set of linearly independent, continuously differentiable functions. Assume that $\phi_j^x(0) = 0, \forall\, 1 \le j \le S$.

Assumption 3.5 *Provided that $x \in \Omega_x$, then,*

$$x_i \in \operatorname{span}\{\phi_1^x(z), \cdots, \phi_S^x(z)\},$$

where x_i is the ith component of x.

Assumption 3.5 suggests there exist a set of weights $\{\theta_{i,1}^x, \cdots, \theta_{i,S}^x\}$ for $1 \le i \le n$, such that

$$x = \left[\sum_{j=1}^S \theta_{1,j}^x \phi_j^x(z), \ldots, \sum_{j=1}^S \theta_{n,j}^x \phi_j^x(z)\right]^\top.$$

For brevity, we denote

$$\begin{aligned}
\Theta_i^V &= [\theta_{i,1}^V, \ldots, \theta_{i,N}^V] \in \mathbb{R}^N \\
\Theta_i^{Vg}(x) &= [\theta_{i,1}^{Vg}, \ldots, \theta_{i,q}^{Vg}] \in \mathbb{R}^q \\
\Theta_i^x(z) &= [\theta_{i,1}^x, \ldots, \theta_{i,S}^x] \in \mathbb{R}^S \\
\Phi^V(x) &= [\phi_1^V(x), \ldots, \phi_N^V(x)]^\top \in \mathbb{R}^N \\
\Phi^{Vg}(x) &= [\phi_1^{Vg}(x), \ldots, \phi_q^{Vg}(x)]^\top \in \mathbb{R}^{q\times m} \\
\Phi^x(z) &= [\phi_1^x(z), \ldots, \phi_S^x(z)]^\top \in \mathbb{R}^S.
\end{aligned}$$

Hence, $V_i(x) = \Theta^V \Phi^V(x), u_i(x) = \Theta^{Vg}\Phi^{Vg}(x)$, and $x = \Theta^x \Phi^x(z)$, where

$$\Theta^x = \begin{bmatrix} \Theta_1^x \\ \vdots \\ \Theta_n^x \end{bmatrix} = \begin{bmatrix} \theta_{1,1}^x & \cdots & \theta_{1,S}^x \\ \vdots & \ddots & \vdots \\ \theta_{n,1}^x & \cdots & \theta_{n,S}^x \end{bmatrix}.$$

Next we re-parameterize $\Phi^V(x), \Phi^{Vg}(x)$ as functions of $\Phi^x(z)$. For illustration purpose, assume that each component of $\Phi^V(x), \Phi^{Vg}(x)$ is a polynomial function of x, for instance, $\Phi^V(x) = [x_1^2, \ldots, x_i x_j, \ldots]^\top$. The term $x_i x_j$ takes the following parameterizations

$$x_i x_j = \Theta_i^x \Phi^x(z) \Theta_j^x \Phi^x(z) = (\Theta_i^x \otimes \Theta_j^x)(\Phi^x(z) \otimes \Phi^x(z)),$$

where $\otimes$ represents the Kronecker product. Similarly

$$x_i^k = (\Theta_i^x \otimes \cdots \otimes \Theta_i^x)(\Phi^x(z) \otimes \cdots \otimes \Phi^x(z)).$$

As a result, $\Phi^V(x)$ takes the following representation

$$\Phi^V(x) = [x_1^2, \ldots x_i x_j, \ldots, x_1^4, \ldots]^\top = \Theta^{V,z}\Phi^{V,z}(z),$$

where

$$\Theta^{V,z} = \begin{bmatrix} \Theta_1^x \otimes \Theta_1^x & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ \Theta_i^x \otimes \Theta_i^x & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \Theta_1^x \otimes \Theta_1^x \otimes \Theta_1^x \otimes \Theta_1^x & \vdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & * \end{bmatrix}$$

$$\Phi^{V,z}(z) = \begin{bmatrix} \Phi^x(z) \otimes \Phi^x(z) \\ \Phi^x(z) \otimes \Phi^x(z) \otimes \Phi^x(z) \otimes \Phi^x(z) \\ \vdots \\ \Phi^x(z) \otimes \cdots \otimes \Phi^x(z) \end{bmatrix}.$$

