JJMIE

Volume 6, Number 2, April 2012
ISSN 1995-6665
Pages 155 - 161

Jordan Journal of Mechanical and Industrial Engineering

Optimal Controller Design Algorithm For Non-Affine in Input Discrete-Time Nonlinear System

A. Al-Tamimi[a,*]

[a]Department of Mechatronic Engineering, Hashemite University 13115 Jordan

Abstract

Convergence is proven of the value-iteration-based algorithm to find the optimal controller in the case of general non-affine in input nonlinear systems. That is, it is shown that algorithm converges to the optimal control and the optimal value function. It is assumed that at each iteration the value and action update equations can be exactly solved. Then two standard neural networks (NN) are used: a critic NN is used to approximate the value function while an action network is used to approximate the optimal control policy.

Keyword: Optimal control; Adaptive critics; Approximate dynamic programming; Hamilton Jacobi Bellman; Value iteration; Policy iteration

1. Introduction

This paper is concerned with the design of optimal controller using the application of approximate dynamic programming techniques (ADP) to solve for the value function, and hence the optimal control policy, in discrete-time non-affine in input nonlinear optimal control problems having continuous state and action spaces. ADP is a reinforcement learning approach Sutton and Barto [20] based on adaptive critics Barto et al. [3], Widrow et al. [25]) to solve dynamic programming problems utilizing function approximation for the value function. ADP techniques can be based on value iterations or policy iterations. In contrast with value iterations, policy iterations require an initial stabilizing control action, Sutton and Barto [20]. Howard [11] proved convergence of policy iteration for Markov Decision Processes with discrete state and action spaces. Lookup tables are used to store the value function iterations at each state. Watkins [21] developed Q-learning for discrete state and action MDPs, where a 'Q function' is stored for each state/action pair, and model dynamics are not needed to compute the control action..

ADP was proposed by Werbos [22], [23], [24] for discrete-time dynamical systems having continuous state and action spaces as a way to solve optimal control problems, Lewis and Syrmos [14], forward in time. Bertsekas and Tsitsiklis [4] provide a treatment of Neurodynamic programming, where neural networks (NN) are used to approximate the value function. Cao [26] presents a general theory for learning and optimization.

ADP for linear systems has received ample attention. An off-line policy iteration scheme for discrete-time systems with known dynamics was given in [10] to solve the discrete-time Riccati equation. In [5] Bradtke et al. implemented an (online) Q-learning policy iteration method for discrete-time linear quadratic regulator (LQR) optimal control problems. A convergence proof was given. Hagen [9] discussed, for the LQR case, the relation between the Q-learning method and model-based adaptive control with system identification. Landelius [13] applied HDP, DHP, ADHDP and ADDHP value iteration techniques, called greedy policy iterations therein, to the discrete-time LQR problem and verified their convergence. It was shown that these iterations are in fact equivalent to iterative solution of an underlying algebraic Riccati equation, which is known to converge (Lancaster and Rodman [12]). Liu and Balakrishnan [16] showed convergence of DHP for the LQR case.

In this paper a full, rigorous proof of convergence of the online value-iteration algorithm, to solve the DT HJB equation of the optimal control problem for general non-affine in input nonlinear discrete-time systems is provided. It is assumed that at each iteration, the value update and policy update equations can be exactly solved. Note that this is true in the specific case of the LQR, where the action is linear and the value quadratic in the states. For implementation, two NN are used- the critic NN to approximate the value and the action NN to approximate the control. As a value iteration based algorithm, of course, an initial stabilizing policy is not needed.

Section II of the paper starts by introducing the non-affine in-input nonlinear discrete-time optimal control problem. Section III demonstrates how to setup the HDP algorithm to solve for the nonlinear discrete-time optimal control problem. In Section IV, the proof the convergence of HDP value iterations to the solution of the DT HJB equation is presented. In Section V, two neural network parametric structures to approximate the optimal value function and policy is introduced. As is known, this

* Corresponding author. e-mail: altamimi@hu.edu.jo

provides a procedure for implementing the HDP algorithm. Finally, Section VI presents example that show the practical effectiveness of the ADP technique. The example considers a nonlinear system and the results are compared to solutions based on State Dependent Riccati Equations (SDRE).

2. The Discrete-Time Hub Equation

Consider a non-affine in input nonlinear dynamical-system of the form

$$x_{k+1} = f(x_k, u(x_k)) \tag{1}$$

where $x \in \mathbb{R}^n$, $f(x) \in \mathbb{R}^n$, and the input $u \in \mathbb{R}^m$. Suppose the system is drift-free and, without loss of generality, that $x = 0$ is an equilibrium state, e.g. $f(0) = 0$, Assume that the system **Error! Reference source not found.** is stabilizable on a prescribed compact set $\Omega \in \mathbb{R}^n$.

Definition 1. Stabilizable system: A nonlinear dynamical system is defined to be stabilizable on a compact set $\Omega \in \mathbb{R}^n$ if there exists a control input $u \in \mathbb{R}^m$ such that, for all initial conditions $x_0 \in \Omega$ the state $x_k \to 0$ as $k \to \infty$.

