Proceedings of the 11th
KACC, October 1996

Estimation of Learning Gain in Iterative Learning Control Using Neural Networks

Jin Young Choi and Hyun Joo Park

School of Electrical Eng., Seoul National University, Kwanak Ku, Seoul 151-742, KOREA
Tel: +82-2-880-8372; Fax: +82-2-885-6620; E-mail: jychoi@asri.snu.ac.kr

Abstracts This paper presents an approach to estimation of learning gain in iterative learning control for discrete-time affine nonlinear systems. In iterative learning control, to determine learning gain satisfying the convergence condition, we have to know the system model. In the proposed method, the input-output equation of a system is identified by neural network refered to as Piecewise Linearly Trained Network (PLTN). Then from the input-output equation, the learning gain in iterative learning law is estimated. The validity of our method is demonstrated by simulations.

Keywords Iterative Learning Control, Learning Gain, Input Update Law, PLTN

1. Introduction

Iterative learning control is a technique for improving the performance of systems that operate repetitively over a fixed time interval. The purpose of iterative learning control is to find the desired control trajectory which makes the system output to follow the desired trajectory exactly over a finite time interval. By observing the behavior of a system in each trial, the input at the next trial is calculated by iterative learning law.

Open loop iterative control has demerit that the system output can be extremely large during the learning procedure. For this reason, iterative learning control for feedback nonliear system was proposed, and showed good result for two link robot maninpulator [1]. Jang [2] found out the convergence condition in iterative learning control for feedback nonlinear system. However in practical system, we cannot know whether the convergence condition is satisfied if we don't know the system model. If there is modeling error, or the change of system parameters during the process, the convergence condition can be broken and the system output may not converge to the desired trajectory. Hence it is desirable to find iterative learning gain satisfying the convergence condition even if we don't know the system dynamics.

In this paper, we propose a method to estimate learning gain in the iterative learning law from only input and output measurements. PLTN(piecewise linearly trained network) [3] is used to estimate learning gain. In the proposed method, we don't need to know the system dynamics except for system dimension and relative degree. Our method not only guarantees the convergence in iterative learning control for nonliear feedback system, but also shows fast convergence. In section 2, the convergence condition of iterative learning control for feedback nonlinear systems is described. Section 3 and 4 present the method of estimating learning gain from input-output data. In section 5, we will show the validity of the proposed method by simulations. Conclusion is given in section 6.

2. Iterative Learning Control for Nonlinear Feedback Systems

Consider a discrete-time affine nonlinear system with relative degree d.

$$\begin{aligned} \mathbf{x}(i+1) &= F(\mathbf{x}(i)) + G(\mathbf{x}(i))u(i), \\ y(i) &= h(\mathbf{x}(i)). \end{aligned} \tag{1}$$

where $u, y \in \Re$ are the input and output respectively, and $\mathbf{x} = [x_1, \cdots, x_n]^T \in \Re^n$ is a state vector. Since the relative degree of this system is d, the input signal $u(k)$ influences the output $y(k+d)$. That is,

$$\begin{aligned} &\frac{\partial}{\partial u} h \circ F^j(F(\mathbf{x}) + G(\mathbf{x})u) = 0, \quad j < d-1 \\ &J(\mathbf{x}, u) := \frac{\partial}{\partial u} h \circ F^{d-1}(F(\mathbf{x}) + G(\mathbf{x})u) \neq 0 \end{aligned} \tag{2}$$

The structure of iterative learning control for a nonlinear feedback system is shown in Fig 1. In this figure, u_{ff}^k is feedforward input which is obtained by iterative learning. If the feedforward input becomes the desired input which makes the system output the same as the the desired output by iterative learning, the input by feedback controller becomes zero.

Iterative learning procedure to obtain the desired feedforward input u_{ff} is illustrated in the following. Here k represents iteration step, $u_{ff}^k(i)$ is feedforward input by iterative learning control at step k, and $u_{fb}^k(i)$ is input by feedback control.

1) Design a feedback controller which stabilizes a nonlinear system.

