WCCI 2012 IEEE World Congress on Computational Intelligence
June, 10-15, 2012 - Brisbane, Australia IJCNN

Neural and Fuzzy Dynamic Programming for Under-actuated Systems

Dongbin Zhao, Yuanheng Zhu
State Key Laboratory of Management and Control for Complex Systems
Institute of Automation, Chinese Academy of Sciences
Beijing 100190, China
e-mail: dongbin.zhao@gmail.com, zyh7716155@163.com

Haibo He
Department of Electrical, Computer and Biomedical Engineering
University of Rhode Island
Kingston, RI 02881, USA
e-mail: he@ele.uri.edu

Abstract—**This paper aims to integrate the fuzzy control with adaptive dynamic programming (ADP) scheme, to provide an optimized fuzzy control performance, together with faster convergence of ADP for the help of the fuzzy prior knowledge. ADP usually consists of two neural networks, one is the Actor as the controller, the other is the Critic as the performance evaluator. A fuzzy controller applied in many fields can be used instead as the Actor to speed up the learning convergence, because of its simplicity and prior information on fuzzy membership and rules. The parameters of the fuzzy rules are learned by ADP scheme to approach optimal control performance. The feature of fuzzy controller makes the system steady and robust to system states and uncertainties. Simulations on under-actuated systems, a cart-pole plant and a pendubot plant, are implemented. It is verified that the proposed scheme is capable of balancing under-actuated systems and has a wider control zone.**

Keywords-adaptive dynamic programming; fuzzy control; nerual network; underactutaed sysmem; cart-pole plant; pendubot;

I. INTRODUCTION

For many years, a lot of applications in engineering, economics, operation and other fields have implemented dynamic programming, which is really a great method solving nonlinear stochastic dynamic problems [1~4]. The Bellman equation, developed by Bellman in 1953, makes the foundation of optimization over time in the stochastic case [5]. The main concept of adaptive dynamic programming (ADP) is to develop a parameterized model which is trained to approximate the Bellman J function (the "value function"), and train the controller to minimize the function J [6]. In general, the ADP algorithm is divided into two parts, the "Critic" and the "Actor", separately served as an approximator to J and a system controller.

Fuzzy control has played a key role in both the area of research and the practical applications of industrial processes. Regardless of accurate input-output relations, fuzzy control is much like human thinking and natural language, which is distinctive to conventional methods. Based on fuzzy logic, the fuzzy controller is proposed as a method to convert "a linguistic control strategy based on expert knowledge into an automatic control strategy" [7]. Fuzzy control rules are the core of fuzzy control, which can be described in the following form:

IF x is A and y is B, THEN z is C,

where x, y and z are fuzzy control variables, and A, B and C are the fuzzy values in the universe of discourses X, Y and Z, respectively [8]. A typical fuzzy control system is shown in Fig.1. In most cases, the parameters of fuzzy IF-THEN rules are provided according to the experience and knowledge of human experts. If the provided fuzzy rules have bad control performance, then an adaptive law is added to update the parameters, which is also called "adaptive fuzzy control" [9].

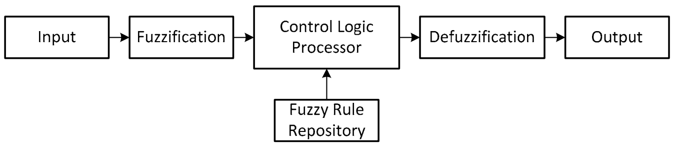

Figure 1. A typical schematic diagram of fuzzy control system

The above mentioned methods both had their own superior features, it was worthy to combine fuzzy control and ADP together to solve kinds of problems [10~13]. In this paper, we use the ADP algorithm to train the fuzzy rules, which are initialized randomly and have no prior knowledge about the controlled object. Finally, those trained rules keep the fuzzy controller playing a really great performance. For simulations, under-actuated systems are adopted as the control objective to observe the performance of the proposed method. In Section II, we present a general description of the ADP with fuzzy controller. In Section III, the method is applied to a cart-pole plant to observe the learning performance. In Section IV, a more complex system, pendubot, is simulated using the new method, and some discussions are presented. In the end, section V gives some conclusions.

