Heuristic Dynamic Programming Nonlinear Optimal Controller

Asma Al-tamimi, Murad Abu-Khalaf and Frank Lewis
The Hashemite University, Math work, The University of Texas at Arlington
Jordan, USA

1. Introduction

This chapter is concerned with the application of approximate dynamic programming techniques (ADP) to solve for the value function, and hence the optimal control policy, in discrete-time nonlinear optimal control problems having continuous state and action spaces. ADP is a reinforcement learning approach (Sutton & Barto, 1998) based on adaptive critics (Barto et al., 1983), (Widrow et al., 1973) 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 & Barto, 1998). (Howard, 1960) 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, 1989) 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, 1990,1991,1992) for discrete-time dynamical systems having continuous state and action spaces as a way to solve optimal control problems, (Lewis & Syrmos, 1995), forward in time. (Bertsekas & Tsitsiklis, 1996) provide a treatment of Neurodynamic programming, where neural networks (NN) are used to approximate the value function. (Cao, 2002) presents a general theory for learning and optimization.
(Werbos, 1992) classified approximate dynamic programming approaches into four main schemes: Heuristic Dynamic Programming (HDP), Dual Heuristic Dynamic Programming (DHP), Action Dependent Heuristic Dynamic Programming (ADHDP), (a continuous-state-space generalization of Q-learning (Watkins, 1989)), and Action Dependent Dual Heuristic Dynamic Programming (ADDHP). Neural networks are used to approximate the value function (the critic NN) and the control (the action NN), and backpropagation is used to tune the weights until convergence at each iteration of the ADP algorithm. An overview of ADP is given in (Si et al., 2004) (e.g. (Ferrari & Stengel, 2004), and also (Prokhorov & Wunsch, 1997), who deployed new ADP schemes known as Globalized-DHP (GDHP) and ADGDHP.
ADP for linear systems has received ample attention. An off-line policy iteration scheme for discrete-time systems with known dynamics was given in (Hewer, 1971) to solve the discrete-time Riccati equation. In (Bradtke et al, 1994) implemented an (online) Q-learning policy iteration method for discrete-time linear quadratic regulator (LQR) optimal control

Source: Machine Learning, Book edited by: Abdelhamid Mellouk and Abdennacer Chebira,
ISBN 978-3-902613-56-1, pp. 450, February 2009, I-Tech, Vienna, Austria

problems. A convergence proof was given. (Hagen, 1998) discussed, for the LQR case, the relation between the Q-learning method and model-based adaptive control with system identification. (Landelius, 1997) 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 & Rodman, 1995). (Lu & Balakrishnan, 2000) showed convergence of DHP for the LQR case.

(Morimoto et al, 2003) developed differential dynamic programming, a Q-learning method, to solve optimal zero-sum game problems for nonlinear systems by taking the second-order approximation to the Q function. This effectively provides an exact Q-learning formulation for linear systems with minimax value functions. In our previous work (Al-tamimi et al, 2007), we studied ADP value iteration techniques to solve the zero-sum game problem for linear discrete-time dynamical systems using quadratic minimax cost. HDP, DHP, ADHDP and ADDHP formulations were developed for zero-sum games, and convergence was proven by showing the equivalence of these ADP methods to iterative solution of an underlying Game Algebraic Riccati Equation, which is known to converge. Applications were made to H-infinity control.

For nonlinear systems with continuous state and action spaces, solution methods for the dynamic programming problem are more sparse. Policy iteration methods for optimal control for continuous-time systems with continuous state space and action spaces were given in (Abu-khalaf & Lewis, 2005) (Abu-Khalaf at el, 2004), but complete knowledge of the plant dynamics is required. The discrete-time nonlinear optimal control solution relies on solving the discrete-time (DT) Hamilton-Jacobi-Bellman (HJB) equation (Lewis & Syrmos, 1995), exact solution of which is generally impossible for nonlinear systems. Solutions to the DT HJB equation with known dynamics and continuous state space and action space were given in (Huang, 1999), where the coefficients of the Taylor series expansion of the value function are systematically computed. In (Chen & Jagannathan, 2005), the authors show that under certain conditions a second-order approximation of the discrete-time (DT) Hamilton-Jacobi-Bellman (HJB) equation can be considered; under those conditions discussed in that paper, the authors solve for the value function that satisfies the second order expansion of the DT HJB instead of solving for the original DT HJB. The authors apply a policy iteration scheme on this second order DT HJB and require an initially stable policy to start the iterations scheme. The authors also used a single (critic) neural network to approximate the value function of the second order DT HJB. These are all off-line methods for solving the HJB equations that require full knowledge of the system dynamics.

Convergence proofs for the on-line value-iteration based ADP techniques for nonlinear discrete-time systems are even more limited. (Prokhorov & Wunsch, 1997) use NN to approximate both the value (e.g. a critic NN) and the control action. Least mean squares is used to tune the critic NN weights and the action NN weights. Stochastic approximation is used to show that, at each iteration of the ADP algorithm, the critic weights converge. Likewise, at each iteration the action NN weights converge, but overall convergence of the ADP algorithm to the optimal solution is not demonstrated. A similar approach was used in (Si et al., 2004).

In (He & Jagannathan, 2005), a generalized or asynchronous version of ADP (in the sense of (Sutton & Barto, 1998) was used whereby the updates of the critic NN and action NN are interleaved, each NN being updated at each time step. Tuning was performed online. A Lyapunov approach was used to show that the method yields uniform ultimate bounded stability and that the weight estimation errors are bounded, though convergence to the exact optimal value and control was not shown. The input coupling function must be positive definite.

