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

Near-optimal control of motor drives via approximate dynamic programming

Wang, Y.; Chakrabarty, A.; Zhou, M.; Zhang, J.

TR2019-116 October 18, 2019

Abstract

Data-driven methods for learning near-optimal control policies through approximate dynamic programming (ADP) have garnered widespread attention. In this paper, we investigate how data-driven control methods can be leveraged to imbue near-optimal performance in a core component in modern factory systems: the electric motor drive. We apply policy iteration-based ADP on an induction motor model in order to construct a state feedback control policy for a given cost functional. Approximate error convergence properties of policy iteration methods imply that the learned control policy is near-optimal. We demonstrate that carefully selecting a cost functional and initial control policy yields a near-optimal control policy that outperforms both a baseline nonlinear control policy based on backstepping, as well as the initial control policy.

IEEE International Conference on Systems, Man, and Cybernetics

This work may not be copied or reproduced in whole or in part for any commercial purpose. Permission to copy in whole or in part without payment of fee is granted for nonprofit educational and research purposes provided that all such whole or partial copies include the following: a notice that such copying is by permission of Mitsubishi Electric Research Laboratories, Inc.; an acknowledgment of the authors and individual contributions to the work; and all applicable portions of the copyright notice. Copying, reproduction, or republishing for any other purpose shall require a license with payment of fee to Mitsubishi Electric Research Laboratories, Inc. All rights reserved.

Copyright © Mitsubishi Electric Research Laboratories, Inc., 2019
201 Broadway, Cambridge, Massachusetts 02139

Near-optimal control of motor drives via approximate dynamic programming

Yebin Wang[1], *Senior Member, IEEE*, Ankush Chakrabarty[1], *Member, IEEE*, Meng-Chu Zhou[2], *Fellow, IEEE*, Jinyun Zhang[1], *Fellow, IEEE*

***Abstract*—Data-driven methods for learning near-optimal control policies through approximate dynamic programming (ADP) have garnered widespread attention. In this paper, we investigate how data-driven control methods can be leveraged to imbue near-optimal performance in a core component in modern factory systems: the electric motor drive. We apply policy iteration-based ADP on an induction motor model in order to construct a state feedback control policy for a given cost functional. Approximate error convergence properties of policy iteration methods imply that the learned control policy is near-optimal. We demonstrate that carefully selecting a cost functional and initial control policy yields a near-optimal control policy that outperforms both a baseline nonlinear control policy based on backstepping, as well as the initial control policy.**

I. Introduction

Induction motors (IMs) have been limited to low and mid-end industrial applications due to their inferior performance compared to permanent magnetic motors. Therefore, improving the operational efficiency of IMs is expected to greatly expand their market share. Largely, existing control designs such as sliding mode or nonlinear control [1], [2] focus on guaranteeing system stability without ensuring high-performance and adaptation mechanisms or 'machine intelligence'. Until recently, most researchers focused on optimizing transient performance of IMs. For example, [5] utilized model predictive control for torque regulation, where both the active voltage vector and the duty cycle were optimized in a receding horizon manner. The authors in [6] proposed to replace current-loop controllers in conventional vector control [7] with two neural approximations of optimal control policies. Also, [8] devised optimal flux references to minimize the net energy loss of the motor during operation.

This work investigates the state feedback optimal control design for IMs operating at a constant operation point, for example: constant speed, flux, and load torque. This problem serves as a good starting point and can be extended to more involved scenarios such as output feedback and torque regulation, with or without full knowledge of the IM model. Since the IM dynamics are nonlinear, it is extremely challenging to synthesize a state-feedback optimal control policy. This is owing to the fact that constructing optimal controllers for nonlinear systems involves solving the Hamilton-Jacobi-Bellman (HJB) equations, given by

$$\frac{\partial V^*(x,t)}{\partial t} = \min_{u\in U}\left\{L(x,u) + \frac{\partial V^*(x,t)}{\partial x} f(x,u)\right\}, \quad (1)$$

where $x \in \mathbb{R}^n$ denotes the system state, $u \in U \subset \mathbb{R}^m$ denotes the control action in the admissible control set U, $V^*(x)$ denotes a value function, $L(x,u)$ is a stage cost, and $f(x,u)$ is the vector field appearing in the system dynamics; that is, $\dot{x} = f(x,u)$.

Motivated by recent progress on data-driven (near-)optimal control theory, we investigate the potential of implementing approximate dynamic programming (ADP) [9], [10] to ensure high-performance in induction motor drives. Two common algorithms used in ADP are policy iteration (PI) and value iteration. Both policy and value iteration leverage operational data obtained from different scenarios to construct near-optimal control policies via function approximators. Policy and value iteration have been particularly lauded for linear time-invariant (LTI) systems, for example, as demonstrated in [11]–[13], [16]. For nonlinear systems, the application of ADP has been largely limited to the state feedback case [10], [14], [17]–[19], with a recent exception being [20].

To the best of our knowledge, this work constitutes the first study on optimal feedback control of IMs via ADP. Although restricting ourselves to a seemingly straightforward case: assuming state feedback and exact knowledge of the IM model, we expose challenging hurdles that need to be surmounted in order to incorporate ADP into practical applications. One of our major findings is that the choice of the initial stabilizing control policy greatly affects the convergence of policy iteration algorithm; and there is a necessity to estimate the region of attraction associated with a control policy. Indeed, loss of guaranteed convergence may occur unless each consequent control policy has a domain of attraction that is a strict subset of the previous one. Although this study is at a preliminary stage, it identifies major challenges before ADP can contest standard control algorithms such as model predictive control and PID in industrial settings.

The remainder of this paper is organized as follows. In section II, we formulate a speed regulation problem and present preliminaries on data-driven optimal control. Section III describes our policy iteration-based state feedback optimal control synthesis based on a well-known IM state-space model. The results of our control policy are reported

[1]Y. Wang, A. Chakrabarty and J. Zhang are with Mitsubishi Electric Research Laboratories, Cambridge, MA 02139, USA. Email: {yebinwang,chakrabarty,jzhang}@merl.com.

[2]M. C. Zhou is with Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102, USA. Email: zhou@njit.edu.

in Section IV. Concluding remarks and future work are consigned to Section V.

