Reinforcement Learning Based Control System And Its Stability Analysis

Dipak M. Adhyaru

Instrumentation and Control Engg. Department
Institute of Technology, Nirma University, Ahmedabad, Gujarat,India-382481
dipak.adhyaru@inrmauni.ac.in

***Abstract*— In present paper Reinforcement learning (RL) based control is implemented with Actor-critic based technique. Generation of reinforcement signal is described earlier with binary sign function for Actor-Critic based Reinforcement Learning. In the present paper two more discriminate functions introduced to generate reinforcement signal. It is proved with the results that these functions have faster convergence then earlier method to obtain reinforcement signal. In this paper, 2-link robot manipulator system's dynamics is used to simulate results. It is assumed that the system has a certain 'canonical' structure. Present paper also shows reinforcement learning based controller achieves faster convergence then conventional controller based on loop gains. Convergence proof of proposed algorithm is defined with Lyapunov stability theory. Also we have proposed generalized Lyapunov function for any discriminant functions and system expressed in canonical form.**

***Keywords*—Reinforcement learning, Neural Network, Lyapunov Stability, Discriminate function**

I. INTRODUCTION

Reinforcement learning (RL) is analogues to how human learn by experience. RL is based on the common sense notion that if an action gives satisfactory results or improves results then, tendency to produce similar action should be strengthened, i.e. reinforced. There are many techniques available which implement this concept, like Q-learning, Temporal Difference based learning, SARSA, Actor-Critic based learning etc. Reinforcement learning (RL) is one of the major NN approaches to learning control. The ability to generate the correct control actions nevertheless makes RL important in situations where lack of sufficient structure in the task definition makes it difficult to define a priori the desired outputs for each input, as required for supervised learning control and direct inverse control. In this paper, we will explore a new type of RL algorithm, in which the learning signal is merely a binary, "+1" or "-1", from a critic rather than an instructive correction signal. It was proposed by Lewis [1].We modify this reinforcement generating signal with other discreminant functions. We have also explored stability proof for general case (i.e. any discriminant functions and system which can be expressed in canonical format). We view RL based methods as computationally simple, direct approaches to adaptive-learning control.

In section **2** we will review stability of systems and some fundamentals of NN. Then section 3 presents RL control architecture and rigorous mathematical analysis on the tracking stability of the closed-loop system. In section 4 convergence proof of the algorithm is given using generalized Lyapunov function. At the last, present algorithm is applied to simulate control of a highly nonlinear 2-link robot system.

II. BACKGROUND

Let denote $\mathbb{R}$ the real numbers, $\mathbb{R}^n$ the real n-vectors, $\mathbb{R}^{mxn}$ and the real m x n matrices. We define the norm of a vector $X \in \mathbb{R}^n$ as

$$\|x\| = \left(\sum_{j=1}^{n} x_j^2 \right)^{1/2}$$

The absolute value is denoted as $\|.\|$, and $(.)^T$ denotes the transpose of a vector or a matrix. The trace of A written tr(A),is the sum of diagonal elements, and satisfies tr(A)= $\text{tr}(A)^T$ for a matrix $A=[a_{ij}] \in \mathbb{R}^{mxn}$

Given A= $[a_{ij}]$ and $B \in \mathbb{R}^{mxn}$, the Frobenius norm is defined by $\|A\|_F^2 = tr(A^T A) = \sum_{ij} a_{ij}^2$ with tr(.) the trace operator. The associated inner product is $(A,B)_F$ $=\text{tr}(A^T B)$. The Frobenius norm is compatible with the 2-norm so that $\|Ax\| \le \|A\|_F \|x\|_2$, with $A \in \mathbb{R}^{mxn}$ and $X \in \mathbb{R}^n$.

A. *Practical Stability of Nonlinear Systems*

Given $x(t) \in \mathbb{R}^n$ and a nonlinear function $f(x,t): \mathbb{R}^n x \mathbb{R} \to \mathbb{R}^n$, the differential equation $\dot{x} = f(x,t).....,t_0 \le t, x(t_0) = x_0$ has a differentiable solution $x(t)$ if *f(x,t)* is continuous in $x(t)$ and *t.*

The solution is $x(t)$ is said to be *uniformly ultimately bounded* (UUB) if there exists a compact set $U \subseteq \mathbb{R}^n$

B. *Stability of Systems*

Given $X \in \mathbb{R}^n$ and a nonlinear function $h(x,t): \mathbb{R}^n X \mathbb{R} \to \mathbb{R}^n$,the differential equation $\dot{x} = h(x,t); t_0 \le t, x(t_0) = x_0$

Has a differential solution x(t) if $h(x,t)$ is continuous in $x(t)$ and t.. The solution $x(t)$ is said to be Uniformly Ultimately bounded (UUB) if there exists a compact set $U \subseteq \mathbb{R}^n$ .Such that for all, there exists a $\delta > 0$ and a number $T(\delta, x_0)$ such that $\|X\| < \delta$ for all $\sigma(.)$.

