A New Control Method of Nonlinear Systems Based on Impulse Responses of Universal Learning Networks

Hirasawa, Kotaro
Department of Electrical and Electronic Systems Engineering, Kyushu University

Hu, Jinglu
Department of Electrical and Electronic Systems Engineering, Kyushu University

Murata, Junichi
Department of Electrical and Electronic Systems Engineering, Kyushu University

Jin, ChunZhi
Department of Control Engineering and Science, Kyushu Institute of Technology

https://doi.org/10.15017/1498352

出版情報：九州大学大学院システム情報科学紀要. 3 (2), pp.159-172, 1998-06-22. 九州大学大学院システム情報科学研究院
バージョン：
権利関係：

九州大学大学院
システム情報科学研究科報告
第3巻 第2号 平成10年9月

Research Reports on Information Science and
Electrical Engineering of Kyushu University
Vol.3, No.2, September 1998

A New Control Method of Nonlinear Systems Based on Impulse Responses of Universal Learning Networks

Kotaro HIRASAWA* , Jinglu HU* , Junichi MURATA* and ChunZhi JIN**

(Received June 22, 1998)

Abstract: A new control method of nonlinear dynamic systems is proposed based on impulse responses of Universal Learning Networks (ULNs). ULNs form a superset of neural networks. They consist of a number of inter-connected nodes where the nodes may have any continuously differentiable nonlinear functions in them and each pair of nodes can be connected by multiple branches with arbitrary (positive, zero, or even negative) time delays. A generalized learning algorithm is derived for the ULNs, in which both the first order derivatives (gradients) and the higher order derivatives are incorporated. The derivatives are calculated by using forward or backward propagation schemes. The algorithm can also be used in a unified manner for almost all kinds of learning networks. In this paper, not only the controlled object but also its controller are described by the ULNs and the controller is constructed by using the higher order derivatives of ULNs. The main feature of the proposed control method is to use impulse response defined by the higher order derivatives as a criterion function of the network. By using the impulse response, nonlinear dynamics with not only quick response but also quick damping can be more easily obtained than the conventional nonlinear control systems with quadratic form criterion functions of control variables.

Keywords: Large-scale complicated systems, Learning networks, Neural networks, Optimization, Control systems, Higher order derivatives, Quick response, Quick damping

1. Introduction

Since the first proposal of a neuron model by McCulloch and Pitts in 1940's, especially after the revitalization of artificial neural networks in 1980's, a variety of neural networks have been devised and are now applied in many fields. The vast majority of neural networks in use are those networks whose parameters or weights are tuned by gradient-based supervised learning. This category includes feedforward neural networks or multi-layer perceptrons, various types of recurrent neural networks, radial basis function networks, and some networks with special architectures, such as time delay neural networks. These networks seemingly have different architectures and are trained by distinguishable training algorithms. In essence, however, they can be unified in a single framework in regard to both their architectures and learning algorithms.

Universal Learning Networks (ULNs)[1] have been proposed, as the name indicates, to provide a universal framework for the class of neural networks. Unification of a variety of neural network architectures and their learning algorithms is an objective of ULNs; this provides a consistent viewpoint for the various kinds of neural networks. However, there is another benefit that is expected for ULNs. Generalization of the architectures and the gradient-based learning algorithms does not only give a unified description but attains new abilities of the networks. Allowing high degrees of freedom in their architectures gives them more flexibility and representing abilities, and therefore the ULNs can be useful and effective tools for modeling and controlling large-scale nonlinear complex systems including physical, social and economical phenomena. In addition to calculation of the first order derivatives of the signals flowing in the networks that are necessary in gradient-based learning, the generalized ULN learning algorithm is equipped with a systematic mechanism that calculates their second or higher order derivatives. This allows elaborate criterion functions to be used in their learning, which enables more sophisticated specification of the network performance.

This paper describes the ULNs and their application to controller design problems. In control systems, it is generally required to design a controller that meets three specifications namely quick response, quick damping and small steady state er-

* Department of Electrical and Electronic Systems Engineering

** Department of Control Engineering and Science, Kyushu Institute of Technology

ror. In the conventional control systems, even in the neural network control systems, the criterion function which is made up of quadratic form of state and control vectors is usually adopted in order to realize these specifications. But it has been recently recognized that the above mentioned criterion function is not sufficient to satisfy all the specifications required by the designer. In this paper, utilizing one of the features of ULNs, systematic calculation of the higher order derivatives, a new method employing the impulse response defined on ULNs as an extended part of the criterion function is proposed to overcome the problems in the conventional control. Before going into the details, it should be worth giving to the readers, as preliminary knowledge, the salient features of the ULNs and their differences from the existing alternatives.

A ULN consists of a number of inter-connected nodes. They may have any continuously differentiable functions in them, and each pair of nodes can be connected by multiple branches with arbitrary (positive, zero or even negative) time delays. Networks having time delays of multiple-length can be converted to those having unit-length time delays only by inserting an appropriate number of additional nodes between the nodes. However, zero or negative time delays can not be represented by the unit time delays. A ULN including negative time delays between the nodes can be used to model the systems with prediction mechanisms.

Learning of a ULN is defined as determination of its optimal parameter values that minimize a criterion function of the network. Generalized learning algorithms are derived based on two ideas. One is that static networks are merely special dynamic networks, and the other is that the derivatives of the node outputs can be obtained by either propagating the changes of the node outputs caused by the changes in the parameters through the network forward in time/space or propagating the changes caused by the changes in the other node outputs backward in time/space.

The most important feature of the learning of ULNs is use of the higher order derivatives of the criterion function with respect to parameters. Buntine [2] summarized the methods for calculating the second order derivatives in feedforward neural networks but in ULNs a generalized calculation method of the n-th order derivatives is proposed.

The ULNs, when applied to controller design problems, will exhibit the following advantages.

The ULNs allows any nonlinear functions to be embedded in their nodes. Therefore, both a controlled system and its controller can be represented by a single ULN. If the controlled system is unknown, its identification can also be done by the ULN learning algorithm [3]. Then an optimal controller can be obtained by off-line gradient learning.

The ULN controllers have a similarity with the traditional nonlinear optimal controllers in the sense that both are constructed by minimizing criterion functions or performance indexes. However, the control laws obtained by traditional controller design method are, in most cases, feedforward control laws[4], and the control signals are given as functions of time. On the other hand, the ULN controllers form feedback control schemes and their parameters are determined by optimization. Therefore, although the ULN controllers do not attain better performance indexes because of their structural constraint, they exhibit better robustness against external disturbances due to their feedback configuration.

The higher order derivative calculation mechanism of the ULNs renders additional advantages to the ULN controllers. As a criterion function, one may employ a standard control performance index, for example, tracking error of the controlled system. In addition to this, one can put other terms to his/her criterion functions. Let us assume that our additional term is the sensitivity of the standard criterion function with respect to the changes in the plant parameters. Minimization of the sensitivity by a gradient method requires the second order derivatives of the standard criterion function since the sensitivity itself is the first order derivative. This derivative can be calculated by the proposed mechanism. Minimizing this extended criterion function will result in a robust controlled system[5][6] that is not much affected by changes in the plant parameters. When a Lyapunov exponent of the ULN is adopted as the extended criterion function, its chaotic behavior can be easily realized[7].

The proposed control method aiming at both quick response and quick damping at the same time also employs an additional term based on the impulse response of the ULNs, which is again defined by the higher order derivative.

In Section 2, the structure and the learning algorithms of ULNs are described. The specific procedures for calculating the higher order derivatives are presented in Sections 3 and 4. Section 3 presents the backward propagation procedure, while Section 4

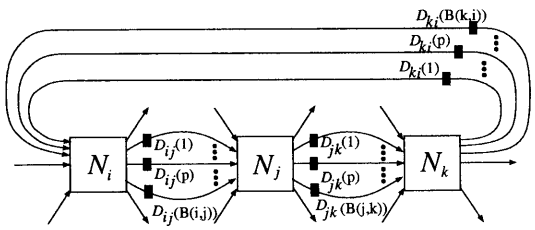


Fig.1 Architecture of Universal Learning Network

describes the forward propagation scheme and compares the computational complexities of the both procedures. In Section 5, a new control method of nonlinear systems based on the impulse response of ULNs is presented. Section 6 gives the simulation results of a nonlinear crane control system, where a good controller in terms of steady state error, response and damping speed is presented and the usefulness of the proposed method is illustrated.

2. Universal Learning Networks[1)8)9)]

In this section, we discuss the structure and learning algorithms of ULNs.

2.1 Structure of ULNs

As shown in **Fig.1**, a ULN consists of a number of nodes and branches for inter-connecting the nodes. The nodes may have any continuously differentiable nonlinear functions in them, and each pair of nodes can be connected each other by multiple branches with arbitrary (positive, zero or even negative) time delays[10)11)]. ULNs including negative time delays can be used to model systems that have prediction mechanisms in them.

