Proceedings of the 11th KACC, October 1996

Robust Feedback Error Learning Neural Networks Control of Robot Systems with Guaranteed Stability

Sung-Woo Kim

Production Engineering Research Lab.
Hyundai Electronics Ind. Co. Ltd., KOREA
E-mail: swkim11@hei.co.kr

Abstract — **This paper considers feedback error learning neural networks for robot manipulator control. Feedback error learning proposed by Kawato [2,3,5] is a useful learning control scheme, if nonlinear subsystems (or basis functions) consisting of the robot dynamic equation are known exactly. However, in practice, unmodeled uncertainties and disturbances deteriorate the control performance. Hence, we presents a robust feedback error learning scheme which add robustifying control signal to overcome such effects. After the learning rule is derived, the stability is analyzed using Lyapunov method.**

1. Introduction

Among the NNC schemes so far, feedback error learning (FEL) has several useful advantages over others such as no desired outputs of the neural network as the learning signal, no error back-propagation through the plant [8], simultaneous learning and control, etc. However, it also has problems that it requires *a priori* knowledge on the robot dynamics model and the system stability. The stability problem is essential in FEL, since the learning signal extraction from the feedback controller becomes meaningless, once the system trajectory starts diverging or drifting away.

Thus, in this paper, we mainly focus on the problems of FEL and its countermeasures. The rest of the paper is organized as follows: After briefly reviewing some problems of FEL such as linear parameterization, learning rule derivation, and stability in Section 2, we present countermeasures of such problems in Section 3. There, the neural network learning and control scheme with guaranteed performance is proposed. Simulation are carried out in Section 4. Finally, concluding comments follow.

2. Feedback Error Learning Neural Networks

Kawato *et al.* [2,3] and Miyamoto *et al.* [5] proposed a neural network of the form

$$\tau = W\phi(q, \dot{q}, \ddot{q}) \tag{1}$$

where the basis function, $\phi(\cdot)$, is constructed using the subsystems of robot dynamics. This equation means that the robot system is parameterized with a signal vector ϕ and a parameter matrix W. For example, consider the two-link robot manipulator of the dynamic equation:

Table 1: Basis functions and corresponding weights of each joint

		Joint 1	Joint 2
i	$\phi_i(x)$	w_{1i}	w_{2i}
1	$\ddot{q}_1$	$(m_1+m_2)\ell_1^2 + m_2\ell_2^2$	$m_2\ell_2^2$
2	$\ddot{q}_2$	$m_2\ell_2^2$	$m_2\ell_2^2$
3	$\ddot{q}_1 \cos q_2$	$2m_2\ell_1\ell_2$	$m_1\ell_1\ell_2$
4	$\ddot{q}_2 \cos q_2$	$m_1\ell_1\ell_2$	0
5	$\dot{q}_1^2 \sin q_2$	0	$m_2\ell_1\ell_2$
6	$\dot{q}_1\dot{q}_2 \sin q_2$	$-2m_2\ell_1\ell_2$	0
7	$\dot{q}_2^2 \sin q_2$	$-m_2\ell_1\ell_2$	0
8	$\cos q_1$	$(m_1+m_2)g\ell_1$	0
9	$\cos(q_1+q_2)$	$m_2g\ell_2$	$m_2g\ell_2$

$$\begin{aligned} \tau_1 = {} & [(m_1+m_2)\ell_1^2 + m_2\ell_2^2 + 2m_2\ell_1\ell_2\cos q_2]\ddot{q}_1 \\ & + [m_2\ell_2^2 + m_1\ell_1\ell_2\cos q_2]\ddot{q}_2 \\ & - m_2\ell_1\ell_2(2\dot{q}_1\dot{q}_2 + \dot{q}_2^2)\sin q_2 \\ & + (m_1+m_2)g\ell_1\cos q_1 \\ & + m_2g\ell_2\cos(q_1+q_2) \end{aligned} \tag{2}$$

$$\begin{aligned} \tau_2 = {} & [m_2\ell_2^2 + m_1\ell_1\ell_2\cos q_2]\ddot{q}_1 \\ & + m_2\ell_2^2\ddot{q}_2 + m_2\ell_1\ell_2\dot{q}_1^2\sin q_2 \\ & + m_2g\ell_2\cos(q_1+q_2). \end{aligned} \tag{3}$$

A possible set of basis functions (or subsystems) for each joint can be built as shown in Table 1.

