Paper

A New Minimax Control Method for Nonlinear Systems Using Universal Learning Networks

Non-member	Hongping CHEN	(Kyushu University)
Member	Kotaro HIRASAWA	(Kyushu University)
Member	Jinglu HU	(Kyushu University)
Member	Junichi MURATA	(Kyushu University)

In neural network based control systems, if system environments, that is, system parameters and disturbances at training stage, are much different from those at control stage, performances of control systems may become worse. To solve this problem, robust control design is needed. In this paper, we propose a new minimax control method using Universal Learning Networks, in which the criterion function is evaluated at several specific operating points, and at each training step the worst criterion function among the operating points is optimized. Moreover, a sensitivity term calculated on the operating point is included in the criterion function in order to improve the performance of the control system between two operating points. The minimax control method including sensitivity term is shown to have better robustness against the changes of system parameters.

Keywords: Universal Learning Networks, Neural networks, Minimax control, Robust control, Neural network control, Sensitivity

1. Introduction

Because artificial neural networks can approximate any continuous nonlinear function to any desired accuracy, neural networks are becoming an effective tool in optimal control of nonlinear systems. Many neural network based controllers have been developed for compensating the effects of nonlinearities and for handling system uncertainties in control systems. A neural network based controller can be constructed by either online adaptive control design methods or off-line robust control methods. Many approaches have been proposed to use neural networks for adaptive control of nonlinear systems [(1)~(3)]. In this paper, we focus on off-line robust controller design.

In a conventional off-line design method, the criterion function is usually evaluated at a specific operating point of system parameters [(4)]. This arises a problem that system performance at the evaluation point can be ensured, and there is, however, no guarantee at other points. It means that the obtained off-line solution will be restricted to the case of a small change in the environments. If system environment variables at control stage are much different from those at training stage, the performance of control systems becomes worse, and systems may become unstable.

A natural way to solve this problem is to increase the evaluation points, that is, the use of multi-point evaluation method. The weighted multi-point evaluation method is one of the simplest implementations [(5)]. However, it is usually difficult to choose appropriate weights. As a better alternative of multi-point evaluation methods, in this paper we propose a new minimax control method. In the new method, the criterion function is evaluated at several specific operating points of the system parameters, and at each training step the worst criterion function among the operating points is optimized. By using the minimax control method, the criterion function at each special operating point can be ensured to be small.

On the other hand, since one can practically only choose limited evaluation points, there is no guarantee for system performance between two specific operating points. To solve this problem, we further introduce a sensitivity term calculated on the operating points to the criterion function based on a similar concept introduced in feedback system theory for preventing undesirable effects caused by parameter variations in the control system [(6)]. It has been demonstrated that low sensitivity to parameter variations is one of the necessary properties to secure a robust control. In this paper, the *sensitivity* is defined as a change of criterion function caused by a small change in the system parameters around a nominal value. In our method, an extended criterion function containing sensitivity term has been considered for controller design using ULN (Universal Learning Network) learning [(4) (7)]. This allows us to obtain a controller that minimizes the worst criterion function and the sensitivity calculated on the operating point. A controller obtained in this way is expected to have better robustness for the change of system parameters. The effectiveness of the proposed

method is demonstrated by applying it to a nonlinear crane control system.

This paper is organized as follows: Section 2 gives a brief review on the ULN. Section 3 discusses ULN based minimax control method. Section 4 carries out a numerical simulation to demonstrate the effectiveness of the proposed method. Finally, Section 5 gives concluding remarks.

2. Universal Learning Network

Universal learning network (ULN) has been proposed, as its name indicates, to provide a universal framework for a class of neural networks and moreover, to model and control complex systems based on the idea that most of general complex systems in real world can be modeled by networks whose nodes represent the processing elements and branches among the nodes describe their relations [7].

It is generally recognized that any dynamic system can be described by a set of related equations [8]. Depending on prior knowledge available about systems, equations may be fully known, partly known, or totally unknown. To model such dynamic systems, we introduce a learning network consisting of two kinds of element: nodes and branches. The nodes correspond to equations and branches to their relations. The nodes may have continuously differentiable nonlinear functions including a function realized by a subnetwork, e.g., sigmoidal neural networks or neuro-fuzzy networks, in addition the branches for inter-connecting nodes can have arbitrary time delays and adjustable weights on them. Such learning networks are called ULN. When no prior knowledge is used and only sigmoidal functions are used in all nodes, the learning network is reduced to a conventional sigmoidal neural network.

One of the distinctive features of the ULN is that it incorporates prior knowledge, if available, in the modeling. For example, the dynamics described by differential or difference equations can be treated in the ULN, and prior knowledge of systems expressed in fuzzy rules may be incorporated in the ULN via a fuzzy network. It, therefore, is expected to have performances superior to conventional neural networks in the applications of system modeling and control.

The generic equation that describes ULN behavior is expressed as follows:

$$\begin{aligned} h_j(t) = f_j(&\{h_i(t - D_{ij}(p)) | i \in JF(j), p \in B(i,j)\}, \\ &\{r_n(t) | n \in N(j)\}, \{\lambda_m | m \in M(j)\}, \\ &\{f_g | g \in G\}) \quad t \in T \end{aligned} \cdots\cdots\cdots\cdots\cdots (1)$$

where f_j is the nonlinear node function of node j, $h_j(t)$ the output value of node j at time t, $JF(j)$ the set of suffixes of the nodes connecting to nodes j, $r_n(t)$ the n-th external input, λ_m the m-th adjustable parameter, $D_{ij}(p)$ the time delay of p-th branch from node i to node j, f_g the value of g-th system parameter, G the set of suffixes of system parameters f_g, t the sampling instant, T the discrete set of sampling instants, and $N(j)$, $M(j)$ the sets of suffixes of external inputs and adjustable parameters related to node j, respectively. **Figure 1** shows the architecture of ULN.