II. Preliminaries

A. Problem statement and design objective

A typical IM model consists of five states: two stator currents (or fluxes), two rotor fluxes, and a rotor speed. The representation of the state dynamics depends on the reference frame we choose. Consider a reference frame rotating at an angular velocity ω_1. Consequently, the IM state-space model is given by

$$\dot{i}_{ds} = -\gamma i_{ds} + \omega_1 i_{qs} + \beta(\alpha\phi_{dr} + \omega\phi_{qr}) + \frac{u_{ds}}{\sigma} \tag{2a}$$

$$\dot{i}_{qs} = -\omega_1 i_{ds} - \gamma i_{qs} + \beta(\alpha\phi_{qr} - \omega\phi_{dr}) + \frac{u_{qs}}{\sigma} \tag{2b}$$

$$\dot{\phi}_{dr} = -\alpha\phi_{dr} + (\omega_1 - \omega)\phi_{qr} + \alpha L_m i_{ds} \tag{2c}$$

$$\dot{\phi}_{qr} = -\alpha\phi_{qr} - (\omega_1 - \omega)\phi_{dr} + \alpha L_m i_{qs} \tag{2d}$$

$$\dot{\omega} = \frac{\mu}{J}(\phi_{dr} i_{qs} - \phi_{qr} i_{ds}) - \frac{T_l}{J}, \tag{2e}$$

where notation is reported in Table I.

TABLE I
Description of states and parameters of the IM model

Notation	Description
i_{ds}, i_{qs}	stator currents in d- and q-axis
ϕ_{dr}, ϕ_{qr}	rotor fluxes in d- and q-axis
ω, ω^*	rotor angular speed and its reference
u_{ds}, u_{qs}	stator voltages in d- and q-axis
ω_1	angular speed of a rotating reference frame
ϕ^*	rotor flux amplitude reference
i^*_{ds}, i^*_{qs}	references of stator currents in d- and q-axis
T_l, T_l^*	load torque and its reference
J	moment of inertia
L_s, L_m, L_r	stator, mutual, and rotor inductances
R_s, R_r	stator and rotor resistances
σ	$\frac{L_s L_r - L_m^2}{L_r}$
α	R_r/L_r
β	$L_m/(\sigma L_r)$
γ	$R_s/\sigma + \alpha\beta L_m$
μ	pL_m/L_r
p	number of pole pairs

For a reference frame rotating at an angular velocity $\omega_1 = \omega + \alpha L_m i_{qs}/\phi_{dr}$, we know that the q-axis rotor flux vanishes. That is, $\phi_{qr} = 0$. The horizontal axis of such a rotating reference frame is always aligned with the rotor flux vector, which is why it is commonly referred to as the 'field-oriented' frame. This frame is of particular interest for controller design because the motor operates at constant speed, flux, and torque, implying the existence of an equilibrium state. This enables us to pose the controller design as a classical state-feedback stabilization problem.

The motor dynamics in the field-oriented frame are represented as follows:

$$\dot{i}_{ds} = -\gamma i_{ds} + \left(\omega + \frac{\alpha L_m i_{qs}}{\phi_{dr}}\right) i_{qs} + \beta\alpha\phi_{dr} + \frac{u_{ds}}{\sigma} \tag{3a}$$

$$\dot{i}_{qs} = -\left(\omega + \frac{\alpha L_m i_{qs}}{\phi_{dr}}\right) i_{ds} - \gamma i_{qs} - \beta\omega\phi_{dr} + \frac{u_{qs}}{\sigma} \tag{3b}$$

$$\dot{\phi}_{dr} = -\alpha\phi_{dr} + \alpha L_m i_{ds} \tag{3c}$$

$$\dot{\phi}_{qr} = 0 \tag{3d}$$

$$\dot{\omega} = \frac{\mu}{J}\phi_{dr} i_{qs} - \frac{T_l}{J}, \tag{3e}$$

For simplicity, we refer to the states in both models (2) and (3) using the same notation (albeit, with abuse of notation).

Suppose that the equilibrium induced by the rotating reference frame results in the IM operating at a constant speed ω^*, a constant rotor flux ϕ^*, and a constant load torque T_l^*. Accordingly, the equilibrium state corresponding to these constant operational modes can be written as:

$$\begin{aligned} x^e &= \begin{bmatrix} i^e_{ds} & i^e_{qs} & \phi^e_{dr} & \phi^e_{qs} & \omega^e \end{bmatrix}^\top \\ &= \begin{bmatrix} \frac{\phi^*}{L_m} & \frac{T_l^*}{\mu\phi^*} & \phi^* & 0 & \omega^* \end{bmatrix}^\top. \end{aligned} \tag{4}$$

The corresponding equilibrium control action associated with x^e is given by

$$u^e = \begin{bmatrix} u^e_{ds} \\ u^e_{qs} \end{bmatrix} = \sigma \begin{bmatrix} \gamma\, i^e_{ds} - \left(\omega^* + \frac{\alpha L_m i^e_{qs}}{\phi^*}\right) i^e_{qs} - \beta\alpha\phi^* \\ \left(\omega^* + \frac{\alpha L_m i^e_{qs}}{\phi^*}\right) i^e_{ds} + \gamma\, i^e_{qs} + \beta\omega^*\phi^* \end{bmatrix}. \tag{5}$$

Our **objective** is to design a state-feedback control policy to ensure optimal IM performance during the transient and subsequent regulation of the equilibrium x^e, based on a given cost function. Concretely, for a control policy u_{ds} and u_{qs}, we employ the following cost functional to evaluate the transient performance:

$$\mathcal{C}(u) = \int_0^\infty \left(y^\top Q y + u^\top R u\right)\, \mathrm{dt}, \tag{6}$$

where

$$y = \begin{bmatrix} i_{ds} - i^e_{ds} \\ i_{qs} - i^e_{qs} \\ \omega - \omega^e \end{bmatrix}, \qquad u = \begin{bmatrix} u_{ds} - u^e_{ds} \\ u_{qs} - u^e_{qs} \end{bmatrix},$$

and $Q = Q^\top$ and $R = R^\top$ are positive definite matrices that weight the output tracking and control costs, respectively.

B. Policy iteration for optimal state-feedback stabilization

Designing optimal state feedback control policies for nonlinear systems is extremely challenging. In this work, we propose the use of ADP [9] to tackle this problem. The main steps involved in ADP is described below to make this paper self-contained.

In the most general setting, we consider a class of nonlinear systems modeled by

$$\dot{x} = f(x, u), \quad x(0) = x_0, \tag{7}$$

where $x \in \Omega_x \subset \mathbb{R}^n$ is the vector of system state, Ω_x a compact set containing the origin in its interior, $u \in \mathbb{R}^m$ is a vector of control inputs, and $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is a locally Lipschitz vector field in x and u to ensure uniqueness and existence of solutions to the differential equation, and $f(0, 0) = 0$. ADP is employed to compute a state-feedback

control policy $u^\star(x)$ that minimizes a general cost functional

$$\mathcal{C}_L(u) = \int_0^\infty L(x,u)\,\mathrm{dt}, \tag{8}$$

where $L(x,u)$ is positive definite for all x and u and $L(0,0)=0$.

One embodiment of ADP is policy iteration [9], [21]–[23], which involves iteratively improving an initial admissible control policy using operational data until convergence to the optimal policy. A prerequisite of policy iteration is the availability of an initial admissible control policy $u_0(x)$. By admissible, we mean that the control policy $u_0(x)$ stabilizes the system (7) and incurs a finite cost $\mathcal{C}_L < \infty$. Typical implementations of policy iteration involve policy evaluation and policy improvement, which are discussed next.