C. Neural Networks

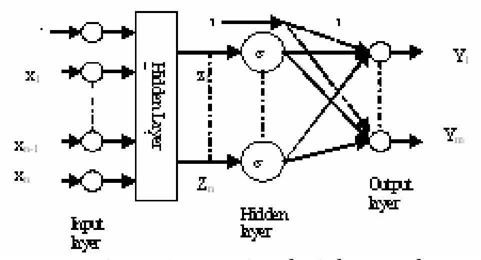

Figure 1: Functional Link Neural Network

Figure 1 depicts a functional link neural net (FLNN) which can be considered as "1"-layer feed forward neural net with input pre-processing element. This architecture is inspired by the work on high-order nets by Giles and Maxwell [11] and by the work on functional-link nets by Pao [10]. The FLNN architecture appears to be identical to that of the conventional "2"-layer neural net, except for the critical differences that only weights in the output layer will be adjusted. Note that the FLNN architecture is different from the functional link net in [10]. The ability of such neural nets to approximate continuous functions has been widely studied [3], [10].An FLNN can be used to approximate a nonlinear mapping $y(x): X^n \to \psi^m$,where $X^n \subset \mathbb{R}^n$ is the application specific n-dimensional input space and $\psi^m \subset \mathbb{R}^m$ in the application specific output space. If $\sigma(.)$ is a continuous discreminant function, then finite sums of the form

$$y_j(x) = \sum_{i=1}^{n} \{w_{ji}\sigma(z_i + \theta_i) + \theta_j\} + \varepsilon_j(x), j = 1,2...m.........(1)$$

are dense in the space of continuous functions defined on hypercube or a compact set, where $w_{ij} \in \mathbb{R}$ are output layer weight values, $\theta_i, \theta_j \in \mathbb{R}$ are threshold values and $z_i \in \mathbb{R}$ is the ith input to hidden neurons. $\varepsilon_j(x)$: is the sigmoidal activation function and N and is the number of units (also called nodes and neurons) in the hidden layer. $\sigma(.)$. is the output of the sigmoidal function. Note that the activation functions $\sigma(.)$ for each neuron are not necessarily the same. Finally $\varepsilon(x)$ is called as the *neural-net functional reconstruction error*.

III. Reinforcement Learning in Control

This is where the RL techniques show to have a major edge over the other methods as the RL techniques has the capability to learn in the absence of a teacher, i.e. with out a goal directed data pattern. Of all the existing structures of RL, the Actor-Critic/Adaptive-Critic structure seems to have a very close connection with the well-established structure of Adaptive control. Like in Adaptive control scheme, the Adaptive-Critic structure has an Adjustment Mechanism unit, a Critic, and an action-taking controller, an Actor.

In the Actor-Critic structure of RL, the actor takes actions that drive the state of system. The critic gives a scalar evaluation of the new state of the system. The scalar evaluation provided by the critic conveys much less information than the desired outputs required in the supervised learning. This critic information helps the actor in updating its parameters to produce a better action in the future that yields an improved system performance. Though the Actor-Critic structure is basically developed in game theory perspective, its link to control is very clear. Infect, its links with Bellman's Dynamic Programming is proved in the literature [5]. The control view point of Actor-Critic structure is presented in the next section.

A. Adaptive-Critic Structure

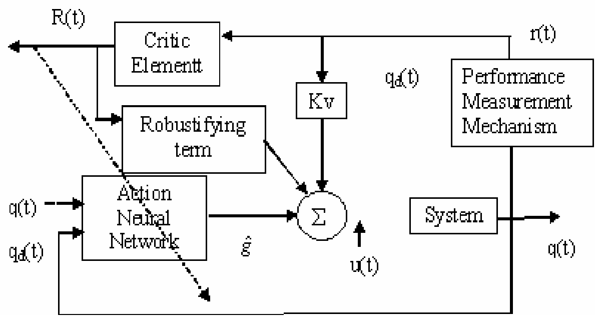

Figure 2: Actor-critic based reinforcement learning

A two link robot manipulator is used as a system for illustrating the operation of the present control structure. The details of error dynamics of the two link manipulator are given in section -4. The critic unit monitors the filtered error, and produces a reinforcement signal R(t) is a function of r(t), where *r(t)* is the filtered error as defined in (section-III-B). In the present work we have implemented signal R(t) with three different discriminant functions, as described below.