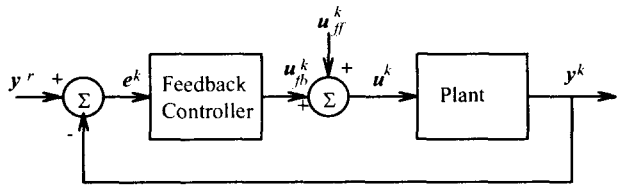

Figure 1: The structure of iterative learning control for a feedback nonlinear system

2) At the first iteration step, i.e., $k = 0$, only feedback controller is used. Hence $u_{ff}^0(i) \equiv 0$
3) Apply input trajectory $u^k(i) = u_{ff}^k(i) + u_{fb}^k(i)$ to the system, and obtain the output trajectory $y^k(i)$.
4) Calculate the next step input trajectory from $u^k(i)$ and output error $e^k(i) = y^r(i) - y^k(i)$.

$$u_{ff}^{k+1}(i) = l(i, u^k(\cdot), e^k(\cdot)) \tag{3}$$

5) Repeat step 3)-4) until the output error becomes smaller than the permitted error for $i \in [0, N]$.

In the above procedure, what is important is how to obtain the input trajectory of the next trial, u_{ff}^{k+1} from the output error and input trajectory, i.e., how to obtain equation (3). One of the method to find the next input trajectory was suggested by Jang [2].

Theorem 1
Let the input update equation in the iterative learning control for feedback nonlinear system (1) be like the following.

$$u_{ff}^{k+1}(i) = u^k(i) + S^k(i)e^k(i+d), i \in [0, N], \tag{4}$$

where, $S^k : [0, N] \rightarrow \Re$ is a bounded function, and $e^k(i) = y^r(i) - y^k(i)$, $i \in [0, N+d]$ is the output trajectory error.
If, $S^k(i)$ satisfies the following inequality for $i \in [0, N]$,

$$\|I - S^k(i)J(\mathbf{x}^k(i))\| \leq \rho < 1 \tag{5}$$

then, as $k \rightarrow \infty$ the output trajectory error converges to zero uniformly for $i \in [0, N]$, i.e.,

$$\lim_{k \rightarrow \infty} \|e^k(i)\| = 0, \quad \forall i \in [0, N] \tag{6}$$

□

We call $J(\mathbf{x}, u)$ in (2) and (5) decoupling matrix in the MIMO system. From the convergence condition in (5) $S^k(i)$ can be chosen as the following.

$$S^k(i) = J^{-1}(\mathbf{x}^k(i)) \tag{7}$$

When equation (7) holds, the convergence of the system output to the desired output is very fast. If we know the state equation of a system, from the definition of J in equation (2) we can calculate $S^k(i) = J^{-1}(\mathbf{x}^k(i))$. However, when the state equation is unknown, $J(\mathbf{x}^k(i))$ cannot be obtained directly. In the next section, we suggest the method to find $J(\mathbf{x}^k(i))$ when the state equation of the system is unknown.

3. Input Update Law In Case Of Unknown State Equation

The system (1) can be expressed in input-output equation form as the following.

$$\begin{aligned} y(i+1) &= f_0(\bar{\mathbf{x}}(i)) + g_0(\bar{\mathbf{x}}(i))u(i-d+1) \\ \bar{\mathbf{x}}(i) &= [y(i-n+1), \cdots, y(i), \\ &\quad u(i-d-m+1), \cdots, u(i-d)] \end{aligned} \tag{8}$$

Let us define state variable $\mathbf{x}$ like the following.

$$\begin{aligned} x_j(i) &= y(i-n+j), \quad j = 1, 2, \cdots, n \\ x_{n+j}(i) &= u(i-m-d+j), \quad j = 1, 2, \cdots, m+d-1 \\ \mathbf{x}(i) &= [x_1(i), \cdots, x_{n+m+d-1}(i)] \end{aligned} \tag{9}$$

For the state variable defined in the above equations, the following relation is held.