1) Policy evaluation: Let $i \in \mathbb{N}\cup\{0\}$. Policy evaluation involves solving for a positive definite Lyapunov (or reward) function $V_i(x)$ satisfying

$$\nabla V_i(x) f\big(x,u_i(x)\big) + L\big(x,u_i(x)\big) = 0, \tag{9}$$

for every $x \in \Omega_x$, where $\nabla V_i(x) = \partial V_i(x)/\partial x$ is a row vector.

2) Policy improvement: Policy improvement leverages the updated reward function V_i in order to obtain an improved control policy according to

$$u_{i+1}(x) = \arg\min_{u\in U}\{L(x,u) + \nabla V_i f(x,u)\}, \tag{10}$$

for every $x \in \Omega_x$, where $U \subset \mathbb{R}^m$ is the set of all admissible control policies.

The ith control policy $u_i(x)$ and reward function $V_i(x)$ depend only on the state to be stabilized. Herein, we omit the argument x to simplify notations. The policy evaluation step solves (9) to compute a reward function V_i or its gradient ∇V_i corresponding to the current control policy u_i. As a system of first-order linear partial differential equations, a closed-form solution of (9) is difficult to obtain, and typically, an approximate solution is more practical to compute and yields satisfactory performance. By properly parameterizing the functional forms of u_i and V_i (for example, via basis function representations or neural approximators), one can reduce (9) to a finite number of algebraic equations, enabling the computation of an approximate solution. Specifically, with linear parameterizations of V_i, for a sampled state $x \in \Omega_x$, the equation (9) is reduced to a linear algebraic equation. The two steps (9)-(10) are repeated until convergence.

Remark 1. If control policy u_i globally stabilizes a closed-loop system, an exact solution V_i of (9) is always positive definite. This is because a) it satisfies $\dot{V}_i = -L(x,u_i)$ and thus $V_i(x(t)) = V_i(\infty) + \int_t^\infty L(x,u_i)\mathrm{d}t$; b) given u_i stabilizing the system, $x(\infty)=0$ and $V_i(\infty)=0$. One can further verify that V_i is also a Lyapunov function of the closed-loop system with control policy u_{i+1}. Eventually, we conclude that starting with a stabilizing control policy u_0, policy iteration will produce a sequence of stabilizing control policies $\{u_{i+1}\}, i \in \mathbb{N}\cup\{0\}$.

Remark 2. The values of the cost functional (8) associated with the sequence of control policies $\{u_i\}, i \in \mathbb{N}\cup\{0\}$ monotonically decrease. In other words, the cost of the closed-loop system with improved control policy u_{i+1} is no greater than that associated with control policy u_i.

Remark 3. The above discussion motivates why these types of controllers are sometimes referred to as 'near-optimal' controllers [3], [4] or 'approximate optimal' [14], [15], since optimality properties are a consequence of an infinite number of basis functions used in the parameterization and the class of functions being approximated.

We know that policy iteration algorithms exhibit the following convergence property [11].

Theorem 1. *Consider system* (7) *and cost functional* (8)*. Suppose that $u_0(x) \in U$ is an admissible initial control policy, and a positive definite solution V_i of* (9) *exists for every $i = 0, 1, \cdots$. Then,*

1) $u_{i+1} \in U$ *for* $i \geq 0$*;*
2) $\mathcal{C}_L(u_{i+1}) \leq \mathcal{C}_L(u_i)$ *for* $i \geq 0$*;*
3) $\lim_{i\to\infty} \mathcal{C}_L(u_i) = \mathcal{C}_L^*$ *with* $\mathcal{C}_L^* \in [0,\infty)$.

III. Optimal State Feedback Control Synthesis

This section discusses how to obtain initial admissible control policies, effective parameterizations of reward functions and control policies, along with implementation details specific to the induction motor problem.

A. Constructing initial admissible control policies

1) Local linear quadratic regulator policy: Policy iteration is used to synthesize a feedback control policy that stabilizes the origin $x=0$ of nonlinear systems of the form (7). The induction motor speed control problem is to regulate its outputs to certain values $(\omega^*, \phi^*, T_l^*)$, which, as discussed in (4) and (5), is tantamount to stabilizing the system (3) the equilibrium pair (x^e, u^e). To this end, we formulate the tracking error dynamics

$$\dot{e} = f(e, x^e) + g(u + u^e), \tag{11}$$

where $e := x - x^e = \begin{bmatrix} e_{id} & e_{iq} & e_{\phi_d} & e_{\phi_q} & e_\omega \end{bmatrix}^\top$, u^e is defined in (5), and $f(e,x^e) = [f_1,\ldots,f_5]^\top$ with

$$\begin{aligned}
f_1 &= -\gamma(e_{id} + i_{ds}^e) + \beta\alpha(e_{\phi_d} + \phi^*) + \frac{e_{iq} + i_{qs}^e}{e_{\phi_d} + \phi^*} \\
&\quad \times \big(L_m\alpha e_{iq} + L_m\alpha i_{qs}^e + e_{\phi_d}e_\omega + e_{\phi_d}\omega^* + e_\omega\phi^* + \omega^*\phi^*\big) \\
f_2 &= -\gamma(e_{iq} + i_{qs}^e) - \beta(e_\omega + \omega^*)(e_{\phi_d} + \phi^*) - \frac{e_{id} + i_{ds}^e}{e_{\phi_d} + \phi^*} \\
&\quad \times (L_m\alpha e_{iq} + L_m\alpha i_{qs}^e + e_{\phi_d}e_\omega + e_{\phi_d}\omega^* + e_\omega\phi^* + \omega^*\phi^*) \\
f_3 &= L_m\alpha e_{id} + L_m\alpha i_{ds}^e - \alpha e_{\phi_d} - \alpha\phi^* \\
f_4 &= 0 \\
f_5 &= \frac{1}{J}(e_{iq}e_{\phi_d}\mu + e_{iq}\mu\phi^* + e_{\phi_d}i_{qs}^e\mu + i_{qs}^e\mu\phi^* - T_l^*),
\end{aligned}$$

and $g = (1/\sigma)\begin{bmatrix} I_{2\times 2} & 0_{2\times 3}\end{bmatrix}^\top$.