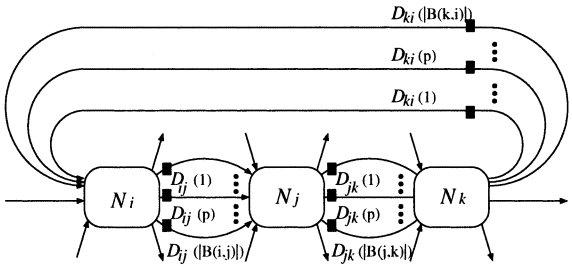


Fig. 1. Structure of universal learning network

Because a ULN allows any nonlinear function to be embedded in its nodes, both a controlled system and its controller can be represented by a single ULN when the ULN is applied to controller design problems. If the controlled system is unknown, one should first perform an identification [9], then design the ULN control system using the ULN learning. Therefore the fundamental difference of a ULN control from a conventional control is that controller design can be regarded as a parameter optimization, which makes design problem simple.

Furthermore, the ULN is equipped with a generalized learning algorithm, in which second or higher order derivatives can be used. The higher order derivative calculation mechanism of the ULN renders additional advantages to 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 also put other terms to the criterion functions. For example, the additional term is the sensitivity of the standard criterion function with respect to the changes in the plant parameters. Minimizing a sensitivity using a gradient method requires the second order derivatives since the sensitivity itself is the first order derivative.

3. ULN Based Minimax Control Method

3.1 Minimax Approach Minimax approach is one of the major techniques for designing systems robust to system uncertainties, in which the goal is to optimize the worst-case performance. Let $\lambda = \{\lambda_1, \ldots, \lambda_m, \ldots, \lambda_M\}$ denote the controller parameters, $f = \{f_1, \ldots, f_g, \ldots, f_G\}$ the system parameters, $E(\lambda, f)$ the criterion function. The optimization of a minimax approach can be described by

$$\sigma = \inf_{\lambda} \sup_{f} E(\lambda, f). \cdots\cdots\cdots\cdots\cdots\cdots (2)$$

Introducing $e(\lambda)$ representing the maximum risk associated with the control λ, defined by

$$e(\lambda) = \sup_{f} E(\lambda, f), \cdots\cdots\cdots\cdots\cdots\cdots (3)$$

(2) can be expressed as

$$\sigma = e(\hat{\lambda}). \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots (4)$$

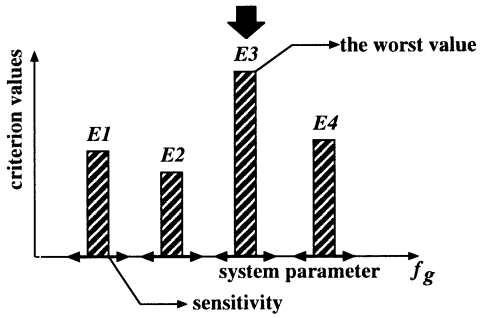


Fig. 2. Basic idea of ULN based minimax control method

In order to obtain a controller robust against changes of system parameters, we apply the above minimax approach to ULN control system. Let the criterion function be evaluated at several specific operating points, and at each step the worst criterion function among the operating points be optimized. The criterion function for minimax approach is described by

$$E_R = \underset{l \in L}{Max} \{E_l\} \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots (5)$$

where E_l is the standard criterion function depending on discrete system parameter vector l, and $l = (l_1, l_2, .., l_{|G|}) \in L$ is the vector in discrete system parameter space, and L is the set. Minimizing (5) ensures that the criterion at each special operating point l to be small. However, in the minimax method only discrete operating points can practically be chosen, there is no guarantee for the performance between two specific operating points. Further improvement is needed.

3.2 Sensitivity Term To solve the problem, we introduce a sensitivity term to the standard criterion function E_l. As shown in Fig.2, in the minimax approach the optimization at each step is performed such that the criterion function is minimized at the specific operating point where it has the worst criterion values among all specific operating points. Obviously, if the sensitivity at the corresponding operating points is also minimized, then it may guarantee that the criterion function not only at the operating point but also on a small area around the operating point is minimized. When the number of specific operating points is chosen to be sufficient large, we may expect that the obtained control system has good performance also between two specific operating points. The criterion function including sensitivity term is described by

$$E_R = \underset{l \in L}{Max} \left\{ E_l + k_f \left(\sum_{g \in G} \frac{\partial^\dagger E_l}{\partial f_g} \right)^2 \right\} \quad \cdots\cdots (6)$$

where $g \in G$ is the suffix of system parameters, f_g the value of gth system parameter, and k_f the sensitivity coefficient assigned an appropriate positive number. In (6), the sensitivity is defined as an ordered derivative $\frac{\partial^\dagger E_l}{\partial f_g}$, which is the change of criterion function E_l caused by change of system parameter f_g with other variables fixed [10], see Appendix A for more details about the ordered derivative.