In this chapter, we provide a full, rigorous proof of convergence of the online value-iteration based HDP algorithm, to solve the DT HJB equation of the optimal control problem for general nonlinear discrete-time systems. 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. Full knowledge of the system dynamics is not needed to implement the HDP algorithm; in fact, the internal dynamics information is not needed. As a value iteration based algorithm, of course, an initial stabilizing policy is not needed for HDP.

The point is stressed that these results also hold for the special LQR case of linear systems $\dot{x} = Ax + Bu$ and quadratic utility. In the general folklore of HDP for the LQR case, only a single NN is used, namely a critic NN, and the action is updated using a standard matrix equation derived from the stationarity condition (Lewis & Syrmos1995). In the DT case, this equation requires the use of both the plant matrix A, e.g. the internal dynamics, and the control input coupling matrix B. However, by using a second action NN, the knowledge of the A matrix is not needed. This important issue is clarified herein.

Section two of the chapter starts by introducing the nonlinear discrete-time optimal control problem. Section three demonstrates how to setup the HDP algorithm to solve for the nonlinear discrete-time optimal control problem. In Section four, we prove the convergence of HDP value iterations to the solution of the DT HJB equation. In Section five, we introduce two neural network parametric structures to approximate the optimal value function and policy. As is known, this provides a procedure for implementing the HDP algorithm. We also discuss in that section how we implement the algorithm without having to know the plant internal dynamics. Finally, Section six presents two examples that show the practical effectiveness of the ADP technique. The first example in fact is a LQR example which uses HDP with two NNs to solve the Riccati equation online without knowing the A matrix. The second example considers a nonlinear system and the results are compared to solutions based on State Dependent Riccati Equations (SDRE).

2. The discrete-time HJB equation

Consider an affine in input nonlinear dynamical-system of the form

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

where $x \in \mathbb{R}^n$, $f(x) \in \mathbb{R}^n$, $g(x) \in \mathbb{R}^{n \times m}$ 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. $f(0) = 0$, $g(0) = 0$. Assume that the system (1) 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 (Abu-Khalaf & Lewis, 2005).

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 (1) 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 (Lewis & Syrmos, 1995), 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

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

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

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

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 (Werbos, 1990) is a method to solve the DT HJB online. In this section, a proof of convergence of the HDP algorithm in the general nonlinear discrete-time setting is presented.

3.1 The HDP algorithm

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

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

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

$$\begin{aligned} V_1(x_k) &= x_k^TQx_k+u_0^T(x_k)Ru_0(x_k)+V_0(f(x_k)+g(x_k)u_0(x_k)) \\ &= x_k^TQx_k+u_0^T(x_k)Ru_0(x_k)+V_0(x_{k+1}) \end{aligned} \tag{9}$$

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

$$\begin{aligned} u_i(x_k) &= \arg\min_u(x_k^TQx_k+u^TRu+V_i(x_{k+1})) \\ &= \arg\min_u(x_k^TQx_k+u^TRu+V_i(f(x_k)+g(x_k)u)) \\ &= \frac{1}{2}R^{-1}g(x_k)^T\frac{\partial V_i(x_{k+1})}{\partial x_{k+1}} \end{aligned} \tag{10}$$

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

$$\begin{aligned} V_{i+1}(x_k) &= \min_u(x_k^TQx_k+u^TRu+V_i(x_{k+1})) \\ &= x_k^TQx_k+u_i^T(x_k)Ru_i(x_k)+V_i(f(x_k)+g(x_k)u_i(x_k)) \end{aligned} \tag{11}$$

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 (Hewer, 1971), 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 four 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.

3.2 The special case of linear systems

Note that for the special case of linear systems, it can be shown that the HDP algorithm is one way to solve the Discrete-Time Algebraic Riccati Equation (DARE) (Landelius, 1997)). Particularly, for the discrete-time linear system

$$x_{k+1} = Ax_k + Bu_k \tag{12}$$

the DT HJB equation (7) becomes the DARE

$$P = A^T PA + Q - A^T PB(R + B^T PB)^{-1} B^T PA \tag{13}$$

with $V^*(x_k) = x_k^T Px_k$.
In the linear case, the policy update (10) is

$$u_i(x_k) = -(R + B^T P_i B)^{-1} B^T P_i Ax_k \tag{14}$$

Substituting this into (11), one sees that the HDP algorithm (10), (11) is equivalent to

$$\begin{aligned} &P_{i+1} = A^T P_i A + Q - A^T P_i B(R + B^T P_i B)^{-1} B^T P_i A \\ &P_0 = 0 \end{aligned} \tag{15}$$

It should be noted that the HDP algorithm (15) solves the DARE forward in time, whereas the dynamic programming recursion appearing in finite-horizon optimal control [21] develops backward in time

$$\begin{aligned} &P_k = A^T P_{k+1} A + Q - A^T P_{k+1} B(R + B^T P_{k+1} B)^{-1} B^T P_{k+1} A \\ &P_N = 0 \end{aligned} \tag{16}$$

where N represents the terminal time. Both equations (15) and (16) will produce the same sequence of P_i and P_k respectively. It has been shown in (Lewis & Syrmos, 1995) and (Lancaster, 1995) that this sequence converges to the solution of the DARE after enough iterations.
It is very important to point out the difference between equations (14) and (15) resulting from HDP value iterations with

$$u_i(x_k) = \underbrace{-(R + B^T P_i B)^{-1} B^T P_i A}_{K_i} x_k \tag{17}$$

$$\begin{aligned} &(A + BK_i)^T P_{i+1}(A + BK_i) - P_{i+1} = -Q - K_i^T RK_i \\ &(P_0, u_0): \text{ Initial stable control policy with corresponding Lyapunov function} \end{aligned} \tag{18}$$

resulting from policy iterations, those in(Hewer, 1971). Unlike P_i in (15), the sequence P_i in (18) is a sequence of Lyapunov functions. Similarly the sequence of control policies in (17) is stabilizing unlike the sequence in (14).