II. ADAPTIVE DYNAMIC PROGRAMMING WITH FUZZY CONTROLLER

A. The Mechanism of ADP

In [14] and [15], ADP approaches are classified into several categories: "heuristic dynamic programming (HDP), action-dependent HDP (ADHDP; note the prefix "action-dependent" (AD) used hereafter), also known as Q-learning, dual heuristic dynamic programming (DHP), ADDHP, globalized DHP (GDHP), and ADGDHP". In our

This work was supported partly by National Natural Science Foundation of China (Nos. 60874043, 60921061, 61034002), and Beijing Natural Science Foundation (No. 4122083)
This work was also supported in part by the National Science Foundation (NSF) under Grant ECCS 1053717

research, the adopted approach proposed by Si and Wang in [16], is closely related to ADHDP. Fig. 2 is a schematic diagram of the ADP algorithm.

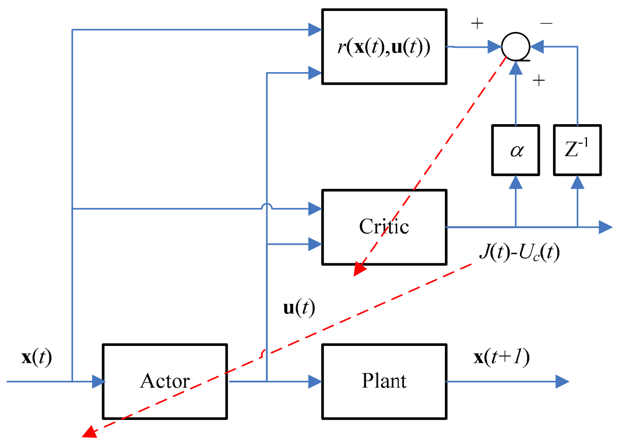

Figure 2. A schematic diagram of ADP. Signal flow is transmitted along the solid lines while parameters are trained along the dashed lines.

In this paper, the Actor is a fuzzy controller, and the Critic is a network with the input of states $X(t)$ and action $u(t)$. The reinforcement signal $r(t)$ is either a "0" or a "-1" corresponding to "success" or "failure", respectively. The output of the Critic is expected to estimate the cost-to-go $J(t)$ in the Bellman equation. For the Critic, the prediction error and the objective function are defined as

$$e_c(t) = \alpha J(t) - \left[J(t-1) - r(t)\right] \tag{1}$$

$$E_c(t) = \frac{1}{2}e_c^2(t). \tag{2}$$

where α is a discount factor for the infinite-horizon problem $(1 < \alpha < 1)$ which is set 0.95 in this paper.

The objective of training the Critic network is to reduce $E_c(t)$ close to zero, then following equation can be derived:

$$J(t) = \sum_{k=t+1}^{\infty} \alpha^{k-t-1} r(k) \tag{3}$$

The above equation is exactly the form as that defined in the Bellman equation. During the training process, many methods can be applied such as gradient descent method (GD) [16] and particle swarm optimization (PSO) [17] which are both adopted in this paper for comparison.

Similar to the Critic, the Actor is expected to provide optimal action signal u(t) which is to minimize the error between the desired ultimate objective, denoted by U_c , and the function J. According to our above definition, U_c is set "0" for "success". So the error $e_a(t)$ and the performance error $E_a(t)$ are defined as

$$e_a(t) = J(t) \tag{4}$$

$$E_a(t) = \frac{1}{2}e_a^2(t) \tag{5}$$

Likewise, the Actor is trained to minimize $E_a(t)$ so that both gradient descent and PSO are adopted respectively for comparison.

The whole process of the ADP algorithm is summarized as follows. Firstly, the Actor and the Critic are both initialized randomly. Then the Actor generates an action signal $u(t)$ based on states $X(t)$ and the function $J(t)$ is calculated by the Critic, with the input of $u(t)$ and $X(t)$. Combined with the previous $J(t$-1) and the reinforcement signal $r(t)$, $e_c(t)$ and $E_c(t)$ are derived. Then the Critic is adapted according to $E_c(t)$ with the gradient descent method or PSO, or some other methods, until meeting some criterion. Afterwards, the Actor is modified with $E_a(t)$ in the same way. The next states $X(t$+1) are calculated from the system with the action signal $u(t)$ generated by the modified Actor. Then, the iteration process continues for the next cycle [18~19].