The generic equation that describes the behavior of ULNs is expressed as follows,

$$\begin{aligned} &h_j(t) \\ &= f_j\left[\{h_i(t - D_{ij}(p)) | i \in JF(j), p \in B(i,j)\},\right. \\ &\quad \left.\{r_n(t) \mid n \in N(j)\}, \{\lambda_m(t) \mid m \in M(j)\}\right], \quad (1) \\ &\qquad\qquad j \in J, t \in T, \end{aligned}$$

where

$h_j(t)$: output value of node j at time t,
$r_n(t)$: value of n-th external input variable at time t,
$\lambda_m(t)$: value of m-th parameter at time t,
f_j : nonlinear function of node j,
$D_{ij}(p)$: time delay of p-th branch from node i to node j,
J : set of nodes $\{j\}$,
$JF(j)$: set of nodes which are connected to node j,
$JB(j)$: set of nodes which are connected from node j,
N : set of external input variables $\{n\}$,
$N(j)$: $\{n | r_n$ is fed to node $j\}$,
M : set of parameters $\{m\}$,
$M(j)$: $\{m | h_j$ is partially differentiable with respect to $\lambda_m\}$,
$B(i,j)$: set of branches from node i to node j,
T : discrete set of sampling times.

The ULNs operate on a discrete-time basis. Each pair of nodes i and j may have multiple branches between them i.e. set $B(i,j)$ may contain more than one elements, and parameters λ_m can be time-varying. Functions $f_j(\cdot)$ that govern the operation of the nodes can be any continuously differentiable functions; typically, sigmoid functions can be employed. And in that case, Eq.(1) can be expressed specifically as,

$$h_j(t) = f_j(\alpha_j(t)) = \frac{1}{1 + e^{-\phi_j \alpha_j(t)}}, \quad (2)$$

$$\alpha_j(t) = \sum_{i \in JF(j)} \sum_{p \in B(i,j)} \omega_{ij}(p) h_i(t - D_{ij}(p)) + \theta_j, \quad (3)$$

where

$w_{ij}(p)$: weight parameter of p-th branch from node i to node j,
θ_j: threshold parameter of node j,
ϕ_j: slope parameter of node j.

2.2 Learning of ULNs

Learning of ULNs is realized by minimizing a criterion function L based on the gradient method:

$$\lambda_1 \leftarrow \lambda_1 - \gamma \frac{\partial^\dagger L}{\partial \lambda_1}, \quad (4)$$

where γ is the learning coefficient assigned with a small positive value and $\partial^\dagger L / \partial \lambda_1$ is the ordered derivative defined by Werbos in 12) which means the change of the criterion function L caused by the change of λ_1 with other variables being fixed.

The criterion function L usually consists of two parts, a fundamental part E and an extended part E_x:

$$L = E + E_x. \quad (5)$$

The fundamental part E is defined as a function of any node outputs, any parameters and any time

instants:

$$E = E(\{h_r(s)\}, \{\lambda_m(s)\}), \quad r \in J_0, m \in M_0, s \in T_0, \tag{6}$$

where

J_0: $\{r|h_r$ is related to evaluation of criterion function$\}$,

M_0: $\{m|\lambda_m$ is related to evaluation of criterion function$\}$,

T_0: $\{s|$ time instant s is related to evaluation of criterion function$\}$.

A typical choice of E is sum of errors between the network outputs and their desired values.

The extended part E_x is a function of derivatives of node output $h_r(s)$. Therefore, this part is related to some notions that can not be represented by the fundamental criterion E, such as, smoothness, robustness, and stability, and thus it renders the ULNs a variety of advantageous features. The cost that one might have to pay for these advantages is computational complexity in the gradient-based optimization of the generalized criterion function L. The gradient of L includes not only the first order derivatives of node output $h_r(s)$ but also its higher order derivatives since E_x in L itself contains the derivatives of $h_r(s)$. In the subsequent sections, we will describe systematic calculation methods of the derivatives. Of course we have to endure the additional computational cost for calculation of the higher order derivatives. However, the calculation itself is not complicated and is easily implemented. Since E is a differentiable function of node outputs $h_r(s)$, calculation of the derivatives of E and computation of those of $h_r(s)$ are essentially identical. Therefore, for clarity, in the following two sections we will discuss the first and higher order derivatives of E only. Their computational complexity will be evaluated and compared in Section 4.3.

3. Derivatives Calculation Method by Backward Propagation

In this section, a backward calculation method of the first order derivatives and the second order derivatives of the criterion function E (Eq.(6)) with respect to the parameters will be presented, which can be easily extended to calculation of the n-th order derivatives.

3.1 The First Order Derivatives

Let us evaluate the change of criterion function E caused by a change of a time-varying parameter $\lambda_1(t)$ at time $t = t_1$. It should be noted that the parameter is changed at time t_1 only with keeping its values at other time instants unchanged. The parameter may influence the function directly or indirectly. The direct influence can be evaluated by a partial derivative $\partial E/\partial\lambda_1(t_1)$. To evaluate the indirect influence, we need to take the roles of some intermediate variables into account that bridge the function E and the parameter $\lambda_1(t_1)$. We can adopt node outputs $h_d(t_1)$ which are directly affected by the parameter as those intermediate variables. Then, the first order derivative $\partial^\dagger E/\partial\lambda_1(t_1)$ can be represented as follows,

$$\frac{\partial^\dagger E}{\partial\lambda_1(t_1)} = \sum_{d\in JD(\lambda_1)}\left[\frac{\partial^\dagger E}{\partial h_d(t_1)}\frac{\partial h_d(t_1)}{\partial\lambda_1(t_1)}\right] + \frac{\partial E}{\partial\lambda_1(t_1)}, \tag{7}$$

where

t_1 : a certain specified time instant,

$\lambda_1(t_1)$: value of parameter at time t_1,

$JD(\lambda_1)$: $\{d|h_d(t_1)$ is partially differentiable with respect to $\lambda_1(t_1)\}$.

Since node outputs $h_d(t_1)$ may affect E directly or indirectly, we need to consider the ordered derivative $\partial^\dagger E/\partial h_d(t_1)$ here. Now, let us denote the ordered derivative as $\delta_1(j,t)$:

$$\delta_1(j,t) = \frac{\partial^\dagger E}{\partial h_j(t)}. \tag{8}$$

Its evaluation again requires the help of some intermediate variables, which leads to the following back-propagation algorithm,

$$\delta_1(j,t) = \sum_{k\in JB(j)}\sum_{p\in B(j,k)} \left[\frac{\partial h_k(t+D_{jk}(p))}{\partial h_j(t)}\delta_1(k,t+D_{jk}(p))\right] + \frac{\partial E}{\partial h_j(t)}, \quad j\in J, t\in T. \tag{9}$$

The first term in the right-hand side indicates the indirect effect which is calculated taking the outputs of the *downstream* nodes k as the intermediate variables, and the second term shows the direct effect of node output $h_j(t)$ on the criterion E.

Substituting this signal $\delta_1(j,t)$ into Eq.(7), we can derive the first order derivative of E with respective to a time-varying parameter $\lambda_1(t)$ at time t_1, and also the derivative with respective to a time-invariant parameter λ_1 can be obtained in a similar way. For static networks, the formulas are reduced to simpler forms. The calculation procedures of the first order derivatives for time-varying parameters,

time-invariant parameters and static networks are summarized as follows,

- For time-varying parameters,

$$\frac{\partial^\dagger E}{\partial \lambda_1(t_1)} = \sum_{d \in JD(\lambda_1)} [\frac{\partial h_d(t_1)}{\partial \lambda_1(t_1)} \delta_1(d, t_1)] + \frac{\partial E}{\partial \lambda_1(t_1)}, \quad (10)$$

- For time-invariant parameters,

$$\frac{\partial^\dagger E}{\partial \lambda_1} = \sum_{t' \in T} \sum_{d \in JD(\lambda_1)} [\frac{\partial h_d(t')}{\partial \lambda_1} \delta_1(d, t')] + \frac{\partial E}{\partial \lambda_1}, \quad (11)$$

- For static networks,

$$\frac{\partial^\dagger E}{\partial \lambda_1} = \sum_{d \in JD(\lambda_1)} [\frac{\partial h_d}{\partial \lambda_1} \delta_1(d)] + \frac{\partial E}{\partial \lambda_1}, \quad (12)$$

$$\delta_1(j) = \sum_{k \in JB(j)} [\frac{\partial h_k}{\partial h_j} \delta_1(k)] + \frac{\partial E}{\partial h_j}, \quad j \in J. \quad (13)$$

The summation over $d \in JD(\lambda_1)$ in Eqs.(10), (11) and (12) reflects the fact that there are a number of nodes whose outputs are partially differentiable with respect to a certain λ_1. Although a change of a time-varying parameter at a specific time instant t_1 directly affects node outputs $h_d(t)$ at time $t = t_1$ only, a change in a time-invariant parameter directly causes change of $h_d(t)$ at any time. Therefore, we need to consider $\partial h_d(t')/\partial \lambda_1$ at any time t' and sum them up. This is the reason why Eq.(11) only contains a summation over t'. In the algorithm (9) - (13), the derivative $\partial^\dagger E/\partial \lambda_1$ is calculated by propagating δ_1 or the derivative of the function E with respect to the node outputs backward in time (for dynamic networks) or backward in space (for static networks), which is, in essence, identical to the back-propagation algorithm for ordinary neural networks. In other words, the learning algorithm includes the ordinary back-propagation as one of its special examples.