One can similarly re-parameterize $\Phi^{Vg}(x) = \Theta^{Vg,z}\Phi^{Vg,z}(z)$. In the end, re-parameterizations of u and V arrive at the following formula

$$\begin{aligned}
u(z) &= -\frac{1}{2}R^{-1}\Theta^{Vg}\Phi^{Vg}(x) = -\frac{1}{2}R^{-1}\bar{\Theta}^{Vg}\bar{\Phi}^{Vg}(z) \\
V &= \Theta^V \Phi(x) = \bar{\Theta}^V \bar{\Phi}^V(z),
\end{aligned}$$

where $\bar{\Theta}^{Vg} = \Theta^{Vg}\Theta^{Vg,z}, \bar{\Theta}^V = \Theta^V \Theta^{V,z}$.

3.3 Data-Driven PI

With aforementioned re-parameterizations, we can rewrite (4)-(5), where the newly parameterized equations comprise of unknown parameters and z. More precisely, linear parameterizations permit us to reduce the newly parameterized equations to a system of linear equations. Assume that system (1) is subject to control

$$u(z) = \underbrace{-\frac{1}{2}R^{-1}\bar{\Theta}^{Vg}\bar{\Phi}^{Vg}(z)}_{K(z)} + v(t),$$

where $v(t) \in \mathbb{R}^m$. The resultant closed-loop system is

$$\dot{x} = f(x) + g(x)K(z) + g(x)v(t). \tag{6}$$

Remark 3.6 *The cost function of the closed-loop system* (6) *is time-varying and takes the following values at t and $t + \delta$*

$$\begin{aligned}
V_v(t, x(t)) &= \int_t^\infty (y^\top Q y + u^\top R u)\mathrm{d}t \\
V_v(t, x(t+\delta)) &= \int_{t+\delta}^\infty (y^\top Q y + u^\top R u)\mathrm{d}t.
\end{aligned}$$

The cost function V_v does not satisfy (4)*, due to the time-varying signal $v(t)$. Instead, it satisfies the following time-varying differential equations*

$$-\frac{\partial V_v}{\partial t} = \nabla V_v(f+gK+gv)+y^\top Qy+(K+v)^\top R(K+v),$$

which is non-trivial to solve. Also, solving V_v is not helpful to derive the control policy in the form of $u(z)$.

To synthesize the output feedback optimal control policy $u^*(z^*)$ via ADP, it is relevant to solve the value function $V(z)$, which corresponds to the closed-loop system with the control $u = K(z)$, i.e.,

$$\dot{x} = f(x) + g(x)K(z). \tag{7}$$

This fact implies the constraint that $K(z)$ and V satisfy the following equations

$$\begin{aligned}\nabla V(f(x) + g(x)K(z)) = \\ -(y^\top Qy + (K(z))^\top RK(z)).\end{aligned} \tag{8}$$

Basic idea of the following derivation is to infer $V(z)$ and ∇Vg from output trajectories of the closed-loop system (6). Along the trajectory of the closed-loop system (6), the change of V during the time interval $[t, t+\delta]$ is given by

$$\begin{aligned}\mathrm{d}V(t) &= V(x(t+\delta)) - V(x(t)) \\ &= \bar{\Theta}^V\{\bar{\Phi}^V(z(t+\delta)) - \bar{\Phi}^V(z(t))\} \\ &= \int_t^{t+\delta} (\nabla V(x)(f(x)+g(x)K(z)+g(x)v(t)))\mathrm{d}t\end{aligned}$$

Taking (8) into account, we rearrange $\mathrm{d}V(t)$ and have

$$\begin{aligned}&\bar{\Theta}^V\{\bar{\Phi}^V(z(t+\delta)) - \bar{\Phi}^V(z(t))\} \\ &= \int_t^{t+\delta} (\nabla Vg(x)v(t) - (y^\top Qy + (K(z))^\top RK(z)))\mathrm{d}t \\ &= \bar{\Theta}^{Vg}\int_t^{t+\delta} \bar{\Phi}^{Vg}(z)v(t)\mathrm{d}t \\ &\quad - \int_t^{t+\delta} (y^\top Qy + (K(z))^\top RK(z))\mathrm{d}t.\end{aligned}$$