4. Convergence of the HDP algorithm

In this section, we present a proof of convergence for nonlinear HDP. That is, we prove convergence of the iteration (10) and (11) 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 (Lancaster, 1995) 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

$$\Lambda_{i+1}(x_k) = Q(x_k) + \mu_i^T R\mu_i + \Lambda_i(\underbrace{f(x_k) + g(x_k)\mu_i(x_k)}_{x_{k+1}}) . \tag{19}$$

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 $V_i(x_k) \le \Lambda_i(x_k)\ \ \forall i$.

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 $V_i(x_k) \le \Lambda_i(x_k)\ \ \forall i$. ■

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 $0 \le V_i(x_k) \le Y(x_k)\ \ \forall i$.

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

$$\begin{aligned} Z_{i+1}(x_k) &= Q(x_k) + \eta^T(x_k)R\eta(x_k) + Z_i(x_{k+1}) \\ x_{k+1} &= f(x_k) + g(x_k)\eta(x_k) \end{aligned} . \tag{20}$$

It follows that the difference

$$\begin{aligned} Z_{i+1}(x_k) - Z_i(x_k) &= Z_i(x_{k+1}) - Z_{i-1}(x_{k+1}) \\ &= Z_{i-1}(x_{k+2}) - Z_{i-2}(x_{k+2}) \\ &= Z_{i-2}(x_{k+3}) - Z_{i-3}(x_{k+3}) \\ &\ \ . \\ &\ \ . \\ &\ \ . \\ &= Z_1(x_{k+i}) - Z_0(x_{k+i}) \end{aligned} \tag{21}$$

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

$$\begin{aligned} Z_{i+1}(x_k) &= Z_1(x_{k+i}) + Z_i(x_k) \\ &= Z_1(x_{k+i}) + Z_1(x_{k+i-1}) + Z_{i-1}(x_k) \\ &= Z_1(x_{k+i}) + Z_1(x_{k+i-1}) + Z_1(x_{k+i-1}) + Z_{i-2}(x_k) \\ &= Z_1(x_{k+i}) + Z_1(x_{k+i-1}) + Z_1(x_{k+i-2}) + ... + Z_1(x_k) \end{aligned} \tag{22}$$

and equation (22) can be written as

$$\begin{aligned} Z_{i+1}(x_k) &= \sum_{n=0}^{i} Z_1(x_{k+n}) \\ &= \sum_{n=0}^{i} (Q(x_{k+n}) + \eta^T(x_{k+n})R\eta(x_{k+n})) \\ &\le \sum_{n=0}^{\infty} (Q(x_{k+n}) + \eta^T(x_{k+n})R\eta(x_{k+n})) \end{aligned} \tag{23}$$

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

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

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 $\eta(x_k)$. ■

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

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

and as $i \to \infty$

$$V_i \to V^*, \; u_i \to u^*$$

that is 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

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

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

$$\begin{aligned} \Lambda_{i+1}(x_k) &= Q(x_k) + \mu_i^T R \mu_i + \Lambda_i(f(x_k) + g(x_k)\mu_i(x_k)) \\ &= Q(x_k) + u_{i+1}^T R u_{i+1} + \Lambda_i(f(x_k) + g(x_k)u_{i+1}(x_k)) \end{aligned} \tag{24}$$

and consider

$$V_{i+1}(x_k) = Q(x_k) + u_i^T R u_i + V_i(f(x_k) + g(x_k)u_i(x_k)) \tag{25}$$

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

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

Now assume that $V_i(x_k) \ge \Lambda_{i-1}(x_k)$, then subtracting (24) from (25) it follows that

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

and this completes the proof that $\Lambda_i(x_k) \le V_{i+1}(x_k)$.
From $\Lambda_i(x_k) \le V_{i+1}(x_k)$ and $V_i(x_k) \le \Lambda_i(x_k)$, it then follows that

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

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

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

and hence

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

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 $V_\infty(x_k) = Y(x_k) \ge V^*(x_k)$. This implies that $V^*(x_k) \le V_\infty(x_k) \le V^*(x_k)$ and hence $V_\infty(x_k) = V^*(x_k)$, $u_\infty(x_k) = u^*(x_k)$. ■

5. Neural network approximation for Value and Action

We have just proven that the nonlinear HDP algorithm converges to the value function of the DT HJB equation that appears in the 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 (Werbos, 1990) and (Lewis & Jaganathan, 1999). The important point is stressed that the use of two NN, a critic for value function approximation and an action NN for the control, allows the implementation of HDP in the LQR case *without knowing* the system internal dynamics matrix A. This point is not generally appreciated in the folklore of ADP.