3.2 The Second Order Derivatives

The first order derivative calculation discussed in the preceding subsection can be easily extended to calculation of the n-th order derivatives. Details of the n-th order derivative calculation method by backward propagation can be found in 8). In this subsection, we give the calculation method of the second order derivatives as an example.

Let t_1 and t_2 be certain specified time instants, and $\lambda_1(t)$ and $\lambda_2(t)$ be parameters. Then by differentiating Eq.(10) with respect to $\lambda_2(t_2)$, we have the second order derivative $\partial^{\dagger 2} E/\partial \lambda_1(t_1) \partial \lambda_2(t_2)$ as,

$$\begin{aligned}
&\frac{\partial^{\dagger 2} E}{\partial \lambda_1(t_1) \partial \lambda_2(t_2)} \\
&= \sum_{d \in JD(\lambda_1)} [\frac{\partial h_d(t_1)}{\partial \lambda_1(t_1)} \frac{\partial^\dagger \delta_1(d, t_1)}{\partial \lambda_2(t_2)}] \\
&\quad + \sum_{d \in JD(\lambda_1)} [\frac{\partial^\dagger (\frac{\partial h_d(t_1)}{\partial \lambda_1(t_1)})}{\partial \lambda_2(t_2)} \delta_1(d, t_1)] \\
&\quad + \frac{\partial^\dagger (\frac{\partial E}{\partial \lambda_1(t_1)})}{\partial \lambda_2(t_2)}. \quad (14)
\end{aligned}$$

Let us define $\delta_{12}(d, t_1) = \partial^\dagger \delta_1(d, t_1)/\partial \lambda_2(t_2) = \partial^\dagger (\partial^\dagger E/\partial h_d(t_1))/\partial \lambda_2(t_2)$. It can be calculated by differentiating (9) with respect to $\lambda_2(t_2)$:

$$\begin{aligned}
\delta_{12}(j, t) = &\sum_{k \in JB(j)} \sum_{p \in B(j,k)} [\frac{\partial h_k(t + D_{jk}(p))}{\partial h_j(t)} \\
&\times \delta_{12}(k, t + D_{jk}(p))] \\
&+ \sum_{k \in JB(j)} \sum_{p \in B(j,k)} [\frac{\partial^\dagger (\frac{\partial h_k(t + D_{jk}(p))}{\partial h_j(t)})}{\partial \lambda_2(t_2)} \\
&\times \delta_1(k, t + D_{jk}(p))] \\
&+ \frac{\partial^\dagger (\frac{\partial E}{\partial h_j(t)})}{\partial \lambda_2(t_2)}, \quad j \in J, \ t \in T. \quad (15)
\end{aligned}$$

In Eqs.(14) and (15), $\partial^\dagger (\partial h_d(t_1)/\partial \lambda_1(t_1))/\partial \lambda_2(t_2)$, $\partial^\dagger (\partial E/\partial \lambda_1(t_1))/\partial \lambda_2(t_2)$, $\partial^\dagger (\partial h_k(t + D_{jk}(p))/\partial h_j(t))/\partial \lambda_2(t_2)$ and $\partial^\dagger (\partial E/\partial h_j(t))/\partial \lambda_2(t_2)$ can be calculated by applying the calculation procedure of the first order derivatives, with E being replaced with $\partial h_d(t_1)/\partial \lambda_1(t_1)$, $\partial E/\partial \lambda_1(t_1)$, $\partial h_k(t + D_{jk}(p))/\partial h_j(t)$ and $\partial E/\partial h_j(t)$, respectively.

For time-invariant parameters and static ULNs, the calculation procedure is rewritten as follows:

- For time-invariant parameters,

$$\begin{aligned}
\frac{\partial^{\dagger 2} E}{\partial \lambda_1 \partial \lambda_2} = &\sum_{t' \in T} \sum_{d \in JD(\lambda_1)} [\frac{\partial h_d(t')}{\partial \lambda_1} \delta_{12}(d, t')] \\
&+ \sum_{t' \in T} \sum_{d \in JD(\lambda_1)} [\frac{\partial^\dagger (\frac{\partial h_d(t')}{\partial \lambda_1})}{\partial \lambda_2} \delta_1(d, t')] \\
&+ \frac{\partial^\dagger (\frac{\partial E}{\partial \lambda_1})}{\partial \lambda_2}, \quad (16)
\end{aligned}$$

$$\begin{aligned}
\delta_{12}(j, t) = &\sum_{k \in JB(j)} \sum_{p \in B(j,k)} [\frac{\partial h_k(t + D_{jk}(p))}{\partial h_j(t)} \\
&\times \delta_{12}(k, t + D_{jk}(p))]
\end{aligned}$$

$$+\sum_{k\in JB(j)}\sum_{p\in B(j,k)}\left[\frac{\partial^{\dagger}\left(\frac{\partial h_k(t+D_{jk}(p))}{\partial h_j(t)}\right)}{\partial\lambda_2}\right.$$
$$\left.\times\delta_1(k,t+D_{jk}(p))\right]$$
$$+\frac{\partial^{\dagger}\left(\frac{\partial E}{\partial h_j(t)}\right)}{\partial\lambda_2},\quad j\in J,\ t\in T. \tag{17}$$

- For static networks,

$$\frac{\partial^{\dagger 2}E}{\partial\lambda_1\partial\lambda_2}=\sum_{d\in JD(\lambda_1)}\left[\frac{\partial h_d}{\partial\lambda_1}\delta_{12}(d)\right]$$
$$+\sum_{d\in JD(\lambda_1)}\left[\frac{\partial^{\dagger}\left(\frac{\partial h_d}{\partial\lambda_1}\right)}{\partial\lambda_2}\delta_1(d)\right]+\frac{\partial^{\dagger}\left(\frac{\partial E}{\partial\lambda_1}\right)}{\partial\lambda_2}, \tag{18}$$

$$\delta_{12}(j)=\sum_{k\in JB(j)}\left[\frac{\partial h_k}{\partial h_j}\delta_{12}(k)\right]$$
$$+\sum_{k\in JB(j)}\left[\frac{\partial^{\dagger}\left(\frac{\partial h_k}{\partial h_j}\right)}{\partial\lambda_2}\delta_1(k)\right]+\frac{\partial^{\dagger}\left(\frac{\partial E}{\partial h_j}\right)}{\partial\lambda_2},\quad j\in J. \tag{19}$$

The n-th order derivative $\frac{\partial^{\dagger n}E}{\partial\lambda_1(t_1)\partial\lambda_2(t_2)\cdots\partial\lambda_n(t_n)}$ can be obtained by evaluating iterative equations for δ_1, δ_{12}, δ_{13}, $\cdots$, $\delta_{12\cdots n-1}$, $\delta_{12\cdots n}=\frac{\partial^{\dagger}\delta_{12\cdots n-1}}{\partial\lambda_n}$. Details of the calculation method of n-th order derivatives can be found in 8).

4. Derivatives Calculation Method by Forward Propagation

In this section, we will introduce a new calculation scheme for the n-th order derivatives of the criterion function E by using forward propagation method. In the forward propagation calculation, the change of node outputs with respect to changes in the parameters propagate forwards, whereas the changes of criterion function with respective to changes in node outputs propagate backwards in the backward propagation calculation. Details of the n-th order derivatives calculation by forward propagation method can be found in 9). In this section, only the calculation of the first and second order derivatives is discussed as examples.

4.1 The First Order Derivatives

In contrast to the backward propagation procedure described in the previous section, let us think of evaluating the indirect effect of the changes in the parameter on the criterion function E by considering the node outputs that directly influence E as the intermediate variables. Then, the first order derivative of the function E with respect to a time-varying parameter $\lambda_1(t)$ at time t_1 can be calculated as follows,

$$\frac{\partial^{\dagger}E}{\partial\lambda_1(t_1)}=\sum_{r\in J_0}\sum_{s\in T_0}\left[\frac{\partial E}{\partial h_r(s)}\frac{\partial^{\dagger}h_r(s)}{\partial\lambda_1(t_1)}\right]+\frac{\partial E}{\partial\lambda_1(t_1)}, \tag{20}$$

where

t_1 : a certain specified time instant,
$\lambda_1(t_1)$: value of parameter at time t_1,
J_0 : $\{r|h_r$ is related to evaluation of $E\}$,
T_0 : $\{s|$time instant s at which E is evaluated$\}$.