This is reduced to one linear equation

$$\bar{\Theta}\Psi(t) = \rho(t),$$

where $\bar{\Theta} = [\bar{\Theta}^V, \bar{\Theta}^{Vg}]$, and

$$\begin{aligned}\Psi(t) &= \begin{bmatrix}\Delta\Phi^V \\ \psi(t)\end{bmatrix} \\ \rho(t) &= \int_t^{t+\delta} (y^\top Qy + (K(z))^\top RK(z))\mathrm{d}t \\ \Delta\Phi^V(t) &= \bar{\Phi}^V(z(t)) - \bar{\Phi}^V(z(t+\delta)) \\ \psi(t) &= \int_t^{t+\delta} \bar{\Phi}^{Vg}(z)v(t)\mathrm{d}t.\end{aligned} \tag{9}$$

$\bar{\Theta}$ is a vector of unknown parameters. By collecting data during intervals $[t, t+\delta,], [t+\delta, t+2\delta], \ldots, [t+(M_j-1)\delta, t+M_j\delta]$ with $N+q \le M_j < \infty$, one can form a system of linear equations

$$\bar{\Theta}\Psi = \rho, \tag{10}$$

where $\Psi = [\Psi(t), \Psi(t+\delta), \ldots, \Psi(t+M_j\delta)]$ and $\rho = [\rho(t), \rho(t+\delta), \ldots, \rho(t+M_j\delta)]$. As long as $\Psi\Psi^\top$ is non-singular, $\bar{\Theta}$ is uniquely determined as $\bar{\Theta} = \rho\Psi^\top(\Psi\Psi^\top)^{-1}$.

Algorithm 1 provides detailed steps of data-driven PI.

Algorithm 1: Data-driven PI

```
Initialize $i = 0, j = 0, t_0 = 0, \delta, M_j, M_i, \bar{\Theta}_0^{Vg}$;
Define an initial admissible output feedback control policy
    $K_0(z) = -\frac{1}{2}R^{-1}\bar{\Theta}_0^{Vg}\Phi^{Vg}(z)$.
$t_s = t_0$;
for $i \le M_i$ do
    Initialize $t_s = t_s + j\delta$;
    $j = 0, \Psi = \emptyset, \rho = \emptyset$;
    while $j \le M_j$ do
        $t = t_s + j\delta$;
        Let $v_j(t)$ be a vector of small constants over $[t, t+\delta]$;
        $u_i(z) = K_i(z) + v_i(t)$;
        calculate $\Psi(t), \rho(t)$ according to (9);
        Update $\Psi = [\Psi, \Psi(t)], \rho = [\rho, \rho(t)]$;
        $j = j + 1$;
        if $\det(\Psi(\Psi)^\top) \ne 0$ then
            Solve (10) for $\bar{\Theta}$;
            Update $K_{i+1} = -\frac{1}{2}R^{-1}\bar{\Theta}_i^{Vg}\Phi^{Vg}(z)$;
            break;
    $i = i + 1$;
    if $|\bar{\Theta}_{i-1} - \bar{\Theta}_i| < \epsilon$ then
        break;
return $\bar{\Theta}_{i-1}$;
```

In Algorithm 1, i is the index for PI, M_i is the maximum number of iterations, j tracks episodes of measurements to form well-conditioned linear equations (10),and M_j indicates the maximum number of episodes.

Algorithm 1 determines $\bar{\Theta}^V$ and $\bar{\Theta}^{Vg}$ jointly. Intuitively, the deduction of $\bar{\Theta}^V$ and $\bar{\Theta}^{Vg}$ relies on decoupling impacts of $K(z)$ and v on $\rho(t)$ into $\Delta\Phi^V(t)$ and $\psi(t)$. Although $K(z)$ is a stabilizing control policy, $v_i(t)$ ought to be small enough, compared with $K(z)$, to avoid de-stabilize the closed-loop system (6). This means that $\Delta\Phi^V(t)$ might be much larger than $\psi(t)$. Eventually the matrix Ψ may suffer from ill-conditionedness.