B. The Fuzzy Controller

The fuzzy controller is adopted with the simplest and most common one, which is composed of membership functions and fuzzy rules. In this paper, the membership functions for each variable are two Monotonics, representing "Positive" and "Negative" respectively, as shown in Fig. 3. During the learning process, the shape of the membership is fixed, and only the fuzzy rules are trained.

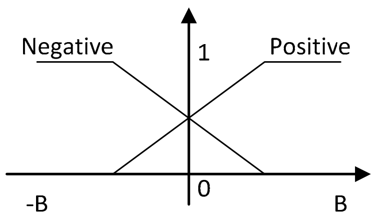

Figure 3. Monotonic membership functions

Suppose there are n input variables, and after fuzzification, the fuzzy variables of the input are calculated as

$$\begin{aligned} \mu_{i,N} &= \begin{cases} 1, & X_i < -B_i \\ \dfrac{B_i - X_i}{2B_i}, & -B_i < X_i < B_i \\ 0, & B_i < X_i \end{cases} \\ \mu_{i,P} &= \begin{cases} 0, & X_i < -B_i \\ \dfrac{B_i + X_i}{2B_i}, & -B_i < X_i < B_i \\ 1, & B_i < X_i \end{cases} \end{aligned} \tag{6}$$

where i =1⋯n and B_i is the boundary of the ith membership function. Then, for each fuzzy control rules, the weights are calculated as

$$\omega_r = \prod_{i=1}^{n} \mu_{i,j_i} \tag{7}$$

where $j_i = N$ or P, $r = 1 \cdots 2^n$. With fuzzy control rules R, the output of fuzzy controller $u(t)$ is generated by

$$u(t) = \sum_{r=1}^{2^n} \omega_r \cdot R_r \tag{8}$$

The generated output signal $u(t)$ usually is a normalized one. In some cases, $u(t)$ is multiplied by a gain K_s to provide a exaggerated continuous signal on the plant. While in other cases, $u(t)$ is transformed to a signed constant signal which is also called a bang-bang control strategy. In following sections, those two control strategies are both adopted.

C. The Critic Network

The network in this paper has the same structure as in [16~20] with one hidden layer, which is shown in Fig. 4.

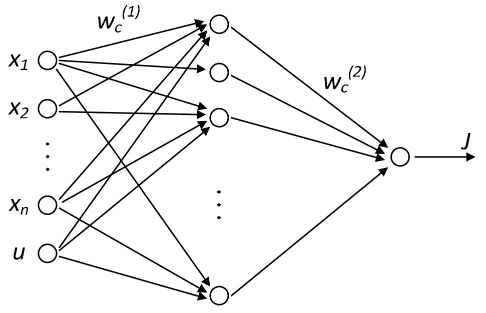

Figure 4. A nonlinear critic network with one hidden layer [15]

In the Critic network, the output $J(t)$ is formulated by:

$$J(t) = \sum_{i=1}^{N_h} \omega_{c_i}^{(2)}(t) p_i(t) \tag{9}$$

$$p_i(t) = \frac{1 - \exp^{-q_i(t)}}{1 + \exp^{-q_i(t)}}, \qquad i = 1, \cdots, N_h \tag{10}$$

$$q_i(t) = \sum_{j=1}^{n+1} \omega_{c_{ij}}^{(1)}(t) x_j(t), \qquad i = 1, \cdots, N_h \tag{11}$$

where w_c is the weight vector; q_i is the input of the ith hidden node; p_i is the output of the ith hidden node; N_h is the number of hidden nodes; $n+1$ is the number of the input into the Critic network.

D. Training with Gradient Descent (GD) Method

For GD method, the adaptation of the Critic is summarized as follows.