It is desired to find the control action $u(x_k)$ which minimizes the infinite-horizon cost function given as

$$V(x_k) = \sum_{n=k}^{\infty} Q(x_n) + u^T(x_n) R u(x_n) \tag{2}$$

for all x_k, where $Q(x) > 0$ and $R > 0 \in \mathbb{R}^{m \times m}$. The class of controllers needs to be stable and also guarantee that (2) is finite, *i.e.* the control must be admissible [1].

Definition 2 Admissible Control: A control $u(x_k)$ is defined to be admissible with respect to (2) on Ω if $u(x_k)$ is continuous on a compact set $\Omega \in \mathbb{R}^n$, $u(0) = 0$, u stabilizes **Error! Reference source not found.** on Ω, and $\forall x_0 \in \Omega$, $V(x_0)$ is finite.

Equation (2) can be written as

$$\begin{aligned} V(x_k) &= x_k^T Q x_k + u_k^T R u_k + \sum_{n=k+1}^{\infty} x_n^T Q x_n + u_n^T R u_n \\ &= x_k^T Q x_k + u_k^T R u_k + V(x_{k+1}) \end{aligned} \tag{3}$$

where we require the boundary condition $V(x=0) = 0$ so that $V(x_k)$ serves as a Lyapunov function. From Bellman's optimality principle [14], it is known that for the infinite-horizon optimization case, the value function $V^*(x_k)$ is time-invariant and satisfies the discrete-time Hamilton-Jacobi-Bellman (HJB) equation

$$V^*(x_k) = \min_{u_k}(x_k^T Q x_k + u_k^T R u_k + V^*(x_{k+1})) \tag{4}$$

Note that the discrete-time HJB equation develops backward-in time.

The optimal control u^* satisfies the first order necessary condition, given by the gradient of the right hand side of (4) with respect to u as

$$\frac{\partial(x_k^T Q x_k + u_k^T R u_k)}{\partial u_k} + \frac{\partial x_{k+1}}{\partial u_k}^T \frac{\partial V^*(x_{k+1})}{\partial x_{k+1}} = 0 \tag{5}$$

and therefore

$$u^*(x_k) = -\frac{1}{2} R^{-1} \frac{\partial x_{k+1}}{\partial u_k}^T \frac{\partial V^*(x_{k+1})}{\partial x_{k+1}} \tag{6}$$

Where

$$\frac{\partial^2 V(x_k)}{\partial u^2(x_k)} \rangle 0$$

Substituting (6) in (4), one may write the discrete-time HJB as

$$\begin{aligned} V^*(x_k) = x_k^T Q x_k + \frac{1}{4} \frac{\partial V^{*T}(x_{k+1})}{\partial x_{k+1}} \frac{\partial x_{k+1}}{\partial u_k} R^{-1} \frac{\partial x_{k+1}}{\partial u_k}^T \frac{\partial V^*(x_{k+1})}{\partial x_{k+1}} \\ + V^*(x_{k+1}) \end{aligned} \tag{7}$$

where $V^*(x_k)$ is the value function corresponding to the optimal control policy $u^*(x_k)$. This equation reduces to the Riccati equation in the linear quadratic regulator (LQR) case, which can be efficiently solved. In the general nonlinear case, the HJB cannot be solved exactly.

In the next sections we apply the HDP algorithm to solve for the value function V^* of the HJB equation (7) and present a convergence proof.

3. The HDP Algorithm

The HDP value iteration algorithm [22] is a method to solve the DT HJB online. In this section, a proof of convergence of the HDP algorithm for the non-affine in-input nonlinear discrete-time setting is presented.

In the HDP algorithm, one starts with an initial value, e.g. $V_0(x) = 0$ and then solves for u_0 as follows

$$u_o(x_k) = \arg\min_u (x_k^T Q x_k + u^T R u + V_0(x_{k+1})) \tag{8}$$

Once the policy u_0 is determined, iteration on the value is performed by computing

$$\begin{aligned} V_1(x_k) &= x_k^T Q x_k + u_0^T(x_k) R u_0(x_k) + V_0(f(x_k), u_0(x_k))) \\ &= x_k^T Q x_k + u_0^T(x_k) R u_0(x_k) + V_0(x_{k+1}) \end{aligned} \tag{9}$$

The HDP value iteration scheme therefore is a form of incremental optimization that requires iterating between a sequence of action policies $u_i(x)$ determined by the greedy update

$$\begin{aligned} u_i(x_k) &= \arg\min_u (x_k^T Q x_k + u^T R u + V_i(x_{k+1})) \\ u_i(x_k) &= \arg\min_u (x_k^T Q x_k + u^T R u + V_i(f(x_k, u))) \end{aligned} \tag{10}$$

and a sequence $V_i(x) \geq 0$
where

$$\begin{aligned} V_{i+1}(x_k) &= \min_u (x_k^T Q x_k + u^T R u + V_i(x_{k+1})) \\ &= x_k^T Q x_k + u_i^T(x_k) R u_i(x_k) + V_i(f(x_k, u_i(x_k))) \end{aligned} \quad (11)$$

with initial condition $V_0(x_k) = 0$.