Eq.(20) shows that the change of the criterion function caused by the change of parameters can be reduced to the change of the outputs of the nodes related to evaluation of the function. The ordered derivative of node output $h_j(t)$ with respect to the time-varying parameter $\lambda_1(t_1)$,

$$P_1(j,t,\lambda_1(t_1))=\frac{\partial^{\dagger}h_j(t)}{\partial\lambda_1(t_1)} \tag{21}$$

can be calculated using the following forward propagation procedure:

$$P_1(j,t,\lambda_1(t_1))$$
$$=\sum_{i\in JF(j)}\sum_{p\in B(i,j)}\left[\frac{\partial h_j(t)}{\partial h_i(t-D_{ij}(p))}\right.$$
$$\left.\times P_1(i,t-D_{ij}(p),\lambda_1(t_1))\right]+\frac{\partial h_j(t)}{\partial\lambda_1(t_1)},\quad j\in J,t\in T. \tag{22}$$

The first term in the right-hand side indicates the indirect effect which is calculated taking the outputs of the *upstream* nodes i as the intermediate variables, while the second term shows the direct effect of parameter value $\lambda_1(t_1)$ on node output $h_j(t)$. For time-invariant parameters and static networks, Eq.(20) and (22) can be reduced to the following simpler forms.

- For time-invariant parameters,

$$\frac{\partial^{\dagger}E}{\partial\lambda_1}=\sum_{r\in J_0}\sum_{s\in T_0}\left[\frac{\partial E}{\partial h_r(s)}P_1(r,s,\lambda_1)\right]+\frac{\partial E}{\partial\lambda_1}, \tag{23}$$

$$P_1(j,t,\lambda_1)=\sum_{i\in JF(j)}\sum_{p\in B(i,j)}\left[\frac{\partial h_j(t)}{\partial h_i(t-D_{ij}(p))}\right.$$
$$\left.\times P_1(i,t-D_{ij}(p),\lambda_1)\right]+\frac{\partial h_j(t)}{\partial\lambda_1},\quad j\in J,t\in T, \tag{24}$$

where $P_1(j,t,\lambda_1)=\partial^{\dagger}h_j(t)/\partial\lambda_1$.

- For static networks,

$$\frac{\partial^{\dagger}E}{\partial\lambda_1}=\sum_{r\in J_0}\left[\frac{\partial E}{\partial h_r}P_1(r,\lambda_1)\right]+\frac{\partial E}{\partial\lambda_1}, \tag{25}$$

$$P_1(j,\lambda_1)=\sum_{i\in JF(j)}\left[\frac{\partial h_j}{\partial h_i}P_1(i,\lambda_1)\right]+\frac{\partial h_j}{\partial \lambda_1},\quad j\in J, \tag{26}$$

where $P_1(j,\lambda_1)=\partial^\dagger h_j/\partial\lambda_1$.

In the backward propagation scheme, the calculation of the first order derivative of E with respec to a time-invariant parameter λ_1 needed a summation over $t'\in T$ (see Eq.(11)). This was because a change in a time-invariant parameter directly causes change of $h_d(t)$ at any time. In contrast to this, the forward propagation procedure for the time-invariant parameters does not contain such a summation. The reason for this is that, in the forward propagation scheme, the sensitivity of a node output $h_r(s)$ with respect to the change in a parameter λ_1 is evaluated by an ordered derivative which incorporates all the direct and indirect effects. The indirect effects are calculated by following a chain of spatial and temporal causes-and-effects as shown in Eqs. (22), (24) and (26). Therefore, $P_1(j,t,\lambda_1)$ includes all the effects at any time $t'\le t$ caused by the change of λ_1.

4.2 The Second Order Derivatives

Let t_1 and t_2 be certain specified time instants, and $\lambda_1(t)$ and $\lambda_2(t)$ be time-varying parameters. Then the second order derivative $\partial^{\dagger 2}E/\partial\lambda_1(t_1)\partial\lambda_2(t_2)$ can be obtained by differentiating Eq.(20) with respect to $\lambda_2(t_2)$:

$$\begin{aligned}&\frac{\partial^{\dagger 2}E}{\partial\lambda_1(t_1)\partial\lambda_2(t_2)}\\&=\sum_{r\in J_0}\sum_{s\in T_0}\left[\frac{\partial^\dagger(\frac{\partial E}{\partial h_r(s)})}{\partial\lambda_2(t_2)}\frac{\partial^\dagger h_r(s)}{\partial\lambda_1(t_1)}\right.\\&\left.+\frac{\partial E}{\partial h_r(s)}\frac{\partial^{\dagger 2}h_r(s)}{\partial\lambda_1(t_1)\partial\lambda_2(t_2)}\right]+\frac{\partial^\dagger(\frac{\partial E}{\partial\lambda_1(t_1)})}{\partial\lambda_2(t_2)}.\end{aligned} \tag{27}$$

Then introducing

$$\begin{aligned}&P_2(j,t,\lambda_1(t_1),\lambda_2(t_2))\\&=\frac{\partial^{\dagger 2}h_j(t)}{\partial\lambda_1(t_1)\lambda_2(t_2)}=\frac{\partial^\dagger P_1(j,t,\lambda_1(t_1))}{\lambda_2(t_2)},\end{aligned} \tag{28}$$

and differentiating Eq.(22) with respect to $\lambda_2(t_2)$, the following iterative formula for P_2 can be obtained:

$$\begin{aligned}&P_2(j,t,\lambda_1(t_1),\lambda_2(t_2))\\&=\sum_{i\in JF(j)}\sum_{p\in B(i,j)}\left[\frac{\partial^\dagger(\frac{\partial h_j(t)}{\partial h_i(t-D_{ij}(p))})}{\partial\lambda_2(t_2)}\right.\\&\left.\times P_1(i,t-D_{ij}(p),\lambda_1(t_1))\right]\\&+\sum_{i\in JF(j)}\sum_{p\in B(i,j)}\left[\frac{\partial h_j(t)}{\partial h_i(t-D_{ij}(p))}\right.\\&\left.\times P_2(i,t-D_{ij}(p),\lambda_1(t_1),\lambda_2(t_2))\right]\\&+\frac{\partial^\dagger(\frac{\partial h_j(t)}{\partial\lambda_1(t_1)})}{\partial\lambda_2(t_2)},\quad j\in J,t\in T.\end{aligned} \tag{29}$$

The first order derivatives in Eqs.(27) and (29), for example $\partial^\dagger(\partial E/\partial h_r(s))/\partial\lambda_2(t_2)$, can be calculated by replacing $\partial E/\partial h_r(s)$ with E and again using the first order derivative calculation.

Furthermore, as in the preceding section, Eqs.(27) and (29) can be transformed to the following simplified expressions for time-invariant parameters and static networks.

- For time-invariant parameters,

$$\begin{aligned}\frac{\partial^{\dagger 2}E}{\partial\lambda_1\partial\lambda_2}&=\sum_{r\in J_0}\sum_{s\in T_0}\left[\frac{\partial^\dagger(\frac{\partial E}{\partial h_r(s)})}{\partial\lambda_2}P_1(r,s,\lambda_1)\right.\\&\left.+\frac{\partial E}{\partial h_r(s)}P_2(r,s,\lambda_1,\lambda_2)\right]+\frac{\partial^\dagger(\frac{\partial E}{\partial\lambda_1})}{\partial\lambda_2},\end{aligned} \tag{30}$$

$$\begin{aligned}&P_2(j,t,\lambda_1,\lambda_2)\\&=\sum_{i\in JF(j)}\sum_{p\in B(i,j)}\left[\frac{\partial^\dagger(\frac{\partial h_j(t)}{\partial h_i(t-D_{ij}(p))})}{\partial\lambda_2}\right.\\&\left.\times P_1(i,t-D_{ij}(p),\lambda_1)\right]\\&+\sum_{i\in JF(j)}\sum_{p\in B(i,j)}\left[\frac{\partial h_j(t)}{\partial h_i(t-D_{ij}(p))}\right.\\&\left.\times P_2(i,t-D_{ij}(p),\lambda_1,\lambda_2)\right]\\&+\frac{\partial^\dagger(\frac{\partial h_j(t)}{\partial\lambda_1})}{\partial\lambda_2},\quad j\in J,\ t\in T,\end{aligned} \tag{31}$$

where $P_2(j,t,\lambda_1,\lambda_2)=\partial^{\dagger 2}h_j(t)/\partial\lambda_1\partial\lambda_2$.

