Control of Nonminimum Phase Systems with Neural Networks and Genetic Algorithm

Lae-Jeong Park*, Sangbong Park*, Zeungnam Bien*, and Cheol Hoon Park*

ABSTRACT

It is well known that, for nominimum phase systems, a conventional linear controller of PID type or an adaptive controller of this structure shows limitation in achieving a satisfactory performance under tight specifications. In this paper, we combine a neuro-controller with a PI-controller with off-line learning capability provided by the Genetic Algorithm to propose a novel neuro-controller to control nonminimum phase systems effectively. The simulation results show that our proposed model is more efficient with faster rising time and less undershoot effect when the performances of the proposed controller and a conventional form are compared.

I. Introduction

Due to various capabilities of neural networks such as massive parallelism, self-organization, and adaptive learning, neural networks have been paid a lot of attention as a possible means to solve some of outstanding problems for which conventional approaches may not work well in providing high performance. Industrial control systems in practice are mainly of PID type due to various advantages which such a simple structure renders. However, fine tuning of controller parameters might be a time consuming job, and obtaining a high control quality requires experts' knowledge about both control theory and process dynamics. Even though a lot of research has been conducted to design an adaptive control system when the system is linear, time invariant, and minimum phase, practical use of such theoretical results is limited to few industrial systems, and not much is known for dynamic systems with nonminimum phase characteristics and uncertainty.

Earlier works can be found in [1] and [2], where H. Elliot proposed a direct adaptive control structure for nonminimum phase systems by adjoining extra parameters in terms of partial state predictor[1], and L. Praly presented a method which estimates both model and controller parameters with conceptual least-square criterion minimization and without any extra condition[2]. These methods, however, are not capable of solving the original problems of nonminimum phase systems. Recently, a method with fuzzy logic was proposed which has been successfully applied to steam generator[3]. It is well known that controlling dynamic systems with nominimum phase characteristics, or with inverse response[4] is a very difficult task if only a linear controller such as PID is used for good performance and robustness.

*Department of Electrical Engineering Korea Advanced Institute of Science and Technology

Naturally, it is desirable to adopt a nonlinear controller in addition to the linear controller to alleviate the nonminimum phase effect and improve performance. Here, good performance refers to the time domain criteria such as less undershoot, faster rising time, and small steady state errors.

In this paper, we propose a novel neuro-controller with a PI controller with off-line learning capability provided by Genetic Algorithm to control nonminimum phase systems effectively.

Ⅱ. Nonminimum phase systems

When a nonlinear system is modeled and linearized at each operating point, it often happens that the linear system has one or more zeros which lie in the right half s-plane. Such a linear system is called a nonminimum phase system. For the nonminimum phase system, the transfer function may not have a stable inverse because of zeros in the right half s-plane.

For instance, consider the following system with stable poles. Since it has a zero at s = 1,

$$G(s)=\frac{(s-1)}{(s+1)\cdot(s+2)} \tag{1}$$

it has the effect of inverse response. Fig. 1 shows a general block diagram of PI controller where the controller parameters are denoted as K_p and K_i.

$$T(s)=\frac{(s-1)\cdot(K_p s+K_i)}{s(s+1)(s+2)+(K_p s+K_i)(s-1)} \tag{2}$$

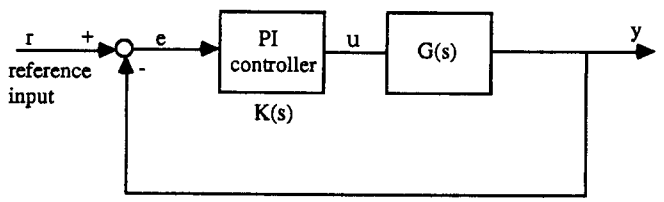

Fig. 1 General diagram of a PI controller, $K(s)=K_P+\frac{K_I}{s}=\frac{K_P s+K_I}{s}$.

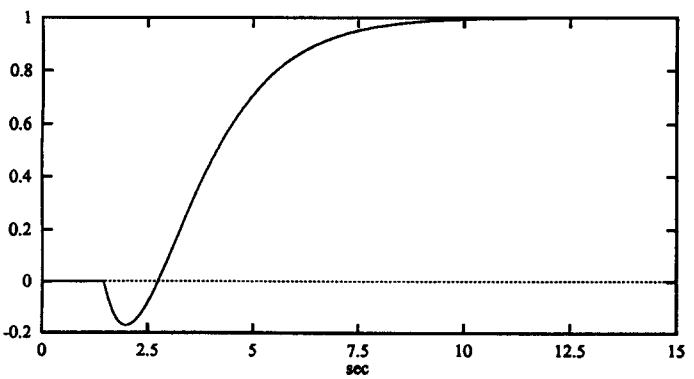

Fig. 2 A step response of a nonminimum phase system with a PI controller.

As shown in Eq. (2), the poles of the closed-loop system can be placed at desired positions. But the zero of the given system does not change by feedback of the closed-loop. Fig. 2 shows a step response of a nonminimum phase system with a PI controller. The system response has a little bit undershoot effect and a slow rising time. One may adjust the PI gains to obtain fast rising time and small undershoot, but the response then shows large overshoot and lengthy settling time due to chattering. These phenomena arise largely from the nonminimum phase characteristics.