Note that, as a value-iteration algorithm, HDP *does not require an initial stabilizing gain*. This is important as stabilizing gains are difficult to find for general nonlinear systems.

Note that i is the value iterations index, while k is the time index. The HDP algorithm results in an incremental optimization that is implemented forward in time and online. Note that unlike the case for policy iterations in [10], the sequence $V_i(x_k)$ is not a sequence of cost functions and are therefore not Lyapunov functions for the corresponding policies $u_i(x_k)$ which are in turn not necessarily stabilizing. In Section IV it is shown that $V_i(x_k)$ and $u_i(x_k)$ converges to the value function of the optimal control problem and to the corresponding optimal control policy respectively.

4. Convergence of The HDP Algorithm

In this section, the proof of convergence for nonlinear HDP is presented. That is, the iteration (10) and (11) converges to the optimal value, *i.e.* $V_i \to V^*$ and $u_i \to u^*$ as $i \to \infty$. The linear quadratic case has been proven by [12] for the case of known system dynamics.

Lemma 1. Let μ_i be any arbitrary sequence of control policies and Λ_i be defined by

$$\Lambda_{i+1}(x_k) = Q(x_k) + \mu_i^T R \mu_i + \Lambda_i(\underbrace{f(x_k, \mu_i(x_k))}_{x_{k+1}}) \quad (12)$$

Let u_i and V_i be the sequences defined by (10) and (11). If $V_0(x_k) = \Lambda_0(x_k) = 0$, then $V_i(x_k) \leq \Lambda_i(x_k) \ \ \forall i$.

Proof: Since $u_i(x_k)$ minimizes the right hand side of equation (11) with respect to the control u, and since $V_0(x_k) = \Lambda_0(x_k) = 0$, then by induction it follows that $V_i(x_k) \leq \Lambda_i(x_k) \ \ \forall i$.

Lemma 2. Let the sequence V_i be defined as in (11). If the system is controllable, then:

- There exists an upper bound $Y(x_k)$ such that $0 \leq V_i(x_k) \leq Y(x_k) \ \ \forall i$.
- If the optimal control problem (4) is solvable, there exists a least upper bound $V^*(x_k) \leq Y(x_k)$ where $V^*(x_k)$ solves (7), and that $\forall i: 0 \leq V_i(x_k) \leq V^*(x_k) \leq Y(x_k)$.

Proof: see [27]

Now the main result.

Theorem 1. Consider the sequence V_i and u_i defined by (11) and (10) respectively. If $V_0(x_k) = 0$, then it follows that V_i is a non-decreasing sequence $\forall i: V_{i+1}(x_k) \geq V_i(x_k)$. Moreover, as $i \to \infty$, $V_i \to V^*$, $u_i \to u^*$ and hence the sequence V_i converges to the solution of the DT HJB (7).

Proof: From Lemma 1, let μ_i be any arbitrary sequence of control policies and Λ_i be defined by

$$\Lambda_{i+1}(x_k) = Q(x_k) + \mu_i^T R \mu_i + \Lambda_i(\underbrace{f(x_k, \mu_i(x_k))}_{x_{k+1}})$$

If $V_0(x_k) = \Lambda_0(x_k) = 0$, it follows that $V_i(x_k) \leq \Lambda_i(x_k) \ \ \forall i$. Now assume that $\mu_i(x_k) = u_{i+1}(x_k)$ such that

$$\begin{aligned} \Lambda_{i+1}(x_k) &= Q(x_k) + \mu_i^T R \mu_i + \Lambda_i(f(x_k, \mu_i(x_k))) \\ &= Q(x_k) + u_{i+1}^T R u_{i+1} + \Lambda_i(f(x_k, u_{i+1}(x_k))) \end{aligned} \quad (13)$$

and consider

$$V_{i+1}(x_k) = Q(x_k) + u_i^T R u_i + V_i(f(x_k, u_i(x_k))) \quad (14)$$

It will next be proven by induction that if $V_0(x_k) = \Lambda_0(x_k) = 0$, then $\Lambda_i(x_k) \leq V_{i+1}(x_k)$. Induction is initialized by letting $V_0(x_k) = \Lambda_0(x_k) = 0$ and hence

$$\begin{aligned} V_1(x_k) - \Lambda_0(x_k) &= Q(x_k) \\ &\geq 0 \\ V_1(x_k) &\geq \Lambda_0(x_k) \end{aligned}$$

Now assume that $V_i(x_k) \geq \Lambda_{i-1}(x_k)$, then subtracting (13) from (14) it follows that

$$V_{i+1}(x_k) - \Lambda_i(x_k) = V_i(x_{k+1}) - \Lambda_{i-1}(x_{k+1}) \geq 0$$

and this completes the proof that $\Lambda_i(x_k) \leq V_{i+1}(x_k)$.