- For static networks,

$$\begin{aligned}&\frac{\partial^{\dagger 2}E}{\partial\lambda_1\partial\lambda_2}\\&=\sum_{r\in J_0}\left[\frac{\partial^\dagger(\frac{\partial E}{\partial h_r})}{\partial\lambda_2}P_1(r,\lambda_1)+\frac{\partial E}{\partial h_r}P_2(r,\lambda_1,\lambda_2)\right]\\&+\frac{\partial^\dagger(\frac{\partial E}{\partial\lambda_1})}{\partial\lambda_2},\end{aligned} \tag{32}$$

$$\begin{aligned}&P_2(j,\lambda_1,\lambda_2)\\&=\sum_{i\in JF(j)}\left[\frac{\partial^\dagger(\frac{\partial h_j}{\partial h_i})}{\partial\lambda_2}P_1(i,\lambda_1)\right]\\&+\sum_{i\in JF(j)}\left[\frac{\partial h_j}{\partial h_i}P_2(i,\lambda_1,\lambda_2)\right]+\frac{\partial^\dagger(\frac{\partial h_j}{\partial\lambda_1})}{\partial\lambda_2},\\&j\in J,\end{aligned} \tag{33}$$

where $P_2(j,\lambda_1,\lambda_2)=\partial^{\dagger 2}h_j/\partial\lambda_1\partial\lambda_2$.

The n-th order derivative $\frac{\partial^{\dagger n} E}{\partial\lambda_1(t_1)\partial\lambda_2(t_2)\cdots\partial\lambda_n(t_n)}$ can be obtained by calculating $P_1, P_2, \cdots, P_n$ iteratively, where

$$P_3(j,t,\lambda_1(t_1),\lambda_2(t_2),\lambda_3(t_3)) = \frac{\partial^{\dagger 3} h_j(t)}{\partial\lambda_1(t_1)\partial\lambda_2(t_2)\partial\lambda_3(t_3)}, \tag{34}$$

$$\vdots$$

$$P_n(j,t,\lambda_1(t_1),\ \cdots\ ,\lambda_n(t_n)) = \frac{\partial^{\dagger n} h_j(t)}{\partial\lambda_1(t_1)\ \cdots\ \partial\lambda_n(t_n)}. \tag{35}$$

Details of calculation of the n-th order derivatives can be found in 9).

4.3 Computational Complexity of Backward and Forward Propagation Schemes

Comparing the forward propagation scheme with the backward propagation scheme discussed in the previous section, we can see that they are similar in form. However, the computation loads required in these two schemes are very different. Since $P_1(j,t,\lambda_1(t_1))$ is a function of parameter $\lambda_1(t_1)$, we need to calculate P_1 for every parameter. On the other hand, $\delta_1(j,t)$ in the backward propagation scheme does not depend on parameters, and therefore we only need to calculate a single time function $\delta_1(j,t)$ for a given node j. This results in, for calculation of the first order derivatives, a larger computational load of the forward propagation scheme by a factor of the number of parameters involved. For this reason, the forward propagation algorithm is rarely used for the calculation of the first order derivatives.

The situations are different for calculation of the higher order derivatives. The forward propagation scheme requires lower computational load when applied to calculation of the higher order derivatives. Let $|J|$ be the number of nodes, $|T|$ be the number of sampling instants and $|M|$ be the number of parameters. As explained in 8)9), in the cases of time-invariant parameter systems, the computational load of the n-th order derivatives by forward propagation is proportional to $|J|^2|T||M|^n$, whereas it is proportional to $|J|^{2n}|T|^n$ by backward propagation. In practical applications, for instance controller design, a large number of sampling times are usually involved. Therefore, when the higher order derivatives are required, it is highly recommended to use the forward propagation scheme.

5. Nonlinear Control Method Based on Impulse Response

In control systems, it is generally required to design a controller which meets the specifications such as small steady state error, quick response and quick damping. In the conventional linear control theory, those requirements were pursued by either pole assignment[13] or optimal control which minimizes quadratic form of the state and control vectors[14]. Recently, Neural Networks are widely used to control nonlinear systems[15]. But even in the neural network control systems, the criterion function is usually made up of quadratic form of the state vector and control vector, which means minimization of the tracking error of the controlled system and energy consumption of the system.

One of the assertion of this paper is that as far as the above mentioned criterion function of quadratic form is employed, it is fairly difficult to satisfy all the specifications required by the designer. So, in this paper, a new method employing the impulse response defined on ULNs as an extended part of the criterion function is proposed in order to realize the quick response and quick damping at the same time, and also to minimize the steady state error of the system.

The reason why impulse response is more preferable than quadratic form of the control vector as an extended part of the criterion function is that the dynamics of each node of the system can be controlled more precisely by using the impulse response.

The impulse response is defined by using the higher order derivatives or approximately by the first order derivatives of ULNs, therefore, calculation method of higher order derivatives stated in the preceding sections is necessary to train the controller described by ULNs.

5.1 Criterion Function Based on Impulse Response

As the ULNs are discrete time recurrent networks, the impulse response $\Delta h_{nk}(t)$ of node k at time t when n-th external input $r_n(t)$ has changed by $\Delta r_n(t)$ at time t_0 is described using Taylor's series as follows,

$$\Delta h_{nk}(t) = \sum_{\ell=1}^{\infty} \frac{1}{\ell} \frac{\partial^{\dagger \ell} h_k(t)}{\partial r_n(t_0)^{\ell}} \Delta r_n(t_0), \quad n \in N, k \in J', \tag{36}$$

where

N : set of external input variables $\{n\}$,
J' : $\{j \mid h_j$ is the node of controlled system$\}$.

For simplicity, from now on, the following approximate definition of the impulse response by the first order derivative is used,

$$\Delta h_{nk}(t) = \frac{\partial^\dagger h_k(t)}{\partial r_n(t_0)} \Delta r_n(t_0) \quad n \in N, k \in J'. \quad (37)$$

As shown in Eq.(5), the criterion function L usually consists of two parts, a fundamental part E and an extended part E_x.

The proposed method adopts the following criterion functions using the above impulse response,

$$L = E + E_{xp}, \quad (38)$$

$$E = \sum_{t \in T} \sum_{k \in J'} Q_k (h_k(t) - h_k^0(t))^2, \quad (39)$$

$$E_{xp} = \sum_{t \in T} \sum_{k \in J'} \sum_{n \in N} Q_{nk} \left[\frac{\partial^\dagger h_k(t)}{\partial r_n(t_0)} \Delta r_n(t_0) - g_{nk}(t) \right]^2, \quad (40)$$

where

$h_k^0(t)$: target value of $h_k(t)$,
$g_{nk}(t)$: target value of $\frac{\partial^\dagger h_k(t)}{\partial r_n(t_0)} \Delta r_n(t_0)$,
Q_k, Q_{nk} : coefficients,
T : discrete set of sampling times.

On the other hand, commonly used neural network control systems adopt the following conventional criterion function,

$$L = E + E_{xc}, \quad (41)$$

$$E = \sum_{t \in T} \sum_{k \in J'} Q_k (h_k(t) - h_k^0(t))^2, \quad (42)$$

$$E_{xc} = \sum_{t \in T} \sum_{r \in R} R_r (u_r(t))^2, \quad (43)$$

where

R : set of control variables $\{r\}$,
R_r : coefficient.

The proposed method is different in its extended criterion function from the conventional method.

It is obvious from Eq.(40) and Eq.(43) that the output of each node in the system can be more easily controlled by the proposed method because $g_{nk}(t)$ can be freely chosen for all $h_k(t)$ $k \in J'$ and all $r_n(t_0)$ $n \in N$.

In this paper, $g_{nk}(t)$, the target value of $\frac{\partial^\dagger h_k(t)}{\partial r_n(t_0)} \Delta r_n(t_0)$, is supposed to be given refering to the following response of the second order linear systems.

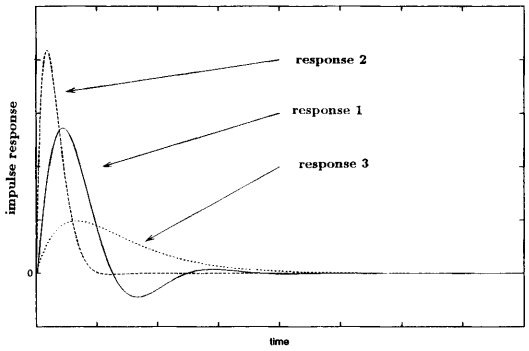


Fig.2 Specification of impulse response

<In the case of specifying response and damping characteristic>

$$g_{nk}(t) = K_{nk} e^{-\xi_{nk}\omega_{nk}t} \sin(\omega_{nk}\sqrt{1-\xi_{nk}^2}\,t) = K_{nk} e^{-a_{nk}t} \sin b_{nk} t, \quad (44)$$

where

$$a_{nk} = \xi_{nk}\omega_{nk}, \qquad b_{nk} = \omega_{nk}\sqrt{1-\xi_{nk}^2}.$$

< In the case of specifying damping characteristics only >

$$g_{nk}(t) = K_{nk} t e^{-\xi_{nk}t}, \quad (45)$$

where

K_{nk} : gain of the system,
ω_{nk} : natural angular frequency,
ξ_{nk} : damping coefficient.