To be specific, let us rewrite G(s) in Eq. (1) in the time-domain:

$$\frac{d^2y}{dt^2} + 3\frac{dy}{dt} + 2y = \frac{du}{dt} - u \tag{3}$$

Note that subtraction of the input terms in the right hands of Eq. (3) has key effect on the output of the system when the reference input changes. There exists a trade-off between undershoot and rising time. But in case that system has both nonminimum zero(s) and nonminimum pole(s), it much more difficult to apply a PI controller to satisfy the given performance specification.

Ⅲ. Controller design based on neural networks

In this section, we show several computational approaches for training a neuro-controller. Existing training methods are largely classified into two classes; feedforward approach and feedback one. Fig. 3 shows three different approaches of realizing a feedforward neuro-controller with system inverse model.

The simplest approach for obtaining the system inverse model of an unknown system is shown in Fig. 3.1. The system inverse model identified by a neural network is set in the opposite input-output direction to that of the system, as shown by the arrow. The neuro-controller is trained by the error signal given as the difference between the target output and the actual output for the given input. This approach to acquire a system inverse model is referred to as direct inverse modeling by neuro-identifier[5, 6]. Even though this model may be able to acquire inverse system dynamics of the given system in case that its identification is accurate, it is necessary to train the model with all possible data of input space, and this approach seems to have a drawback that it does not necessarily achieve a particular target trajectory, that is, it does not ensure which input range of the neural network is appropriate to generate a particular output; the learning is not goal-directed[5].

Fig. 3.2 shows the scheme of combining a forward model and the inverse model[7]. First, the forward model of an nuknown system may be identified by a neural network if system dynamic equation is not known. Second, by using reference input and its output, the neuro-controller is trained with errors through either a Jacobian of the system dynamics or the neuro-identifier by the backpropagation of the error signal between the target output and actual output. Many local minima and lengthy training time of the neuro-controller, however, make it difficult to train the neuro-controller.

Fig. 3.3 shows feedback error learning, which provides good control results for robot manipulators, proposed by Kawato[8]. This approach has advantage to perform the control and learning simultaneously, although the difficulty in achieving the feedback controller of the forward kinematics of system exists.

Fig. 4 shows a feedback approach which replaces a conventional controller with a neuro-controller. In this case, a neural network is adopted not as a system inverse model of the given system, but only as a nonlinear controller.

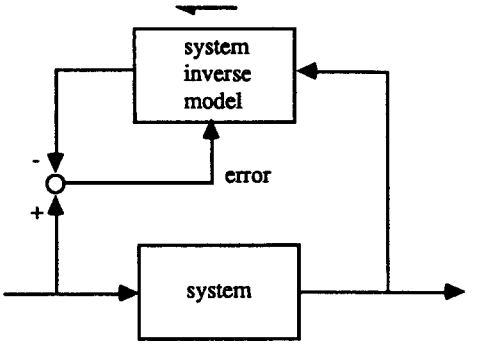

Fig. 3.1 Direct inverse modeling.

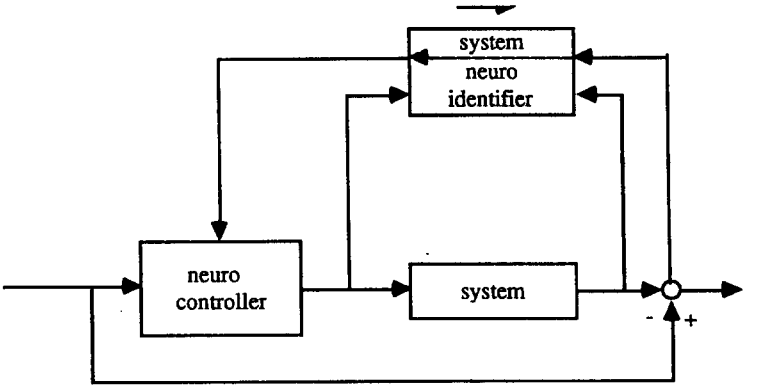

Fig. 3.2 Forward and inverse modeling.

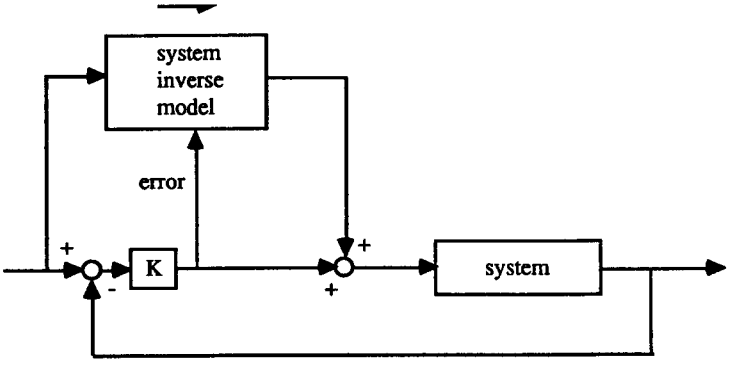

Fig. 3.3 Feedback error learning.

Fig. 3 Three computational approaches for learning the inverse model of a controlled plant.