$$\begin{aligned} x_j(i+1) &= x_{j+1}(i), \quad j = 1, 2, \cdots, n-1 \\ x_n(i+1) &= f_0[\bar{\mathbf{x}}(i)] + g_0[\bar{\mathbf{x}}(i)]x_{n+m+1}(i) \\ x_{n+j}(i+1) &= x_{n+j+1}(i), \\ &\quad j = 1, 2, \cdots, m+d-2 \\ x_{n+m+d-1}(i+1) &= u(i) \end{aligned} \tag{10}$$

The above equations can be expressed in the form of (1), and the result is,

$$\begin{aligned} \mathbf{x}(i+1) &= F(\mathbf{x}(i)) + G(\mathbf{x}(i))u(i) \\ y(i) &= h(\mathbf{x}(i)) \end{aligned} \tag{11}$$

where,

$$\begin{aligned} F(\mathbf{x}(i)) &= \begin{bmatrix} x_2(i) \\ x_3(i) \\ \vdots \\ f_0[\bar{\mathbf{x}}(i)] + g_0[\bar{\mathbf{x}}(i)]x_{n+m+1}(i) \\ x_{n+2}(i) \\ \vdots \\ x_{n+m+d-1}(i) \\ 0 \end{bmatrix} \\ G(\mathbf{x}(i)) &= \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \\ h(\mathbf{x}(i)) &= x_n(i) \end{aligned} \tag{12}$$

From the definition of J in (2), we can obtain J.

$$\begin{aligned} J(\mathbf{x}(i), u(i)) &= \frac{\partial}{\partial u(i)} h \circ F^{d-1}(F(\mathbf{x}(i)) + G(\mathbf{x}(i))u(i)) \\ &= g_0[\bar{\mathbf{x}}(i+d-1)] \\ &= g_0[y(i+d-n), \cdots, y(i+d-1), \\ &\quad u(i-m), \cdots, u(i-1)] \end{aligned} \tag{13}$$

From (4),(7) and (13), the input update equation which makes the output trajectory converge fast to the

desired trajectory is expressed in the following equation.

$$\begin{aligned} u_{ff}^{k+1}(i) &= u^k(i) + J^{-1}(\mathbf{x}^k(i))\epsilon^k(i+d) \\ &= u^k(i) + (g_0[\bar{\mathbf{x}}(i+d-1)])^{-1}\epsilon^k(i+d) \\ & \quad i \in [0, N] \end{aligned} \tag{14}$$

In case that we don't know the system dynamics, we can approximate the input-output equation using neural networks. From the approximated input-output equation we can obtain approximated $g_0(\cdot)$, i.e., $g_0(\cdot)$, inverse of the estimated learning gain. We will show the details in the next section.

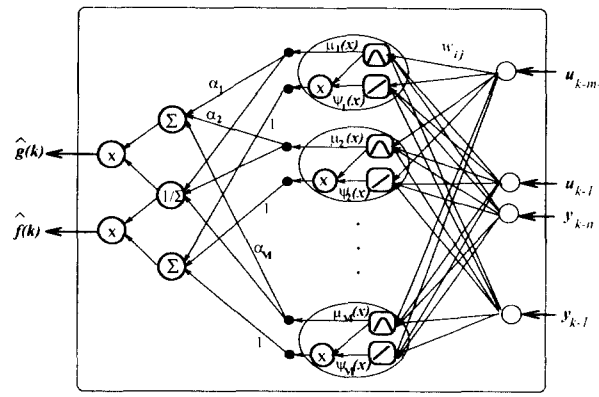

Figure 2: Neural Network to estimate J

4. Estimation of J using PLTN

To estimate $J = g_0(\cdot)$ only from input and output data, we use PLTN(Piecewise Linearly Trained Network) . The structure of PLTN is shown in figure 2.