It is known that the bigger the ω_{nk} is, the quicker response is obtained, and that also the bigger the ξ_{nk} is, the more attenuated response the system shows.

When designing the controller based on the impulse response, $K_{nk}, \omega_{nk}, \xi_{nk}$ can be determined by the following procedure(refer to **Fig.2**):

1. As is executed in the usual design of neural controller, parameters of the controller are trained by using the conventional criterion functions, that is, Eq.(41).
2. Evaluate the impulse response of the system obtained by the procedure 1 (let us assume our response is response 1 in **Fig.2**).
3. $K_{nk}, \omega_{nk}, \xi_{nk}$ are determined according to, *e.g.* whether quicker and attenuated response should be obtained (response 2) or less quick response is required (response 3).

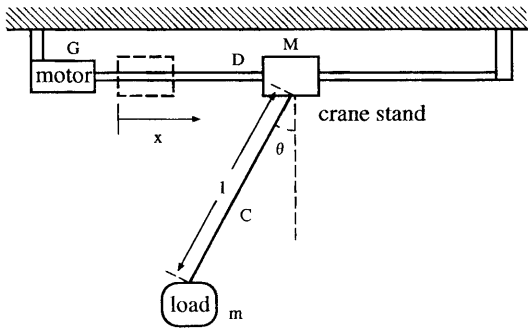


Fig.3 Structure of a nonlinear crane system

5.2 Training of Parameters

Training of the parameters of the controller can be done by using the higher order derivatives of ULNs. In ULNs, since not only the controller but also the controlled system can be described by the network, the gradient method for training parameters can be easily and usefully put to use.

To putting it concretely, parameter λ_m of the controller is trained as follows,

$$\lambda_m \leftarrow \lambda_m - \gamma \frac{\partial^\dagger L}{\partial \lambda_m}, \tag{46}$$

$$\frac{\partial^\dagger L}{\partial \lambda_m} = \frac{\partial^\dagger E}{\partial \lambda_m} + \frac{\partial^\dagger E_{xp}}{\partial \lambda_m}, \tag{47}$$

$$\frac{\partial^\dagger E}{\partial \lambda_m} = 2 \sum_{t \in T} \sum_{k \in J'} Q_k (h_k(t) - h_k^0(t)) \frac{\partial h_k^\dagger(t)}{\partial \lambda_m}, \tag{48}$$

$$\frac{\partial^\dagger E_{xp}}{\partial \lambda_m} = 2 \sum_{t \in T} \sum_{k \in J'} \sum_{n \in N} Q_{nk} \times \left[\frac{\partial^\dagger h_k(t)}{\partial r_n(t_0)} \Delta r_n(t_0) - g_{nk}(t) \right] \times \frac{\partial^{\dagger 2} h_k(t)}{\partial r_n(t_0) \partial \lambda_m} \Delta r_n(t_0), \tag{49}$$

where γ is learning coefficient assigned with a small positive number.

As was stated in Section 4.3, because the forward propagation algorithm is preferable in control applications where a large number of sampling times are involved, $\frac{\partial^\dagger L}{\partial \lambda_m}$ was calculated by using the first order and second order derivatives calculation method in Section 4.

Therefore, $\frac{\partial h_k^\dagger(t)}{\partial \lambda_m}$ in Eq.(48), $\frac{\partial^\dagger h_k(t)}{\partial r_n(t_0)}$ and $\frac{\partial^{\dagger 2} h_k(t)}{\partial r_n(t_0) \partial \lambda_m}$ in Eq.(49) can be calculated by using Eq.(24), Eq.(22) and Eq.(29) with $\lambda_1, \lambda_1(t_1)$ and $(\lambda_1(t_1), \lambda_2(t_2))$ being replaced with $\lambda_m, r_n(t_o)$ and $(r_n(t_0), \lambda_m)$ respectively.

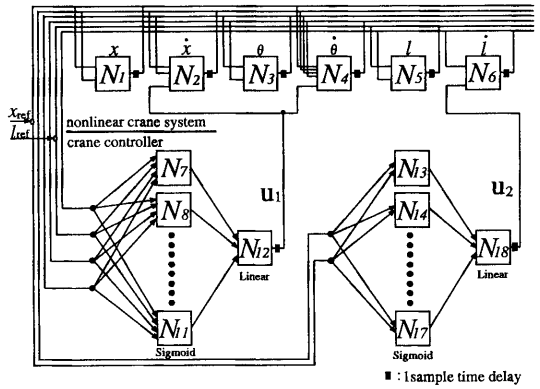


Fig.4 Control model of a nonlinear crane system using ULN

6. Simulation

6.1 Nonlinear Crane Control System

The controlled system is a nonlinear crane system shown in **Fig.3**. The position of the crane stand, the angle between the rope and the vertical line, and the position of the load are denoted by x, θ and ℓ, respectively. Then the nonlinear crane system is described as follows,

$$\begin{aligned} \frac{d^2x}{dt^2} &= -\frac{mg}{M}\theta - \frac{D+G}{M}\frac{dx}{dt} + \frac{G}{M}u_1, \\ \frac{d^2\theta}{dt^2} &= -\frac{M+m}{\ell M}g\theta - \frac{D+G}{\ell M}\frac{dx}{dt} + \frac{G}{\ell M}u_1, \\ \frac{d^2\ell}{dt^2} &= -\frac{C+G_m}{m}\frac{d\ell}{dt} + \frac{G_m}{m}u_2, \end{aligned} \tag{50}$$

where, u_1 and u_2 are input voltages for moving the crane stand and for rolling up the load, respectively, and C, G, G_m, D, M and m are appropriate system parameters.

Eq.(50) is transformed to the discrete time forms:

$$\begin{aligned} h_1(t) &= a_{11}h_1(t-1) + a_{21}h_2(t-1), \\ h_2(t) &= a_{22}h_2(t-1) + a_{32}h_3(t-1) + b_1u_1(t-1), \\ h_3(t) &= a_{33}h_3(t-1) + a_{43}h_4(t-1), \\ h_4(t) &= a_{24}\frac{h_2(t-1)}{h_5(t-1)} + a_{34}\frac{h_3(t-1)}{h_5(t-1)} \\ &\quad + a_{44}h_4(t-1) + \frac{b_1}{h_5(t-1)}u_1(t-1), \\ h_5(t) &= a_{55}h_5(t-1) + a_{65}h_6(t-1), \\ h_6(t) &= a_{66}h_6(t-1) + b_2u_2(t-1), \end{aligned} \tag{51}$$

where

$$h_1(t) = x(t),\ h_2(t) = \dot{x}(t),\ h_3(t) = \theta(t),$$
$$h_4(t) = \dot{\theta}(t),\ h_5(t) = \ell(t),\ h_6(t) = \dot{\ell}(t).$$

The ULN based control system for the nonlinear crane system is shown in **Fig.4**. Each of the control input u_1 and u_2 is composed of a feed forward neural network whose intermediate layer is made up of

sigmoid nodes with a range of $(-1.0, 1.0)$ and output layer has a summation node. All the branches have one-sampling time delay.

The following physical parameter values are used: $M = 40$[kg], $m = 2$[kg], $g = 9.8$[m/sec^2], $C = 0.42$[kg/sec], $G = 700$[N/V], $G_m = 0.98$[N/V] and $D = 300$[kg/sec]. The reference of the crane stand position (x_{ref}) is 1[m], and the reference of the load height (ℓ_{ref}) is 0.5[m], assuming the initial positions of the crane stand and the load to be 0[m] and 1[m], respectively.