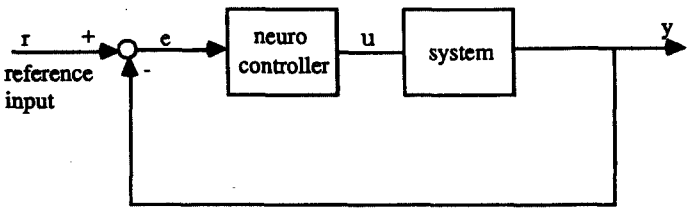

Fig. 4 Learning model for a feedback controller.

Ⅳ. Nonminimum phase systems and neuro-controller

Although we have tried to train neural networks with such learning methods as shown in Fig. 3 and 4, we have not been successful in training the neuro-controller with the conventional learning algorithms. The trained neuro-controller tended to blow up after some iterations or the controller seemed not to be trained at all. Simulation results of a system with a nonminimum phase zero using the direct forward/inverse modeling controller and feedback modeling controller respectively show that the undershoot effect is not decreased and, instead, increased even more with a little bit fast rising time and the step response eventually blows up. We have found some problems in training a neuro-controller with the conventional approaches as follows.

First, to train the neuro-controller with direct inverse model and forward/inverse model, the calculation of the errors at the output of the neuro-controller is required. For conventional approaches, the output errors of the neuro-controller are unkown because of the unknown desired control action and the system's output errors and backpropagated through the system Jacobian or the neuro-identifier. However, the system itself has unstable inverse model, which means that it might be difficult to update the interconnection weights of the neuro-controller through the backpropagation of system's output errors.

Second, the conventional approaches are based on instantaneous evaluation of system performance, which is the judgement of performance with respect to the relation between the input and its system output at each moment. As for general control command, if large error is detected, the controller commands to make more large action(controller's output), but for such controller's output, nonminimum phase systems show the negative effect as mentioned in section Ⅱ, at the range of undershoot: the more control command, the more the undershoot effect appears, and the undershoot affects the whole system performance, and the whole system blows up! These phenomena may be dangerous for some applications.

Therefore, we need a new learning algorithm based on global evaluation(not instantaneous performance evaluation) and/or controller output errors(not system output errors) to train the neuro-controller of nonminimum phase systems.

Ⅴ. Genetic Algorithm

A. History and algorithm

The underlying principles and mathematical framework of Genetic Algorithm(GA) were developed by J. Holland in 1960's, and it has been used in various application fields such as classifier systems, function optimization, and combinatorial optimization[9]. Diverse application capability is fundamentally attribu-

table to the GA's outstanding features such as robustness. Recently research on GA as a training algorithm of neural networks has been actively carried out and has shown good performance in some cases, where conventional training algorithm such as backpropagation(BP) can not be used directly.

GA is a stochastic search algorithm based on natural selection and natural genetics. GA is significantly different from other search techniques in that it mimic adaptation mechanism or survival of the fittest organisms in natural systems and adopts the coding technique of genetics. Compared to other search algorithms, the significant features of GA are considered as follows[10]: First, it works with the coded finite-length strings of parameters, chromosomes, instead of real parameters. Second, it searches for an optimal solution with maintenance of not a single point but many points in the search space. Third, it is guided in a probabilistic fashion, not deterministic one.

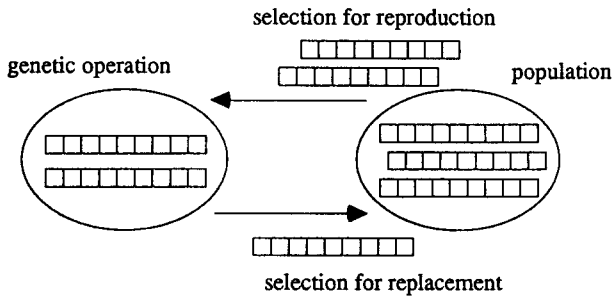

Fig. 5 The conceptual diagram of Genetic Algorithm.

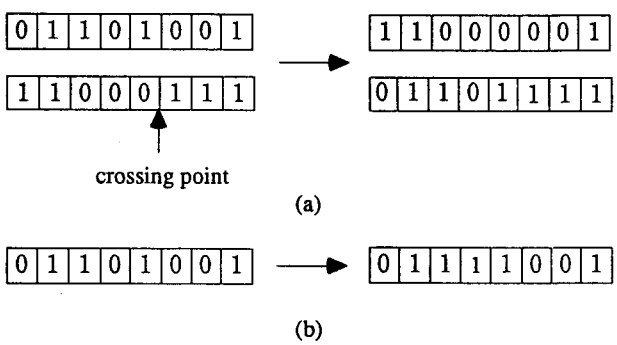

Fig. 6 Genetic operators: (a) crossover, (b) mutation

Fig. 5 shows a conceptual diagram of GAs. First, two chromosomes in the population that represent two solutions of the given problem respectively are selected according to their appropriateness. Parents, the selected chromosomes, are processed with genetic operators such as crossover and mutation in order to reproduce new chromosomes. New offsprings are evaluated by a criterion function that measures their fitness, and then are inserted into the population. Thereafter, in order to maintain the same number of chromosomes in the population, chromosomes having worse fitness are deleted during selection for replacement. Until GA finds the optimal solution, this procedure is repeated.