At the tracking equilibrium state $(e,u)=0$, the tracking error dynamics satisfy $f(0,x^e) + gu^e = 0$. The control objective is to stabilize the zero solution of the tracking

$$A_e = \begin{bmatrix} -\gamma & \frac{2L_m\alpha i^e_{qs}+\omega^*\phi^*}{\phi^*} & -\frac{\alpha(L_m(i^e_{qs})^2-\beta(\phi^*)^2)}{(\phi^*)^2} & i^e_{qs} \\ -\frac{L_m\alpha i^e_{qs}+\omega^*\phi^*}{\phi^*} & -\gamma-\frac{L_m\alpha i^e_{ds}}{\phi^*} & \frac{L_m\alpha i^e_{ds} i^e_{qs}-\beta\omega^*(\phi^*)^2}{(\phi^*)^2} & -\beta\phi^*-i^e_{ds} \\ L_m\alpha & 0 & -\alpha & 0 \\ 0 & \frac{\mu\phi^*}{J} & \frac{i^e_{qs}\mu}{J} & 0 \end{bmatrix}. \tag{13}$$

error dynamics, that is, $e = 0$. We observe that $f(e, x^e)$ is nonlinear in e, and $f(e, x^e)$ does not possess any structure that can easily be exploited for control synthesis. Both facts pose challenges to design a global state feedback control policy $u(e)$ to stabilize the tracking error dynamics (11).

To proceed, we linearize the tracking error dynamics at $e = 0$ and $u = 0$, and calculate the linear quadratic regulator (LQR) policy based on the linearized error dynamics. Linearization of (11) around $(e, u) = 0$ gives the following linear system

$$\dot{e} = A_e e + g\, u, \tag{12}$$

where A_e is described in (13). LQR design for (12) gives a linear control policy $u_0^{\mathrm{LQR}}(e) = Ke$ by solving the continuous-time algebraic Riccati equation.

2) Nonlinear state-feedback control policy: A common shortcoming of linear control, when used to stabilize nonlinear systems, is that it renders a limited region of attraction. Since policy iteration has to be carried out within the region of attraction, one has to construct the region of attraction for every control policy $u_i, i = 0, 1, \cdots$. This is not only very difficult to ensure, but also an undesirable property of a control policy, limiting its usefulness in practical systems. One way of mitigating this issue is to use a global control policy as described in [2] via backstepping which results in closed-loop tracking error dynamics that are globally exponentially stable. We proceed as follows.

The speed control policy is given by

$$i^*_{qs} = \frac{1}{\mu\phi_{dr}}\left(-k_\omega e_\omega + T^*_l\right), \tag{14}$$

where k_ω is constant, and $e_\omega = \omega - \omega^*$. Without loss of generality, we do not introduce flux control to regulate ϕ_{dr} to ϕ^*. Instead, we regulate i_{ds} to $i^*_{ds} = \phi^*/L_m$, which necessarily implies $\phi_{dr}(t) \to \phi^*$ as $t \to \infty$. With reference currents given as i^*_{qs}, i^*_{ds}, the current control policy is derived as follows

$$\begin{aligned} u_{ds} &= \sigma\left[-k_d e_{id} - \omega_1 i_{qs} - \alpha\beta\phi_{dr} + \gamma i^*_{ds} + \dot{i}^*_{ds}\right] \\ u_{qs} &= \sigma\left[-k_q e_{iq} + \omega_1 i_{ds} + \beta\omega\phi_{dr} + \gamma i^*_{qs} + \dot{i}^*_{qs}\right], \end{aligned} \tag{15}$$

where k_d and k_q are constant, and $e_{id} = i_{ds} - i^*_{ds}$ and $e_{iq} = i_{qs} - i^*_{qs}$ are the tracking errors of stator currents, respectively. The control policy (14)-(15) consists of feedforward and feedback terms. Since policy iteration needs an initial control policy in the form of $u(e)$, we need to reparameterize (14)-(15) as a function of the tracking errors. Redefining the tracking error as

$$\begin{aligned} e &= (e_{id}, e_{iq}, e_{\phi_d}, e_{\phi_q}, e_\omega) \\ &= (i_{ds} - i^*_{ds}, i_{qs} - i^*_{qs}, \phi_{dr} - \phi^*, 0, \omega - \omega^*), \end{aligned} \tag{16}$$

we derive its dynamics

$$\dot{e} = f_e(x, \dot{i}^*_{ds}, \dot{i}^*_{qs}) + g_e \begin{bmatrix} u_{ds} \\ u_{qs} \end{bmatrix}, \tag{17}$$

where $f_e = f(x) - [\dot{i}^*_{ds}, \dot{i}^*_{qs}, 0, 0, 0]^\top$ and $g_e = g$. To facilitate policy iteration, we need to represent f_e and u_{ds}, u_{qs} as functions of e and $i^*_{ds}, i^*_{qs}, \phi^*, \omega^*$. It is clear that $\dot{i}^*_{ds} = 0$. Reparametrizing ω_1 with e yields

$$\omega_1 = e_\omega + \omega^* + \alpha L_m \frac{e_{iq} + i^*_{qs}}{e_{\phi_d} + \phi^*}. \tag{18}$$

Substituting (18) into (15), we represent f_e in terms of $(i_{ds}, i_{qs}, \phi_{dr}, \omega, i^*_{ds}, i^*_{qs}, \phi^*, \omega^*, \dot{i}^*_{qs})$. Considering the following formula

$$\begin{aligned} &i_{qs} = e_{iq} + i^*_{qs}, \quad i^*_{qs} = \frac{-k_\omega e_\omega + T^*_l}{\mu(e_{\phi_d} + \phi^*)}, \\ &i_{ds} = e_{id} + i^*_{ds}, \quad \phi_{dr} = e_{\phi_d} + \phi^*, \quad \omega = e_\omega + \omega^*, \end{aligned}$$

we express f_e as a function of e and $(i^*_{ds}, i^*_{qs}, \phi^*, \omega^*, \dot{i}^*_{qs})$. Similarly, we reparameterize the nonlinear control policy

$$\begin{aligned} u_{ds}(e) &= \sigma\left(\gamma i^*_{ds} - k_d e_{id} - \alpha\beta(e_{\phi_d} + \phi^*) + \rho_d\right) \\ u_{qs}(e) &= \sigma\left(\gamma\frac{-k_\omega e_\omega + T^*_l}{\mu(e_{\phi_d} + \phi^*)} + \dot{i}^*_{qs} - k_q e_{iq}\right. \\ &\qquad \left. +\beta(e_\omega + \omega^*)(e_{\phi_d} + \phi^*) + \rho_q\right), \end{aligned} \tag{19}$$

where

$$\begin{aligned} \rho_d = {} & \frac{(e_{iq}e_{\phi_d}\mu + e_{iq}\mu\phi^* - k_\omega e_\omega + T^*_l)}{\mu^2(e_{\phi_d} + \phi^*)^3}\left(-L_m\alpha e_{iq}e_{\phi_d}\mu\right. \\ & -L_m\alpha e_{iq}\mu\phi^* + k_\omega L_m\alpha e_\omega - e^2_{\phi_d}e_\omega\mu - e^2_{\phi_d}\mu\omega^* \\ & -2e_{\phi_d}e_\omega\mu\phi^* - 2e_{\phi_d}\mu\omega^*\phi^* - e_\omega\mu(\phi^*)^2 - \mu\omega^*(\phi^*)^2 \\ & \left.-L_m T^*_l\alpha\right) \\ \rho_q = {} & \frac{-(e_{id} + i^*_{ds})}{\mu(e_{\phi_d} + \phi^*)^2}\left(-L_m\alpha e_{iq}e_{\phi_d}\mu - L_m\alpha e_{iq}\mu\phi^*\right. \\ & +k_\omega L_m\alpha e_\omega - e^2_{\phi_d}e_\omega\mu - e^2_{\phi_d}\mu\omega^* - 2e_{\phi_d}e_\omega\mu\phi^* \\ & \left.-2e_{\phi_d}\mu\omega^*\phi^* - e_\omega\mu(\phi^*)^2 - \mu\omega^*(\phi^*)^2 - L_m T^*_l\alpha\right). \end{aligned}$$