$$\begin{aligned}\Delta\omega_{c_i}^{(2)}(t) &= l_c(t)\left[-\frac{\partial E_c(t)}{\partial \omega_{c_i}^{(2)}(t)}\right] \\ &= -l_c(t)\frac{\partial E_c(t)}{\partial J(t)}\frac{\partial J(t)}{\partial \omega_{c_i}^{(2)}(t)} = -l_c(t)\alpha e_c(t) p_i(t).\end{aligned} \tag{12}$$

$$\begin{aligned}\Delta\omega_{c_{ij}}^{(1)}(t) &= l_c(t)\left[-\frac{\partial E_c(t)}{\partial \omega_{c_{ij}}^{(1)}(t)}\right] \\ &= -l_c(t)\frac{\partial E_c(t)}{\partial J(t)}\frac{\partial J(t)}{\partial p_i(t)}\frac{\partial p_i(t)}{\partial q_i(t)}\frac{\partial q_i(t)}{\partial \omega_{c_{ij}}^{(1)}(t)} \\ &= -l_c(t)\alpha e_c(t)\omega_{c_i}^{(2)}(t)\left[\frac{1}{2}\left(1 - p_i^2(t)\right)\right]x_j(t).\end{aligned} \tag{13}$$

The adaptation of the Actor (fuzzy controller) is as follows:

$$\begin{aligned}\Delta R_r(t) &= l_a(t)\left[-\frac{\partial E_a(t)}{\partial R_r(t)}\right] \\ &= -l_a(t)\frac{\partial E_a(t)}{\partial J(t)}\frac{\partial J(t)}{\partial u(t)}\frac{\partial u(t)}{\partial R_r(t)} \\ &= -l_a(t)e_a(t)\omega_r\sum_{i=1}^{N_h}\left[\omega_{c_i}^{(2)}(t)\frac{1}{2}\left(1 - p_i^2(t)\right)\omega_{c_{i,n+1}}^{(1)}(t)\right].\end{aligned} \tag{14}$$

So, the new weights of the Critic and the fuzzy control rules are updated as:

$$\omega_c(t+1) = \omega_c(t) + \Delta\omega_c(t) \tag{15}$$

$$R(t+1) = R(t) + \Delta R(t) \tag{16}$$

E. Training with Particle Swarm Optimization (PSO) Method

For the PSO algorithm, the whole learning process is summarized as follows.

1) Initialize particles, including x_{id} (position of the current particle), p_{id} (the best solution for each particle ever achieved), p_{gd} (a global best solution for all particles ever achieved);

2) Calculate the fitness for each particle x_{id};

3) Update the best fitness solution p_{id} for each particle;

4) Update the global best fitness solution p_{gd} for all particles;

5) Calculate particle velocity v_{id} and update position x_{id} of each particle;

6) The iteration process continues for the next cycle until maximum iterations or some criterions are met.

The mentioned updating formulas for x_{id} are shown as follows:

$$v_{id} = \omega v_{id} + c_1 r_1 (p_{id} - x_{id}) + c_2 r_2 (p_{gd} - x_{id}) \tag{17}$$

$$x_{id} = x_{id} + v_{id} \tag{18}$$

$$\omega = \omega_{\min} + iter / iter_{\max} \left(\omega_{\max} - \omega_{\min} \right) \tag{19}$$

where c_1 and c_2 represent learning factors, r_1 and r_2 are random numbers between (0,1), ω_{min} and ω_{max} define the minimum and maximum values for the inertia weight, *iter* represents the iteration process steps. The fitness here is adopted as $\exp[-E_a(t)]$ and $\exp[-E_c(t)]$. x_{id} consists of the parameters needed to be learned, such as w_c^1 and w_c^2 or R_r.

After learning, normalization is necessary to avoid the parameters being updated beyond the scope for both methods.

III. ADP WITH FUZZY CONTROL FOR A CART-POLE PLANT

A. The Cart-Pole Balancing Problem

The proposed method combined ADP with fuzzy control is now simulated on a single cart-pole plant. The plant is a single pole mounted on a cart, and the objective of the ADP algorithm is to train the fuzzy controller from initially random rules to balance the pole. The cart-pole plant used here can also be found in [16] and [20], shown in Fig. 5, and the model is summarized in the following.