Input-output equation (8) for output $y(i+d)$ is,

$$\begin{aligned} y(i+d) &= f_0(\bar{\mathbf{x}}(i+d-1)) + g_0(\bar{\mathbf{x}}(i+d-1))u(i) \\ \bar{\mathbf{x}}(i+d-1) &= [y(i+d-n), \cdots, y(i+d-1), \\ & \quad u(i-m), \cdots, u(i-1)] \end{aligned} \tag{15}$$

Equation (15)can be approximated using PLTN like the following method.

$$\begin{aligned} y(i+d) &= \sum_{j=1}^{M} \{\bar{\mu}_j(\bar{\mathbf{x}}(i+d-1)) \\ & \quad (\mathbf{w}_j^T \bar{\mathbf{x}}(i+d-1) + \alpha_j u(i) + \beta_j)\} \\ &= \sum_{j=1}^{M} \bar{\mu}_j(\bar{\mathbf{x}}(i+d-1))(\mathbf{w}_j^T \bar{\mathbf{x}}(i+d-1) + \beta_j) \\ & \quad + \sum_{j=1}^{M} \bar{\mu}_j(\bar{\mathbf{x}}(i+d-1))\alpha_j u(i) \\ &= f_0(\bar{\mathbf{x}}(i+d-1)) + g_0(\bar{\mathbf{x}}(i+d-1))u(i) \end{aligned} \tag{16}$$

where

$$\begin{aligned} f_0(\bar{\mathbf{x}}(k)) &= \sum_{j=1}^{M} \bar{\mu}_j(\bar{\mathbf{x}}(k))(\mathbf{w}_j^T \bar{\mathbf{x}}(k) + \beta_j) \\ g_0(\bar{\mathbf{x}}(k)) &= \sum_{j=1}^{M} \bar{\mu}_j(\bar{\mathbf{x}}(k))\alpha_j, \end{aligned} \tag{17}$$

where $\bar{\mathbf{w}}_j$ is a (n+m+d+1)-dimensional parameter vector and β_j is a bias constant. This function is equivalent to the McCulloch-Pitts neuron model [4] except the nonlinearity of threshold. The $\bar{\mu}_j(\bar{\mathbf{x}})$ is the localizing function based on the radial basis function (RBF):

$$\mu_j(\bar{\mathbf{x}}) = \exp(-||\bar{\mathbf{x}} - \mathbf{c}_j||^2/\gamma), \quad j = 1, \cdots, M \tag{18}$$

$$\bar{\mu}_j(\bar{\mathbf{x}}) = \mu_j(\bar{\mathbf{x}}) / \sum_{m=1}^{M} \mu_m(\bar{\mathbf{x}}), \tag{19}$$

where $||\bar{\mathbf{x}} - \mathbf{c}_j||$ is Euclidean distance between the input $\bar{\mathbf{x}}$ and the centeral point $\mathbf{c}_j$ of $\mu(\bar{\mathbf{x}})$, and M is the total number of processing units. The vigilance parameter γ determines the size of the local neighborhood.

Now the input update equation can be expressed like the following ,where $g_0(\cdot)$ in (14) is changed with the approximated function $g_0(\cdot)$.

$$u_{ff}^{k+1}(i) = u^k(i) + (g_0[\bar{\mathbf{x}}(i+d-1)])^{-1}\epsilon^k(i+d) \tag{20}$$

To train the PLTN (16), we use a training method with self-organizing and linear fitting technique described in our previous work [3]. The unsupervised learning for the Receptive Field (RF) node [1] is highlighted by the self-organizing capability representing the automatic recruitment of new processing units whenever there is no RF node covering the new state. The parameters of Linear Fitting (LF) node [2] is updated by the recursive least squares method [5]. Since only the winning LF node [3] updates its parameters, the computational complexity for training is similar to that of the linear system identification. It is interesting to note that this computational complexity is much smaller than that of multilayer feedforward neural network. Furthermore its training speed is fast due to linear fitting approach.

5. Simulation

The proposed method was applied to the following discrete-time nonliear system with relative degree $d = 3$:

$$\begin{aligned} y(i+3) &= \frac{y(i+2) + y(i+1)}{1 + y^2(i+2) + y^2(i+1)} \\ & \quad + (1 + y^2(i+2))u(i), \end{aligned} \tag{21}$$

The following signal was used for the desired output.

$$y^r(i) = \sin(\pi i/30), \quad i \in [0, 60] \tag{22}$$

[1] RF node generates the function $\mu_j(\cdot)$.