(1) The signum function is defined as $\mathrm{sgn}(x) = \begin{cases} +1 & if \quad x \geq 0 \\ -1 & otherwise \end{cases}$ So the critic signal is a mere binary signal, a +1 or -1, depending upon the filtered error information.

i.e $R_1(t) = \mathrm{sgn}(r(t))$ (2)

(2) $R_2(t) = R^+(t) + R^-(t) = \sigma^+(t) + \sigma^-(t)$ (3)

Where $\sigma^+(t) = \alpha^+ / (1 - e^{-\alpha + r(t)})$

(3) $R_3(t) = R^+(t) + R^-(t) = \sigma^+(t) + \sigma^-(t)$ (4)

Where $\sigma^+(t) = \alpha^+ \{e^{+\alpha + r(t)} - e^{-\alpha + r(t)} / (e^{+\alpha + r(t)} + e^{-\alpha + r(t)})$

The time derivative of the measured performance signal can be written as

$$\dot{r} = g(x, x_d) + u(t) + d(t) \quad (5)$$

Where $g(x, x_d)$ is fairly complex function of *x* and x_d. The control input to the plant is selected as

$$u(t) = -K_v r - \hat{g}(x, x_d) + v(t) \quad (6)$$

Where $\widehat{g}(x, x_d)$ is provided by the NN, the performance measurement gain matrix $K_v = K_v^T > 0$ and *v(t)* is a robustifying parameter that will be dealt later.

The NN approximation of $g(x, x_d)$ can be given as,

$$\widehat{g}(x, x_d) = W^T \sigma(\chi) + \varepsilon(x) \ \ldots, \ \varepsilon(x) \leq \varepsilon_N \tag{7}$$

where the input $\chi = V^T x$, *V* is first layer weights of the NN, *x* is the input to the NN and $\varepsilon(x)$ is the approximation error ,bounded by known constant.

The different parameters are chosen as following so that the filtered error *r* and the error in weights W are *Uniformly Ultimately Bounded* by known positive values[8] :

$\|W\|_F \leq W_M$, where W_M is known value.

The robustifying parameter $v(t) = -k_z\, R / \|R\|$. .(8)

Where $k_z \geq b_d$, an upper bound on the disturbance and R is the reinforcement signal.

The weight updating law $\dot{\hat{W}} = F\, \sigma(\chi)\, R^T - \kappa F \hat{W}$.(9)

with $F = F^T > 0$ and the learning rate $\kappa > 0$ for the speed of convergence.

Using equations (8) to (12) we have following performance measure dynamics, which is useful for stability proof(section-V).

$$\dot{r}(t) = -K_v r(t) + \tilde{W}^T \sigma(x) + \varepsilon(x) + d(t) - v(t). \tag{10}$$

Here weight estimation error is $\tilde{W} = W - \hat{W}$. The justification for selection of the parameters as above is done with the help of Lyapunov theory[1] in section -IV.

B. System and Tracking Error Dynamics Section Headings

The Robot manipulator dynamics, in general, can be expressed in state-space representation in the Brunovsky canonical form (BCF) as

$$\dot{x}_1 = x_2; \dot{x}_2 = x_3; \ldots\ldots; \dot{x}_n = f(x) + g(x)u$$

$$y = x_1$$

Feedback linearization will be used to perform output tracking, whose objective can be described as the following: Given a desired output, *$y_d(t)$*, find a control action *u(t)*, so that *$y(t) = x_1(t)$* follows the desired trajectory with an acceptable accuracy (i.e. bounded error tracking) while all states and control remain bounded. To design a tracking controller, a mild assumption is made. Define the trajectory vector as

$$x_d(t) = \left[y_d \ \dot{y}_d \ \ldots \ y_d^{\,n-1} \right]^T .$$

Assumption: The desired trajectory vector *$x_d(t)$* is continuous, available for measurement, and $\|x_d(t)\| \leq Q$,with Q a known bound.