From $\Lambda_i(x_k) \leq V_{i+1}(x_k)$ and $V_i(x_k) \leq \Lambda_i(x_k)$, it then follows that

$$\forall i: V_i(x_k) \leq V_{i+1}(x_k).$$

From part a) in Lemma 2 and the fact that V_i is a non-decreasing sequence, it follows that $V_i \to V_\infty$ as $i \to \infty$. From part b) of Lemma 2, it also follows that

$$V_\infty(x_k) \leq V^*(x_k).$$

It now remains to show that in fact V_∞ is V^*. To see this, note that from (11) it follows that

$$V_\infty(x_k) = x_k^T Q x_k + u_\infty^T(x_k) R u_\infty(x_k) + V_\infty(f(x_k, u_\infty(x_k)))$$

and hence

$$V_\infty(f(x_k, u_\infty(x_k))) - V_\infty(x_k) = -x_k^T Q x_k - u_\infty^T(x_k) R u_\infty(x_k)$$

and therefore $V_\infty(x_k)$ is a Lyapunov function for a stabilizing and admissible policy $u_\infty(x_k) = \eta(x_k)$. Using part b) of Lemma 2 it follows that $V_\infty(x_k) = Y(x_k) \geq V^*(x_k)$. This implies that $V^*(x_k) \leq V_\infty(x_k) \leq V^*(x_k)$ and hence $V_\infty(x_k) = V^*(x_k)$, $u_\infty(x_k) = u^*(x_k)$.

5. Neural Network Approximation for Value and Action

It has just been proven that the nonlinear HDP algorithm converges to the value function of the DT HJB equation that appears in the non-affine in-input nonlinear discrete-time optimal control. It was assumed that the action and value update equations (10), (11) can be exactly solved at each iteration. In fact, these equations are difficult to solve for general nonlinear systems. Therefore, for implementation purposes, one needs to approximate u_i, V_i at each iteration. This allows approximate solution of (10), (11).

In this section, we review how to implement the HDP value iterations algorithm with two parametric structures such as neural networks [24] [15]. NN Approximation for Implementation of HDP Algorithm for Nonlinear Systems

It is well known that neural networks can be used to approximate smooth functions on prescribed compact sets [28]. Therefore, to solve (11) and (10), $V_i(x)$ is approximated at each step by a critic NN

$$\hat{V}_i(x) = \sum_{j=1}^{L} w_{vi}^j \phi_j(x) = W_{Vi}^T \boldsymbol{\phi}(x) \tag{15}$$

and $u_i(x)$ by an action NN

$$\hat{u}_i(x) = \sum_{j=1}^{M} w_{ui}^j \sigma_j(x) = W_{ui}^T \boldsymbol{\sigma}(x) \tag{16}$$

where the activation functions are $\phi_j(x), \sigma_j(x) \in C^1(\Omega)$ respectively. Since it is required that $V_i(0) = 0$ and $u_i(0) = 0$, we select activation functions with $\phi_j(0) = 0, \sigma_j(0) = 0$. Moreover, since it is known that V^* is a Lyapunov function, and Lyapunov proofs are convenient if the Lyapunov function is symmetric and positive definite, it is convenient to also require that the activation functions for the critic NN be symmetric, i.e. $\phi_j(x) = \phi_j(-x)$.

The neural network weights in the critic NN (15) are w_{vi}^j. L is the number of hidden-layer neurons. The vector $\boldsymbol{\phi}(x) \equiv [\phi_1(x)\ \phi_2(x) \cdots \phi_L(x)]^T$ is the vector activation function and $W_{Vi} \equiv \left[w_{vi}^1\ w_{vi}^2 \cdots w_{vi}^L \right]^T$ is the weight vector at iteration i. Similarly, the weights of the neural network in (16) are w_{ui}^j. M is the number of hidden-layer neurons. $\boldsymbol{\sigma}(x) \equiv [\sigma_1(x)\ \sigma_2(x) \cdots \sigma_L(x)]^T$ is the vector activation function, and $W_{ui} \equiv \left[w_{ui}^1\ w_{ui}^2 \cdots w_{ui}^L \right]^T$ is the vector weight.

According to (11), the critic weights are tuned at each iteration of HDP to minimize the residual error between $\hat{V}_{i+1}(x_k)$ and the target function defined in equation (17) in a least-squares sense for a set of states x_k sampled from a compact set $\Omega \subset \mathbb{R}^n$.

$$\begin{aligned} d(x_k, x_{k+1}, W_{Vi}, W_{ui}) &= x_k^T Q x_k + \hat{u}_i^T(x_k) R \hat{u}_i(x_k) + \hat{V}_i(x_{k+1}) \\ &= x_k^T Q x_k + \hat{u}_i^T(x_k) R \hat{u}_i(x_k) + W_{Vi}^T \boldsymbol{\phi}(x_{k+1}) \end{aligned} \tag{17}$$

The residual error (c.f. temporal difference error) becomes

$$\left(W_{Vi+1}^T \boldsymbol{\phi}(x_k) - d(x_k, x_{k+1}, W_{Vi}, W_{ui}) \right) = e_L(x) \tag{18}$$