We propose Algorithm 2 as an alternative to implement the data-driven PI more reliably. Basic idea is to determine $V(z)$ through output trajectories of the closed-loop system (7); and then work out ∇Vg by utilizing output trajectories of the closed-loop system (7). Algorithm 2 splits data-driven PI into three steps: for $i = 0, 1, \ldots$

(i) Policy evaluation: apply $u_i(z) = K_i(z)$ and measure the output of system (7) to construct linear equations

$$\bar{\Theta}_i^V\Delta\Phi^V = \rho, \tag{11}$$

where $\Delta\Phi^V = [\Delta\Phi^V, \Delta\Phi^V(t)], \rho = [\rho, \rho(t)]$; solve (11) for $\bar{\Theta}_i^V$;

(ii) Gradient determination: resolve $\bar{\Theta}_i^{Vg}$ by forming and solving the following linear equations

$$\bar{\Theta}_i^{Vg}\Psi^{Vg} = \rho - \bar{\Theta}_i^V\Delta\Phi^V, \tag{12}$$

where $\Psi^{Vg} = [\psi(t), \ldots, \psi(t + M_j\delta)]$ and $\Delta\Phi^V = [\Delta\Phi^V(t), \ldots, \Delta\Phi^V(t+M_j\delta)]$ are generated by output of system (6);

(iii) Policy improvement: update the control policy:

$$K_{i+1}(z) = -\frac{1}{2}R^{-1}\bar{\Theta}_i^{Vg}\Phi^{Vg}(z). \tag{13}$$

Readers are referred to Algorithm 2 for details.

Algorithm 2: Decomposition-based Data-driven PI

```
Initialize $i = 0, j = 0, t_0 = 0, \delta, M_j, M_i, \bar{\Theta}_0^{Vg}$;
Define an initial stabilizing control policy
    $K_0 = -\frac{1}{2}R^{-1}\bar{\Theta}_0^{Vg}\Phi^{Vg}(z)$.
$t_s = t_0$;
for $i \le M_i$ do
    Initialize $t_s = t_s + j\delta$;
    $j = 0, \rho = \emptyset, \Delta\Phi^V = \emptyset$;
    $u_i(z) = K_i(z)$;
    while $j \le M_j$ do
        $t = t_s + j\delta$;
        Calculate $\Delta\Phi^V(t), \rho(t)$ according to (9);
        Update $\Delta\Phi^V = [\Delta\Phi^V, \Delta\Phi^V(t)], \rho = [\rho, \rho(t)]$;
        if $\det(\Delta\Phi(\Delta\Phi)^\top) \neq 0$ then
            Solve (11) for $\bar{\Theta}_i^V$;
            break;
        $j = j + 1$;
    $j = 0, \Psi^{Vg} = \emptyset, \rho = \emptyset, \Delta\Phi^V = \emptyset$;
    while $j \le M_j$ do
        Let $v_i(t)$ be a vector of small constants over $[t, t+\delta]$;
        $u_i(z,t) = K_i(z) + v_i(t)$;
        $t = t_s + j\delta$;
        Calculate $\Delta\Phi^V(t), \Psi(t), \rho(t)$ according to (9);
        Update $\Psi^{Vg} = [\Psi^{Vg}, \psi(t)], \Delta\Phi^V =$
        $[\Delta\Phi^V, \Delta\Phi^V(t)], \rho = [\rho, \rho(t)]$;
        $j = j + 1$;
        if $\det(\Psi^{Vg}(\Psi^{Vg})^\top) \neq 0$ then
            Solve (12) for $\bar{\Theta}_i^{Vg}$;
            Update the control policy according to (13);
            break;
    $i = i + 1$;
    if $|\bar{\Theta}_{i-1} - \bar{\Theta}_i| < \epsilon$ then
        break;
return $\bar{\Theta}_{i-1}$;
```