5.1 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 (Hornik & Stinchcombe, 1990). Therefore, to solve (11) and (10), $V_i(x)$ is approximated at each step by a critic NN

$$\hat{V}_i(x) = \sum_{j=1}^{L} w_{vi}^j \phi_j(x) = W_{Vi}^T \boldsymbol{\phi}(x) \tag{26}$$

and $u_i(x)$ by an action NN

$$\hat{u}_i(x) = \sum_{j=1}^{M} w_{ui}^j \sigma_j(x) = W_{ui}^T \boldsymbol{\sigma}(x) \tag{27}$$

where the activation functions are respectively $\phi_j(x), \sigma_j(x) \in C^1(\Omega)$. Since it is required that $V_i(x=0)=0$ and $u_i(x=0)=0$, 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 (26) 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 (27) 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 (28) in a least-squares sense for a set of states x_k sampled from a compact set $\Omega \subset \mathbb{R}^n$.

$$\begin{aligned} d(x_k, x_{k+1}, W_{Vi}, W_{ui}) &= x_k^T Q x_k + \hat{u}_i^T(x_k) R \hat{u}_i(x_k) + \hat{V}_i(x_{k+1}) \\ &= x_k^T Q x_k + \hat{u}_i^T(x_k) R \hat{u}_i(x_k) + W_{Vi}^T \boldsymbol{\phi}(x_{k+1}) \end{aligned} \tag{28}$$

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

$$\left(W_{Vi+1}^T \boldsymbol{\phi}(x_k) - d(x_k, x_{k+1}, W_{Vi}, W_{ui})\right) = e_L(x). \tag{29}$$

Note that the residual error in (29) 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 [11]. 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.*

$$\left\langle \frac{de_L(x)}{dW_{Vi+1}}, e_L(x) \right\rangle = 0, \tag{30}$$

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

$$0 = \int_\Omega \phi(x_k)\left(\phi^T(x_k) W_{Vi+1} - d^T(x_k, x_{k+1}, W_{Vi}, W_{ui})\right) dx_k \tag{31}$$

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

$$W_{Vi+1} = \left(\int_\Omega \phi(x_k) \phi(x_k)^T dx \right)^{-1} \int_\Omega \phi(x_k) d^T(\phi(x_k), W_{Vi}, W_{ui}) dx \tag{32}$$

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 (32) exists.

The action NN weights are tuned to solve (10) at each iteration. The use of $\hat{u}_i(x_k, W_{ui})$ from (27) allows the rewriting of equation (10) as

$$W_{ui} = \arg\min_w \left(x_k^T Q x_k + \hat{u}_i^T(x_k, w) R \hat{u}_i(x_k, w) + \hat{V}_i(x_{k+1}^i) \right)\Big|_\Omega \tag{33}$$

where $x_{k+1}^i = f(x_k) + g(x_k)\hat{u}_i(x_k, w)$ 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 (33) 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

$$W_{ui}\big|_{m+1} = W_{ui}\big|_m - \alpha \frac{\partial (x_k^T Q x_k + \hat{u}_i^T(x_k, W_{ui}|_m) R \hat{u}_i(x_k, W_{ui}|_m) + \hat{V}_i(x_{k+1}))}{\partial W_{ui}}\Bigg|_{W_{ui}|_m}$$
$$W_{ui}\big|_{m+1} = W_{ui}\big|_m - \alpha\sigma(x_k)\left(2R\hat{u}_i(x_k, W_{ui}|_m) + g(x_k)^T \frac{\partial \phi(x_{k+1})}{\partial x_{k+1}} W_{Vi} \right)^T \tag{34}$$

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 $W_{ui}\big|_m \Rightarrow W_{ui}$ as $m \Rightarrow \infty$, and satisfy (33). Note that one can use alternative tuning methods such as Newton's method and Levenberg-Marquardt in order to solve (33).

In Figure 1, the flow chart of the HDP iteration is shown. Note that because of the neural network used to approximate the control policy the internal dynamics, *i.e.* $f(x_k)$ is not needed. That is, the internal dynamics can be unknown.

Remark. Neither $f(x)$ nor $g(x)$ is needed to update the critic neural network weights using (32). Only the input coupling term $g(x)$ is needed to update the action neural network weights using (34). Therefore the proposed algorithm works for system with partially unknown dynamics- no knowledge of the internal feedback structure $f(x)$ is needed.

5.2 HDP for Linear Systems Without Knowledge of Internal Dynamics

The general practice in the HDP folklore for linear quadratic systems is to use a critic NN to approximate the value, and update the critic weights using a method such as the batch update (32), or a recursive update method such as LMS. In fact, the critic weights are nothing but the elements of the Riccati matrix and the activation functions are quadratic polynomials in terms of the states. Then, the policy is updated using