Note that the residual error in (18) is explicit, in fact linear, in the tuning parameters W_{Vi+1}. Therefore, to find the least-squares solution, the method of weighted residuals may be used [8]. The weights W_{Vi+1} are determined by projecting the residual error onto $de_L(x)/dW_{Vi+1}$ and setting the result to zero $\forall x \in \Omega$ using the inner product, *i.e.*

$$\left\langle \frac{de_L(x)}{dW_{Vi+1}}, e_L(x) \right\rangle = 0 \tag{19}$$

where $\langle \mathrm{f}, \mathrm{g} \rangle = \int_\Omega f g^T dx$ is a Lebesgue integral. One has

$$0 = \int_\Omega \phi(x_k) \left(\phi^T(x_k) W_{Vi+1} - d^T(x_k, x_{k+1}, W_{Vi}, W_{ui}) \right) dx_k \tag{20}$$

Therefore a unique solution for W_{Vi+1} exists and is computed as

$$W_{Vi+1} = \left(\int_\Omega \phi(x_k) \phi(x_k)^T dx \right)^{-1} \int_\Omega \phi(x_k) d^T(\phi(x_k), W_{Vi}, W_{ui}) dx \tag{21}$$

To use this solution, it is required that the outer product integral be positive definite. This is known as a persistence of excitation condition in system theory. The next assumption is standard in selecting the NN activation functions as a basis set.

Assumption 1. The selected activation functions

$\{\phi_j(x)\}^L$ are linearly independent on the compact set $\Omega \subset \mathbb{R}^n$.

Assumption 1 guarantees that excitation condition is satisfied and hence $\int_\Omega \phi(x_k)\phi(x_k)^T dx$ is of full rank and invertible and a unique solution for (21) exists.

The action NN weights are tuned to solve (10) at each iteration. The use of $\hat{u}_i(x_k, W_{ui})$ from (16) allows the rewriting of equation (16) as

$$W_{ui} = \arg\min_w \left(x_k^T Q x_k + \hat{u}_i^T(x_k, w) R \hat{u}_i(x_k, w) + \hat{V}_i(x_{k+1}^i) \right)\Big|_\Omega \quad (22)$$

where $x_{k+1}^i = f(x_k, \hat{u}_i(x_k, w))$ and the notation means minimization for a set of points x_k selected from the compact set $\Omega \in \mathbb{R}^n$.

Note that the control weights W_{ui} appear in (22) in an implicit fashion, *i.e.* it is difficult to solve explicitly for the weights since the current control weights determine x_{k+1}. Therefore, one can use an LMS algorithm on a training set constructed from Ω. The weight update is therefore

$$\begin{aligned} W_{ui}|_{m+1} &= W_{ui}|_m - \\ &\alpha \frac{\partial (x_k^T Q x_k + \hat{u}_i^T(x_k, W_{ui}|_m) R \hat{u}_i(x_k, W_{ui}|_m) + \hat{V}_i(x_{k+1}))}{\partial W_{ui}}\Big|_{W_{ui}|_m} \\ &= W_{ui}|_m - \alpha \sigma(x_k) \left(2R\hat{u}_i(x_k, W_{ui}|_m) + \frac{\partial x_{k+1}}{\partial u_k}^T \frac{\partial \phi(x_{k+1})}{\partial x_{k+1}} W_{Vi} \right)^T \end{aligned} \quad (23)$$

where α is a positive step size and m is the iteration number for the LMS algorithm. By a stochastic approximation type argument, the weights $W_{ui}|_m \Rightarrow W_{ui}$ as $m \Rightarrow \infty$, and satisfy (22). Note that one can use alternative tuning methods such as Newton's method and Levenberg-Marquardt in order to solve (22).

6. Simulation Examples

In this section, an examples are provided to demonstrate the solution of the DT HJB equation. The example is for a DT non-affine in-input nonlinear system. MATLAB is used in the simulations to implement some of the functions discussed in the paper.

Consider the following affine in input nonlinear system

$$x_{k+1} = f(x_k, u_k) \quad (24)$$

Where

$$f(x_k, u_k) = \begin{bmatrix} 0.2 x_k(1) \exp(x_k^2(2)) \\ .3 x_k^3(2) \end{bmatrix} + \begin{bmatrix} 0 \\ -.2 \end{bmatrix} u_k^2$$

The approximation of the value function is given as

$$\hat{V}_{i+1}(x_k, W_{Vi+1}) = W_{Vi+1}^T \phi(x_k)$$

The vector activation function is selected as

$$\begin{aligned} \phi(x) = [& x_1^2 \quad x_1 x_2 \quad x_2^2 \quad x_1^4 \quad x_1^3 x_2 \\ & x_1^2 x_2^2 \quad x_1 x_2^3 \quad x_2^4 \quad x_1^6 \quad x_1^5 x_2 \quad x_1^4 x_2^2 \\ & x_1^3 x_2^3 \quad x_1^2 x_2^4 \quad x_1 x_2^5 \quad x_2^6] \end{aligned}$$

and the weight vector is

$$W_V^T = \begin{bmatrix} w_v^1 & w_v^2 & w_v^3 & w_v^4 & ..... & w_v^{15} \end{bmatrix}.$$