The performance of GA considerably depends upon the following three components: coding technique, selection for reproduction and replacement, and genetic operator. Even though several coding techniques are available, depending on the given problem, the binary coding technique is widely used in which all chromosomes are composed of 0's and 1's. During reproduction and replacement, each chromosome is usually selected with probability proportional to its fitness. This selection operation has a significant effect on the GA's performance. Specifically speaking, if chromosomes with higher fitness always survive during selection operation, GA might find local solutions. Otherwise, or if randomly selected regardless of their fitness, GA is not different from random search algorithms. Therefore, it is important to adopt the bias tendency of selection operation adequately. The genetic operators are analogous to the biological ones of crossover and mutation whose functions are shown in Fig. 6.

B. GA for training of neuro-controller

Since conventional training algorithms of neuro-controller do not work well when the plant has nonminimum phase characteristics, GA is adopted as a training algorithm of the neuro-controller. It has merits and weaknesses as a training algorithm of a neuro-controller[11, 12]. It is generally impossible to train a neuro-controller on-line because GA can not utilize performance evaluation of the neuro-controller at each instant. If the given dynamic system enables us to make a decision whether the network output in each moment is appropriate or not, we do not have to use GA or other search algorithms in order to train the neuro-controller. Such dynamic systems can be controlled by conventional training methods in which a good control action in each moment do not have bad influence on performance later.

However, neuro-controllers of a nonminimum phase system can not be easily designed by conventional training algorithms based on instantaneous judgments because of the characteristics of the systems. So, it is necessary to devise a new training method which enables us to evaluate global performance of the neuro-controller, that is, the appropriateness of the control at each instant in the context with a global basis such as response curve of the plant. In this sense, GA may be an effective training algorithm when an instantaneous evaluation method can not be applied to train neuro-controller successfully. This is a main advantage of GA as a training algorithm even though on-line training is not possible. We explain how we have modified each component of GA to train our neuro-controller successfully.

A. Coding: all interconnection weights of a neuro-controller without bias thresholds are represented by a chromosome which is composed of several substrings. A coded substring that represents a specific interconnection weight of neuro-controller has 16 bit length and a value between -10 and 10. For example, if the structure of the neuro-controller has 5-10-1 neuron for input, hidden, and output, respectively, the length of a chromosome is equal to 960 bits. Because GA should find adequate parameters(interconnection weights) of the dynamic system(the neuro-controller), each interconnection weight should have different dynamic range, that is, because the neuro-controller has delays of its output as inputs, the interconnection weights through which such inputs propagate should have the limited dynamic range so that the neuro-controller is stable. Otherwise, the output of neuro-controller may blow up. We restricted the dynamic range of such interconnection weights to lie between -0.5 and 0.5.

B. Population: when the number of chromosomes in the population is not large, GA often shows the phenomenon that all chromosomes in the population become very similar to each other before it finds the optimal solution of a given optimization problem. Therefore, until GA finds are optimal solution, maintenance of diverse genes in the population without disturbance of guiding a search path is necessary. To combat premature gene loss, many heuristic methods have been suggested. Among many methods, parallel GA(PGA) reportedly shows good performance compared to serial GA(conventional GA). In

PGA, each subpopulation operates in the same fashion like serial GA, and it exchanges its best solution with neighboring subpopulations' best ones at each predetermined epoch. Neighboring subpopulations are decided according to a given topological configuration. We use 20 subpopulations with ring configuration, each subpopulation has 30 chromosomes, and exchange of the best's is performed at every 60 epoch.

C. Crossover and mutation: as shown in Fig. 6, one-point crossover selects a crossover point randomly and then crosses the two strings at the point. However, when chromosomes are long, one-point crossover can not process useful schemata effectively because new chromosomes having various schemata can not be generated sufficiently[6]. So, we use a modified crossover. At first, one third of the total bits in a chromosome are randomly selected, and then the genes of the parent chromosomes are exchanged at the selected sites. This crossover can generate much new chromosomes having diverse genes than conventional one-point crossover. And a conventional mutation is used.

D. Fitness: GA necessarily uses a cost function which enables us to evaluate the extent to which each chromosome is suitable to the given objective. Our objective is to reduce the undershoot and to achieve fast rising time simultaneously. It is, however, not easy to find an appropriate cost function that can evaluate fast rising time and less undershoot using only the acceptable sequared errors between the actual output and the desired output of the plant. Such performance specification must be fuzzy and such a mathematical cost function can not be derived easily. Therefore, as in other techniques, we use a new cost function by a linear combination of two cost functions: the first one represents the sum of squared errors, while the other the undershoot effect as shown Eq. (4).

$$E_1 = \int_0^T (T_d(t) - T_a(t))^2 \, dt \tag{4.1}$$

$$E_2 = \int_0^{T_u} (T_d(t) - T_a(t))^2 \, dt \tag{4.2}$$

$$E = \alpha \times E_1 + \beta \times E_2 \tag{4.3}$$

where $T_d(t)$ and $T_a(t)$ are the desired step response and actual one respectively, and T and T_u are training time interval and the duration of undershoot respectively.

As expected, in this case the characteristics of the obtained response curve depend considerably on the control parameters, α and β. It is difficult to find the optimal values of two parameters for GA to search for the best solution because the performance specification in itself can not be exactly described. So we find suitable values of α and β by trial and error. In computer simulation α is 1, and β 60.