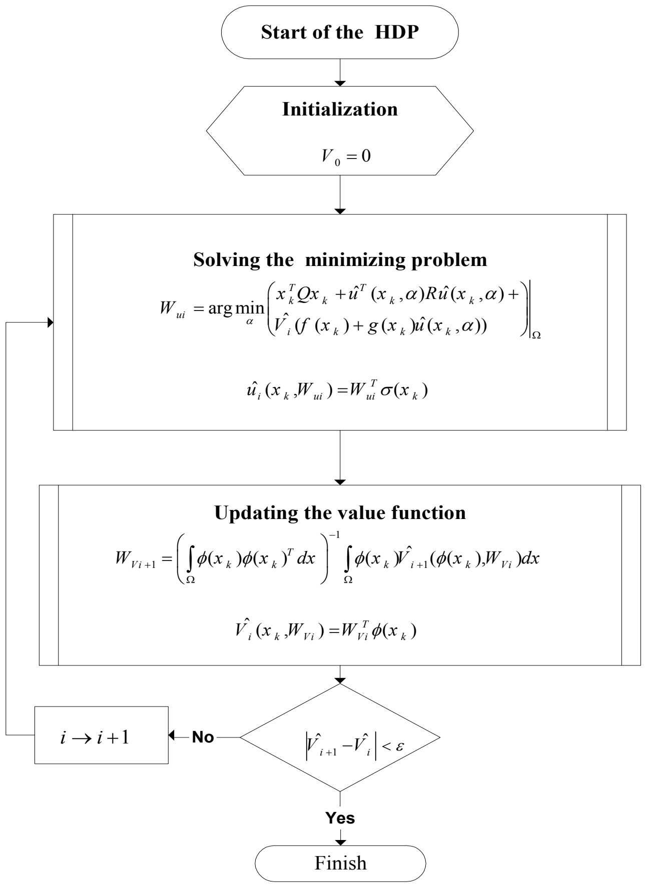

Start of the HDP
Initialization
$V_0 = 0$
Solving the minimizing problem
$W_{ui} = \arg\min_{\alpha} \left(\begin{matrix} x_k^T Q x_k + \hat{u}^T(x_k,\alpha) R \hat{u}(x_k,\alpha) + \\ \hat{V}_i(f(x_k) + g(x_k)\hat{u}(x_k,\alpha)) \end{matrix} \right) \Bigg|_{\Omega}$
$\hat{u}_i(x_k, W_{ui}) = W_{ui}^T \sigma(x_k)$
Updating the value function
$W_{Vi+1} = \left(\int_{\Omega} \phi(x_k)\phi(x_k)^T dx \right)^{-1} \int_{\Omega} \phi(x_k) \hat{V}_{i+1}(\phi(x_k), W_{Vi}) dx$
$\hat{V}_i(x_k, W_{Vi}) = W_{Vi}^T \phi(x_k)$
$i \rightarrow i+1$
No
$\left| \hat{V}_{i+1} - \hat{V}_i \right| < \varepsilon$
Yes
Finish

Fig. 1. Flow chart shows the proposed algorithm

$$u_i(x_k) = -(R + B^T P_i B)^{-1} B^T P_i A x_k \tag{35}$$

Note that this equation requires the full knowledge of both the internal dynamics matrix A and the control weighting matrix B . However, we have just seen (see remark above) that the knowledge of the A matrix can be avoided by using, instead of the action update(35), a second NN for the action

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

In fact the action NN approximates the effects of A and B given in (35), and so effectively learns the A matrix.

That is, using two NN even in the LQR case avoids the need to know the internal dynamics *A*. In fact, in the next section we give a LQR example, and only the input coupling matrix *B* is needed for the HDP algorithm. Nevertheless, the HDP converges to the correct LQR Riccati solution matrix *P*.

6. Simulation examples

In this section, two examples are provided to demonstrate the solution of the DT HJB equation. The first example will be a linear quadratic regulator, which is a special case of the nonlinear system. It is shown that using two NN allows one to compute the optimal value and control (i.e. the Riccati equation solution) online *without knowing the system matrix A* . The second example is for a DT nonlinear system. MATLAB is used in the simulations to implement some of the functions discussed in the chapter.

6.1 Unstable multi-input linear system example

In this example we show the power of the proposed method by using an unstable multi-input linear system. We also emphasize that the method does not require knowledge of the system A matrix, since two neural networks are used, one to provide the action. = This is in contrast to normal methods of HDP for linear quadratic control used in the literature, where the A matrix is needed to update the control policy.
Consider the linear system

$$x_{k+1} = Ax_k + Bu_k \ . \tag{36}$$

It is known that the solution of the optimal control problem for the linear system is quadratic in the state and given as

$$V^*(x_k) = x_k^T P x_k$$

where P is the solution of the ARE. This example is taken from (Stevens & Lewis, 2003), a linearized model of the short-period dynamics of an advanced (CCV-type) fighter aircraft. The state vector is

$$x = [\alpha \quad q \quad \gamma \quad \delta_e \quad \delta_f]^T$$

where the state components are, respectively, angel of attack, pitch rate, flight-path, elevator deflection and flaperon deflection. The control input are the elevator and the flaperon and given as

$$u = [\delta_{ec} \quad \delta_{fc}]^T$$

The plant model is a discretized version of a continuous-time model given in (Bradtke & Ydestie, 1994)]

$$A = \begin{bmatrix} 1.0722 & 0.0954 & 0 & -0.0541 & -0.0153 \\ 4.1534 & 1.1175 & 0 & -0.8000 & -0.1010 \\ 0.1359 & 0.0071 & 1.0 & 0.0039 & 0.0097 \\ 0 & 0 & 0 & 0.1353 & 0 \\ 0 & 0 & 0 & 0 & 0.1353 \end{bmatrix}$$

$$B=\begin{bmatrix} -0.0453 & -0.0175 \\ -1.0042 & -0.1131 \\ 0.0075 & 0.0134 \\ 0.8647 & 0 \\ 0 & 0.8647 \end{bmatrix}$$

Note that system is not stable and with two control inputs. The proposed algorithm does not require a stable initial control policy. The ARE solution for the given linear system is

$$P=\begin{bmatrix} 55.8348 & 7.6670 & 16.0470 & -4.6754 & -0.7265 \\ 7.6670 & 2.3168 & 1.4987 & -0.8309 & -0.1215 \\ 16.0470 & 1.4987 & 25.3586 & -0.6709 & 0.0464 \\ -4.6754 & -0.8309 & -0.6709 & 1.5394 & 0.0782 \\ -0.7265 & -0.1215 & 0.0464 & 0.0782 & 1.0240 \end{bmatrix} \tag{37}$$

and the optimal control $u_k^* = Lx_k$, where L is

$$L=\begin{bmatrix} -4.1136 & -0.7170 & -0.3847 & 0.5277 & 0.0707 \\ -0.6315 & -0.1003 & 0.1236 & 0.0653 & 0.0798 \end{bmatrix} \tag{38}$$