Define a state error vector as, $e = x - x_d$ and a filtered tracking error as $r = \Lambda^T e$ (11)

where $\Lambda = [\lambda_1 \ \lambda_2 \ \lambda_3 \ \ldots \ \lambda_{n-1} \ 1]^T$ is an appropriately chosen coefficient vector so that $e \rightarrow 0$ exponentially as $r \rightarrow 0$, (i.e. $s^{n-1} + \lambda_{n-1} s^{n-2} + \cdots + \lambda_1$ is asymptotically stable).

The time derivative of the filtered error can be written as

$$\dot{r} = f(x) + g(x)u + Y_d \tag{12}$$

Where

$$Y_d = -x_d^{(n)} + \sum_{i=1}^{n-1} \lambda_i e_{i+1} = -x_d^{(n)} + [0 \quad \Lambda^T] e$$

is a known signal.

Using the error dynamics given by equation (11), a controller can be constructed using NN that keeps *r(t)* bounded. Then since equation (10) is a stable system, *e(t)* is bounded, thus, the tracking performance is achieved. The mathematical model for the 2-link robot manipulator is shown below

$$M(q)\ddot{q} + Vm(q,\dot{q})\dot{q} + F(\dot{q}) + G(q) + \tau_d = \tau \tag{13}$$

The robot dynamics can be expressed in terms of filtered error as

$$M\dot{r} = -V_m r + f(x) + \tau_d - \tau \ ;$$

where unknown nonlinear robot function is defined as,

$$f(x) = M(q)(\ddot{q}_d + \Lambda\dot{e}) + Vm(q,\dot{q})(\ddot{q}_d + \Lambda\dot{e}) + F(\dot{q}) + G(q).$$

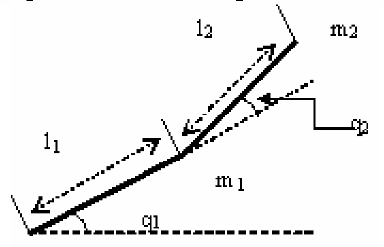

Figure 3: Robot Manipulator

IV. Stability Analysis

There are many person who had given convergence proof RL based control algorithm.[7],[8],[9].Here we have used Lyapunov based approach to prove convergence.

We have proposed a generalized Lyapunov function for the stability analysis.

It is $L(r,\tilde{W}) = \int R(r(t))dt + \frac{1}{2} tr(\tilde{W} F^{-1} \tilde{W})$. (14)

Then, the time derivative of Lyapunov function is given by

$$\dot{L}(r,\tilde{W}) = R(t)\dot{r}(t) + tr(\tilde{W} F^{-1} \dot{\tilde{W}}). \tag{15}$$

Evaluating (15) along the trajectories of (10) yields

$$\dot{L}(r,\tilde{W}) = -R(t)K_v r(t) + R(t)\{\varepsilon(x) + d(t) - v(t)\} + R(t)\tilde{W}^T \sigma(z) + tr(\tilde{W}^T F^{-1} \dot{\tilde{W}}). \tag{16}$$

And applying (8) and by the property of trace operator, we have

$$\dot{L}(r,\tilde{W}) \leq -R^T(t)K_v r(t) + R^T(t)\varepsilon(x) + tr\{\tilde{W}^T (F^{-1}\dot{\tilde{W}} + \sigma(x)R^T(t)\} \tag{17}$$

Here following property of trace operator is used

$$tr(D^T a b^T) = a^T D b \tag{18}$$

$a \in \mathbb{R}^n, b \in \mathbb{R}^l$ and $D \in \mathbb{R}^{n \times l}$

Inserting the reinforcement learning rule (9) into (17) yields

$$\dot{L}(r,\tilde{W}) \leq -R^T(t)K_v r(t) + R^T(t)\varepsilon(x) + \kappa tr(\tilde{W}^T(W_M - \tilde{W}^T). \quad (19)$$

Using the inequality

$$tr(\tilde{W}^T\hat{W}) = tr\{\tilde{W}^T(W - \tilde{W})\} \leq \|\tilde{W}\|_F (W_M - \|\tilde{W}\|_F) \qquad (20)$$

And $\|R(t)\| \leq k1$ we obtain

$$\dot{L}(r,\tilde{W}) \leq \text{-}k1(K_v)\|r(t)\| - \kappa(\|\tilde{W}\|_F - W_M/2)^2 + \kappa W_M^2/4 + \text{k}1\varepsilon_N \, (21)$$

The time derivative of L is guaranteed to be negative as long as either (22) or (23)hold.

$$\|r(t)\| \geq \frac{\kappa W_M^2/4 + \text{k}1\varepsilon_N}{k1(K_v)} \equiv B_r \qquad (22)$$

$$\|\tilde{W}\|_F \geq W_M/2 + \sqrt{W_M^2/4 + k1\varepsilon_N/\kappa} \equiv B_{\tilde{W}} \qquad (23)$$

Where B_r and $B_{\tilde{W}}$ are the convergence regions for the performance measure (error) and the weight estimation error respectively. Therefore $\dot{L}(r,\tilde{W})$ is negative outside a compact set. According to a standard Lyapunov theory extension, this demonstrates the UUB of both $\|r(t)\|$ and $\|\tilde{W}\|_F$.