[2] LF node generates the function $\psi_j(\cdot)$.

[3] The winning LF node is the LF node corresponding to the RF node with the largest function value for the current state $\mathbf{x}(k)$.

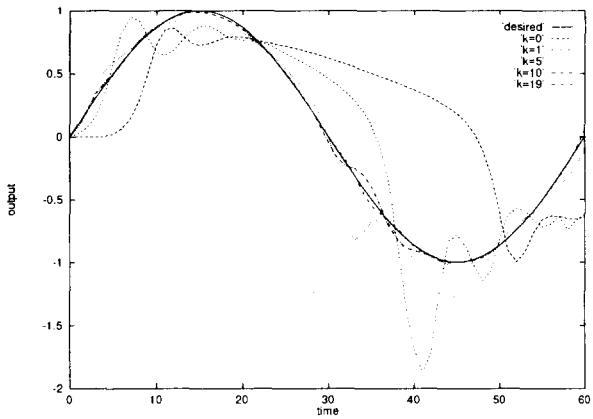

Figure 3: Learning procedure of the desired trajectory

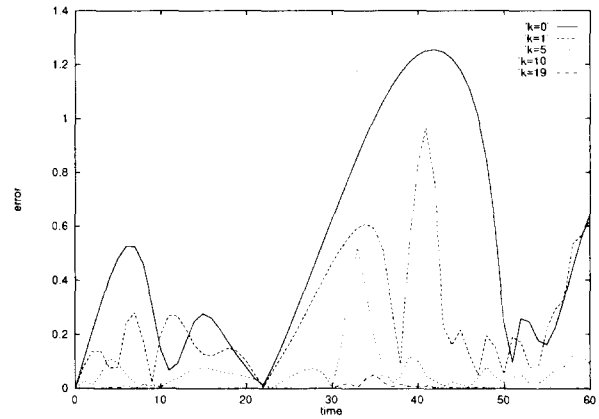

Figure 4: Error is decreasing as iteration step goes

We designed proportional controller with gain 0.2 as the stabilizing controller. Fig 3 and 4 shows the result of simulation for system (21). These figures show that as the iterative learning process goes, the output trajectory of the system converges to the desired trajectory. To prevent the output from becoming large during iterative learning, we multiplied $S^k(i)$ by a positive constant smaller than 1. In the system (21) used in the simulation, the output diverges if the input range becomes large. That is, system (21) is open-loop unstable system. Hence, we cannot train the neural networks to approximate $g_0(\cdot)$ for large input. Therefore if the input becomes large during the control, the estimated function $g_0(\cdot)$ becomes very different from $g_0(\cdot)$. In this reason, we multiplied $S^k(i)$ by a positive constant smaller than 1.

6. Conclusion

In this paper, we proposed iterative learning control for systems with unknown dynamics. From the input and output data, learning gain in input update law is estimated using neural networks. We used PLTN as an approximating neural network. By using the proposed method, we can make the output trajectory of the unknown system except for system dimension and relative degree converge to the desired trajectory.

References

[1] J. McIntyre C. G. Atkeson. Robot trajectory learning through practice. *Proc. IEEE Conf. Robotics Autom.*, pages 1737-1742, 1986.

[2] T. J. Jang. Iterative learning control for nonlinear feedback systems. *Seoul National Univ. Ph.D. Thesis*, 1994.

[3] J. Y. Choi and R. M. Kil. Nonlinear estimation network with versatile bump shaping units for function approximations. *IEEE International Conference on Neural Networks (ICNN)*, pages 1359-1363, June 1994.

[4] W.S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bull. Math. Biophys.*, 21:115-133, 1943.

[5] G.C. Goodwin and R.L. Payne. *Dynamic System Identification: Experiment Design and Data Analysis*. Academic Press, 1977.