3.3 Controller Design In a ULN based control system, both objective system and its controller are treated as a unified network called ULN. The controller design becomes the ULN learning. The ULN learning is performed by minimizing the criterion function (5) or (6) depending on whether the sensitivity term is included. A gradient-based implementation of the minimization is described by

$$\lambda_m \longleftarrow \lambda_m - \gamma \frac{\partial^\dagger E_R}{\partial \lambda_m} \quad \cdots\cdots\cdots\cdots\cdots\cdots (7)$$

where $\frac{\partial^\dagger E_R}{\partial \lambda_m}$ is the order derivative of E_R with respect to λ_m.

However, calculating $\frac{\partial^\dagger E_R}{\partial \lambda_m}$ is not trivial. In a minimax optimization, even if the standard criterion function E_l is differentiable for all operating point l's, E_R is non-differentiable in general. Therefore gradient-based methods can not be directly used for the optimization. To overcome this difficulty, several methods have been proposed in the literature [11]. For example, the problem can be replaced by the one using a differentiable objective function which is a uniformly close approximation to the original objective function. The new problem can then be treated using gradient-based methods. This only guarantees a near-optimal solution of the original minimax optimization problem, and the algorithm is usually very time-consuming.

For simplicity, in this paper we only approximately calculate $\frac{\partial^\dagger E_R}{\partial \lambda_m}$ by

$$\frac{\partial^\dagger E_R}{\partial \lambda_m} \simeq \frac{\partial^\dagger E_{l_{\max}}}{\partial \lambda_m} \quad \cdots\cdots\cdots\cdots\cdots\cdots (8)$$

for the criterion function without including sensitivity term, and

$$\frac{\partial^\dagger E_R}{\partial \lambda_m} \simeq \frac{\partial^\dagger}{\partial \lambda_m} \left\{ \left[E_{l_{\max}} + k_f \left(\sum_{g \in G} \frac{\partial^\dagger E_{l_{\max}}}{\partial f_g} \right)^2 \right] \right\} \quad \cdots\cdots\cdots\cdots\cdots\cdots (9)$$

for the criterion function including sensitivity term, where $E_{l_{\max}} = \mathrm{Max}_{l \in L} E_l$ is the E_l with largest value determined at each learning step. Obviously, this approximation will cause large error when algorithm closes to minimum points of E_R, and may result in oscillation of the algorithm. To avoid the oscillation, we let the learning coefficient $\gamma \rightarrow 0$ as the training step $t \rightarrow \infty$. Further research is needed to develop algorithm for calculating $\frac{\partial^\dagger E_R}{\partial \lambda_m}$ more efficiently.

Because the sensitivity itself is the first order derivative, calculating (9) requires the second order derivatives of the standard criterion function. See Appendix B for the calculation of the first and second order derivatives of E_l.

4. Simulation Studies

The proposed minimax approach is applied to a non-

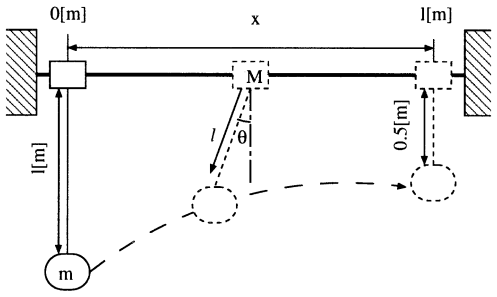


Fig. 3. Structure of nonlinear crane system

linear crane control system to demonstrate the effectiveness.

4.1 Nonlinear Crane System The objective to be controlled is a nonlinear crane system shown in **Fig.3**. Let x, θ and l denote the position of crane stand, the angle between rope and vertical line, and the position of load, respectively. Then the nonlinear crane system is described by

$$\frac{d^2x}{dt^2} = -\frac{mg}{M}\theta - \frac{D+G}{M}\frac{dx}{dt} + \frac{G}{lM}u_1 \quad \cdots\cdots \quad (10)$$

$$\frac{d^2\theta}{dt^2} = -\frac{M+m}{lM}g\theta - \frac{D+G}{lM}\frac{dx}{dt} + \frac{G}{lM}u_1 \quad \cdots \quad (11)$$

$$\frac{d^2l}{dt^2} = -\frac{C+G_m}{m}\frac{dl}{dt} + \frac{G_m}{m}u_2 \quad \cdots\cdots\cdots\cdots \quad (12)$$

where u_1 ,u_2 are the input voltages for moving the crane stand and for rolling up the load, M the mass of the crane stand, m the mass of the load, C, D the friction coefficients, G, G_m the coefficient of transforming voltage to torque, and g the gravity acceleration.