Remark 4. Even though i^*_{qs} is a function of e, we treat it as a known time-varying signal to simplify the aforementioned derivation of $u(e)$. Tracking error (16) is different from that used in the derivation of LQR. For the LQR case, $e = x - x^e$ and the tracking error dynamics do not involve $\dot{x}^e$.

Note that the control policy (19), which depends on the tracking error e and the reference $(\omega^*, \phi^*, T^*_l)$, does not vanish at $e = 0$. Therefore, the cost functional (6) with control policy (19) is not finite. Hence, (19) cannot be the initial control policy of policy iteration. To eliminate this issue, we remove the non-zero feedforward term from the

control policy. Rewriting (19), we get

$$u_{ds}(e) = u_{ds}^v + u_{ds}^c \tag{20a}$$
$$u_{qs}(e) = u_{ds}^v + u_{qs}^c, \tag{20b}$$

where u_{ds}^v and u_{qs}^v vanish as $e \to 0$, and u_{ds}^c and u_{qs}^c include terms independent from e. Therefore, we can use u_{ds}^v and u_{qs}^v as the initial control policy of policy iteration.

B. Selecting the reward function and learned control policies

Essentially, policy iteration involves solving the partial differential equation (9). A computationally efficient method of solving this involves parameterizing the i-th reward function V_i and i-th control policy u_i via basis function expansions. To this end, let $\{\varphi_j(x)\}_{j=1}^N$ with $\varphi_j(\cdot) : \mathbb{R}^n \to \mathbb{R}$ and $\{\psi_j(x)\}_{j=1}^q$ with $\psi_j(\cdot) : \mathbb{R}^n \to \mathbb{R}^m$ be two sets of linearly independent, continuously differentiable functions and vector fields, respectively. In addition, we assume that $\varphi_j(0) = 0$, $\forall\ 1 \le j \le N$ and $\psi_j(0) = 0$, $\forall\ 1 \le j \le q$.

Assumption 1. *Given $u_i(x) \in U$ and*

$$u_i(x) \in \operatorname{span}\{\psi_1(x), \cdots, \psi_q(x)\},$$

assume that

$$V_i(x) \in \operatorname{span}\{\varphi_1(x), \cdots, \varphi_N(x)\},$$
$$u_{i+1}(x) \in \operatorname{span}\{\psi_1(x), \cdots, \psi_q(x)\},$$

where $V_i(x)$ and $u_{i+1}(x)$ are obtained from (9) *and* (10)*.*

Assumption 1 ensures that one can find three sets of weights $\{w_{i,k}\}_{k=1}^N$, $\{c_{i,k}\}_{k=1}^q$, and $\{c_{i+1,k}\}_{k=1}^q$, such that the current control policy $u_i(x) = \sum_{j=1}^q c_{i,j}\psi_j(x)$, the current reward function $V_i(x) = \sum_{j=1}^N w_{i,j}\varphi_j(x)$, and the improved control policy $u_{i+1}(x) = \sum_{j=1}^q c_{i+1,j}\psi_j(x)$ can be adequately represented in the same functional space.

Remark 5. In general, Assumption 1 is difficult to verify, because it essentially requires to solve the exact solution of (9). On the other hand, even though Assumption 1 is not satisfied or not verifiable, the approximation of weights can still be numerically obtained based on methods, such as the off-line approximation using Galerkin's method [24]. In addition, for uncertain nonlinear systems, these weights can be trained by using approximate-dynamic-programming-based online learning methods [25], [26]. When approximation methods are used, the region of attraction Ω_x, or Ω_e for tracking error dynamics, is required to be a compact set to guarantee the boundedness of the approximation error [15].

Remark 6. Intuitively, one expects that a large number of basis functions φ_i and ψ_j will be required to offer satisfactory approximations of an initial control policy, reward function and subsequent control policies. The selection of approximators for reward function and control policy emerges as one of major challenges in the course of applying ADP to motor drives. This entails deep understanding of partial differential equations and approximation theory, and is beyond the scope of this paper. Alternatively, in order to use u_{ds}^v and u_{qs}^v in (20) as an initial control policy, one can obtain its approximation by training a neural network, which is consigned to the journal extension of this work.

In this work, we select polynomial basis functions. Concretely, for reward function approximation, we choose the basis

$$\{\varphi_j(e)\} := \{e_i^2, e_i e_j, e_i^4, e_i^2 e_j^2, 1 \le i, j \le 5\}.$$

For the policy improvement step, the policy basis functions are selected as

$$\{\psi_j(e)\} := \{e_i, e_i^3, e_i e_j^2, 1 \le i, j \le 5\}.$$

Note that since g is constant, the basis functions required to express any solution of the policy improvement step will be a linear combination of the derivatives of V_i. Therefore, selecting $\{\psi_j\}$ to be derivatives of $\{\varphi_j(e)\}$ ensures that the improved policy is the exact (not approximated) optimum induced by the approximate reward function V_i.

Even though the vanishing control policy $u_{ds}^v(e), u_{qs}^v(e)$ defined in (20) can be used as an initial control policy, its parameterization over polynomial basis functions $\psi(e)$ is difficult to obtain. For the ease of implementation, we take further simplification and extract the linear portion of nonlinear control policy (19) as an initial control policy, i.e.,

$$\begin{aligned} u_{ds}^l &= -k_d e_{id} - K_{d2} e_{iq} + K_{d3} e_{\phi_d} + K_{d5} e_\omega \\ u_{qs}^l &= K_{q1} e_{id} - K_{q2} e_{iq} + K_{q3} e_{\phi_d} + K_{q5} e_\omega, \end{aligned} \tag{21}$$

where

$$\begin{aligned} K_{d2} &= \frac{\mu\omega^*(\phi^*)^2 + 2L_m T_l^* \alpha}{\mu(\phi^*)^2}, \\ K_{d3} &= \frac{-\alpha\beta\mu^2(\phi^*)^4 + T_l^*\mu\omega^*(\phi^*)^2 + 3L_m(T_l^*)^2\alpha}{\mu^2(\phi^*)^4}, \\ K_{d5} &= \frac{k_\omega\mu\omega^*(\phi^*)^2 + 2k_\omega L_m T_l^*\alpha - T_l^*\mu(\phi^*)^2}{\mu^2(\phi^*)^3}, \\ K_{q1} &= \frac{\mu\omega^*(\phi^*)^2 + L_m T_l^*\alpha}{\mu(\phi^*)^2},\ K_{q2} = \frac{k_q\phi^* - L_m\alpha i_{ds}^*}{\phi^*}, \\ K_{q3} &= -\frac{-\beta\mu\omega^*(\phi^*)^3 + 2L_m T_l^*\alpha i_{ds}^* + T_l^*\gamma\phi^*}{\mu(\phi^*)^3}, \\ K_{q5} &= \beta\phi^* - \frac{\gamma k_\omega}{\mu\phi^*} - \frac{(k_\omega L_m\alpha - \mu(\phi^*)^2) i_{ds}^*}{\mu(\phi^*)^2}. \end{aligned}$$