Ⅵ. The architecture of neuro-controller model

A. System configuration

Fig. 7 shows the block diagram of our system configuration. A PI controller and a feedback neuro-controller are simultaneously used to control a nonminimum phase system. We do not design the neuro-controller as a replacement of a conventional linear controller. Instead, our design is based on the principle that it is preferable that nonlinear controllers actively control the controlled plant in local operating region or local environment where linear controllers only can not fulfill the given performance specification.

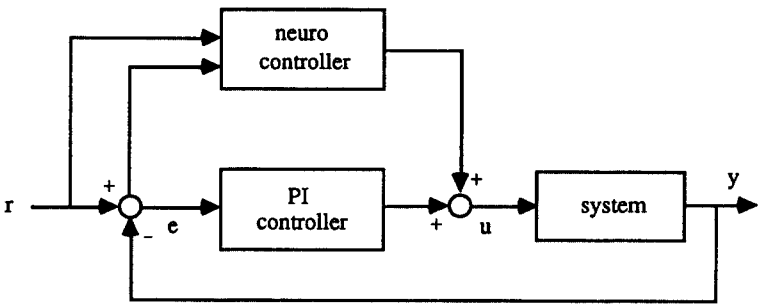

Fig. 7 The block diagram of our system configuration based on the neuro-controller.

In this paper, we use an additional neuro-controller to activate in local regions so that the performance specifications such as fast rising time and less undershoot are satisfied. One reason to use a PI controller in our configuration is to reduce the steady state error.

B. The structure of proposed neuro-controller

How do we activate the neuro-controller only when it is needed? And when is the neuro-controller necessary? The problems are naturally related with quantities which are used as the inputs of the neuro-controller. Fig. 8 and Fig. 9 give answers about the two questions. As shown in Fig. 8, the neural network has seven input neurons and one output neuron(in this paper, we consider only SISO model) whether it has hidden neurons or not.

By analyzing the step response of the plant and the error curve with only a PI controller, we can determine which quantities are used for the input neurons of the neural network. For example, the region indicated by a circle on the error curve shown in Fig. 9 can be regarded as the sign of the undershoot effect of the nonminimum phase system. When the inequality (5) holds, the undershoot effect is marked in effect.

$$\frac{de}{dt} \times e > 0 \tag{5}$$

Using the product terms of the error and the error derivative as the input of the neural network, we can activate the neuro-controller in local regions where the reference input changes without interference with a PI controller so that the step response of the plant gives significant reduction of undershoot. Because we consider only the step response of the nonminimum phase system in this paper, the input

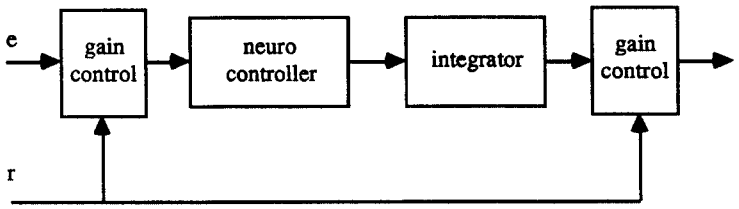

Fig. 8 The block diagram of the neuro-controller.

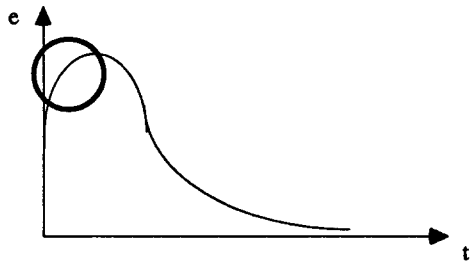

Fig. 9 A error curve of a system output with a PI controller.

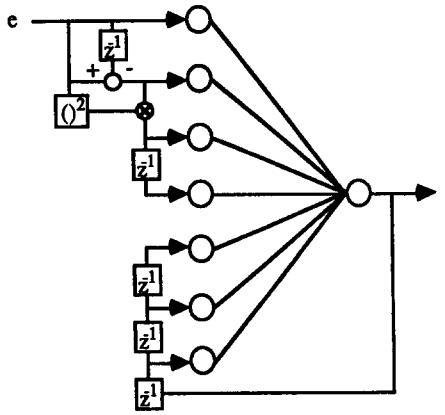

Fig. 10 The architecture of the neuro-controller.

neuron of this neuro-controller such as the high order neuron may need to have different forms if other reference signals were presented or other plants were considered. However, some observation and analysis on the given nonminimum phase system in this manner can, we think, enable us to find appropriate input neurons. Overall architecture of the neuro-controller is shown in Fig. 10. The reason why we use an integrator is to reject the high frequency components of the output of the neural network, and to achieve the amplification on the contribution of local activity of the neuro-controller for reduction of undershoot. Without the integrator, we could not get even small improvement.

The gain controller is used to adapt the neuro-controller to various magnitude of the step reference input successfully. It scales the input and output of the neuro-controller by the change of the current reference signal with respect to the delayed reference one in order that the operating region of the neuro-controller does not change.

Ⅶ. Simulations

For simulation, we have assumed the following:

1. The plants are described by linear models.
2. The systems and PI control parameters are known.

To control nonlinear systems, they should be first modeled and linearized. And as an available way to obtain the parameters of the system and controller, the direct adaptive control scheme may be adopted [2], and/or neuro-identifier may be used. Therefore, our assumptions may not make it impossible to apply our control model to nonlinear systems and/or unknown systems.