Now we can use definition of different reinforcement signal (in equations (2),(3) and (4)) to propose Lyapunov stability proof.

(A) For the reinforcement signal R_1 defined in section 3.1 equation (2)

Consider following Lyapunov-like function

$$L(r,\tilde{W}) = \sum_{i=1}^{m} |r_i| + \frac{1}{2} tr(\tilde{W}F^{-1}\tilde{W}). \qquad (24)$$

(B) For the reinforcement signal R_2 defined in section 3.1 equation (3)

Consider following Lyapunov-like function

$$L(r,\tilde{W}) = \ln(1 + e^{\alpha^+ r(t)}) + \ln(1 + e^{-\alpha^- r(t)}) + \frac{1}{2} tr(\tilde{W}F^{-1}\tilde{W}). \, (24)$$

(C) For the reinforcement signal R_3 defined in section 3.1 equation (4)

Consider following Lyapunov-like function

$$L(r,\tilde{W}) = \ln(e^{\alpha^+ r(t)} + e^{-\alpha^+ r(t)}) + \ln(e^{-\alpha^- r(t)} + e^{\alpha^- r(t)}) + \frac{1}{2} tr(\tilde{W}F^{-1}\tilde{W}). \; (25)$$

V. Results and Discussions

The present controller performance is illustrated by applying it on a two link robot manipulator. The parameters of the problem are as follows: Length $l_1 = l_2 = 1$m and masses $m_1 = m_2 = 1$Kg desired trajectory $q_{1desired}$ = 0.1 * sin (t); $q_{2desired}$ = 0.1*cos (t); Loop gains K_v = diag (40).

The NN parameters used in simulations are as given below:

Number of Hidden neurons: 50, $\alpha^+ = \alpha^- = 30$

Hidden neuron activation function: tansig

Output neuron activation function : Purelin

Learning rate in the weight tuning law: F = diag[1]; κ = 0.01

Inputs to NN: $x = \begin{bmatrix} q^T & \dot{q}^T & e^T & \dot{e}^T & r^T & q^T_{desired} & \dot{q}^T_{desired} \end{bmatrix}^T$, a total of 14 inputs.

Inputs to Hidden neurons: $\chi = V^T x$

First layer weights *V*: randomly initialized and kept constant.

Simulation time: 10 seconds.

The simulation results are presented in four parts: with fixed loop gains only, with the RL controller (using 1st discriminant function) included along with the loop gains, with the RL controller (using 2nd discriminant function) included along with the loop gains, with the RL controller (using 3rd discriminant function) included along with the loop gains .We have also second case under a sudden disturbance for all above four parts.

A . Simple Loop gains

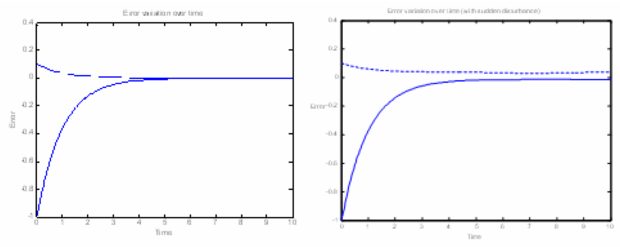

Figure 4: (a) (b)

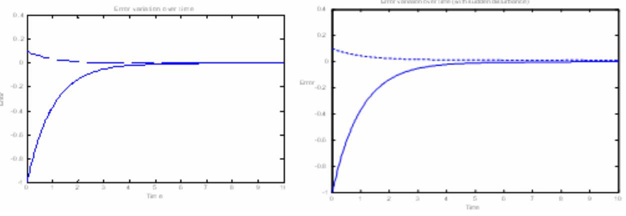

Figure 5: (a) (b)

The output tracking performance of the plant with loop gains alone is shown in Figures 4(a) and 4(b), for a loop gains of diag (40). The performance is not as encouraging as a steady error component is always present in the output. When the loop gains are increased by a substantial amount, to diag[120], the system performance got better as shown in Figures 5(a) and 5(b). But in practice limitations on loop gains always exist because of actuator saturation and unnecessary excitation of higher modes of the plant. Also when we apply sudden disturbance then there is a permanent steady state error of large amount.

