Chinmoy Pal
Ichiro Hagiwara
Nissan Motor Corporation
Research Center, Natsushima-cho
Yokosuka 237, Japan

Naoki Kayaba
Shin Morishita
Yokohama National University
Dept. of Naval Architecture and
Ocean Engineering
156 Tokiwadai, Hodogaya-ku
Yokohama 240, Japan

A Learning Method for Neural Networks Based on a Pseudoinverse Technique

A theoretical formulation of a fast learning method based on a pseudoinverse technique is presented. The efficiency and robustness of the method are verified with the help of an Exclusive OR problem and a dynamic system identification of a linear single degree of freedom mass–spring problem. It is observed that, compared with the conventional backpropagation method, the proposed method has a better convergence rate and a higher degree of learning accuracy with a lower equivalent learning coefficient. It is also found that unlike the steepest descent method, the learning capability of which is dependent on the value of the learning coefficient η, the proposed pseudoinverse based backpropagation algorithm is comparatively robust with respect to its equivalent variable learning coefficient. A combination of the pseudoinverse method and the steepest descent method is proposed for a faster, more accurate learning capability. © *1996 John Wiley & Sons, Inc.*

INTRODUCTION

Neural network models are very efficient in computing problems where many assumptions have to be satisfied in parallel, as in the case of image-recognition problems. This is accomplished, in contrast to classical sequential computers, by using networks of analog neurons with nonlinear behavior and with a high degree of interconnectivity. A neural network is defined by its node characteristics, the learning rules, and the network topology. The learning rules control the improvement of the network performance through appropriate adaptive changes of the weights of the links. Recently, there has been an increasing number of studies on neural networks in the fields of pattern recognition, system identification, control, and other interdisciplinary areas. The pioneering work in the field of neural networks is usually attributed to McCullouch and Pitts (1943), who developed a simplified model of the neuron (the basic unit of the brain) consisting of variable resistors. Seminal contributions were also made by Hebb (1949), Rosenblatt (1962), Minsky and Papert (1969), Werbos (1974), and Grossberg (1976). Interest in the analysis of neural networks was revived in the early 1980s due to the work of Hopfield (1982), and later on to the works of Kohonen (1984), McLelland and Rumelhart (1986), and Mead (1989). Detailed background information can be found in some recent publications by Aleksander and Morton (1990), Khanna

Received August 1, 1995; Accepted December 14, 1995.
Shock and Vibration, Vol. 3, No. 3, pp. 201–209 (1996)

CCC 1070-9622/96/030201-09

(1990), Nelson and Illingworth (1991), and Ngyyen and Widrow (1990).

The unique features of an artificial neural network model are: its ability to perform computations at much higher speeds on massively parallel nets of simpler computation elements, unlike the sequential computations of a Von Neuman machine; its higher degree of robustness or fault tolerance; and its adaptation or learning capability due a greater number of locally connected processing nodes. These attributes make it a versatile tool and an ideal choice in cases where real-time adaptations (Narendra and Parthasarathy, 1991) and fast processing of large amounts of data are key issues.

Neural networks, representing an ideal choice for real-time adaptation (Morishita and Ura, 1993), can be broadly classified into two groups: a layer-type neural network and a fully connected neural network. This article presents a new training algorithm that is applicable only to layer-type neural networks.

The backpropagation method is one of the most frequently used techniques in training a layer-type neural network. Because a neural network has the capability of learning, nonlinear numerical mathematical optimization techniques, such as the steepest descent method or conjugate gradient method, are frequently used to train a neural network by calculating optimum values of its parameters. However, most of these methods have some drawbacks, one of which is their dependency on the value of the learning coefficient. This has spurred further research on the development of more efficient and robust algorithms. Some advances have been made in this connection, for example, optimization of the slope of the sigmoid function (Yamada and Yabuta, 1992; Pal et al., 1994) and introduction of a genetic algorithm for the global optimum (Davis, 1991), an augmented Kalmal filter algorithm (Murase et al., 1991; Iignuni et al., 1992), and an optimal filtering algorithm (Shah et al., 1992). Regarding the learning algorithm as the solution of a nonlinear optimization problem, Newton's method, a well-known iterative technique for solving a general nonlinear problem, can be effectively used in off-line training of a multilayered neural network (Anderson, 1988). Although it converges in fewer iterations than the steepest descent based algorithm, it requires too many computations to be effective and feasible for on-line structural identification and control problems. A simplified second-order algorithm (Parker, 1987) has the drawback of requiring adjustment of many tuning parameters. To achieve acceptable trade-offs among learning speed and accuracy, computational cost, and the simplicity and robustness of the learning process, a new learning algorithm based on a pseudoinverse technique that is used in the analysis of stability of structures and nonlinear structural optimization is proposed here for application to practical on-line problems. The efficiency and robustness of the proposed method are verified with the help of an Exclusive OR (XOR) problem and a dynamic identification of a linear single degree of freedom mass–spring model subjected to an external sinusoidal excitation using time series response data. Finally, a combination of the proposed method and the existing steepest descent method is suggested as a way of obtaining a faster, more accurate learning capability.