Two linear models for controlled plants are used in simulation.

$$\frac{(s-1)}{(s+3)(s+4)} \tag{6}$$

$$\frac{(s-1)}{(s+2)(s-0.128)} \tag{7}$$

The first model has only a nonminimum zero and the second one both nonminimum pole and zero. The sampling time of each model is 0.05 sec and 0.2 sec, respectively. As mentioned before, only PI controllers can not meet the performance specification due to the systems' characteristics and it is more difficult for the second system to find a proper linear controller. The proposed learning algorithm is applied to each model. Fig. 11 and 12 show the actual output of the first system and controller respectively when only PI controller is used. The undershoot effect and slow rising time are shown. In Fig. 13, the performance comparison is given between PI controller and the proposed model; the dotted line is actual output of system with a two-layer neural network and the (short) deshed line is one with a single layer neural network. The proposed model improves performance such as less undershoot and faster rising time. Fig. 14 shows the controller output; solid line is one of neural network controller and deshed line is the total controller output; the sum of the PI controller output and neuro-controller. Fig. 15 shows the capability of generalization of proposed model for several step inputs. Fig. 16-20 show the output of system/controller of the second model, which has both nonminimum pole and zero. The unstable pole of system can be replaced with the desired stable one due to adjustment of PI controller gain, even though the performance of PI controller may be poor. Overall the steady state error can be reduced due to using the PI controller, and the performance is considerably improved by the nonlinear neuro-controller. The neuro-controller only activates during the transition time. However, the chattering of the controller output in Fig. 19 may not be desirable because it is impossible, for instance, to control the feedwater valve of steam generator like Fig. 19. If the penalty term which restricts the high frequency of the controller's output is added to the cost function of GA in Eq. (4), the problem of chattering may be alleviated to some degrees.

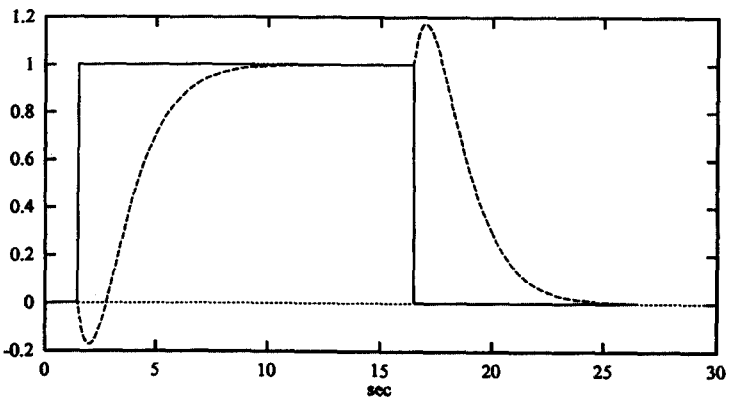

Fig. 11 Step response of G(s) with a PI controller, $G(s) = (s-1)/(s+3)(s+4)$.

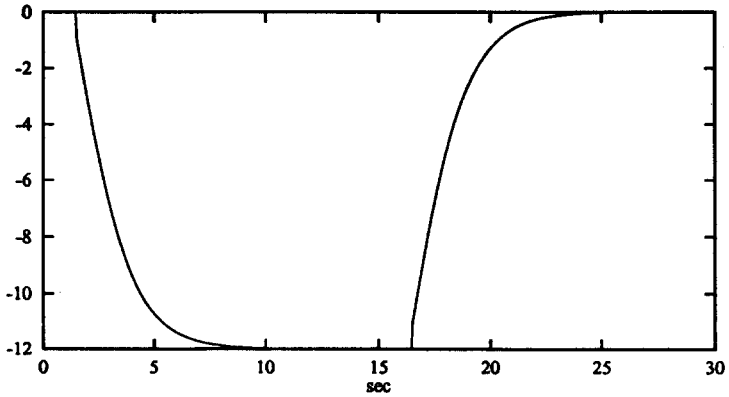

Fig. 12 The output of a PI controller, $G(s) = (s-1)/(s+3)(s+4)$.

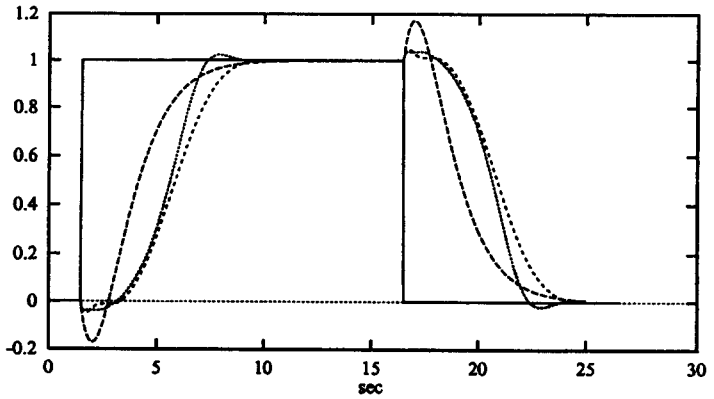

Fig. 13 Performances of the proposed neuro-controller and a PI controller, $G(s) = (s-1)/(s+3)(s+4)$.