For the LQR case the value is quadratic and the control is linear. Therefore, we select linear activation functions for the action NN and quadratic polynomial activations for the critic NN. The control is approximated as follows

$$\hat{u}_i = W_{ui}^T \sigma(x_k) \tag{39}$$

where W_u is the weight vector, and the $\sigma(x_k)$ is the vector activation function and is given by

$$\sigma^T(x) = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 \end{bmatrix}$$

and the weights are

$$W_u^T = \begin{bmatrix} w_u^{1,1} & w_u^{1,2} & w_u^{1,3} & w_u^{1,4} & w_u^{1,5} \\ w_u^{2,1} & w_u^{2,2} & w_u^{2,3} & w_u^{2,4} & w_u^{2,5} \end{bmatrix}$$

The control weights should converge to

$$\begin{bmatrix} w_u^{1,1} & w_u^{1,2} & w_u^{1,3} & w_u^{1,4} & w_u^{1,5} \\ w_u^{2,1} & w_u^{2,2} & w_u^{2,3} & w_u^{2,4} & w_u^{2,5} \end{bmatrix} = -\begin{bmatrix} L_{11} & L_{12} & L_{13} & L_{14} & L_{15} \\ L_{21} & L_{22} & L_{23} & L_{24} & L_{25} \end{bmatrix}$$

The approximation of the value function is given as

$$\hat{V}_i(x_k, W_{Vi}) = W_{Vi}^T \phi(x_k)$$

where W_V is the weight vector of the neural network given by

$$W_V^T = \begin{bmatrix} w_v^1 & w_v^2 & w_v^3 & w_v^4 & w_v^5 & w_v^6 & w_v^7 & w_v^8 & w_v^9 & w_v^{10} & w_v^{11} & w_v^{12} & w_v^{13} & w_v^{14} & w_v^{15} \end{bmatrix}$$

and $\phi(x_k)$ is the vector activation function given by

$$\phi^T(x) = \begin{bmatrix} x_1^2 & x_1x_2 & x_1x_3 & x_1x_4 & x_1x_5 & x_2^2 & x_2x_3 & x_4x_2 & x_2x_5 & x_3^2 & x_3x_4 & x_3x_5 & x_4^2 & x_4x_5 & x_5^2 \end{bmatrix}$$

In the simulation the weights of the value function are related to the P matrix given in (37) as follows

$$\begin{bmatrix} P_{11} & P_{12} & P_{13} & P_{14} & P_{15} \\ P_{21} & P_{22} & P_{23} & P_{24} & P_{25} \\ P_{31} & P_{32} & P_{33} & P_{34} & P_{35} \\ P_{41} & P_{42} & P_{43} & P_{44} & P_{45} \\ P_{51} & P_{52} & P_{53} & P_{54} & P_{55} \end{bmatrix} = \begin{bmatrix} w_v^1 & 0.5w_v^2 & 0.5w_v^3 & 0.5w_v^4 & 0.5w_v^5 \\ 0.5w_v^2 & w_v^6 & 0.5w_v^7 & 0.5w_v^8 & 0.5w_v^9 \\ 0.5w_v^3 & 0.5w_v^7 & w_v^{10} & 0.5w_v^{11} & 0.5w_v^{12} \\ 0.5w_v^4 & 0.5w_v^8 & 0.5w_v^{11} & w_v^{13} & 0.5w_v^{14} \\ 0.5w_v^5 & 0.5w_v^9 & 0.5w_v^{12} & 0.5w_v^{14} & w_v^{15} \end{bmatrix}$$

The value function weights converge to

$$W_V^T = [55.5411 \quad 15.2789 \quad 31.3032 \quad -9.3255 \quad -1.4536 \quad 2.3142 \quad 2.9234 \quad -1.6594 \quad -0.2430$$
$$24.8262 \quad -1.3076 \quad 0.0920 \quad 1.5388 \quad 0.1564 \quad 1.0240]$$

The control weights converge to

$$W_u = \begin{bmatrix} 4.1068 & 0.7164 & 0.3756 & -0.5274 & -0.0707 \\ 0.6330 & 0.1005 & -0.1216 & -0.0653 & -0.0798 \end{bmatrix}$$

Note that the value function weights converge to the solution of the ARE (37), also the control weights converge to the optimal policy (38) as expected.

6.2 Nonlinear system example

Consider the following affine in input nonlinear system

$$x_{k+1} = f(x_k) + g(x_k)u_k \tag{40}$$

where

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

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

$$\phi(x) = [x_1^2 \quad x_1x_2 \quad x_2^2 \quad x_1^4 \quad x_1^3x_2$$
$$x_1^2x_2^2 \quad x_1x_2^3 \quad x_2^4 \quad x_1^6 \quad x_1^5x_2 \quad x_1^4x_2^2$$
$$x_1^3x_2^3 \quad x_1^2x_2^4 \quad x_1x_2^5 \quad x_2^6]$$

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 (34). The critic activations are selected as polynomials to satisfy $\hat{V}_i(x=0)=0$ at each step. Note that then automatically one has $\hat{u}_i(x=0)=0$ as required for admissibility. We decided on 6th order polynomials for VFA after a few simulations, where it came clear that 4th order polynomials are not good enough, yet going to 8th order does not improve the results.
The result of the algorithm is compared to the discrete-time State Dependent Riccati Equation (SDRE) proposed in (Cloutier, 1997).
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 chapter is compared to the SDRE approach. Figure 2 and Figure 3 show the states trajectories for the system for both methods.
In Figure 4, the cost function of the SDRE solution and the cost function of the proposed algorithm in this chapter 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.