The control is approximated by

$$\hat{u}_i = W_{ui}^T \sigma(x_k)$$

where the vector activation function is

$$\begin{aligned} \sigma^T(x) = [& x_1 \quad x_2 \quad x_1^3 \quad x_1^2 x_2 \quad x_1 x_2^2 \\ & x_2^3 \quad x_1^5 \quad x_1^4 x_2 \quad x_1^3 x_2^2 \quad x_1^2 x_2^3 \\ & x_1 x_2^4 \quad x_2^5] \end{aligned}$$

and the weights are

$$W_u^T = \begin{bmatrix} w_u^1 & w_u^2 & w_u^3 & w_u^4 & ..... & w_u^{12} \end{bmatrix}$$

The control NN activation functions are selected as the derivatives of the critic activation functions, since the gradient of the critic activation functions appears in (23). The critic activations are selected as polynomials to satisfy $\hat{V}_i(0) = 0$ at each step. Note that then automatically one has $\hat{u}_i(0) = 0$ as required for admissibility. The result of the algorithm is compared to the discrete-time State Dependent Riccati Equation (SDRE) proposed in [6].

The training sets is $x_1 \in [-2, 2]$, $x_2 \in [-1, 1]$. The value function weights converged to the following

$$\begin{aligned} W_V^T = [& 1.0382 \quad 0 \quad 1.0826 \quad .0028 \quad -0 \quad -.053 \quad 0 \quad -.2792 \\ & -.0004 \quad 0 \quad -.0013 \quad 0 \quad .1549 \quad 0 \quad .3034] \end{aligned}$$

and the control weights converged to

$$W_u^T = [\, 0 \quad -.0004 \quad 0 \quad 0 \quad 0 \quad .0651 \quad 0 \quad 0 \quad 0 \quad -.0003 \quad 0 \quad -.0046]$$

The result of the nonlinear optimal controller derived in this paper is compared to the SDRE approach. Figure 2 and Figure 3 show the states trajectories for the system for both.

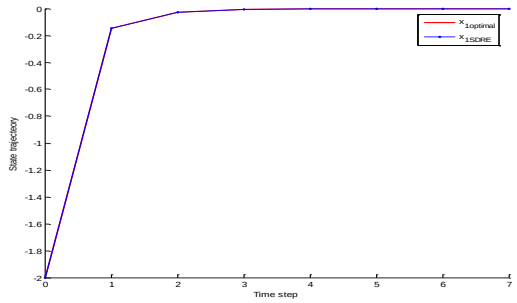

Figure 2: The state trajectory for both methods.

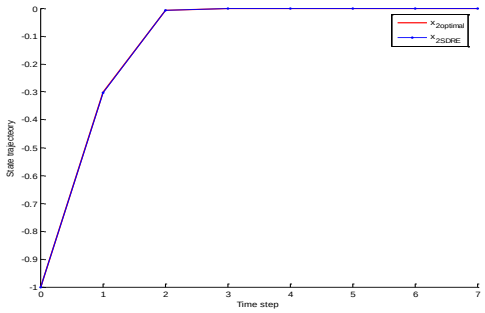

Figure 3: The state trajectory for both methods.

In Figure 4, the cost function of the SDRE solution and the cost function of the proposed algorithm in this paper are compared. It is clear from the simulation that the cost function for the control policy derived from the HDP method is lower than that of the SDRE method. In Figure 5, the control signals for both methods are shown.

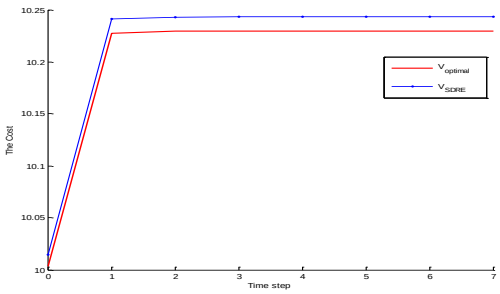

Figure 4: The cost function for both methods.

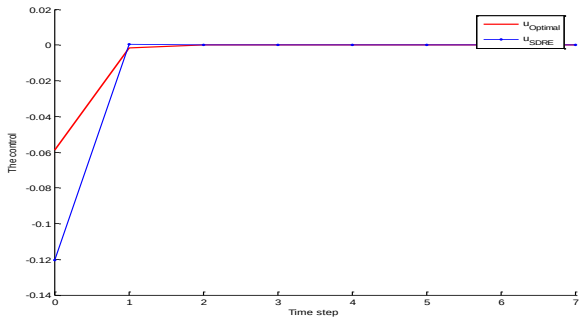

Figure 5: The control signal input for both methods.