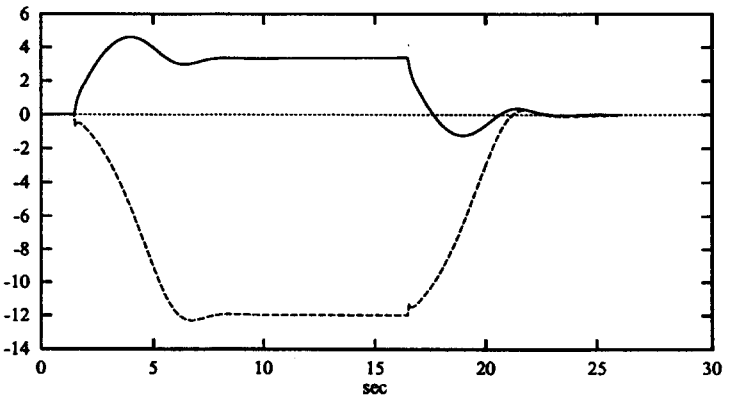

Fig. 14 The output of the proposed neuro-controller, $G(s) = (s-1)/(s+3)(s+4)$.

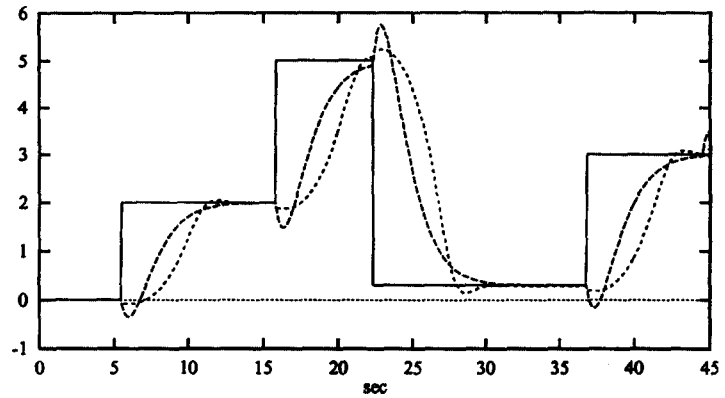

Fig. 15 Generalization capability of the neuro-controller, $G(s) = (s-1)/(s+3)(s+4)$.

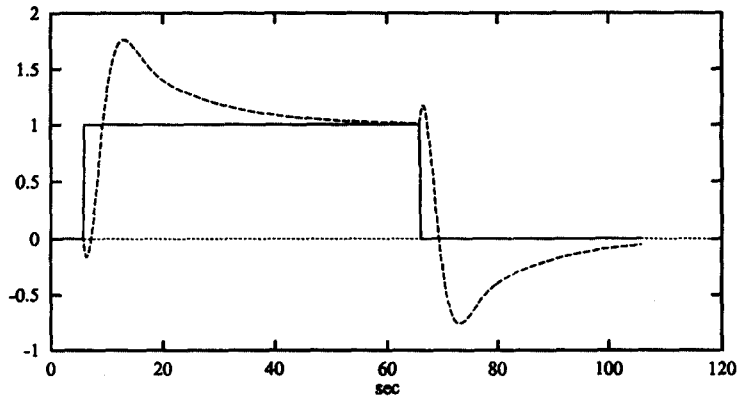

Fig. 16 Step response of G(s) with a PI controller, $G(s) = (s-1)/(s+2)(s-0.128)$.

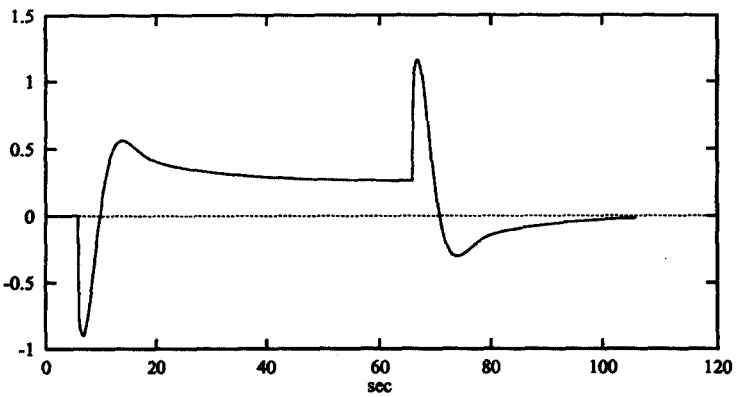

Fig. 17 The output of a PI controller, $G(s) = (s-1)/(s+2)(s-0.128)$.

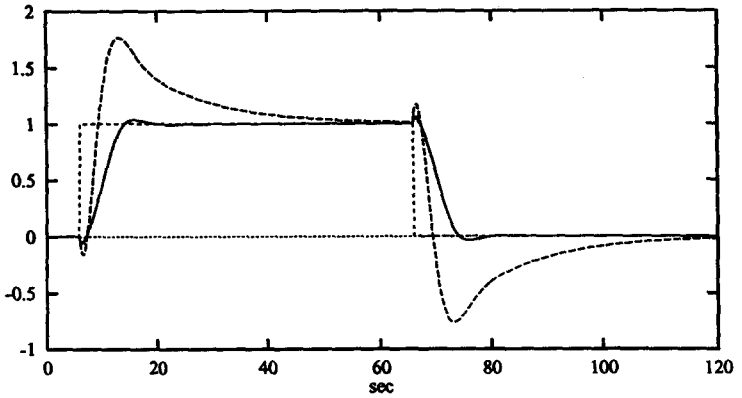

Fig. 18 Performances of the proposed neuro-controller and a PI controller, $G(s) = (s-1)/(s+2)(s-0.128)$.