Let $h_1(t) = x(t)$, $h_3(t) = \theta(t)$, $h_5(t) = l(t)$, $h_2(t) = \frac{dx}{dt}$, $h_4(t) = \frac{d\theta}{dt}$, $h_6(t) = \frac{dl}{dt}$. The discrete-time forms of (10)-(12) are described by

$$h_1(t) = h_1(t-1) + \Delta t \cdot h_2(t-1) \quad \cdots\cdots\cdots \quad (13)$$

$$h_2(t) = \left(1 - \Delta t\frac{D+G}{M}\right)h_2(t-1) - \frac{mg}{M}\Delta t \cdot h_3(t-1) + \frac{G}{M}\Delta t \cdot u_1(t-1) \quad (14)$$

$$h_3(t) = h_4(t-1) + \Delta t \cdot h_4(t-1) \quad \cdots\cdots\cdots \quad (15)$$

$$h_4(t) = h_4(t-1) - \frac{D+G}{M}\Delta t\frac{h_2(t-1)}{h_5(t-1)} - \frac{M+m}{M}g\Delta t\frac{h_3(t-1)}{h_5(t-1)} + \frac{G}{M}\Delta t\frac{u_1(t-1)}{h_5(t-1)} \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \quad (16)$$

$$h_5(t) = h_5(t-1) + \Delta t \cdot h_6(t-1) \quad \cdots\cdots\cdots \quad (17)$$

$$h_6(t) = \left(1 - \frac{C+G_m}{m}\Delta t\right)h_6(t-1) + \frac{G_m}{m}\Delta t \cdot u_2(t-1) \quad \cdots\cdots\cdots\cdots\cdots \quad (18)$$

where Δt is the sampling time.

4.2 ULN Based Crane Control System To control the crane system, we introduce two layered neural network controllers, each of which has 4 hidden nodes with tanh(·) node function, and one linear output node.

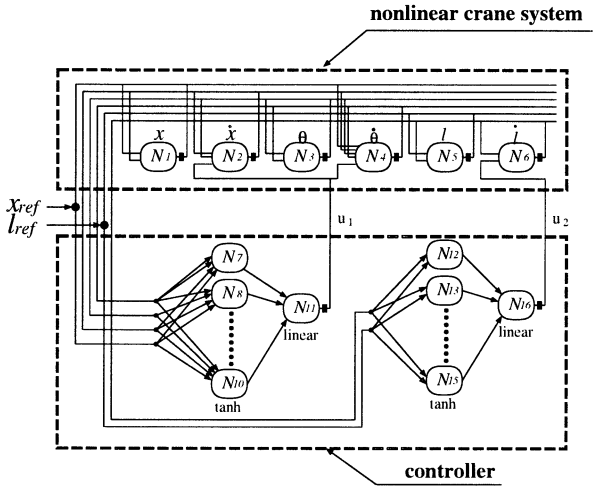


Fig. 4. A feedback control system of nonlinear crane system using a layered controller

Now let us introduce 6 nodes for the six difference equations (13)-(18) describing the crane system, one node for one equation, and many branches for the relation of the equations. Then the crane system is described by a network. This network and the two neural network controllers can be treated as a unified learning network – ULN. **Figure 4** shows the ULN describing the nonlinear crane system and its controllers in a unified network. In this way, the controller design is transformed into the ULN learning.

4.3 Simulation Conditions

- Physical parameter values:
  $M = 40[kg]$, $g = 9.8[m/sec^2]$, $C = 0.42[kg/sec]$, $G = 700[N/V]$, $G_m = 0.98[N/V]$, $D = 300[kg/sec]$, and $\Delta t = 0.003[sec]$.
- Control aim:
  the crane stand position : $0 \rightarrow 1.0$ [m]
  the load height : $1 \rightarrow 0.5$ [m]
- Standard criterion function:

$$E = \frac{1}{2}\Big[\sum_{s\in T}\{Q_1(x_{ref} - x(s))^2\} + Q_2\dot{x}(t_f)^2 + \sum_{s\in T}\{Q_3\theta(s)^2 + Q_4\dot{\theta}(s)^2\} + \sum_{s\in T}\{Q_5(l_{ref} - l(s))^2\} + Q_6\dot{l}(t_f)^2 + \sum_{s\in T}\{R_1u_1(s)^2 + R_2u_2(s)^2\}\Big] \quad \cdots \quad (19)$$

where $Q_1 = Q_2 = Q_3 = Q_4 = Q_5 = 1.0, Q_6 = 5.0$ and $R_1 = R_2 = 0.001$ are the coefficients, $t_f = 7.5[sec]$ the final control instant, and $|T| = 2500$ the set of sampling instants.

4.4 Simulation Results In the simulations, the load mass m is taken as the system parameter f_g. The aim is to design a controller so that the control system is stable even though the load mass m changes in a wide

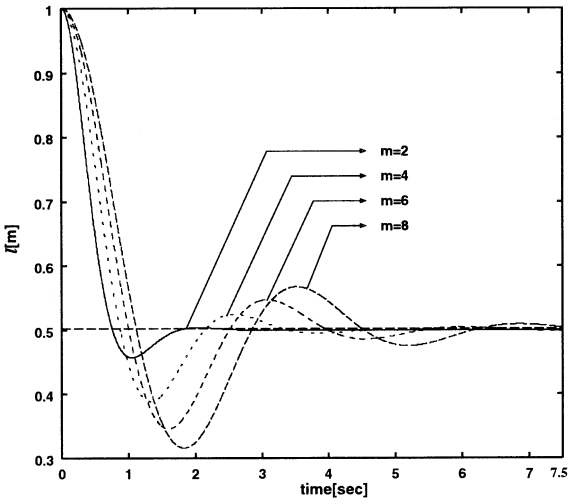


Fig. 5. Results of one-point evaluation

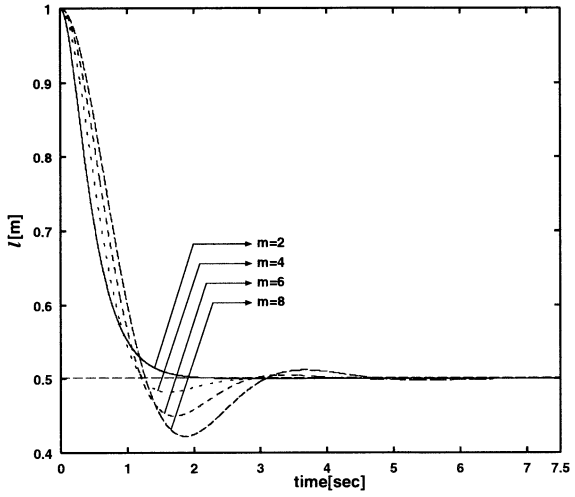


Fig. 6. Results of multi-point evaluation

range.