The control policy (19) can be locally approximated by a summation of the linearized control policy (21) and

$$\begin{aligned} \frac{u_{ds}^c}{\sigma} &= \gamma i_{ds}^* - \alpha\beta\phi^* - \frac{T_l^*}{\mu^2(\phi^*)^3}\left(\mu\omega^*(\phi^*)^2 + L_m\alpha T_l^*\right), \\ \frac{u_{qs}^c}{\sigma} &= \gamma\frac{T_l^*}{\mu\phi^*} + \dot{i}_{qs}^* + \beta\omega^*\phi^* + \frac{i_{ds}^*}{\mu(\phi^*)^2}\left(\mu\omega^*(\phi^*)^2 + L_m\alpha T_l^*\right), \end{aligned}$$

which can be rewritten as

$$\begin{aligned} u_{ds}^c &= \sigma\left(\gamma i_{ds}^* - \alpha\beta\phi^* - \omega_1^* i_{qs}^*\right) \\ u_{qs}^c &= \sigma\left(\gamma i_{qs}^* + \dot{i}_{qs}^* + \beta\omega^*\phi^* + \omega_1^* i_{ds}^*\right). \end{aligned}$$

Note that $u^c = \begin{bmatrix} u_{ds}^c & u_{qs}^c \end{bmatrix}^\top$ is different from u^e in (5).

C. Performing policy iteration with polynomial bases

We will solve (9)–(10) on the basis of the aforementioned parameterizations and tracking error dynamics. When using

the linearized control policy (21), we use the tracking error dynamics (17) in both policy iteration steps. At the ith iteration, the closed-loop system dynamics are given as $\dot{e} = f_e + g_e(u^c + u_i^l(e))$, where u^c is independent from e, and $u_i^l = \sum_{j=1}^{q} c_{i,j}\psi_j(e)$ is purely feedback based on tracking error e. During ith iteration, we take samples e_i in a neighborhood of the origin $e = 0$; for each e_i, (9) yields a linear algebraic equation, the right hand side of which is a known value since it involves computing the known cost function (6) using the tracking error and control input data obtained, and the left hand side of which is

$$\sum_{j=1}^{N} w_{i,j} \frac{\partial \varphi_i(e)}{\partial e}(f_e + g_e(u^c + u_i^l(e)). \tag{22}$$

The linear algebraic equation contains unknown variables $w_{i,j}$, which can be solved by using a sufficiently large number of samples.

Subsequently, we can use this expansion of the reward function V_i to obtain the updated control policy $u_{i+1}^l = -\frac{1}{2}R^{-1}(\nabla V_i(e)\, g_e)^\top$.

IV. Simulation

We verify the effectiveness of the proposed policy iteration methodology via numerical simulations of an induction motor. We compare the performance of three closed-loop systems using three distinct control policies: baseline nonlinear control policy (14)–(15), a combination of u^c and linearized initial control policy (21), and a combination of u^c and the proposed near-optimal control policy generated by policy iteration.

TABLE II
Parameter Values

Notation	Values	Notation	Values
R_s	0.439 Ω	ω^*	5 rad/sec
R_r	0.410 Ω	ϕ^*	0.5 Web
L_m	0.0601 H	T_l^*	1 Nm
L_s	0.0615 H	k_d	200π
L_r	0.0619 H	k_q	200π
J	0.0163 Kgm^2	k_ω	40π

Motor model parameter values, references, and controller gains are provided in Table II. We select an initial condition $x(0) = [0, 0, 0.01, 0, 0]^\top$ for the induction motor and ensure that ϕ_{dr} is non-zero to avoid singularity; note that zero is not our desired equilibrium state. In this work, we do not consider state and input constraints explicitly. Matrices Q and R in the cost function play an important role in the speed tracking performance of the resultant optimal control policy. Here, we select $Q = \text{diag}(1, 0.1, 1000)$ and $R = \text{diag}(0.001, 0.001)$ as the cost function matrices. An initial control policy $u_0(e)$ of the form (21) is given by

$$u_{ds}^l(e) = -628e_{id} - 7e_{iq} - 2029e_{\phi_d} + 814e_\omega,$$
$$u_{qs}^l(e) = 6e_{iid} - 622e_{iq} + 1014e_{\phi_d} - 33419e_\omega.$$

A comparison of the closed-loop dynamics of the three control policies is presented in Figs. 1–3. Fig. 1 illustrates the resulting speed trajectories with the baseline nonlinear control policy, the linear control policy u_0, and the ADP control policy u^*. We observe that the nonlinear control is quite aggressive and thus, results in a quick response but produces a large overshoot in speed tracking. The linearized initial control is less aggressive, and the resulting speed

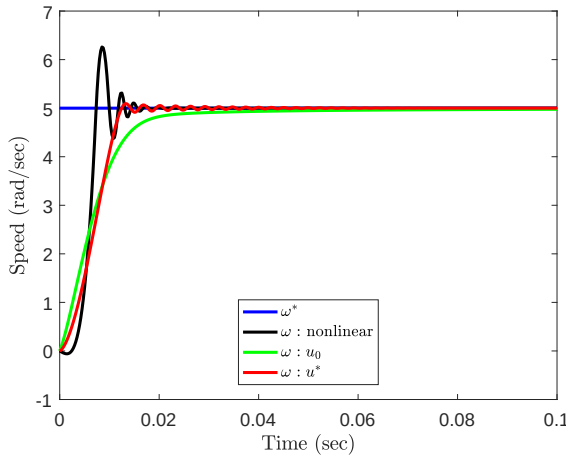

Fig. 1. Comparison of IM closed-loop speed trajectories.