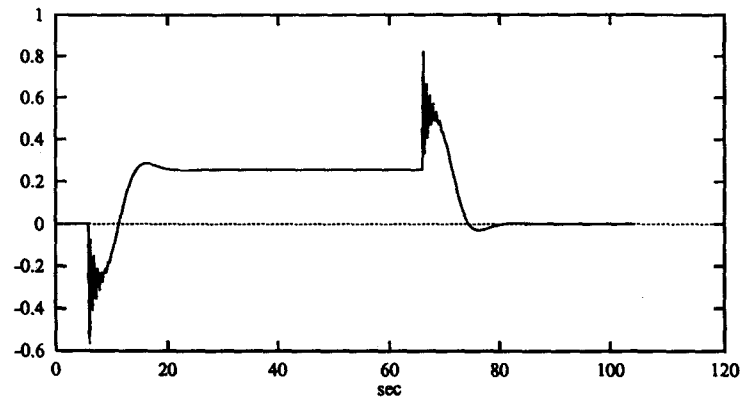

Fig. 19 The output of the proposed neuro-controller, $G(s) = (s-1)/(s+2)(s-0.128)$.

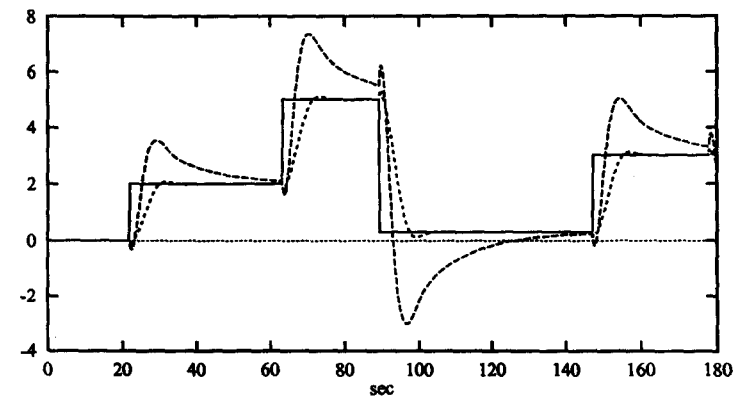

Fig. 20 Generalization capability of the neuro-controller, $G(s) = (s-1)/(s+2)(s-0.128)$.

Ⅶ. Conclusion

In this paper, we have shown that the conventional neuro-control approaches fail to acquire the system inverse dynamics of nonminimum phase systems, so we have proposed a novel parallel architecture with a PI controller and a neural network controller to control nonminimum phase systems and a new learning algorithm based on global evaluation of performance with Genetic Algorithm. GA is adopted to train neuro-controller with off-line learning. In simulation of two linear models, the results show the successful improvement of performance; less undershooot and faster rising time. For further works, it is necessary to find a more effective learning algorithm for nonminimum phase systems as well as minimum phase systems, based on global performance judgment, not instantaneous one.

References

1. H. Elliott, "Direct adaptive pole placement with application to nonminimum phase systems," *IEEE Trans. Automat. Contr.*, vol. AC-27, no. 3, pp. 720-721, June, 1982.
2. L. Praly, "Towards a globally stable direct adaptive control scheme for not necessarily minimum phase systems," *IEEE Trans. Automat. Contr.*, vol. AC-29, no. 10, pp. 946-949, Oct., 1984.
3. N. Na, K. Kwon, and Z. Bien, "A Study on water level control of PWR steam generator at low power operation and transient states," *Journal of Fuzzy Logic and Intelligence Systems*, vol. 3, no. 2, pp. 18-35, June, 1993.
4. C. A. Smith, A. B. Corriplo, "*Principle and practice of Automatic Process Control,*" John Wiley & Sons., 1985.
5. W. T. Miller, R. S. Sutton, and P. J. Werbos, *Neural Networks for Control*, MIT Press, Cambridge, 1990.
6. D. H. Nguyen, B. Widrow, "Neural networks for self-learning control systems," *IEEE Control Systems Mag.*, vol. 10, no. 3, pp. 18-23, 1990.
7. K. S. Narendra, K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Networks*, vol. 1, pp. 4-27, Mar. 1990.
8. M. Kawato, "Feedback-error learning neural network for trajectory control of a robotic manipulator," *Neural Networks*, vol. 1, pp. 251-265, 1988.
9. J. Holland, *Adaptation in Natural and Artificial Systems*, MIT Press, 1992.
10. D. Goldberg, *Genetic Algorithm in search, optimization, and machine learning*, Addison-Wesley, 1989.
11. Y. Ichikawa, T. Sawa, "Neural network application for direct feedback controllers," *IEEE Trans. Neural Networks*, vol. 3, no. 2, pp. 224-231, 1992.
12. B. Yoon, "More efficient genetic algorithm for training layered feedforward neural networks," *Proc. of JCEANF*, pp. 622-629, 1992.