Simulations are carried out to compare control performance of the following four control methods:

• One-point Evaluation

$$E_R = E_m \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \quad (20)$$
$$m = 2$$

• Multi-point Evaluation

$$E_R = \sum_{m \in L} \mu_m E_m \quad \cdots\cdots\cdots\cdots\cdots\cdots \quad (21)$$
$$\mu_m = 1.0$$

• Minimax Evaluation

$$E_R = \underset{m \in L}{Max} \{E_m\} \quad \cdots\cdots\cdots\cdots\cdots\cdots \quad (22)$$

• Minimax Evaluation with Sensitivity

$$E_R = \underset{m \in L}{Max} \left\{ E_m + k_f \left(\frac{\partial^\dagger E_m}{\partial m}\right)^2 \right\} \quad \cdots\cdots \quad (23)$$
$$k_f = 1.0$$

First, we carried out a simulation by using the conventional one-point evaluation method. The criterion function is evaluated at point $m = 2\ kg$, and then we trained the controller. After training of the parameters of the neural network, system responses were calculated at points $m = 2, 4, 6$ and $8\ kg$, the responses with respect to l were shown in **Fig.5**. We can see that the performance at the operating point $m = 2\ kg$ is best, but the larger the value of m is, the worse the system performance is.

Then, we choose $m = 2,\ 4,\ 6$ and $8\ kg$ as the evaluation points, and trained the controllers using other three methods, respectively. After training, system responses were calculated at the evaluation points. **Figure 6, 7** and **8** show the responses for l using multi-point evaluation, minimax evaluation and minimax evaluation with sensitivity, respectively. We can see that the minimax evaluation method has better performance than the multi-point evaluation method, and the minimax evaluation with sensitivity has the best performance. **Fig-**

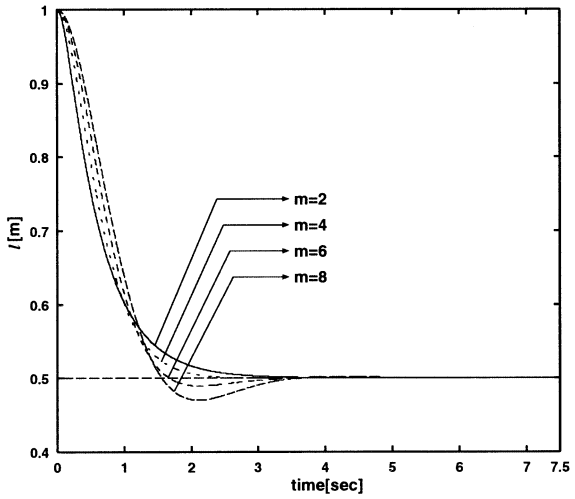


Fig. 7. Results of minimax evaluation

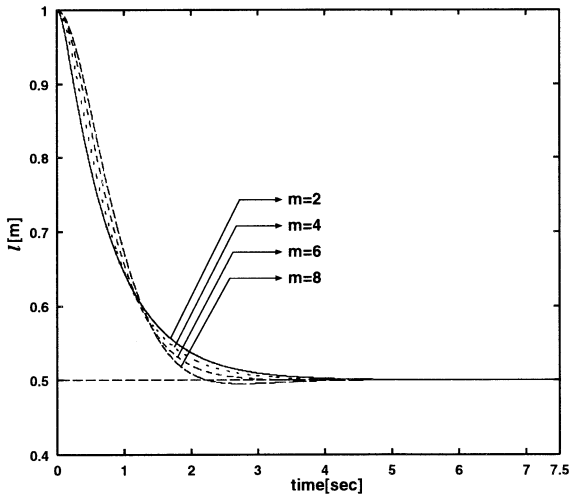


Fig. 8. Results of minimax evaluation with sensitivity control

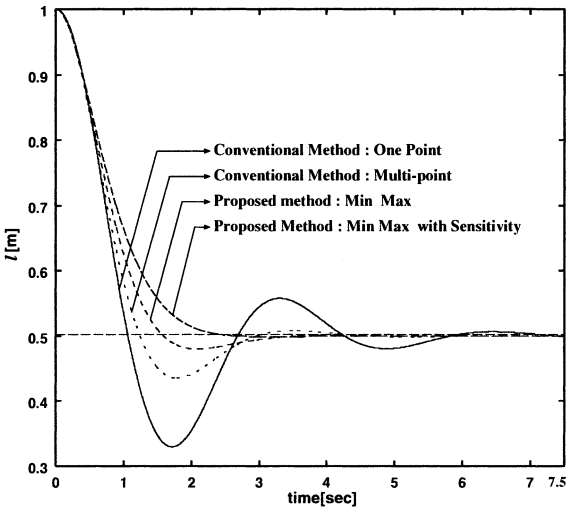


Fig. 9. Comparison of the dynamics among the different methods at evaluation point $m = 8\ kg$