Assuming the parameterized robot model, they applied joint torque using feedback PD controller, τ_{PD}, plus neural network controller, τ_{nn}:

$$\tau = \tau_{PD} + \tau_{nn} \tag{4}$$

where $\tau_{PD} = K_p e + K_v \dot{e}$ and $\tau_{nn} = \hat{W}\phi$ (Fig. 1). The weight of the neural network is updated by

$$\dot{w}_{ij} = \eta\ \tau_{PD,i} \cdot \phi_j, \tag{5}$$

or in matrix form

$$\dot{W}^T = \eta\ \phi\ \tau_{PD}^T, \tag{6}$$

where η is the learning rate. This learning rule is called as *feedback error learning* to emphasize the use of feedback torque as the learning signal [2,3]. It can be understood that the learning rule (5) minimizes the following cost function

$$E = \frac{1}{2}\|\tau - \tau_{nn}\|^2, \tag{7}$$

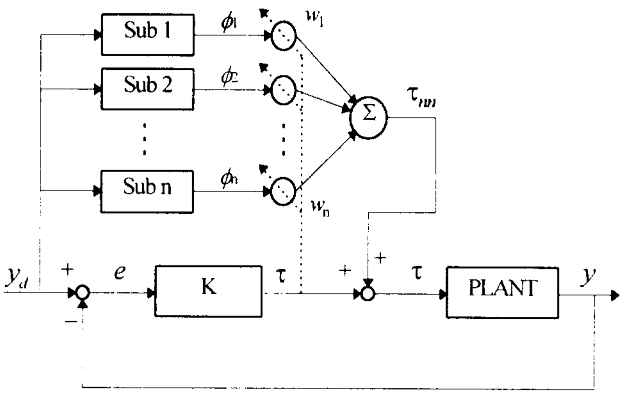

Fig. 1: Feedback error learning neural network control

because the gradient of a weight is written as

$$\frac{\partial E}{\partial w_{ij}} = -(\tau - \tau_{nn})_i \cdot \frac{\partial \tau_{nn,i}}{\partial w_{ij}} = -\tau_{PD,i} \cdot \phi_j. \qquad (8)$$

They applied this learning rule to robot manipulator control and reported that this had worked successfully for trajectory tracking control. However, there are several limitations of neural network control using feedback error learning. First, it is not robust under unstructured uncertainties such as modeling error or disturbance. The subsystem consisting of robot dynamics should be known *a priori* for the system to be written linearly in parameters as in (1). For example, the two-link robot manipulator is controlled well if all of ϕ_i's in Table. 1 are known, but the control performance is degraded seriously even ϕ_8 only is omitted [4]. Second, any stability analysis including neural network learning is not given in original feedback error learning. Hence, we propose *robust feedback error learning* to overcome the indicated limitations.

3. Robust Feedback Error Learning

Consider a general n-dof robot manipulator of

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) + f(\dot{q}) = \tau + d(t) \qquad (9)$$

where $g(q)$ is the gravity vector, $f(\dot{q})$ is the friction vector containing viscous and Coulomb terms, $d(t)$ is the disturbance, and τ is the input torque vector.

Now, apply

$$\tau = \tau_{PD} + \tau_{nn} + \tau_r. \qquad (10)$$

Then, the closed-loop filtered error dynamics is represented as

$$M\dot{s} = -K_D s - Cs + h - \tau_{nn} - \tau_r, \qquad (11)$$

where $h = M\ddot{r} + C\dot{r} + g + f - d$. Assume the function h can be constructed by the neural network as

$$h = W\phi(q,\dot{q},r,\ddot{r}) + \epsilon(q,\dot{q},\dot{r},\ddot{r},d) \qquad (12)$$

where $\epsilon(\cdot)$ represents the function estimation error due to modeling error and disturbance. The bound of ϵ is dependent on the choice of ϕ. If all the subsystems consisting of the robot dynamics are incorporated in ϕ, then theoretically the bound can be $\|\epsilon\| \le \|d\|$. However, if some subsystems are unmodeled and omitted in ϕ, then the bound of ϵ will increase as much as the uncertainty. Here we assume the function ϕ is chosen reasonably so that for a given ϕ, there exist a neural network weight matrix W satisfying

$$\|\epsilon\| \le \rho(q,\dot{q},\dot{r},\ddot{r},d), \qquad (13)$$

which means the estimation error is bounded by a function, $\rho(\cdot)$. As shown in [1], the physical properties of the robot manipulator can be used to show that the estimate error bound, ρ, can be represented as

$$\rho = a_0 + a_1\|\mathbf{e}\| + a_2\|\mathbf{e}\|^2 = Ya, \qquad (14)$$

where $\mathbf{e} = [e^T, \dot{e}^T]^T$, $Y = [1, \|\mathbf{e}\|, \|\mathbf{e}\|^2]^T$, and $a = [a_0, a_1, a_2]^T$ with positive bounding constants a_i's. The bound is not necessarily to be known since it will be estimated using $\hat{\rho}$:

$$\hat{\rho} = \hat{a}_0 + \hat{a}_1\|\mathbf{e}\| + \hat{a}_2\|\mathbf{e}\|^2 = Y\hat{a}. \qquad (15)$$