7. Conclusion

It has been proven the convergence of the algorithm to the value function solution of Hamilton-Jacobi-Bellman equation for non-affine in-input nonlinear dynamical systems, assuming exact solution of value update and the action update at each iteration.

Neural networks are used as parametric structures to approximate at each iteration the value (i.e. critic NN), and the control action. In the special case affine-in input nonlinear system $x_{k+1} = f(x_k) + g(x_k)u_k$, the use of the second neural network to approximate the control policy, the internal dynamics, *i.e.* $f(x_k)$, is not needed to implement HDP. This holds as well for the special LQR case for linear system as shown in [27], where use of two NN avoids the need to know the system internal dynamics. In the non-affine in-input nonlinear example, it is shown that the optimal controller derived from the HDP based value iteration method outperforms suboptimal control methods like those found through the SDRE method.

References

[1] Abu-Khalaf, M., F. L. Lewis, "Nearly Optimal Control Laws for Nonlinear Systems with Saturating Actuators Using a Neural Network HJB Approach," Automatica, vol. 41, pp. 779 – 791, 2005.

[2] Al-Tamimi, A., M. Abu-Khalaf, F. L. Lewis, "Adaptive Critic Designs for Discrete-Time Zero-Sum Games with Application to H-Infinity Control" *IEEE Transactions on Systems, Man, Cybernetics-Part B,* Cybernaetics, Vol 37, No 1, pp 240-247, Feb , 2007.

[3] Barto, A. G., R. S. Sutton, and C. W. Anderson, "Neuronlike elements that can solve difficult learning control problems," *IEEE Trans. Syst., Man, Cybern.,* vol. SMC-13, pp. 835–846, 1983.

[4] Bertsekas, D.P. and J. N. Tsitsiklis, *Neuro-Dynamic Programming*, Athena Scientific, MA, 1996.

[5] Bradtke, S. J., B. E. Ydestie, A. G. Barto, "Adaptive linear quadratic control using policy iteration," Proceedings of the American Control Conference , pp. 3475-3476, Baltmore, Myrland, June, 1994.

[6] Cloutier, J. R.," State –Dependent Riccati equation Techniques: An overview," Proceeding of the American control conference, Albuquerque, NM, June 4-6, 1997, pp 932-936.

[7] Ferrari, S., Stengel, R. "Model-Based Adaptive Critic Designs", pp 64-94, Eds J. Si, A. Barto, W. Powell, D. Wunsch *Handbook of Learning and Approximate Dynamic Programming*, *Wiley*, August 2004.

[8] Finlayson, B. A.. The Method of Weighted Residuals and Variational Principles. Academic Press, New York, 1972

[9] Hagen, S. B Krose, " Linear quadratic Regulation using Reinforcement Learning," Belgian_Dutch Conference on Mechanical Learning, pp. 39-46, 1998.

[10] Hewer, G.A., "An iterative technique for the computation of the steady state gains for the discrete optimal regulator," IEEE Trans. Automatic Control, pp. 382-384, Aug. 1971.

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

[12] Lancaster, P. L. Rodman, *Algebraic Riccati Equations*, Oxford University Press, UK 1995.

[13] Landelius, T., *Reinforcement Learning and Distributed Local Model Synthesis*, PhD Dissertation, Linkoping University, Sweden, 1997.

[14] Lewis, F. L., V. L. Syrmos, *Optimal Control*, 2nd ed., John Wiley, 1995.

[15] Lewis, F. L., Jagannathan, S., & Yesildirek, A.. *Neural Network Control of Robot Manipulators and Nonlinear Systems.* Taylor & Franci, (1999).

[16] Lu, X., S.N. Balakrishnan, "Convergence analysis of adaptive critic based optimal control," Proc. Amer. Control Conf., pp. 1929-1933, Chicago, 2000.
[17] Prokhorov, D., D. Wunsch, "Adaptive critic designs," IEEE Trans. on Neural Networks, vol. 8, no. 5, pp 997-1007, 1997.
[18] Si, Ji. A. Barto, W. Powell, D. Wunsch, *Handbook of Learning and Approximate Dynamic Programming*, John Wiley, New Jersey, 2004.
[19] Stevens B., F. L. Lewis, *Aircraft Control and Simulation*, 2nd edition, John Wiley, New Jersey, 2003.
[20] Sutton, R.S., A.G. Barto, *Reinforcement Learning*, MIT Press, Cambridge, MA, 1998.
[21] Watkins, C., *Learning from Delayed Rewards*, Ph.D. Thesis, Cambridge University, Cambridge, England, 1989.
[22] Werbos, P.J., "A menu of designs for reinforcement learning over time," , Neural Networks for Control, pp. 67-95, ed. W.T. Miller, R.S. Sutton, P.J. Werbos, Cambridge: MIT Press, 1991.
[23] Werbos, P.J., "Approximate dynamic programming for real-time control and neural modeling," Handbook of Intelligent Control, ed. D.A. White and D.A. Sofge, New York: Van Nostrand Reinhold, 1992.
[24] Werbos, P.J., "Neural networks for control and system identification," Heuristics, Vol. 3, No. 1, Spring 1990, pp. 18-27,.1990.
[25] Widrow, B., N. Gupta, and S. Maitra, "Punish/reward: Learning with a critic in adaptive threshold systems," IEEE Trans. Syst., Man, Cybern., vol. SMC-3, pp. 455–465, 1973.
[26] Xi-Ren Cao, "Learning and Optimization—From a SystemsTheoretic Perspective", Proc. of IEEE Conference on Decision and Control, pp. 3367-3371, 2002.
[27] Al-Tamimi, F. L. Lewis, M. Abu-Khalaf, "Discrete-time nonlinear HJB solution using Approximate dynamic programming: Convergence Proof," *IEEE Transactions on Systems, Man, Cybernetics-Part B*, Volume 38, Issue 4, pages 943 – 949, Aug 2008.
[28] Hornik, K., A. Stinch combe, H. White (1990). Universal approximation of unknown mapping and its derivatives using multilayer feed forward networks, Neural Newton, Vol.3, P 551-560.