The criterion function E to achieve the desired dynamics of the system, E_{xp} to specify the impulse response and E_{xc} to reduce the energy consumption are defined as follows respectively,

$$E = \sum_{t\in T}[Q_1(x_{ref} - x(t)^2)] + Q_2(\dot{x}(t_f))^2 + \sum_{t\in T}[Q_3(\theta(t))^2 + Q_4(\dot{\theta}(t))^2] + \sum_{t\in T}[Q_5(\ell_{ref} - \ell(t))^2] + Q_6(\dot{\ell}(t_f))^2, \quad (52)$$

$$E_{xp1} = \sum_{t\in T} Q_{\ell_{ref}\,\ell}[\frac{\partial\dagger\ell(t)}{\partial\ell_{ref}(t_0)}\Delta\ell_{ref}(t_0) - g_{\ell ref\ell}(t)]^2, \quad (53)$$

$$E_{xp2} = \sum_{t\in T} Q_{\ell_{ref}\,\ell}[\frac{\partial\dagger\ell(t)}{\partial\ell_{ref}(t_0)}\Delta\ell_{ref}(t_0) - g_{\ell_{ref}\ell}(t)]^2 + \sum_{t\in T} Q_{x_{ref}\,x}[\frac{\partial\dagger x(t)}{\partial x_{ref}(t_0)}\Delta x_{ref}(t_0) - g_{x_{ref}x}(t)]^2, \quad (54)$$

$$E_{xc} = \sum_{t\in T}[R_1(u_1(t))^2 + R_2(u_2(t))^2], \quad (55)$$

where

T : discrete set of sampling times ($\mid T \mid = 1000$),
t_f : final control time ($10sec$).

6.2 Simulation Results

Simulations were carried out according to the procedure stated in Section 5.1. Therefore, firstly parameters of the neural network controllers in **Fig.4** were trained by using the criterion function of Eq.(52) and Eq.(55). The number of learning iterations is 5000, learning coefficient γ is equal to $0.0001, Q_1, \cdots, Q_6 = 1.0$ and $\Delta r_n(t_0) = 1$. These numbers are used through the simulations.

Fig.5 shows the dynamics of the position $x(t)$ of the crane stand and the position $\ell(t)$ of the load using the controller obtained by the conventional method, and also shows the impulse response $\Delta h_{x_{ref}x}(t)$ and $\Delta h_{\ell_{ref}\ell}(t)$. In simulations, the number of the nodes in the intermediate layer was set to ten and various values of R_i $i = 1, 2$ such as 0.001, 0.01, and 0.1 were studied to investigate whether quick response and quick damping can be achieved at the same time or not by adjusting R_i.

From **Fig.5**(a) and (b), especially **Fig.5**(b), it is evident that the smaller the R_i is, the quicker the response is, but the characteristics of attenuation becomes worse. This means that it is difficult to achieve the quick response concurrently with the quick damping by the conventional method.

Next, as explaind in Section 5.1, evaluating the impulse response in **Fig.5**(c) and (d), especially **Fig.5**(d), the following two extended criterion functions were selected for the proposed impulse response method.

<case 1>

$$E_{xp1} = \sum_{t\in T} Q_{\ell_{ref}\ell}[\frac{\partial\dagger\ell(t)}{\partial\ell_{ref}(t_0)}\Delta\ell_{ref}(t_0) - g_{\ell ref\ell}(t)]^2, \quad (56)$$

$$g_{\ell_{ref}\ell} = 0.1e^{-5t}\sin 3t, \quad (57)$$

$$Q_{\ell_{ref}\ell} = 10000. \quad (58)$$

<case 2>

$$E_{xp2} = \sum_{t\in T} Q_{\ell_{ref}\ell}[\frac{\partial\dagger\ell(t)}{\partial\ell_{ref}(t_0)}\Delta\ell_{ref}(t_0) - g_{\ell_{ref}\ell}(t)]^2 + \sum_{t\in T} Q_{x_{ref}\,x}[\frac{\partial\dagger x(t)}{\partial x_{ref}(t_0)}\Delta x_{ref}(t_0) - g_{x_{ref}x}(t)]^2, \quad (59)$$

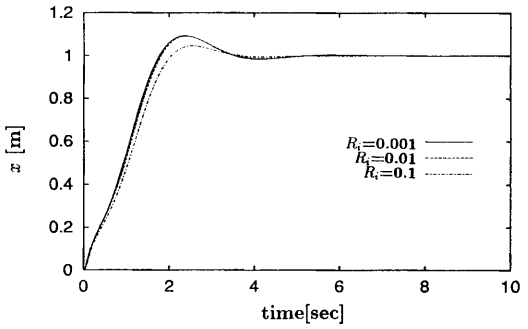
$$g_{\ell_{ref}\ell}(t) = 0.1e^{-5t}\sin 2t, \quad (60)$$

$$g_{x_{ref}x}(t) = 0.06te^{-3t}, \quad (61)$$

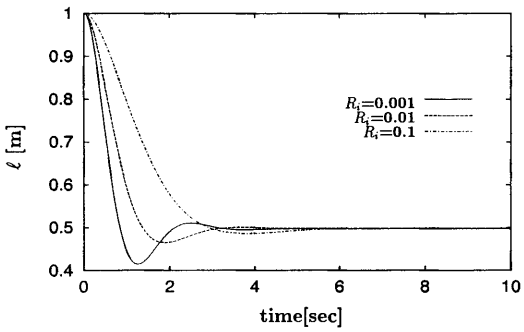
$$Q_{\ell_{ref}\ell} = Q_{x_{ref}x} = 10000, 100000. \quad (62)$$

The criterion function in case 1 was chosen aiming at quicker response and quicker damping of $\ell(t)$ than the ones obtained by the conventional method, and the criterion function in case 2 was established to prove the effectiveness of the proposed method for the multi-input and multi-output system, aiming at quicker response and damping of $\ell(t)$ and also aiming at quicker damping of $x(t)$.

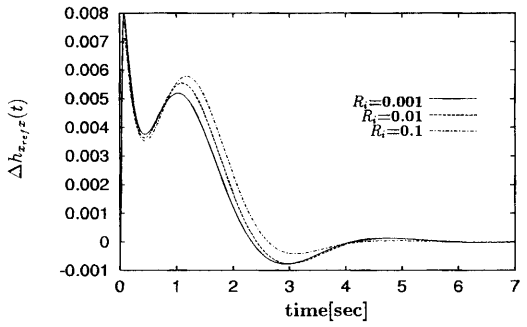
In case 1 the number of the nodes in the intermediate layer is equal to either five or ten, and in case



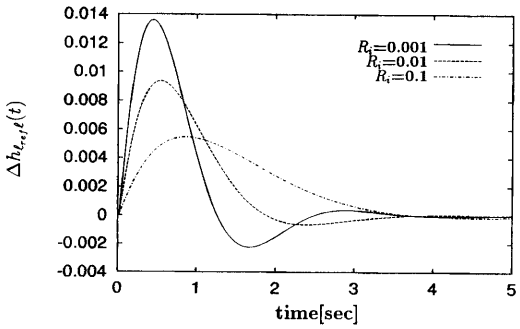
(a) Positions of the crane stand (x)



(b) Positions of the load (ℓ)



(c) $\Delta h_{x_{ref}x}(t)$



(d) $\Delta h_{\ell_{ref}\ell}(t)$

Fig.5 Simulation results in case of criterion function $E + E_{xc}$

2 two cases where $Q = Q_{\ell_{ref}\ell} = Q_{x_{ref}x} = 10000$ and $Q = 100000$ are simulated.

Fig.6 shows the dynamics of $x(t), \ell(t)$ and the impulse response $\Delta h_{\ell_{ref}\ell(t)}$ obtained by using the criterion function of Eq.(52) and (56) comparing with the conventional dynamics.

From **Fig.6** it is clear that the impulse response approaches the target value of Eq.(57) and that quick response is achieved concurrently with quick damping by the proposed method.

The problem is that the steady state error increases as compared with the conventional method, but it is obvious from **Fig.6**(b) that this problem can be overcome by increasing the number of nodes in the controller.

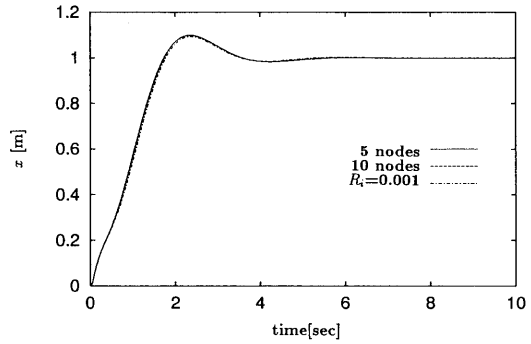
Fig.7 shows the results of the simulations, in which the proposed method is used effectively to enhance the quality of the dynamics of the multi-input and multi-output system. In **Fig.7**(a) and (b), when using $Q = Q_{\ell_{ref}\ell} = Q_{x_{ref}x} = 100000$, quick response and quick attenuation are dramatically achieved for not only $\ell(t)$ but also $x(t)$, but steady state error becomes large.

This is because a large Q strengthened the effect of the extended terms in the criterion function. Therefore, if a slightly smaller $Q(Q = 10000)$ is adopted, the steady state error can be reduced remarkably as shown in **Fig.7**, although the response speed and the attenuation can not improve on those achieved in the case of $Q = 100000$. So, by adjusting the coefficient Q appropriately, the dynamics of higher quality such as quick response, quick attenuation and less steady state error can be achieved by the proposed method.