The difference a and $\hat{a}$ is defined as

$$\tilde{a} = a - \hat{a}. \qquad (16)$$

The neural network controller provides a control input using the current estimated weight $\hat{W}$:

$$\tau_{nn} = \hat{W}\phi. \qquad (17)$$

The weight $\hat{W}$ is updated using the learning rule provided in the following theorem. With the ideal weight W in (12), define the weight estimation error as

$$\tilde{W} = W - \hat{W}. \qquad (18)$$

Now we are ready to derive the learning rule of the neural network weight and robust control law to overcome the uncertainty.

Theorem 1 *Let the desired trajectory $q_d, \dot{q}_d, \ddot{q}_d$ be bounded and the estimation error is bounded as (13). Take the control torque for the robot manipulator as (10) with $\tau_{PD} = K_D s, \tau_{nn} = \hat{W}\phi$, and*

$$\tau_r = \begin{cases} \hat{\rho}\frac{s}{\|s\|} & \text{if } \hat{\rho}\|s\| > \Phi \\ \hat{\rho}^2\frac{s}{\Phi} & \text{if } \hat{\rho}\|s\| \le \Phi \end{cases} \qquad (19)$$

where $\dot{\Phi} = -\alpha\Phi, \Phi(0) > 0$ and α is a positive constant. The neural network weight is updated by the learning rule:

$$\dot{\hat{W}}^T = \Gamma\phi s^T \qquad (20)$$

where $\Gamma > 0$ is a constant matrix determining the learning rate. The bounding estimate is updated by

$$\dot{\hat{a}} = \gamma Y^T\|s\|. \qquad (21)$$

Then the tracking errors, e and $\dot{e}$, converge to zero asymptotically, while all other signals including $\hat{W}$ and $\hat{\rho}$, etc. remain bounded.

Proof: Define the Lyapunov function candidate

$$V = \frac{1}{2}s^T M s + e\Lambda^T K_D e + \frac{1}{2}\mathrm{tr}(\tilde{W}\Gamma^{-1}\tilde{W}^T) + \frac{1}{2}\tilde{a}^T\gamma^{-1}\tilde{a} + \frac{\Phi}{\alpha}. \quad (22)$$

Differentiating the above, substituting (11) and (12), and using (20) yields

$$\dot{V} = -\mathbf{e}^T Q\mathbf{e} + s^T(\epsilon - \tau_r) - \tilde{a}^T\gamma^{-1}\dot{\hat{a}} + \frac{\dot{\Phi}}{\alpha}, \quad (23)$$

where $\mathbf{e}^T = [e^T, \dot{e}^T]^T$, $Q = \mathrm{diag}[\Lambda^T K_D \Lambda, K_D]$. The first term is negative semi-definite since Q is positive definite. Now we have

$$\dot{V} \le -\mathbf{e}^T Q\mathbf{e} - \Phi + \hat{\rho}\|s\| - s^T\tau_r, \quad (24)$$

where (21) is used. The last three terms are considered next. If $\hat{\rho}\|s\| > \Phi$ then $\tau_r = \hat{\rho}s/\|s\|$, and we have

$$-\Phi + \hat{\rho}\|s\| - s^T\tau_r = -\Phi < 0. \quad (25)$$

If $\hat{\rho}\|s\| \le \Phi$ then $\tau_r = \hat{\rho}^2 s/\Phi$, and we have

$$-\Phi + \hat{\rho}\|s\| - s^T\tau_r = -\frac{1}{\Phi}(\hat{\rho}\|s\| - \Phi)^2 - \hat{\rho}\|s\| \le 0 \quad (26)$$

since $\hat{\rho}\|s\| - \Phi \le 0$. Therefore, we obtain

$$\dot{V} \le -\mathbf{e}^T Q\mathbf{e}. \quad (27)$$

We immediately have from the above that $e, s, \hat{a}$, and $\hat{W}$ are bounded and $e, \dot{e}, s \in L_2$. This implies that $e \to 0$ as $t \to \infty$. Moreover, it is obvious from (19) that $\|\tau_r\| \le \hat{\rho}$. Therefore from (11) we see that $\dot{s}$ is bounded, which implies $\dot{e} \to 0$.