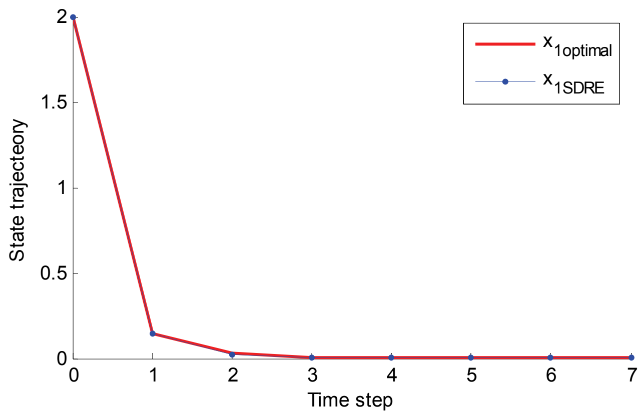


Fig. 2. The state trajectory for both methods

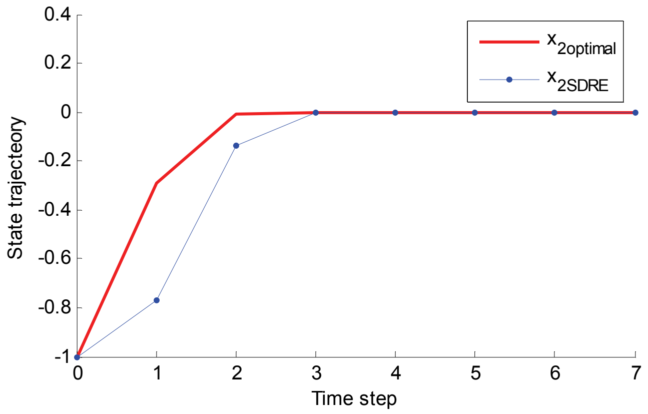


Fig. 3. The state trajectory for both methods

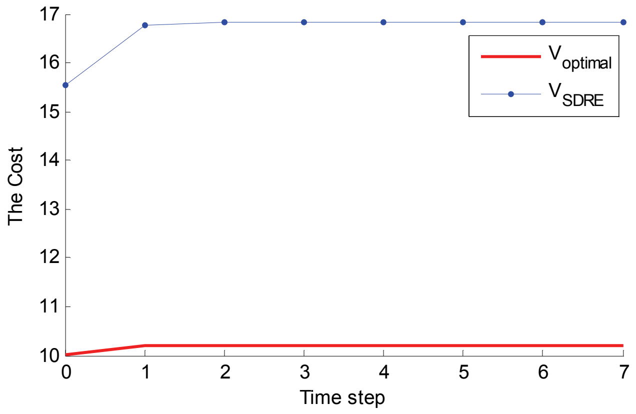


Fig. 4. The cost function for both methods

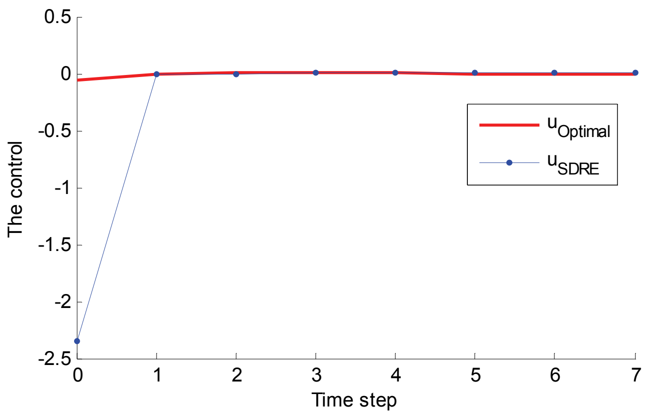


Fig. 5. The control signal input for both methods

7. Conclusion

We have proven convergence of the HDP algorithm to the value function solution of Hamilton-Jacobi-Bellman equation for 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. It is stressed that 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, where use of two NN avoids the need to know the system internal dynamics matrix A. This is not generally appreciated in the folkloric literature of ADP for the LQR. In the simulation examples, it is shown that the linear system critic network converges to the solution of the ARE, and the actor network converges to the optimal policy, without knowing the system matrix A. In the 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.

8. References

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

Abu-Khalaf, M., F. L. Lewis, and J. Huang. (2004).Hamilton-Jacobi-Isaacs formulation for constrained input nonlinear systems. *43rd IEEE Conference on Decision and Control*, 2004, pp. 5034 - 5040 Vol.5.

Al-Tamimi, A. , F. L. Lewis, M. Abu-Khalaf (2007). Model-Free Q-Learning Designs for Discrete-Time Zero-Sum Games with Application to H-Infinity Control. *Automatica*, volume 43, no. 3. pp 473-481.

Al-Tamimi, A., M. Abu-Khalaf, F. L. Lewis. (2007). 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 240-24.

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

Bertsekas, D.P. and J. N. Tsitsiklis.(1996). Neuro-Dynamic Programming. *Athena Scientific*, MA.

Bradtke, S. J., B. E. Ydestie, A. G. Barto (1994). Adaptive linear quadratic control using policy iteration. *Proceedings of the American Control Conference* , pp. 3475-3476, Baltmore, Myrland.

Chen, Z., Jagannathan, S. (2005). Neural Network -based Nearly Optimal Hamilton-Jacobi-Bellman Solution for Affine Nonlinear Discrete-Time Systems. *IEEE CDC 05* ,pp 4123-4128.