Appendix

Lemma 2. Let the sequence V_i be defined as in (11). If the system is controllable, then:

- There exists an upper bound $Y(x_k)$ such that $0 \le V_i(x_k) \le Y(x_k) \ \ \forall i$.
- If the optimal control problem (4) is solvable, there exists a least upper bound $V^*(x_k) \le Y(x_k)$ where $V^*(x_k)$ solves (7), and that $\forall i : 0 \le V_i(x_k) \le V^*(x_k) \le Y(x_k)$.

Proof: Let $\eta(x_k)$ be any stabilizing and admissible control policy, and Let $V_0(x_k) = Z_0(x_k) = 0$ where Z_i is updated as

$$\begin{aligned} Z_{i+1}(x_k) &= Q(x_k) + \eta^T(x_k) R \eta(x_k) + Z_i(x_{k+1}) \\ x_{k+1} &= f(x_k) + g(x_k)\eta(x_k) \end{aligned} \tag{1}$$

It follows that the difference

$$\begin{aligned} Z_{i+1}(x_k) - Z_i(x_k) &= Z_i(x_{k+1}) - Z_{i-1}(x_{k+1}) \\ &= Z_{i-1}(x_{k+2}) - Z_{i-2}(x_{k+2}) \\ &= Z_{i-2}(x_{k+3}) - Z_{i-3}(x_{k+3}) \\ &\ \ . \\ &\ \ . \\ &\ \ . \\ &= Z_1(x_{k+i}) - Z_0(x_{k+i}) \end{aligned} \tag{2}$$

Since $Z_0(x_k) = 0$, it then follows that

$$\begin{aligned} Z_{i+1}(x_k) &= Z_1(x_{k+i}) + Z_i(x_k) \\ &= Z_1(x_{k+i}) + Z_1(x_{k+i-1}) + Z_{i-1}(x_k) \\ &= Z_1(x_{k+i}) + Z_1(x_{k+i-1}) + Z_1(x_{k+i-1}) + Z_{i-2}(x_k) \\ &= Z_1(x_{k+i}) + Z_1(x_{k+i-1}) + Z_1(x_{k+i-2}) + \ldots\ldots + Z_1(x_k) \end{aligned} \tag{3}$$

and equation (2) can be written as

$$\begin{aligned} Z_{i+1}(x_k) &= \sum_{n=0}^{i} Z_1(x_{k+n}) \\ &= \sum_{n=0}^{i} (Q(x_{k+n}) + \eta^T(x_{k+n}) R \eta(x_{k+n})) \\ &\le \sum_{n=0}^{\infty} (Q(x_{k+n}) + \eta^T(x_{k+n}) R \eta(x_{k+n})) \end{aligned}$$

Since $\eta(x_k)$ is an admissible stabilizing controller, $x_{k+n} \to 0$ as $n \to \infty$ and

$$\forall i : \ \ Z_{i+1}(x_k) \le \sum_{i=0}^{\infty} Z_1(x_{k+i}) = Y(x_k)$$

Using Lemma 1 with $\mu_i(x_k) = \eta(x_k)$ and $\Lambda_i(x_k) = Z_i(x_k)$, it follows that

$$\forall i : \ \ V_i(x_k) \le Z_i(x_k) \le Y(x_k)$$

which proves part a). Moreover if $\eta(x_k) = u^*(x_k)$, then

$$\underbrace{\sum_{n=0}^{\infty} (Q(x_{k+n}) + u^{*T}(x_{k+n}) R u^*(x_{k+n}))}_{V^*(x_k)} \le \underbrace{\sum_{n=0}^{\infty} (Q(x_{k+n}) + \eta^T(x_{k+n}) R \eta(x_{k+n}))}_{Y(x_k)}$$

and hence $V^*(x_k) \le Y(x_k)$ which proves part b) and shows that $\forall i : 0 \le V_i(x_k) \le V^*(x_k) \le Y(x_k)$ for any $Y(x_k)$ determined by an admissible stabilizing policy $\eta(x_k)$.