$\square$

Note that the above theorem states that the uncertainty which can not be estimated by the neural network can be overcome by the incorporated robust control term. It allows us a design trade-off between network complexity and robust control. In fact, it allows the possibility of selecting a simplified neural network structure based on the known basis function $\phi(\cdot)$ and compensating the increased magnitude of ρ using the robust control term.

Note that the bounding estimate is likely to drift in the presence of unmodeled dynamics and disturbances and become unbounded since $\dot{\hat{a}}$ is always positive in (21). In order to guarantees that $\hat{a}$ remains bounded we need to modify the update law (21) using σ-modification as

$$\dot{\hat{a}} = \gamma Y^T\|s\| - \sigma\hat{a}. \quad (28)$$

Now, as a special case, consider the estimate error is bounded by a constant, that is,

$$\|\epsilon\| \le \rho_c \ (= const.). \quad (29)$$

Then, feedback error learning neural networks can provide some interesting results without a robust control term. The following theorem states such results.

Theorem 2 *Let the desired trajectory $q_d, \dot{q}_d, \ddot{q}_d$ be bounded and the estimation error bound be a constant, i.e., $\|\epsilon\| \le \rho_c$. Take the control torque for the robot manipulator, $\tau = \tau_{PD} + \tau_{nn}$ with $\tau_{PD} = K_D s, \tau_{nn} = \hat{W}\phi$. The neural network weight is updated by*

$$\dot{\hat{W}}^T = \Gamma\phi s^T \quad (30)$$

where $\Gamma > 0$ is a constant matrix. Then the filtered tracking error $s(t)$ is uniform ultimately bounded (UUB) with a bound

$$\|s\| \le \rho_c/k_D, \quad (31)$$

where k_D is the minimum singular value of K_D.

Proof: Define the Lyapunov function candidate

$$V = \frac{1}{2}s^T M s + \frac{1}{2}\mathrm{tr}(\tilde{W}\Gamma^{-1}\tilde{W}^T). \quad (32)$$

Differentiating and substituting (11) yields

$$\dot{V} = -s^T K_D s + \mathrm{tr}[\tilde{W}(\Gamma^{-1}\dot{\tilde{W}}^T + \phi s^T)] + s^T\epsilon. \quad (33)$$

Using (30), we have

$$\begin{aligned}\dot{V} &= -s^T K_D s + s^T\epsilon \\ &\le -k_D\|s\|^2 + \rho_c\|s\|. \end{aligned} \quad (34)$$

Therefore, $\dot{V} \le 0$ as long as (31) since ρ_c is a constant.

$\square$

Note that boundedness of filtered tracking error $s(t)$ implies

$$\|e\| \le \|s\|/\lambda_{\min} \le \rho_c/(k_D \cdot \lambda_{\min}) \quad (35)$$

where $\lambda_{\min}$ is the smallest element of Λ. Hence, we can guarantee that the tracking error is bounded by a arbitrary small value properly chosen using the constant matrices K_D and Λ. However, such a property is guaranteed only when the function ρ is constant. In case of nonconstant ρ we generally need some additional control inputs such as robust control terms used in Theorem 1.

Theorem 2 states how much the feedback error learning neural network control can do. That is, original feedback error learning is useful only in the case of constant estimate error.

4. Simulation

The performance of the proposed robust feedback error learning neural network control of Theorem 1 is verified through the simulation results of the two-link manipulator. The desired trajectory is the circular one given by

$$\begin{bmatrix} x_d \\ y_d \end{bmatrix} = \begin{bmatrix} 1.0 + 0.2\sin(\frac{2\pi}{T}t) \\ 0.4 + 0.2\cos(\frac{2\pi}{T}t) \end{bmatrix} \quad (36)$$

with $T = 2$sec.

In order to show the robustness of the proposed controller, the friction and disturbance which are not considered in the simulation of feedback error learning neural network control are now included in the following simulations (Fig. 2 and Fig. 3). Fig. 2 depicts the tracking performance. Fig. 3 depicts the control input applied to the first joint.