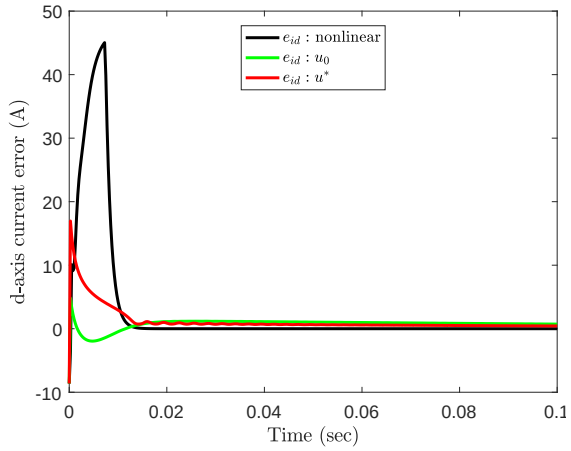

Fig. 2. Comparison of d-axis tracking error trajectories.

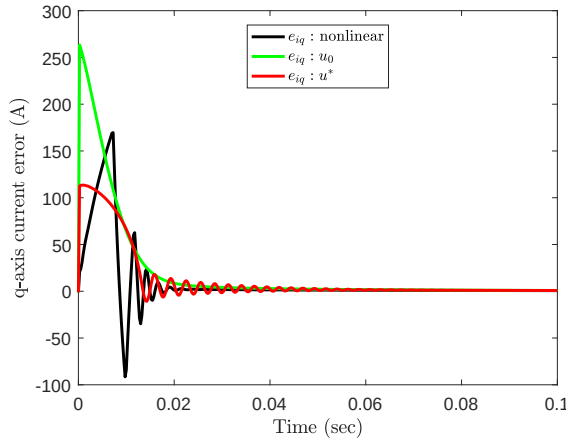

Fig. 3. Comparison of q-axis tracking error trajectories.

trajectory response is smoother, but the settling time is quite large, that is, the tracking error does not converge to a small quantity very quickly. Our near-optimal control policy trades off speed tracking bandwidth and oscillatory behaviour, resulting in a quick but smooth transient and a settling time that is better than its competitors. Fig. 2 shows the d-axis current tracking error trajectories. As expected, the nonlinear controller produces a large peak in the beginning of the transient, but quickly converges to 0. Both linear control and optimal control have similar transient behaviors in e_{id}: lower peak current, which implies safer operational conditions and slower convergence to the equilibrium current. We have observed that the speed of the transient decay of the near-optimal control policy can be improved by reducing the weight on e_{id} through the matrix Q in the cost function (6). Fig. 3 depicts the corresponding q-axis current tracking error trajectories. Note that the near-optimal control policy penalizes large peaks in the q-axis current, resulting in lower q-axis current transient oscillations. This is supported by Fig. 4, which illustrates that the cost function is non-increasing over iterations, as guaranteed by Theorem 1.

Although the trend of the cost is decreasing due to careful selection of cost functions and approximators, this monotonic property does not always hold in practice. Combining extensive simulation with theoretical analysis, we identified the following reasons for this anomaly: (i) The limited region of attraction for a given local control policy. Because the initial control policy locally stabilizes the tracking error dynamics at the origin, one has to sample the state space within its region of attraction to apply policy iteration. In simulation, we skip the step of estimating the region of attraction for simplicity, and thus likely some samples in policy evaluation may lie outside of the region of attraction, resulting in non-stabilizing control actions that lead to an increase in the cost. (ii) The choice of basis functions. On one hand, improper basis functions may violate Assumption 1, which could lead to the failure of policy iteration if implemented naively. An intuitive interpretation is that with improperly selected $\{\varphi_j\}_{j=1}^N$, policy iteration may produce skewed evaluation of a control policy even if the control policy globally stabilizes the tracking error dynamics. On the other hand, bad choices of basis functions may render the algebraic equations, established and solved in policy evaluation, ill-conditioned.

Fig. 5 illustrates the flexibility of Q to tune the closed-loop system performance. We refer to the element Q_ω in the cost matrix Q to represent the weight on the speed tracking error. As expected, larger Q_ω leads to an optimal control policy that gives a faster speed response. Figs. 6–7 provide performance of a sequence of control policies produced by policy iteration with a fixed $Q_\omega = 10^3$. We infer that, over multiple iterations, policy iteration comes up with new policies that improve speed responses consecutively. This phenomenon can be explained by examining Fig. 7, where the peak value of i_{qs} is successively reduced over iterations. This is because the peak value in e_{iq} is such an important contributor to the value of the cost function that, even though its weight is 0.1, reducing the peak brings significant decrements of the cost function.

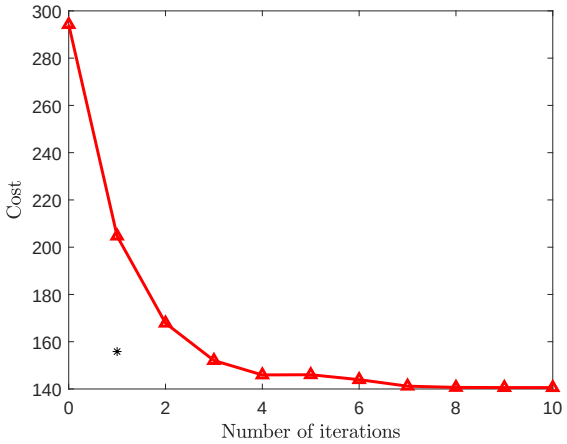

Fig. 4. Stage cost computed iteration-wise.

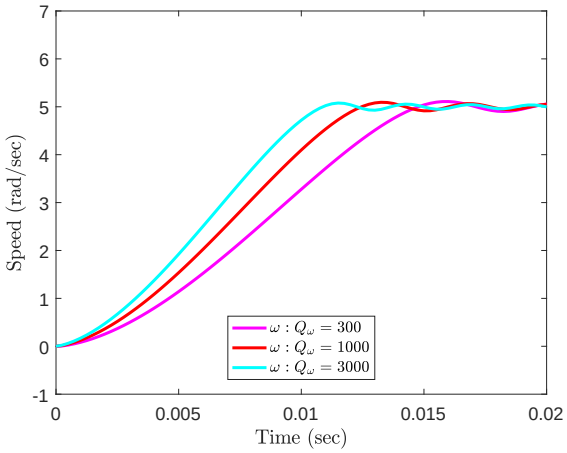

Fig. 5. Comparison of speed trajectories for various speed tracking error penalties (Q_ω).

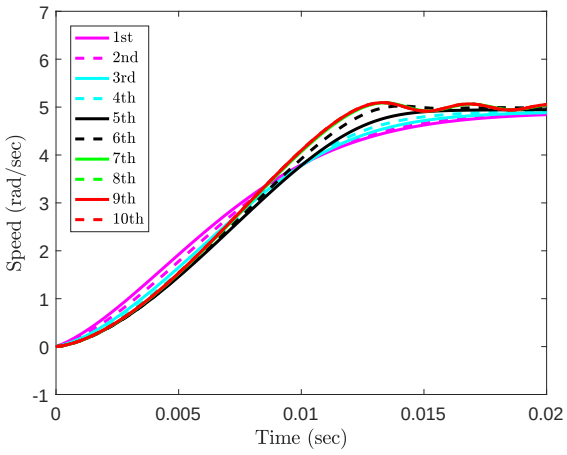

Fig. 6. Speed trajectories resulting from learned policies during multiple ADP iterations.