4 Simulation

Consider a second order nonlinear system

$$\begin{aligned}\dot{x} &= \begin{bmatrix} -x_1 + x_2 \\ -x_2 - dx_2^3 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u \\ y &= Cx,\end{aligned} \tag{14}$$

where $C = [1, 0]$. The objective is to minimize the cost function (2) with the weight $Q = 1000, R = 1$. Let $z = [y, \dot{y}]^\top$. Parameterizations of x over z give

$$x = [y, \dot{y} + y]^\top = \underbrace{\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}}_{\Theta^x} \underbrace{\begin{bmatrix} y \\ \dot{y} \end{bmatrix}}_{\Phi^x(z)}.$$

Assumption 3.3 is verified. Note that Θ^x is not required during the process of data-driven PI. For comparison, we solve both the standard PI and data-driven PI for two cases:

(i) Case 1: $d = 0$, i.e., system (14) is LTI;

(ii) Case 2: $d = 1$, i.e., system (14) is nonlinear.

Both the standard PI and data-driven PI take the same parameterizations of V and u over x, i.e.,

$$\begin{aligned}\Phi^V(x) &= \{x_1^2, x_2^2, x_1x_2, x_1^4, x_2^4, x_1^2x_2^2\} \in \mathbb{R}^6 \\ \Phi^{Vg}(x) &= \{x_1, x_2, x_1^3, x_2^3, x_1x_2^2, x_1^2x_2\} \in \mathbb{R}^6.\end{aligned}$$

Assumption 3.1 holds for the above choices of $\Phi^V(x)$ and $\Phi^{Vg}(x)$. Re-parameterizations for data-driven PI give

$$\begin{aligned}\bar{\Phi}^V(z) &= \begin{bmatrix} \Phi^x(z) \otimes \Phi^x(z) \\ \Phi^x(z) \otimes \Phi^x(z) \otimes \Phi^x(z) \otimes \Phi^x(z) \end{bmatrix} \\ \bar{\Phi}^{Vg}(z) &= \begin{bmatrix} \Phi^x(z) \otimes \Phi^x(z) \\ \Phi^x(z) \otimes \Phi^x(z) \otimes \Phi^x(z) \end{bmatrix},\end{aligned}$$

which can be greatly simplified by taking system properties into account. In this example, considering that x is a linear function of z, we can down select the basis $\bar{\Phi}^V(z)$ as follows

$$\begin{aligned}x_2^4 &\in \text{span}\{z_1^4, z_1z_2^3, z_1^2z_1^2, z_1^3z_2, z_2^4\} \\ x_1^2x_2^2 &\in \text{span}\{z_1^4, z_1^3z_2, z_1^2z_1^2\},\end{aligned}$$

and $\bar{\Phi}^V(z)$ is $\{z_1^2, z_2^2, z_1z_2, z_1^4, z_2^4, z_1z_2^3, z_1^3z_2, z_1^2z_2^2\} \in \mathbb{R}^8$. Similarly we have

$$\begin{aligned}x_2^3 &\in \text{span}\{z_1^3, z_1z_2^2, z_1^2z_2, z_2^3\} \\ x_1x_2^2 &\in \text{span}\{z_1^3, z_1^2z_2, z_1z_2^2\} \\ x_1^2x_2 &\in \text{span}\{z_1^3, z_1^2z_2\},\end{aligned}$$

which implies $\bar{\Phi}^{Vg}(z) = \{z_1, z_2, z_1^3, z_1z_2^2, z_1^2z_2, z_2^3\} \in \mathbb{R}^6$. One can verify Assumption 3.1 for $\Phi^V(z)$ and $\Phi^{Vg}(z)$. With expressions of $\bar{\Phi}^V(z)$ and $\bar{\Phi}^{Vg}(z)$, Algorithm 2 is implemented to solve Problem 2.5.