5. Conclusion

We have presented here the robust feedback error learning neural network control and have shown that if the uncertainty of the robot dynamics is linearly parameterized, then the learning rule of the neural network can be derived directly from the Lyapunov analysis. Thus the closed loop stability can be guaranteed without the passivity properties of the network. Furthermore, the uncertainty which can not be estimated by the neural network can be overcome by the incorporated robust controller.

References

[1] D. M. Dawson, Z. Qu, F. L. Lewis, and J. F. Dorsey, "Robust control for the tracking of robot motion," *Int. J. Control*, vol. 52, pp. 581-595, 1990

[2] M. Kawato, K. Furukawa, and R. Suzuki, "A hierarchical neural-network model for control and learning of voluntary movements," *Biol. Cybern.*, vol. 57, pp. 165-185, 1987

[3] M. Kawato, Y. Uno, M. Isobe, and R. Suzuki, "Hierarchical neural network model for voluntary movement with application to robotics," *IEEE Control Systems Magazine*, pp. 8-16, 1988

[4] Ju-Jang Lee, Sung-Woo Kim and Kang-Bark Park, "Root manipulator control with guaranteed stability using feedback error learning neural networks," *Journal of Robotics and Mechatronics*, vol. 8, no. 4, 1996 (in press)

[5] H. Miyamoto, M. Kawato, T. Setoyama, and R. Suzuki, "Feedback-error-learning neural network for trajectory control of a robotic manipulator," *Neural Networks*, vol. 1, pp. 251-265, 1988

[6] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Networks*, vol. 1, no. 1, pp. 4-27, 1990

[7] T. Ozaki, T. Suzuki, T. Furuhashi, S. Okuma, and Y. Uchikawa, "Trajectory control of robotic manipulators using neural networks," *IEEE Trans. Industrial Electronics*, vol. 38, no. 3, pp. 195-202, 1991

[8] D. Psaltis, A. Sideris, and A. A. Yamamura, "A multilayered neural network controller," *IEEE Control Systems Magazine*, pp. 17-21, 1988

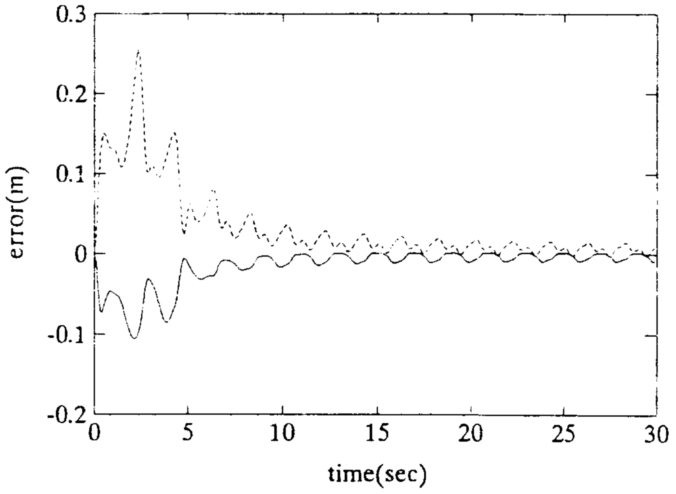

Fig. 2: Tracking performances when Theorem 1 is applied: Solid line (x directional error) and dash line (y directional error).

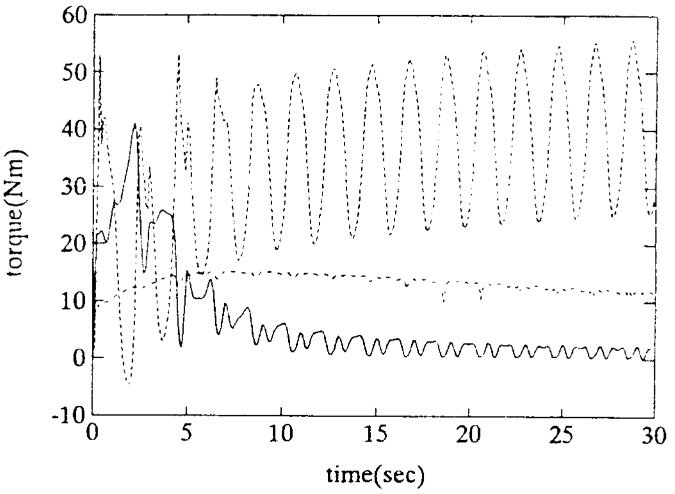

Fig. 3: Control inputs for joint 1: τ_{PD} (solid line), τ_{nn} (dash line), and τ_r (dash-dot line).