ARCHITECTURE AND PARAMETERS OF NEURAL NETWORK

Nomenclature and Symbols

The following are used in Fig. 1:

$f(x_n, u_{0n})$	$1/\{1 + \exp(-u_{0n} \cdot x_n)\}$, $\mu_{0n} = 2/U_0$, sigmoid function
U_0	temperature of a unit which has inverse relation to the slope of the sigmoid function
O_k	$f(S_k, u_{0k})$ = output value of the kth unit of the output layer
H_j	$f(U_j, u_{0j})$ = output value of the jth unit of the hidden layer
I_i	output value of the ith unit at the input layer
x_n	internal potential at the nth unit
V_{kj}	connecting weight joining the jth unit of the hidden layer and kth unit of the output layer
W_{ji}	connecting weight joining the ith unit of the input layer and jth unit of the hidden layer
θ_j	threshold value of the jth unit of the hidden layer
γ_k	threshold value of the kth unit of the output layer
S_k	$\Sigma V_{kj}H_j + \gamma_k$ = the internal potential of the kth unit of the output layer
U_j	$\Sigma W_{ji}I_i + \gamma_k$ = the internal potential of jth unit of the hidden layer
T_k	teaching signal of the kth unit of the output layer

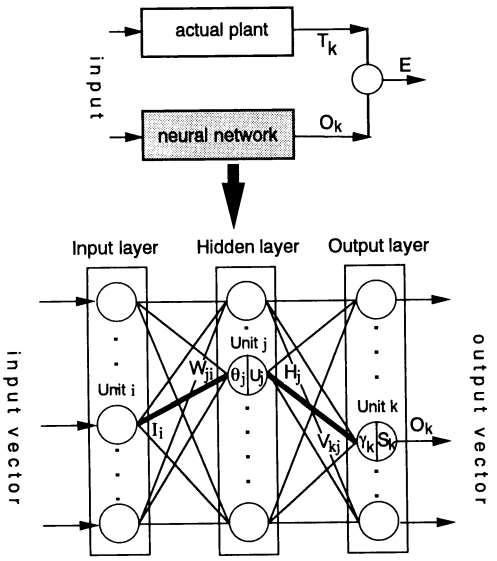

FIGURE 1 Three-layered neural network.

u_{0n}	slope of the sigmoid function of the nth unit
u_{0k}	slope of the sigmoid function of the kth unit of the output layer
u_{0j}	slope of the sigmoid function of the jth unit of the hidden layer
η	learning parameter/coefficient for connecting weights
η_u	learning parameter/coefficient for the slope of the sigmoid function
m	mass
c	damping coefficient
k	spring constant

APPLICATION OF PSEUDOINVERSE METHOD IN TRAINING A LAYER-TYPE NEURAL NETWORK

The present section is devoted to the detailed mathematical formulation of the proposed pseudoinverse based backpropagation method that is used in feedforward training of a neural network. In general, the training of a neural network is carried out by minimizing the error function E_p as calculated from the difference between the output O_k and the teaching signal T_k for the input pattern p.

$$E_p = \frac{1}{2}\sum_k (T_k - O_k)^2. \tag{1}$$

Here, k indexes the units (neurons) and a summation of error is carried out for all the units of the output layer to calculate the total error. With the backpropagation method, a sensitivity analysis of the error surface is performed with respect to the neural network parameters, such as the connecting weights, threshold values, and the slope of the sigmoid function. These parameters are modified or updated with the help of the above-mentioned sensitivity values by multiplying them by an arbitrarily fixed factor, i.e., learning coefficient η, to be chosen properly by the designer. However, the value of this learning coefficient η significantly affects the speed, accuracy, and hence the efficiency of training an untuned neural network.

Although the existing steepest descent method minimizes the error, it does not yield an ideal target value having zero error. With the method proposed here, after making a first-order approximation of the error surface, the parameters can be modified under the assumption that the value of the error will be reduced to zero after the first approximation. The parameters of the neural network can be represented as the components of a vector $\boldsymbol{\psi}$ as defined by the following equation:

$$\boldsymbol{\psi} = \{\psi_1 \psi_2 \cdot \cdot \cdot \psi_n\}^T \tag{2}$$

where n denotes the total number of parameters. The sensitivity matrix of the error function E_p can be represented in matrix form by Eq. (3):