It is noteworthy that even though we conduct policy iteration with LQR being the initial control policy, simulation results are not presented. This is partially because that beginning with LQR and the same polynomial basis functions, policy iteration is much more likely to diverge than

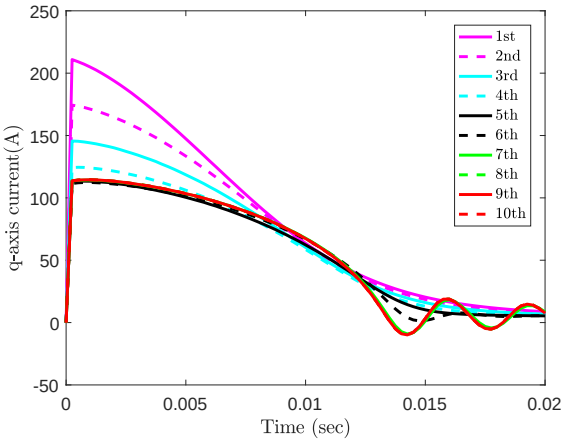

Fig. 7. q-axis current trajectories resulting from learned policies during multiple ADP iterations.

the other case. Specifically, policy evaluation of LQR mostly ends up with a reward function which is indefinite, unless the samples of tracking error e during policy evaluation are drawn from a tiny neighborhood of $e = 0$. We believe that this is because of conditioning issues of the linear algebraic equations, specifically the left hand side (22). Since the left hand side is uniquely determined by basis functions $\{\varphi_j\}_{j=1}^{N}$ and the tracking error dynamics, the ill-conditioning can be alleviated by normalizing the error dynamics, alternative basis functions, as well as using algorithms avoiding the operation of matrix inverse, for example, via recursive least square, to solve algebraic equations in policy evaluation. We consign this to future work.

V. Conclusion and Future Work

This work investigates whether ADP, a well-known data-driven optimal control technique, can improve the transient performance of induction machines in future industrial settings. Our preliminary results demonstrate that an unequivocal 'yes' or 'no' cannot yet be provided. The method is promising because ADP (after careful tuning) can compute a near-optimal control policy that outperforms a baseline nonlinear control or an initial linearized control policy. On the other hand, several pitfalls of ADP have been identified: naively applying policy iteration without an appreciation for these pitfalls will likely result in less-than-useful iterates of ADP-based control policies. These pitfalls include how one chooses the initial control policy and basis functions, where to sample data to get meaningful policy iterates, and how to reliably estimate the region of attraction of each policy that is learned and improved. We conclude that ADP when implemented with a good understanding of function approximation and nonlinear control is indeed useful, but in terms of systematic implementation of ADP with automatic selection of basis functions, asserting regions of attraction, and synthesizing useful and safe control policies using online data, many questions need conclusive answers before ADP becomes an integral part of smart factories.

References

[1] J. Holtz, "Sensorless control of induction motor drives," *Proceedings of the IEEE*, vol. 90, no. 8, pp. 1359–1394, 2002.

[2] R. Marino, P. Tomei, and C. M. Verrelli, *Induction Motor Control Design*. London, UK: Springer, 2010.

[3] H. Zhang, L. Cui, and Y. Luo, "Near-optimal control for nonzero-sum differential games of continuous-time nonlinear systems using single-network ADP." *IEEE Transactions on Cybernetics*, Vol. 43, no. 1, pp.206-216, 2013.

[4] H. Zhang, Y. Luo, and D. Liu, "Neural-network-based near-optimal control for a class of discrete-time affine nonlinear systems with control constraints." *IEEE Transactions on Neural Networks*, Vol. 20, no. 9, pp.1490-1503, 2009.

[5] Y. Zhang and H. Yang, "Model predictive torque control of induction motor drives with optimal duty cycle control," *IEEE Trans. Power Electron.*, vol. 29, no. 12, pp. 6593–6683, 2014.

[6] X. Fu and S. Li, "A novel neural network vector control technique for induction motor drive," *IEEE Trans. Energy Convers.*, vol. 30, no. 4, pp. 1428–1437, Dec. 2015.

[7] F. Blaschke, "The principle of field orientation as applied to the new transvector closed-loop system for rotating-field machines," *Siemens review*, vol. 34, no. 3, pp. 217–220, 1972.

[8] S. Odhano, R. Bojoi, A. Boglietti, G. Roayu, and G. Griva, "Maximum efficiency per torque direct flux vector control of induction motor drives," *IEEE Trans. Ind. Appl.*, vol. 51, no. 6, pp. 4415–4424, 2015.

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

[10] 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.

[11] D. Kleinman, "On an iterative technique for Riccati equation computations," *IEEE Trans. Aut. Control*, Vol. 13, pp. 114–115, 1968.

[12] 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.

[13] 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.

[14] 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 Transactions on Systems, Man, and Cybernetics: Systems*, Vol. 46, no. 5, pp.713-717, 2016.

[15] A. Chakrabarty, V. Dinh, M. Corless, A. Rundell, S. Żak, and G. Buzzard. "Support vector machine informed explicit nonlinear model predictive control using low-discrepancy sequences." *IEEE Transactions on Automatic Control*, Vol. 62, no. 1, pp. pp.135-148, 2017.

[16] 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.

[17] T. Cheng, F. Lewis, and M. 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.

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

[19] Y. Wang, J. Wu, and C. Long, "Policy iteration-based optimal control design for nonlinear descriptor systems," in *Proc. American Control Conference*, pp. 5740–5745, 2016.

[20] Y. Wang, "Data-driven output feedback optimal control for a class of nonlinear systems via adaptive dynamic programming approach: Part I-algorithms," in *Proc. 2018 CCC*, 2018, pp. 2926–2932.

[21] R. A. Howard, *Dynamic Programming and Markov Processes*. Cambridge, MA: MIT Press, 1960.

[22] G. Saridis and C.S.G. Lee, "An approximation theory of optimal control for trainable manipulators," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 9, no. 3, pp. 152–159, 1979.

[23] R. J. Leake and R.-W. Liu, "Construction of suboptimal control sequences," *SIAM Journal on Control*, vol. 5, no. 1, pp. 54–63, 1967.

[24] R. Beard, G. Saridis, and J. Wen, "Galerkin approximations of the generalized HJB equation," *Automatica*, 33(12), pp. 2159–2177, 1997.

[25] 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.

[26] D. Vrabie and F. L. Lewis, "Neural network approach to continuous-time direct adaptive optimal control for partially unknown nonlinear systems," *Neural Networks*, vol. 22, no. 3, pp. 237–246, 2009.