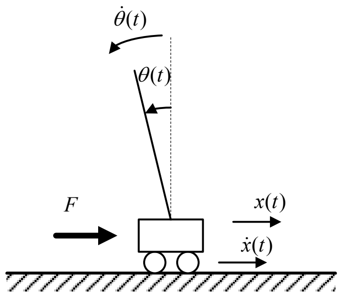

Figure 5. Schematic diagram of the single cart-pole plant

$$\frac{d^2\theta}{d^2t} = \frac{g\sin\theta + \cos\theta\left[-F - ml\dot{\theta}^2\sin\theta + \mu_c\,\mathrm{sgn}(\dot{x})\right] - \dfrac{\mu_p\dot{\theta}}{ml}}{l\left(\dfrac{4}{3} - \dfrac{m\cos^2\theta}{m_c + m}\right)} \tag{20}$$

$$\frac{d^2x}{d^2t} = \frac{F + ml\left[\dot{\theta}^2\sin\theta - \ddot{\theta}\cos\theta\right] - \mu_c\,\mathrm{sgn}(\dot{x})}{m_c + m} \tag{21}$$

where

g	9.8 m/s^2, acceleration due to gravity;
m_c	1.0 kg, mass of cart;
m	0.1 kg, mass of pole;
l	0.5 m, half-pole length;
μ_c	0.0005, coefficient of friction of cart on track;
μ_p	0.000 002, coefficient of friction of pole on cart;

$$\mathrm{sgn}(x) = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{if } x = 0 \\ -1, & \text{if } x < 0. \end{cases}$$

The system states $X(t)$ are constituted of four variables: 1) $x(t)$, position of the cart with the reference to the center of the track; 2) $\theta(t)$, the angle of the pole with the reference to the vertical position; 3) $\dot{x}(t)$, the velocity of the cart; 4) $\dot{\theta}(t)$, the angular velocity of the pole.

During the trials, the controller is considered a failure when $\theta(t)$ or $x(t)$ are out of the predefined scopes. At that moment, the learning process stops and a new trial restarts. So the reinforcement signal is defined as

$$r = \begin{cases} 0 & \text{If } -12° < \theta < 12° \text{ and } -1 < x < 1, \\ -1 & \text{Otherwise.} \end{cases} \tag{22}$$

Here, two control strategies are implemented; respectively bang-bang control with a constant force $\pm 10N$ and continuous control with a conversion gain $K_s = 40$ from u to F.

B. Simulation Results

In this simulation, the GD method is applied to train the Critic network weights and fuzzy rules, while the PSO method has similar performance and is omitted here. Some parameters are defined before the simulations. In ADP algorithm, the learning rate of both the Critic $l_c(t)$ and the Actor $l_a(t)$ is 0.005; the maximum adapting cycles for the Critic N_c and the Actor N_a are 50 and 100, respectively; the training error threshold T_c and T_a are 0.05 and 0.005, respectively. The hidden nodes N_h of the critic network is 6; time step dt is 0.02s. Besides, boundaries B of four membership functions are [1m, 12°, 1.5m/s, 120°/s].

During simulations, there are at most 1000 consecutive trials in a run. If any of the 1000 trials has lasted 6000 steps, it is considered successful and the run stops. Otherwise, if the fuzzy controller still fails after 1000 trials, the run is unsuccessful.

Both bang-bang and continuous control strategies are implemented for 100 runs to calculate their success rates and average trials to success. To be more realistic, sensor noise is added to the state measurements. Two kinds of noise, uniform and Gaussian noise are added to the angle measurements $\theta(t)$. The results are shown in Table I. Besides, [16] presented some simulation results which were generated using a neural network controller with GD method and bang-bang control strategy under the same conditions of our simulation. The comparisons to the proposed fuzzy controller with bang-bang control strategy are listed in Table II.

From the results, it is obvious that the method we proposed provides a 100% success rate on the cart-pole system even the measurements are disturbed by noise. The added noise impacts the number of trials to success. The more noise is added, the more trials are needed. For the cart-pole plant, the bang-bang control strategy is easier and faster to train controller than the continuous strategy, with the fact that bang-bang control is simpler and easier to balance the pole. Besides, with the comparison of fuzzy and neural network controller results, it is obvious that when the noise is small, neural network controller is easier to learn, while at large noise, fuzzy controller shows an advantage with much less training trials. The fuzzy controller shows a perfect robustness with the fact that the success trials are more insusceptible by noise disturbance than neural network controller.