$$\mathbf{Z} = \begin{bmatrix} \frac{\partial E_1}{\partial \psi_1} & \frac{\partial E_1}{\partial \psi_2} & \cdots & \frac{\partial E_1}{\partial \psi_n} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial E_p}{\partial \psi_1} & \frac{\partial E_p}{\partial \psi_2} & \cdots & \frac{\partial E_p}{\partial \psi_n} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial E_m}{\partial \psi_1} & \frac{\partial E_m}{\partial \psi_2} & \cdots & \frac{\partial E_m}{\partial \psi_n} \end{bmatrix}$$

where m is the total number of input patterns. On the other hand, by defining the error vector $\mathbf{e}$ and the parameter modification vector $\Delta\psi$,

$$\mathbf{e} = \{E_1 E_2 \ldots E_p \ldots E_m\}^T \tag{4}$$

$$\Delta\boldsymbol{\psi} = \{\Delta\psi_1 \Delta\psi_2 \ldots \Delta\psi_n\}^T. \tag{5}$$

The following simple relation can be expressed as

$$\mathbf{Z} \cdot \Delta\psi = \Delta\mathbf{e}, \tag{6}$$

$$\Delta\mathbf{e} = -\alpha\mathbf{e}. \tag{7}$$

Equation (6) states that the amount of change or reduction in the value of error is nothing but the product of the error sensitivity matrix $\mathbf{Z}$ and the parameter modification vector $\Delta\Psi$. Here, $\alpha(0 < \alpha < 1)$ stands for the fraction of total error to be minimized at any stage or iteration level during modification of the neural network parameters based on the above linearized assumption defined by Eq. (6). A suitable value of this parameter α can be chosen according to the degree of nonlinearity present, which differs from problem to problem. Parameters can be updated with the help of a successive modification method and a multistep batch modification (single modification for a certain number of input patterns) method. In the case of the successive modification method, as modification is performed for each pattern, $m = 1$ and the above-mentioned sensitivity matrix will be of the $(1 \times n)$ order column vector. On the other hand, with a multistep batch modification method, the size of the rectangular matrix is $(m \times n)$, because modification is carried out after calculating the error for every m input patterns. To calculate the modification vector $\Delta\Psi$ from Eq. (6), one has to determine the inverse of the rectangular sensitivity matrix $\mathbf{Z}$, which cannot be solved by a simple inverse technique. Hence, to accomplish that, we introduce the concept of a generalized pseudoinverse method. The remaining part of the formulation is based on the successive modification method.

Defining the pseudoinverse of a rectangular matrix $\mathbf{Z}$ as $\mathbf{Z}^-$, the following equation can be written for a full rank $\mathbf{Z}$ matrix.

$$\mathbf{Z}^- = \mathbf{Z}^T(\mathbf{Z} \cdot \mathbf{Z}^T)^{-1}. \tag{8}$$

It should be noted that the solution of Eq. (6), i.e., the value of vector $\Delta\Psi$ that satisfies the equation above, may not be unique. Therefore, an approximate optimum solution of the modification vector $\Delta\Psi_0$ can be obtained by minimizing the following expression given by Eq. (9) with respect to $\Delta\Psi$.

$$(\mathbf{Z} \cdot \Delta\psi - \Delta\mathbf{e})^T \cdot (\mathbf{Z} \cdot \Delta\psi - \Delta\mathbf{e}). \tag{9}$$

In a strict mathematical sense, $\Delta\Psi_0$ is a subset of the solution space of all possible $\Delta\Psi$ that satisfy Eq. (10).

$$\begin{aligned}(\mathbf{Z} \cdot \Delta\psi - \Delta\mathbf{e})^T \cdot (\mathbf{Z} \cdot \Delta\psi - \Delta\mathbf{e}) \\ \geq (\mathbf{Z} \cdot \Delta\psi_0 - \Delta\mathbf{e})^T \cdot (\mathbf{Z} \cdot \Delta\psi_0 - \Delta\mathbf{e}).\end{aligned} \tag{10}$$

Hence, in other words, $\Delta\Psi_0$ must satisfy Eq. (11) and (12):

$$\begin{aligned}(\mathbf{Z} \cdot \Delta\psi - \Delta\mathbf{e})^T \cdot (\mathbf{Z} \cdot \Delta\psi - \Delta\mathbf{e}) \\ = (\mathbf{Z} \cdot \Delta\psi_0 - \Delta\mathbf{e})^T \cdot (\mathbf{Z} \cdot \Delta\psi_0 - \Delta\mathbf{e}),\end{aligned} \tag{11}$$

$$\Delta\psi^T \cdot \Delta\psi \geq \Delta\psi_0^T \cdot \Delta\psi_0. \tag{12}$$