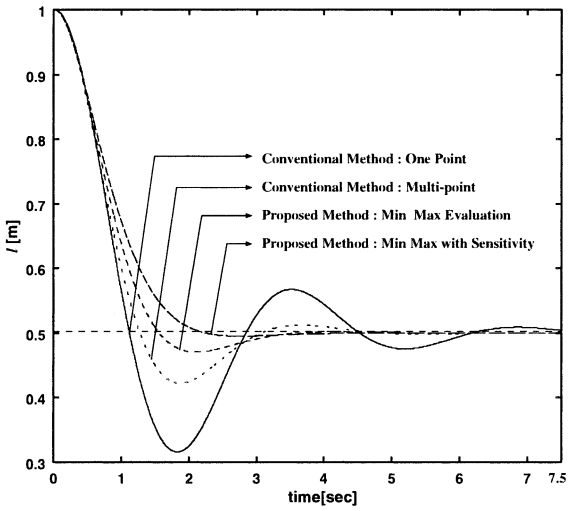


Fig. 10. Comparison of the dynamics among the different methods at non-training point $m = 7\ kg$

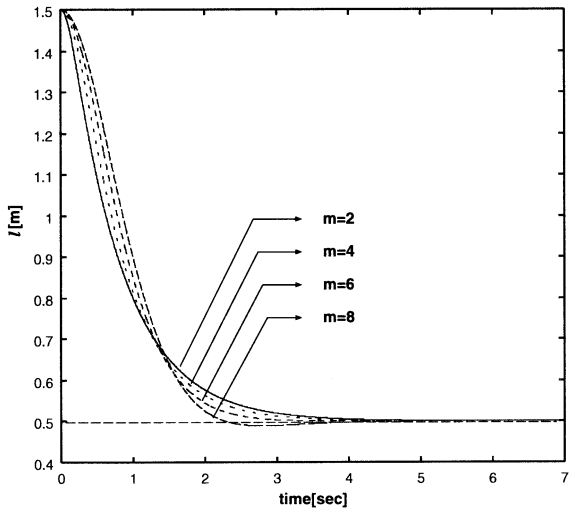


Fig. 11. Responses of control system with a control aim different from the one used for controller design.

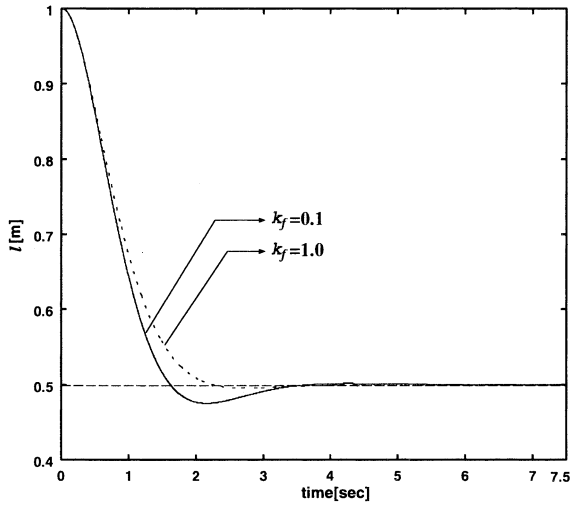


Fig. 12. Influence of sensitivity terms on the dynamics

ure 9 shows a comparison of the dynamics among different methods at evaluation point $m = 8\ kg$. **Figure 10** shows the system responses for l at non-evaluation point $m = 7\ kg$. We can see that the controller performs well even at the non-evaluation points; this means that the trained neural network controllers using the four methods have generalization ability.

To further show the generalization ability, we calculate the control responses with a control aim different from the one for controller design. **Figure 11** shows the control responses of l for the case where the initial position of the load and the reference position of the load height are changed to

the crane stand position : $0 \rightarrow 1.5$ [m]
the load height : $1.5 \rightarrow 0.5$ [m].

It can be seen that a reasonable performance can be guaranteed even though the initial values and reference values of the system have changed because of the generalization ability of neural networks.

Finally, to show the influence of sensitivity terms on the dynamics we carried out simulations with different values of k_f. It has been found that system performance is improved when k_f is larger, however too large k_f causes learning algorithm diverged. Therefore, it is necessary to choose appropriate values of k_f in practical applications. However, how to choose an appropriate k_f for a specific problem is still an open problem. Further research is needed. **Figure 12** shows the dynamics difference between sensitivity coefficient $k_f = 0.1$ and $k_f = 1.0$. We can see that speed of response decreases, but overshoot also decreases when k_f becomes larger. It is clear that sensitivity coefficient k_f is a factor for balancing quick response and robustness.

5. Conclusions

This paper proposed a minimax robust control method. In the proposed method, the criterion function is evaluated at several specific operating points,

and at each learning step the worst criterion function among the specific operating points is optimized. Numerical simulations show that the proposed robust control method has better performance than both the conventional one-point evaluation method and the conventional weighted multi-point evaluation method; and the sensitivity calculated on the specific operating points can be used to further improve the system performance.