Cloutier, J. R. (1997). State –Dependent Riccati equation Techniques: An overview. *Proceeding of the American control conference*, Albuquerque, NM, pp 932-936.

Ferrari, S., Stengel, R.(2004) 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.

Finlayson, B. A. (1972). The *Method of Weighted Residuals and Variational Principles*. Academic Press, New York.

Hagen, S. B Krose. (1998). Linear quadratic Regulation using Reinforcement Learning. *Belgian_Dutch Conference on Mechanical Learning*, pp. 39-46.

He, P. and S. Jagannathan.(2005).Reinforcement learning-basedoutput feedback control of nonlinear systems with input constraints. *IEEE Trans. Systems, Man, and Cybernetics -Part B:Cybernetics*, vol. 35, no.1, pp. 150-154.

Hewer, G. A. (1971). An iterative technique for the computation of the steady state gains for the discrete optimal regulator. *IEEE Trans. Automatic Control*, pp. 382-384.

Hornik, K., M. Stinchcombe, H. White.(1990) .Universal Approximation of an Unknown Mapping and Its Derivatives Using Multilayer Feedforward Networks. *Neural Networks*, vol. 3, pp. 551-560.

Howard, R. (1960). *Dynamic Programming and Markov Processes.*, MIT Press, Cambridge, MA.

Huang, J. (1999). An algorithm to solve the discrete HJI equation arising in the L_2-gain optimization problem. INT. J. Control, Vol 72, No 1, pp 49-57.

Kwon, W. H and S. Han. (2005). *Receding Horizon Control*, Springer-Verlag, London.

Lancaster, P. L. Rodman. (1995). *Algebraic Riccati Equations.* Oxford University Press, UK.

Landelius, T. (1997). *Reinforcement Learning and Distributed Local Model Synthesis.* PhD Dissertation, Linkoping University, Sweden.

Lewis, F. L., V. L. Syrmos. (1995) Optimal Control, 2nd ed., John Wiley.

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

Lin W.,and C. I. Byrnes.(1996).H_∞Control of Discrete-Time Nonlinear System. *IEEE Trans. on Automat. Control* , vol 41, No 4, pp 494-510..

Lu, X., S.N. Balakrishnan. (2000). Convergence analysis of adaptive critic based optimal control. *Proc. Amer. Control Conf.*, pp. 1929-1933, Chicago.

Morimoto, J., G. Zeglin, and C.G. Atkeson. (2003). Minimax differential dynamic programming: application to a biped walking robot. *Proc. IEEE Int. Conf. Intel. Robots and Systems*, pp. 1927-1932, Las Vegas.

Murray J., C. J. Cox, G. G. Lendaris, and R. Saeks.(2002).Adaptive Dynamic Programming. *IEEE Trans. on Sys., Man. and Cyb.*, Vol. 32, No. 2, pp 140-153.

Narendra, K.S. and F.L. Lewis. (2001).Special Issue on Neural Network feedback Control. Automatica, vol. 37, no. 8.

Prokhorov, D., D. Wunsch. (1997). Adaptive critic designs. *IEEE Trans. on Neural Networks*, vol. 8, no. 5, pp 997-1007.

Prokhorov, D., D. Wunsch (1997). Convergence of Critic-Based Training. *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, vol. 4, pp. 3057—3060.

Si, J. and Wang. (2001). On-Line learning by association and reinforcement. *IEEE Trans. Neural Networks*, vol. 12, pp.264-276.

Si, Ji. A. Barto, W. Powell, D. Wunsch.(2004). *Handbook of Learning and Approximate Dynamic Programming.* John Wiley, New Jersey.

Stevens B., F. L. Lewis. (2003). Aircraft Control and Simulation, 2nd edition, John Wiley, New Jersey.

Sutton, R. S., A. G. Barto. (19998). *Reinforcement Learning, MIT Press.* Cambridge, MA .

Watkins, C.(1989). *Learning from Delayed Rewards.* Ph.D. Thesis, Cambridge University, Cambridge, England.

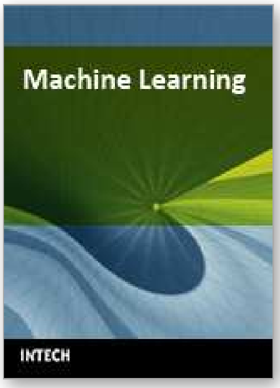
Werbos, P. J. (1991). A menu of designs for reinforcement learning over time. , *Neural Networks for Control*, pp. 67-95, ed. W.T. Miller, R.S. Sutton, P.J. Werbos, Cambridge: MIT Press.

Werbos, P. J. (1992). *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,.

Werbos, P. J. (1990). Neural networks for control and system identification. *Heuristics,* Vol. 3, No. 1, pp. 18-27.

Widrow, B., N. Gupta, and S. Maitra. (1973). Punish/reward: Learning with a critic in adaptive threshold systems. *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-3, pp. 455–465.

Xi-Ren Cao. (2001). Learning and Optimization—From a Systems Theoretic Perspective. *Proc. of IEEE Conference on Decision and Control*, pp. 3367-3371.



Machine Learning

Edited by Abdelhamid Mellouk and Abdennacer Chebira

ISBN 978-953-7619-56-1

Hard cover, 450 pages

Publisher InTech

Published online 01, January, 2009

Published in print edition January, 2009

Machine Learning can be defined in various ways related to a scientific domain concerned with the design and development of theoretical and implementation tools that allow building systems with some Human Like intelligent behavior. Machine learning addresses more specifically the ability to improve automatically through experience.