TABLE I. COMPARISON OF TWO CONTROL STRATEGIES USING GD METHODS FOR CART-POLE PLANT WITH DIFFERENT KINDS OF NOISE BY FUZZY CONTROLLER

Noise type	**Bang-bang**		**Continuous**	
	Success rate	***# of trials***	***Success rate***	***# of trials***
Noise free	100%	30.42	100%	69.67
Uniform 5%	100%	37.42	100%	69.14
Uniform 10%	100%	41.79	100%	73.06
Gaussian$\sigma^2 = 0.1$	100%	45.65	100%	84.54
Gaussian$\sigma^2 = 0.2$	100%	53.28	100%	97.00

TABLE II. COMPARISON OF FUZZY AND NEURAL NETWORK CONTROLLER USING THE SAME BANG-BANG CONTROL STRATEGY AND GD METHOD FOR CART-POLE PLANT WITH DIFFERENT KINDS OF NOISE

Noise type	**Fuzzy+Bang-bang**		**NN+Bang-bang[16]**	
	Success rate	***# of trials***	***Success rate***	***# of trials***
Noise free	100%	30.42	100%	**6**
Uniform 5%	100%	37.42	100%	**32**
Uniform 10%	100%	**41.79**	100%	54
Gaussian$\sigma^2 = 0.1$	100%	**45.65**	100%	164
Gaussian$\sigma^2 = 0.2$	100%	**53.28**	100%	193

IV. ADP WITH FUZZY CONTROL FOR A PENDUBOT PLANT

In this section, a more complex system, pendubot plant is further simulated.

A. The Pendubot Balancing Problem

Pendubot (**pendu**lum ro**bot**) is a two link under-actuated robotic system. Even though with a simple structure, it is featured as a high nonlinear system and is adopted to detect the control performance. Fig. 6 is a schematic diagram of a pendubot. Only the first joint is actuated by the external torque generated from controllers, while the second joint is passive. Under ideal conditions, suppose that there is no friction, the system model is formulated as [21~23].

$$\tau = D(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) \tag{23}$$

where $\tau = [\tau_1\ 0]^T$ is the external torque, $q = [q_1\ q_2]^T$ is the angles of two links. *D*, *C* and *G* represent the inertial, Coriolis, and gravity terms of the system, which also are described as the following.

$$D(q) = \begin{bmatrix} \theta_1 + \theta_2 + 2\theta_3 \cos q_2 & \theta_2 + \theta_3 \cos q_2 \\ \theta_2 + \theta_3 \cos q_2 & \theta_2 \end{bmatrix}, \tag{24}$$

$$C(q,\dot{q}) = \begin{bmatrix} -\theta_3 \dot{q}_2 \sin q_2 & -\theta_3(\dot{q}_2 + \dot{q}_1)\sin q_2 \\ \theta_3 \dot{q}_1 \sin q_2 & 0 \end{bmatrix}, \tag{25}$$

$$G(q) = \begin{bmatrix} -\theta_4 g \sin q_1 - \theta_5 g \sin(q_1 + q_2) \\ -\theta_5 g \sin(q_1 + q_2) \end{bmatrix}, \tag{26}$$

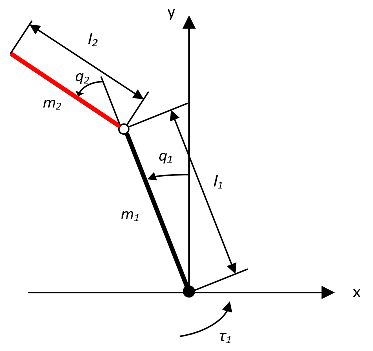

Figure 6. Schematic diagram of pendubot system.

where $\theta_1, \theta_2, \theta_3, \theta_4, \theta_5$ are described as:

$$\theta_1 = m_1 l_{c_1}^2 + m_2 l_1^2 + I_1,$$
$$\theta_2 = m_2 l_{c_2}^2 + I_2,$$
$$\theta_3 = m_2 l_1 l_{c_2},$$
$$\theta_4 = m_1 l_{c_1} + m_2 l_1,$$
$$\theta_5 = m_2 l_{c_2}.$$

The parameters here are adopted the same values as in [17, 23]:

$$\theta_1 = 0.0308$$
$$\theta_2 = 0.0306$$
$$\theta_3 = 0.0095$$
$$\theta_4 = 0.2087$$
$$\theta_5 = 0.0630$$

It is obvious that there are four equilibriums of the system. The objective is to balance the system about the equilibrium where two links are both at top position, namely $q = [0°, 0°]$. In the simulation, the reinforcement signal is defined as:

$$r = \begin{cases} 0 & \text{If } -40° < q_1 < 40° \text{ and } -12° < q_2 < 12°, \\ -1 & \text{Otherwise.} \end{cases} \tag{27}$$

The states consist of the two angles and their angular velocities, namely $X(t) = \{q_1, q_2, \dot{q_1}, \dot{q_2}\}$. Here, two control strategies are applied, respectively bang-bang control with a constant torque $\pm 0.5Nm$ and continuous control with a conversion gain $Ks = 5$ from *u* to τ.