Note that PI assumes the knowledge of f, g and x, whereas data-driven PI merely requires z. For both cases, data-driven PI and standard PI begin with the same initial control policy, albeit with different parameterizations, $u = -2z_1 - 2z_2$ and $u = -2x_2$, respectively. Simulation results for Case 1 and Case 2 are summarized by Figs. 1-4 and Figs. 5-8, respectively. Figs. 1 and 5 show the values of the cost function (2) by simulating the closed-loop systems with control policies generated by PI and data-driven PI, respectively. Figs. 2 and 6 illustrate the errors between the cost function (2) evaluated by simulating the closed-loop systems and the approximate value functions obtained by PI and data-driven PI. Note that when evaluate the cost function, the closed-loop systems start from the same initial conditions $x(0) = [-1, 1]^\top$. Figs. 3 and 7 depict trajectories of x_1 and u of the closed-loop systems with three distinct control policies: the initial control policy $u_0(x)$, the state feedback optimal control policy $u^*(x)$ by PI, and the output feedback optimal control policy $u^*(z)$ by data-driven PI. Figs. 4 and 8 visualize the approximate value functions obtained by PI and data-driven PI.

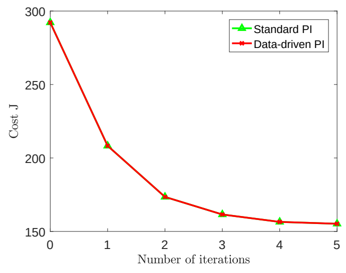

Fig. 1: Case 1: Values of the cost function (2) over iterations

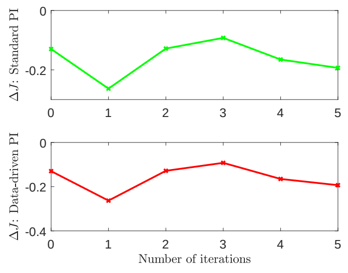

Fig. 2: Case 1: Errors between the cost function (2) and approximate value functions over iterations

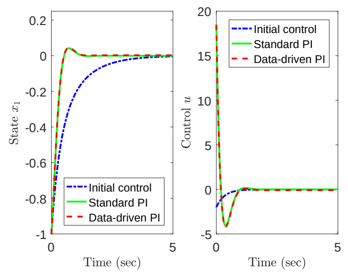

Fig. 3: Case 1: State and control trajectories

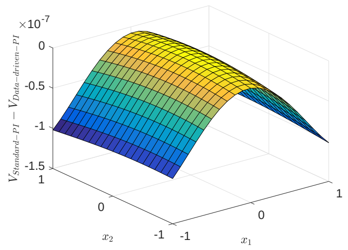

Fig. 4: Case 1: Error between approximate value functions

For both cases, data-driven PI converges within a few iterations. As shown in Figs. 1 and 5, the PI and data-driven PI converge at the same rate for Case 1, while the latter is a little slower than the former for Case 2. All figures corroborate that results of data-driven PI largely coincide with the standard PI. Due to distinct parameterizations, both PI and data-driven PI produce seemingly different results as far as the value function and optimal control policy concerned. Particularly, for Case 1, the data-driven PI outputs parameter values of the approximate value function as follows

$$\bar{\Theta}^V = [253.6421, 6.0806, 61.2778, 0, 0, 0, 0, 0]$$
$$\bar{\Theta}^{Vg} = [61.5654, 12.4323, 0, 0, 0, 0],$$

while the standard PI gives $\Theta_{PI} = [198.4449, 6.0806, 49.1166, 0, 0, 0]$. By applying $z_1 = x_1$ and $z_2 = x_2 - x_1$, one can tell that both standard PI and data-driven PI yield almost the analogous approximate value functions. This fact is also confirmed by Fig. 4, where the error between two approximate value functions obtained by standard PI and data-driven PI is at the order of 10^{-7}. Data-driven PI ends up with the following output feedback control policy

$$u^*(z) = -0.5(61.5654z_1 + 12.4323z_2).$$

Fig. 2 indicates how well the true value function is approximated by the standard PI and data-driven PI. According to Fig. 2, choices of basis Φ^V for both cases fully capture the true value function, a quadratic function of x.