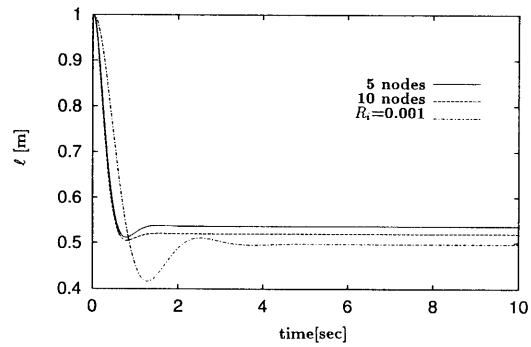
7. Conclusion

Universal Learning Network(ULN) and its application to control systems have been discussed. In regard to architectures and learning algorithms, a class of neural networks are unified in the framework of ULNs, and salient features are added. In an architectural aspect, ULNs have higher degree of freedom: arbitrary continuously differentiable functions in the nodes, multiple branches and arbitrary time delays. The ULN training algorithm, in either backward propagation scheme or forward propagation scheme, is equipped with a mechanism for calculating the higher order derivatives as a novel feature that has not been possessed by common neural networks.

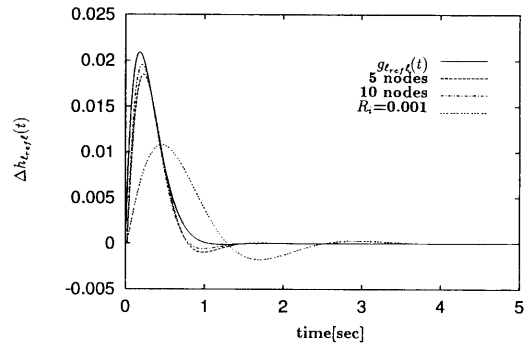
Both of structural and training-algorithmic features enable us to solve sophisticated controller de-



(a) Positions of the crane stand (x)

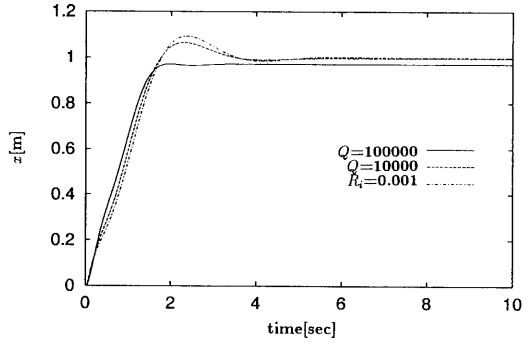


(b) Positions of the load (ℓ)

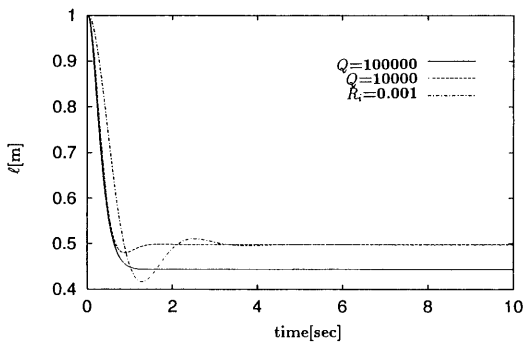


(c) $\Delta h_{\ell_{ref}\ell}(t)$

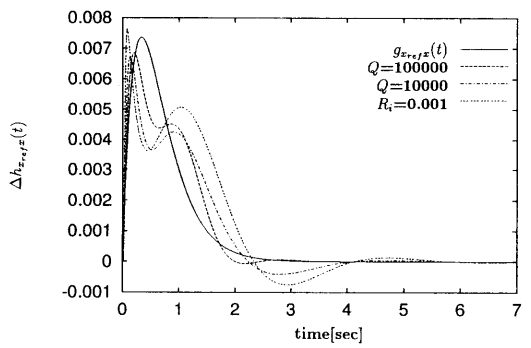
Fig.6 Simulation results in case of criterion function $E + E_{xp1}$



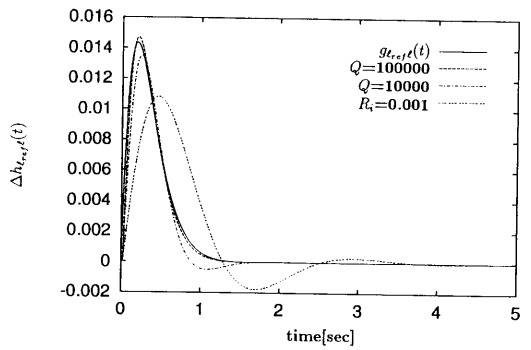
(a) Positions of the crane stand (x)



(b) Positions of the load (ℓ)



(c) $\Delta h_{x_{ref}x}(t)$



(d) $\Delta h_{\ell_{ref}\ell}(t)$

Fig.7 Simulation results in case of criterion function $E + E_{xp2}$

sign problems. As an example problem, a new control method of nonlinear dynamic systems using the feature of higher order derivatives of ULN and the concept of impulse response has been proposed.

From the simulations of a nonlinear crane control system, it has become evident that the dynamics of higher quality can be achieved easily by the proposed control method compared with the conventional control method.

Nonlinear systems have the different characteristics from linear systems in the fact that the dynamics obtained by the proposed method changes depending on the magnitude of the impulse $r_n(t_0)$.

Though $r_n(t_0)$ was equal to one in this article, how the value of $r_n(t_0)$ affects the nature of the proposed method is an open problem.

Acknowledgments

The authors thank Dr. Ohbayashi and their student, Masayuki Hashimoto who did part of the simulations and joined in the discussions.

References

1) K. Hirasawa, J.Hu, M. Obayashi and J. Murata, "Computing higher order derivatives in Universal Learning Networks", Journal of Advanced Computational Intelligence, Vol.2, No.2, 1998
2) W.L. Buntine and A.S. Weigend, "Computing Second Derivatives in Feed-Forward Networks: A Review", IEEE Trans. on Neural Networks, vol. 5, no. 3, 1994.
3) M.Han, K. Hirasawa, M. Ohbayashi and M. Fujita, "Modeling Dynamic Systems Using Universal Learning Network", Proc. IEEE Conference on Systems, Man and Cybernetics, 1996, pp. 1172–1177.
4) A.E. Bryson Jr. and Y.C. Ho, "Applied Optimal Control", Hasisphere Pub. Co., 1975
5) M. Ohbayashi, K. Hirasawa, J. Murata and M. Harada, "Robust Learning Control Using Universal Learning Network", "Proc. IEEE Conference on Neural Networks, 1996, pp. 2208–2213.
6) M. Ohbayashi, K. Hirasawa, M. Hashimoto and J. Murata, "Robust Control Using Second Order Derivative of Universal Learning Network", Proc. IEEE Conference on Systems, Man and Cybernetics, 1996, pp. 1184–1189.
7) M. Koga, K. Hirasawa, M. Ohbayashi and J. Murata, "Chaos Control Using Second Order Derivatives of Universal Learning Network", Proc. IEEE Conference on Neural Networks, 1995, pp. 1287–1292.
8) K. Hirasawa, M. Ohbayashi and J. Murata, "Universal Learning Network and Computation of Its Higher Order Derivatives", Proc. IEEE Conference on Neural Networks, 1995, pp. 1273–1277.
9) K. Hirasawa, M. Ohbayashi, M. Koga, and M. Harada, "Forward Propagation Universal Learning Network", Proc. IEEE Conference on Neural Networks, 1996, pp. 353–358.
10) E.-T. Lin, J.E. Dayhoff and P.A. Ligomenides, "Trajectory Production with the Adaptive Time-Delay Neural Network", Neural Networks, vol. 8, no. 3, pp. 447–461, 1995.
11) T. Waiber, A. Hanazawa, T. Hinton, F. Shikano and K. Lang, "Phoneme Recognition: Neural Networks Versus Hidden Markov Models", Proc. IEEE Conference on Acoust., Speech and Signal Processing, 1988, pp. 107–110.
12) P. Werbos, Beyond Regression: New Tools for Prediction and Analysis in the Behavior Science, Ph.D. thesis, Harvard University, 1974
13) W.M. Wonham, "On Pole Assignment in Multi-Input Controllable Linear Systems", IEEE Trans. on AC, Vol. AC-12, No.6, pp 660-665, 1967.
14) H. Kwaker neak and R. Sivan, "Linear Optimal Control Systems", Wiley-Interscience, 1972.
15) Kumpati S. Narendra, "Neural Networks for Control: Theory and Practice", Proceeding of the IEEE, Vol. 84, No. 10, 1996.