B. Simulation Results

Both GD and PSO methods are adopted. Except for some parameters involved in the previous section, some others are defined here. In PSO algorithm, the particle number N_p is 30; the maximum value of ω, ω_{max} is 0.9; the minimum value ω_{min} is 0.4; the maximum value of v_{id}, v_{max} is 0.4; the learning factor c_1 and c_2 is 2. The boundaries of states are [50°, 20°, 360°/s, 360°/s]. Likewise, a run has 1000 consecutive trials and if any trial lasts 6000 steps, it is considered a success. The simulation results for two control strategies are listed in Table III. For comparison, in the third column, another simulation results are listed, which is generated using the neural network controller with the bang-bang control strategy and trained by the GD method [17].

TABLE III. COMPARISON OF FUZZY CONTROLLER WITH TWO CONTROL STRATEGIES AND TWO TRAINING METHODS AND THE NEURAL NETWORK CONTROLLER FOR PENDUBOT

Control strategy	**Fuzzy+GD**		**Fuzzy+PSO**		**NN+GD[17]**	
	Success rate	***# of trials***	***Success rate***	***# of trials***	***Success rate***	***# of trials***
Bang-bang control	**86%**	347.01	77%	334.93	30%	**240.3**
Continuous control	47%	508.34	**60%**	**380.73**	#	#

According to the results, for the fuzzy controller the continuous control strategy is much harder to learn how to balance the system with lower success rate and more training trials. The GD method needs more learning trials than the PSO method. For the bang-bang control, GD displays its advantage of higher success rate than PSO, while for continuous control the advantage is reverse. Besides, by comparing the results of Fuzzy+GD and Fuzzy+PSO to NN+GD, it is obvious that the both Fuzzy+GD and Fuzzy+PSO outperform NN+GD in the Bang-bang or continuous control strategies. The show higher success rates even though that they need more training trials. However, the fact that too many trials and lower success rate with GD method sets out our further research.

Because of the uncomplicated and simple structure of the fuzzy controller, the internal parameters of the controller are much less than other complex control methods, such as neural networks. For example, there are only 16 parameters of fuzzy rules in above systems to be learned while 30 parameters are trained in [15] and [16]. Besides, when the measurements are polluted by too much noise, the fuzzy controller shows a better robustness than neural network controller with less training trials.

V. CONCLUSIONS

This paper verifies that simple fuzzy controllers can be successfully combined with adaptive dynamic programming, and optimized by adaptive dynamic programming. On the other hand, with the prior knowledge of the fuzzy membership and rules, the learning convergence and successful rate of adaptive dynamic programming are improved greatly, compared to the traditional neural network method. The trained fuzzy controller also shows better performance on robustness. Both GD and PSO are provided for the training process. GD is suitable for continuous models which can generate error back propagation, while PSO is suitable for not only continuous models, but also the models which cannot generate error back propagation, like some fuzzy controllers using tabular look-up method. With the simplicity and easy structure of fuzzy control, the parameters involved are much less than other control methods. The feature of fuzzy control also presents its own superior advantage than other control methods with better control performance. But some other flaws still need further research. In this paper only fuzzy rules are trained by ADP algorithm and the performance is not good enough for complex under-actuated system.

REFERENCES

[1] R. Bellman and S. Dreyfus, Applied Dynamic Programming. Princeton, N. J.: Princeton Univ. Press, 1962.

[2] T. Borgers and R. Sarin, "Learning through reinforcement and replicator dynamics," J. Economic Theory, vol. 77, no. 1, pp. 1–17, 1997.

[3] J. Dalton and S. N. Balakrishnan, "A neighboring optimal adaptive critic for missile guidance," Math. Comput. Modeling, vol. 23, no. 1, pp. 175–188, 1996.