The proposed method has been applied to a nonlinear crane system to demonstrate its effectiveness. From simulations, it is concluded that the proposed method can be effectively applied for the system where the dynamics of its system is known and differentiable to enhance the robustness. If the dynamics of the controlled system are unknown, the proposed method can be applied after the system is identified. Although in this paper simulations are done only for the change of the mass of load, the design of robust control can be done in the same manner for any other system parameters such as disturbances and external inputs.

Appendix

1. Ordered Derivative

Ordered derivative is basically a kind of partial derivative. It is firstly introduced by Werbos [10], which is a partial derivative considering both direct and indirect relationship. J-S.R. Jang and C.T. Sun (1995) [12] explained the difference between the ordered derivative and the ordinary partial derivative in details. Here we borrow the simple example to briefly show the difference, see Ref. (12) for more details.

Consider a problem where z is a function of x and y, and y is in turn a function of x:

$$y = f(x) \quad \cdots\cdots (A1)$$
$$z = g(x, y). \quad \cdots\cdots (A2)$$

For the ordinary partial derivative $\frac{\partial z}{\partial x}$, we assume that all the other variables (in this case, y) are constant:

$$\frac{\partial z}{\partial x} = \frac{\partial g(x,y)}{\partial x}. \quad \cdots\cdots (A3)$$

In other words, we assume the direct variables x and y are independent, without paying attention to the fact that y is actually a function of x. For the ordered derivative, we take this indirect causal relationship into consideration:

$$\frac{\partial^\dagger z}{\partial x} = \frac{\partial g(x,y)}{\partial y}\Big|_{y=f(x)} \cdot \frac{\partial f(x)}{\partial x} + \frac{\partial g(x,y)}{\partial x}\Big|_{y=f(x)} \quad \cdots\cdots (A4)$$

Therefore, the ordered derivative takes into consideration both the direct and indirect paths that lead to the causal relationship.

2. First and Second Order Derivatives

2.1 Calculation of $\frac{\partial^\dagger E_l}{\partial f_g}$ Considering the direct and indirect relationship of E_l and f_g, it is easy to show that $\frac{\partial^\dagger E_l}{\partial f_g}$ is calculated by

$$\frac{\partial^\dagger E_l}{\partial f_g} = \sum_{r\in J_0}\sum_{s\in T_0}\left[\frac{\partial E_l}{h_r(s)}\frac{\partial^\dagger h_r(s)}{\partial f_g}\right] + \frac{\partial E_l}{\partial f_g} \quad \cdots (A5)$$

where J_0 and T_0 are the sets of nodes and time instants directly related to the evaluation.

$\frac{\partial^\dagger h_r(s)}{\partial f_g}$ can be calculated in two ways: backward propagation and forward propagation. In this paper, we use the forward propagation method, that is, it is first computed for the nodes directly related to f_g, then propagated toward the nodes directly related to E_l. Introducing a notation $P_1(j,t,f_g)$ denoting the change of node output $h_j(t)$ for f_g,

$$P_1(j,t,f_g) = \frac{\partial^\dagger h_j(t)}{\partial f_g}, \quad \cdots\cdots (A6)$$

we can express the iterative algorithm by

$$P_1(j,t,f_g) = \sum_{i\in JF(j)}\sum_{p\in B(i,j)}\left[\frac{\partial h_j(t)}{\partial h_i(t-D_{ij}(p))} \times P_1(i,t-D_{ij}(p),f_g)\right] + \frac{\partial h_j(t)}{\partial f_g} \quad (A7)$$

where $JF(j)$ is the set of nodes connected to node j, and $B(i,j)$ is the set of branches from node i to node j.

2.2 Calculation of $\frac{\partial^{\dagger 2} E_l}{\partial f_g \partial \lambda_m}$ By differentiating (A5) with respect to λ_m, we have the second-order derivative $\frac{\partial^{\dagger 2} E_l}{\partial f_g \partial \lambda_m}$ given by

$$\frac{\partial^{\dagger 2} E_l}{\partial f_g \partial \lambda_m} = \sum_{r\in J_0}\sum_{s\in T_0}\left[\frac{\partial^\dagger\left(\frac{\partial E_l}{\partial h_r(s)}\right)}{\partial \lambda_m}\frac{\partial^\dagger h_r(s)}{\partial f_g} + \frac{\partial E_l}{\partial h_r(s)}\frac{\partial^{\dagger 2} h_r(s)}{\partial f_g \partial \lambda_m}\right] + \frac{\partial^\dagger\left(\frac{\partial E_l}{\partial f_g}\right)}{\partial \lambda_m}. \quad (A8)$$

Introducing

$$P_2(j,t,f_g,\lambda_m) = \frac{\partial^{\dagger 2} h_j(t)}{\partial f_g \partial \lambda_m}, \quad \cdots\cdots (A9)$$

the iterative calculation algorithm for $P_2(j,t,f_g,\lambda_m)$ can be obtained by differentiating (A7) for λ_m

$$\begin{aligned} &P_2(j,t,f_g,\lambda_m) \\ &= \sum_{i\in JK(j)}\sum_{p\in B(i,j)}\left[\frac{\partial^\dagger\left(\frac{\partial h_j(t)}{\partial h_i(t-D_{ij}(p))}\right)}{\partial \lambda_m} \times P_1(i,t-D_{ij}(p),f_g)\right] \\ &+ \sum_{i\in JF(j)}\sum_{p\in B(i,j)}\left[\frac{\partial h_j(t)}{\partial h_i(t-D_{ij}(p))} \times P_2(i,t-D_{ij}(p),f_g,\lambda_m)\right] \\ &+ \frac{\partial^\dagger\left(\frac{\partial h_j(t)}{\partial f_g}\right)}{\partial \lambda_m} \quad \cdots\cdots (A10) \end{aligned}$$

(Manuscript received October 24, 2000, revised March 23, 2001)

References

(1) F.C. Chen and C.C. Liu, " Adaptively controlling nonlinear continuous-time systems using multilayer neural networks", *IEEE Transaction on Automatic Control*, Vol. 39, No. 6, pp.1306-1310, June, 1994.