Figs. 5-8 correspond to Case 2. PI and data-driven PI converge to approximate value functions with the following parameters

$$\begin{aligned}\bar{\Theta}^V &= [260.4370, 5.6808, 60.3622, 6.4532,\\ &\quad -1.4969, -1.8005, -1.4759, -0.2357]\\ \bar{\Theta}^{Vg} &= [60.0728, 11.2592, -1.3712,\\ &\quad -3.6079, -4.3631, -0.9119]\\ \Theta_{PI} &= [235.3647, 7.2802, 42.6616,\\ &\quad 4.0927, -0.3399, 0.3626].\end{aligned}$$

Data-driven PI results in the following output feedback optimal control policy

$$\begin{aligned}u^*(z) = -0.5(&60.0728z_1 + 11.2592z_2 - 1.3712z_1^3\\ &- 3.6079z_1^2z_2 - 4.3631z_1z_2^2 - 0.9119z_2^3).\end{aligned}$$

Figs. 6 and 8 expose a surprising but interesting phenomenon: the approximate value function result from data-driven PI gives rise to much less approximation error that the PI case. It is partially ascribed to the fact that $\Phi^V(z)$ is two dimension higher than $\Phi^V(x)$.

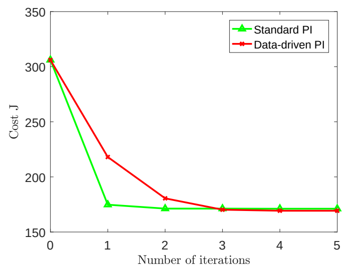

Fig. 5: Case 2: Values of cost function (2) over iterations

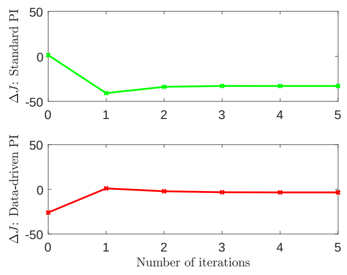

Fig. 6: Case 2: Errors between the cost function (2) and approximate value functions over iterations

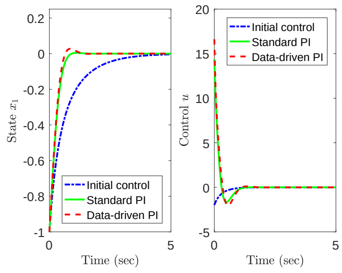

Fig. 7: Case 2: State and control trajectories

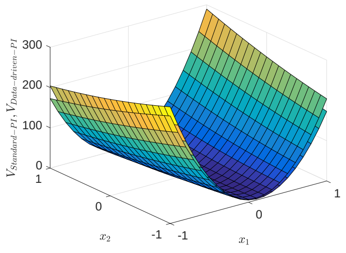

Fig. 8: Case 2: Approximate value functions

5 Conclusion and Future Work

This paper conducted data-driven output feedback optimal control design for a class of nonlinear systems, by the exploitation of approximate/adaptive dynamic programming techniques. It established two sufficient conditions to facilitate data-driven policy iteration algorithm (PI): all admissible output feedback optimal control policies are parameterized over system output and its time derivatives; system state can be represented as functions of system output and its time derivatives. For the case when system state, value function and admissible output feedback control policy can be linearly parameterized over a finite functional space defined over system output and its time derivatives, the data-driven PI is reduced to solve a system of linear equations. Two implementations of data-driven PIs are developed to fulfill the synthesis of data-driven output feedback optimal control. Future work includes optimality/convergence/robustness analysis and relaxing limitations of this work, e.g. , more general parameterizations of all admissible output feedback optimal control policies.

6 Acknowledgement

Author would like to thank Profs. Zhong-Ping Jiang and Weinan Gao for fruitful discussions.

References

[1] K. G. Shin and N. D. Mckay, “A dynamic programming approach to trajectory planning of robotic manipulators,” *IEEE Trans. Automat. Control*, vol. AC-31, no. 6, pp. 491–500, Jun. 1986.

[2] S. Singh and M. C. Leu, “Optimal trajectory generation for robotic manipulators using dynamics programming,” *Trans. ASME, J. Dyn. Sys. Meas. Control*, vol. 109, pp. 88–96, Jun. 1987.