C. RL controller with different discreminant function

The simulation was re-run with the RL controller included as a supplementing controller to the existing loop gain controller, infect for *lower* values of loop gains, at K_v=diag(20) only. The system tracking performance has improved considerably as shown in Figure-6(a).Here system achieves faster steady-state when we use 3rd discreminant function, in comparison with other discreminant functions. Also when we introduce sudden disturbance then RL controller gives better performance with zero steady –state error, as compare to simple loop gains. The results are shown in figure-6(b), figure-7(a) and 7(b).

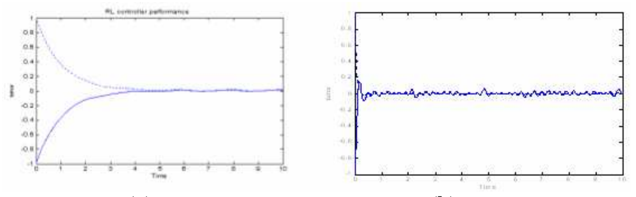

Figure 6: (a) (b)

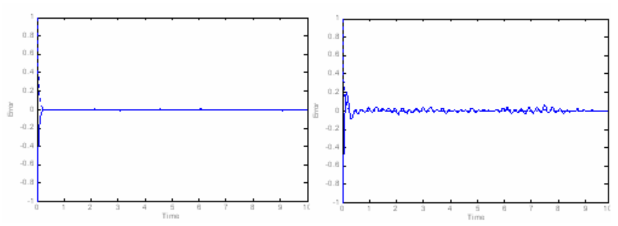

Figure 7: (a) (b)

VI. CONCLUSIONS

In this paper, Actor-critic type RL based control algorithm is simulated on 2-link robot manipulator system. Results shows it gives better performance compared to the controller designed based on loop gain alone. Also we have tried to generate RL signal with different discriminate functions. Results shows R_2 gives best performance among three. Stability proof of all three functions based RL algorithm is explained in section-4.the generalized Lyapunov function for any discriminant function is also given. One can try with other type of function to get better reinforcement signal.

ACKNOWLEDGMENT

Author is very much thankful to Prof. I.N. Kar and Prof. M. Gopal (Indian Institute of Technology-Delhi) for their useful discussion.

REFERENCES

[1] Y.H.Kim and F.L.Lewis, Direct-Reinforcement-Adaptive-Learning Neural network control for Nonlinear systems, Proc. of ACC , pp 1804-1808, 1997.

[2] J.Campos and F.L.Lewis, Adaptive Critic Neural network for feedforward compensation, AACC, pp 2813-2818, 1999.

[3] Simon Haykin, Neural Networks: A Comprehensive Edition, Prentice-Hall of India, 1998

[4] Richard S.Sutton, Andrew G. Barto, and Reinforcement Learning: An Introduction, MIT Press, Cambridge, MA, 1998.

[5] Jennie Si, Andrew G. Barto, Warren Buckler Powell, "Handbook of Learning and Approximate Dynamic Programming", IEEE press series on Computational Intelligence (2005).

[6] Y.H.Kim and F.L.Lewis, Reinforcement-Adaptive-Learning Neural-net-based friction compensation control for high speed and precision, IEEE Trans. On Control System Technology, Vol.8, No.1,January-2000.

[7] C. C. Cheah and Y.C. Sun, Region Reaching Control for Robots with Uncertain Kinematics and Dynamics, Proceedings of the 2006 IEEE International Conference on Robotics and Automation Orlando, Florida - May 2006.

[8] C. C. Cheah, M. Hirano, S. Kawamura, and S. Arimoto, Approximate Jacobian Control With Task-Space Damping, IEEE Trans.on Automatic Control,Vol. 49, No.5,May 2004

[8] V. S. Borkar and S. P. Meyn, The O.D. E. Method for Convergence of Stochastic Approximation and Reinforcement Learning, SIAM Journal on Control and optimization , Volume 38 , Issue 2 (January 2000)

[9] Pao, Y.–H., et al.: Neural-net Computing and Intelligent Control Systems. International Journal of Control 56(2), 263–289 (1992)

[10] Giles, C.L. and T. Maxwell. 1987. Learning, Invariance, and Generalization in High-Order Neural Networks. In: Appl. Optics, v. 26, p.4972