[4] J. N. Tsitsiklis and B. Van Roy, "An analysis of temporal-difference learning with function approximation," IEEE Trans. Automat. Contr., vol. 42, pp. 674–690, May 1997.

[5] P. J. Werbos, "Using adaptive dynamic programming to understand and replicate brain intelligence: The next level design," in Neurodynamics of Higher-Level Cognition and Consciousness, R. Kozma, Ed. Berlin, Germany: Springer-Verlag, 2007.

[6] P. J. Werbos, "Foreword - ADP: the key direction for future research in intelligent control and understanding brain intelligence," IEEE Trans. Syst., Man, Cybern., Part B: Cybern., vol. 38, no. 4, pp. 898–900, 2008.

[7] C.-C. Lee, "Fuzzy logic in control systems: fuzzy logic controller-Part I," IEEE Trans. Syst., Man, Cybern., vol. 20, pp. 404–418, 1990.

[8] A. Katbab, "Fuzzy logic and controller design-a review," in Proc. IEEE Southeastcon, Raleigh, NC, Mar. 1995, pp. 443-449.

[9] L.-X. Wang, "Stable adaptive fuzzy control of nonlinear systems" IEEE Trans. Fuzzy Syst., vol. I , pp. 146–155, Jan. 1993.

[10] T. Shannon, G. Lendaris, "Adaptive critic based approximate dynamic programming for tuning fuzzy controllers," in Proc. IEEE Int. Conf. Fuzzy Systems, San Antonio, May, 2000.

[11] G. G. Lendaris, R. A. Santiago, and M. S. Carroll. "Dual heuristic programming for fuzzy control," Proceeedings of IFSA / NAFIPS Conference,Vancouver,B.C., July 2002.

[12] S. Mohagheghi, G. K. Venayagamoorthy, and R. G. Harley, "Adaptive critic designs based neuro-fuzzy controller for a static compensator in a multimachine power system," IEEE Trans. Power Syst., vol. 21, no. 4, pp. 1744–1754, Nov. 2006.

[13] T. Li, D. B. Zhao, and J. Q. Yi, "Adaptive dynamic neuro-fuzzy system for traffic signal control," in Proc. IEEE International Joint Conference on Neural Networks, Hong Kong, June 2008, pp. 1840–1846.

[14] P. J. Werbos, "Approximate dynamic programming for real-time control and neural modeling," in Handbook of Intelligent Control, D. A. White and D. A. Sofge, Eds. New York: Van Nostrand Reinhold, 1992, ch. 13.

[15] D. V. Prokhorov and D. C. Wunsch, "Adaptive critic designs," IEEE Trans. Neural Networks, vol. 8, no. 5, pp. 997–1007, Sept. 1997.

[16] J. Si and Y. T. Wang, "On-line learning control by association and reinforcement," IEEE Trans. on Neural Networks, vol.12, no.2, pp. 264–276, 2001.

[17] D. B. Zhao, J. Q. Yi, and D. R. Liu, "Particle swarm optimized adaptive dynamic programming," in Proc. IEEE Int. Symp. Approximate Dynamic Programming and Reinforcement Learning, 2007, pp. 32–37.

[18] H. He and B. Liu, "A hierarchical learning architecture with multiple-goal representations based on adaptive dynamic programming," in Proc. IEEE Int. Conf. Networking, Sensing and Control, Chicago, April, 2010, pp. 286–291.

[19] T. Li, D. B. Zhao, and J. Q. Yi. "Heuristic dynamic programming strategy with eligibility traces," in IEEE Proc. American Control Conference. Seattle, USA, 2008. 4535–4540.

[20] A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike adaptive elements that can solve difficult learning control problems," IEEE Trans. Syst., Man. and Cyhern. vol. 13, pp. 834–846, Oct. 1983.

[21] M. J. Zhang and T. J. Tarn, "Hybrid control of the pendubot," IEEE/ASME Trans. on Mechatronics. vol. 7, pp.79–86, 2002.

[22] M. W. Spong and D. J. Block, Pendubot Installation and User Guide . Mechatronic System Inc. 1997.

[23] M. A. Perez-Cisneros, R. Leal-Ascencio, and P. A. Cook, "Reinforcement learning neurocontroller applied to a 2-dof manipulator," in Proc. IEEE Int. Sym. Intel. Contr., Mexico, 2001, pp.56–61.