[3] K. G. Shin and N. D. Mckay, “Minimum-time control of robotic manipulators with geometric path constraints,” *IEEE Trans. Automat. Control*, vol. AC-30, no. 6, pp. 531–541, Jun. 1985.

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

[5] G. Elnagar, M. A. Kazemi, and M. Razzaghi, “The pseudospectral legendre method for discretizing optimal control problems,” *IEEE Trans. Automat. Control*, vol. 40, no. 10, pp. 1793–1796, Oct. 1995.

[6] D. Verscheure, B. Demeulenaere, J. Swevers, J. D. Schutter, and M. Diehl, “Time-optimal path tracking for robots: A convex optimization approach,” *IEEE Trans. Automat. Control*, vol. 54, no. 10, pp. 2318–2327, Oct. 2009.

[7] D. P. Bertsekas, *Dynamic Programming and Optimal Control*. Athena Scientific Belmont, 1995.

[8] F. L. Lewis and D. Vrabie, “Reinforcement learning and adaptive dynamic programming for feedback control,” *IEEE Circuits Syst. Mag.*, vol. 9, no. 3, pp. 40–58, Aug. 2009.

[9] D. Kleinman, “On an iterative technique for Riccati equation computations,” *IEEE Trans. Automat. Control*, vol. AC-13, no. 1, pp. 114–115, 1968.

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

[11] W. Gao, Y. Jiang, Z.-P. Jiang, and T. Chai, “Output feedback adaptive optimal control of interconnected systems based on robust adaptive dynamic programming,” *Automatica*, vol. 72, pp. 37–45, Oct. 2016.

[12] W. Gao, Z.-P. Jiang, F. F. Lewis, and Y. Wang, “Leader-to-formation stability of multi-agent systems: An adaptive optimal control approach,” *IEEE Trans. Automat. Control*, vol. PP, no. 99, 2018.

[13] T. Cheng, F. L. Lewis, and M. Abu-Khalaf, “Fixed-final-time-constrained optimal control of nonlinear systems using neural network HJB approach,” *IEEE Trans. Neural Netw.*, vol. 18, no. 6, pp. 1725–1736, 2007.

[14] Z. Chen and S. Jagannathan, “Generalized Hamilton-Jacobi-Bellman formulation-based neural network control of affine nonlinear discrete time systems,” *IEEE Trans. Neural Netw.*, vol. 19, no. 1, pp. 90–106, 2008.

[15] Y. Jiang, Y. Wang, S. A. Bortoff, and Z.-P. Jiang, “Optimal co-design of nonlinear control systems based on a modified policy iteration method,” *IEEE Trans. on Neural Netw. and Learn. Syst.*, vol. 26, no. 2, pp. 409–414, Feb. 2015.

[16] Y. Wang, J. Wu, and C. Long, “Policy iteration-based optimal control design for nonlinear descriptor systems,” in *Proc. 2016 ACC*, Jul. 2016, pp. 5740–5745.

[17] R. W. Beard, G. N. Saridis, and J. T. Wen, “Galerkin approximations of the generalized Hamilton-Jacobi-Bellman equation,” *Automatica*, vol. 33, no. 12, pp. 2159–2177, 1997.

[18] H. K. Khalil, *Nonlinear Systems*, 3rd ed. Englewood Cliffs, NJ: Prentice-Hall, 2002.

[19] A. Levant, “Robust exact differentiation via sliding mode technique,” *Automatica*, vol. 34, no. 3, pp. 379–384, May 1998.

[20] E. Cruz-Zavala, J. A. Moreno, and L. M. Fridman, “Uniform robust exact differentiator,” *IEEE Trans. Automat. Control*, vol. 56, no. 11, pp. 2727–2733, Nov. 2011.

[21] M. Krstić, I. Kanellakopoulos, and P. Kokotović, *Nonlinear and Adaptive Control Design*. New York, NY: John Wiley and Sons, 1995.

[22] Z.-P. Jiang and Y. Jiang, “Robust adaptive dynamic programming for linear and nonlinear systems: An overview,” *EUR J. of Control*, vol. 19, no. 5, pp. 417–425, 2013.