(2) F.C. Chen and H.K. Khalil, " Adaptive control of a class of nonlinear discrete time systems using neural networks", *IEEE Transaction on Automatic Control*, Vol. 40, No. 5, pp.791-801. May, 1995.

(3) M. S. Ahmed, "Neural-Net-Based Direct Adaptive Control for a Class of Nonlinear Plants", *IEEE Transactions on Automatic Control*, Vol. 45, No. 1, pp.119-124, January 2000.

(4) K. Hirasawa, J. Murata, J. Hu, C. Jin, "Universal Learning Networks and Its Application to Robust Control", *IEEE Trans. Syst., Man, and Cybern., part B*, Vol. 30, No. 3, pp.419-430, June 2000.

(5) M. Obayashi, K. Hirasawa, N. Toshimitsu, J. Murata and J. Hu, "Robust Control for Universal Learning Network Considering Fuzzy Criterion and Second Order Derivatives", *Trans. of SICE*, Vol. 34, No. 9, pp.1246-1254, 1998 (in Japanese).

(6) William R. Perkins, Jose B. Cruz, and Richard L. Gonzales, "Design of Sensitivity Systems", *IEEE Trans. Automatic Control*, Vol AC-13, No.2, pp.159-167, April, 1968.

(7) K. Hirasawa, X. Wang , J. Murata, J. Hu, C. Jin, "Universal Learning Network and its application to chaos control", *Neural Networks*, Vol.13, pp.239-253, 2000.

(8) R. Isermann, S. Ernst, and O. Nelles, "Identification with dynamic neural networks–Architectures, Comparisons, Application," in *Proc. IEEE Int. Conf. Syst.,Man, Cybern.*, 1998, pp.997-1022.

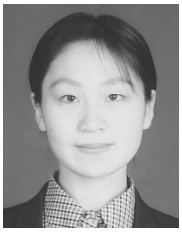
(9) M. Han, K. Hirasawa, M. Ohbayashi, and H. Fujita, "Modeling dynamic systems using universal learning networks," in *Proc. IEEE Int. Conf. Syst., Man, Cyben.,* Beijing, China, 1996, pp. 1172-1177.

(10) P.Werbos; Beyond Regression, "New Tools for Prediction and Analysis in the Behavioral Sciences", Ph.D.dissertation, Harvard University, 1974.

(11) Dipti Deodhare, M.Vidyasager, and S. Sathiya Keerthi, "Synthesis of Fault-Tolerant Feedforward Neural Networks Using Minimax Optimization", *IEEE Trans.Neural Networks*, Vol.9, No.5. pp.891-900, Sept.1998.

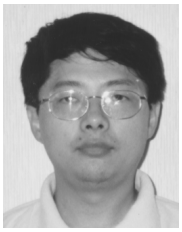
(12) J-S.R. Jang and C.T. Sun, "Neuro-Fuzzy Modeling and Control", *Proceedings of the IEEE*, Vol.83, No.3. pp.378–405, 1995.

Hongping CHEN (Non-member) She received the B.S. degree in Process Control Engineering from Dalian University of Technology, China in 1989. She received the M.S. degree in Electrical and Electronic System Engineering from Kyushu University in 2000. She is currently working toward the Ph.D. degree in Electrical and Electronic System Engineering at Kyushu University. Her research interests include neural networks and their applications in control systems.



Kotaro HIRASAWA (Member) He received the M.S. degree in Electrical Engineering from Kyushu University in 1966. From April 1966, he served in Hitachi Lab. of Hitachi Ltd., and in 1989 he was a vice president of Hitachi Lab.. From August 1991 to November 1992, he served in Omika Factory of Hitachi Ltd.. Since December 1992, he has been a professor in the faculty of Engineering, Kyushu University. Now he belongs to the Graduate School of Information Science, Kyushu University. Dr. Hirasawa is a member of the Society of Instrument and Control Engineers, a member of IEEE.

Jinglu HU (Member) He received the M.Sci. degree in 1986 from Zhongshan University, China and the Ph.D degree in 1997 from Kyushu Institute of Technology. From 1986 to 1993, he was a Research Associate and then a Lecturer in Zhongshan University. Since 1997, he has been Research Associate at Kyushu University. His current research interests are learning network theory, system identification and their applications. Dr. Hu is a member of the Society of Instrument and Control Engineers.



Junichi MURATA (Member) He received the D.Eng degree in Electrical Engineering from Kyushu University in 1986. From 1986 to March 1988, he was Research Associate at Kyushu University. From April 1988, he is an Associate Professor at Kyushu University. Now he belongs to the Graduate School of Information Science and Electrical Engineering, Kyushu University. He is a member of SICE, ISCIE and IEEE.