That particular value of $\Delta\Psi_0$ can be obtained from the following equation:

$$\Delta\psi_0 = \mathbf{Z}^- \cdot \Delta\mathbf{e}. \tag{13}$$

This particular solution minimizes the error with minimal modification of the neural network parameters, such as the connecting weights, threshold values, and the slope of the sigmoid function. A detailed calculation of the above formulations for successive modifications ($m = 1$) reduces to the following equation:

$$\begin{Bmatrix} \Delta\psi_1 \\ \vdots \\ \Delta\psi_n \end{Bmatrix} = -\frac{\alpha E_1}{\sum_{i=1}^{n} \left(\frac{\partial E_1}{\partial \psi_i}\right)^2} \begin{Bmatrix} \frac{\partial E_1}{\partial \psi_1} \\ \vdots \\ \frac{\partial E_1}{\partial \psi_n} \end{Bmatrix}. \tag{14}$$

It should be noted that the coefficient of the right-hand side vector of the above equation corresponds to the learning coefficient η of the steepest descent method; it is directly proportional to the value of the error surface and inversely proportional to the sum of its sensitivities with respect to the design variables, i.e., the neural network parameters. Further, this equivalent value of the learning coefficient changes as learning progresses due to the change in the value of the error and its corresponding sensitivities. This equivalent learning coefficient includes an arbitrary factor α whose effect on learning efficiency is discussed below with the help of numerical

simulations. From the computational point of view in comparison with the steepest descent method, one needs to calculate only those extra operations of addition and division related to the denominator of the above equation, Eq. (14), with the gradients of the error surface calculated with respect to the neural network parameters. Hence, this feature makes it computationally feasible for on-line system applications, such as time series structural identification and control problems.

NUMERICAL SIMULATION

The present algorithm is tested with an XOR problem and a time series response identification problem of a single degree of freedom mass–spring model.

Example 1: XOR Problem

The input–output variables of an XOR problem are given in Table 1. The neural network architecture of the problem is formed with two units in the input layer, two units in the hidden layer, and one unit in the output layer. The learning error of this logical operation is plotted in Fig. 2 with respect to the number of training iterations. Fluctuation of the learning speed for the steepest descent method as shown in Fig. 2(a) largely depends on the learning coefficient parameter η that is varied from a value of 0.01 to 0.9. For example, with $\eta = 0.01$ and $\eta = 0.1$, the learning speed is extremely slow. To achieve the same level of adaptation for $\eta = 0.5$ and $\eta = 0.9$, the learning speed differs by a factor of two. This indicates that selection of a proper value of the learning coefficient η is extremely important in tuning a neural network.

The learning performance of the proposed pseudoinverse algorithm is shown in Fig. 2(b).

Table 1. Input and Output Variables in XOR Problem

Input		Output
0	1	1
1	0	1
0	0	0
1	1	0

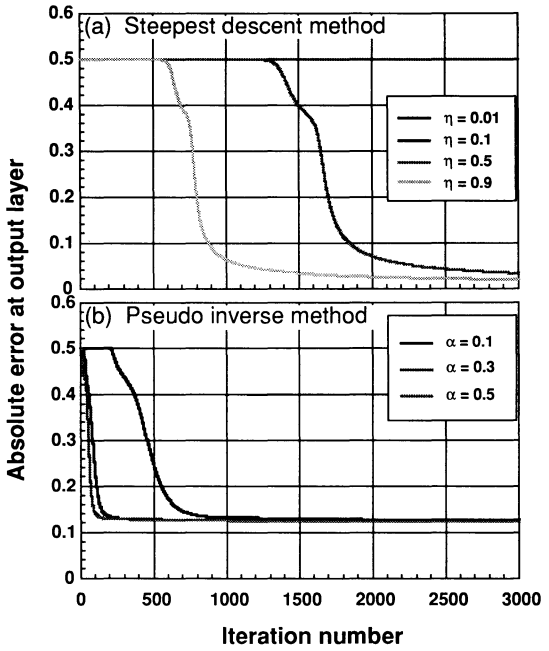

FIGURE 2 History of absolute error of XOR problem.

Compared with the steepest descent method, the number of learning iterations before local convergence occurs is clearly smaller. Although the final error converges to a higher local optimum level compared with that of the steepest descent method, reducing the error to the same level reveals that the variation of learning in performance with respect to α is small and that the learning speed is almost the same for $\alpha = 0.3$ and $\alpha = 0.5$.

Figure 3 shows the change in the equivalent variable learning coefficient parameter in graphical form. It indicates the variation in the equivalent learning coefficient, the squared sum of the gradient of the error surface, and the total error. With the progress of learning, the total error becomes smaller together with the decrease in its gradient calculated with respect to the neural network parameters Ψ. The equivalent learning coefficient increases gradually. In the present example, the final result of the learning performance with the pseudoinverse method is not as good as that of the steepest descent method. This is due to the very large value of the equivalent learning coefficient for the very small value of the gradients, and finally it leads to a local convergence at a higher error level of 0.13. The corresponding

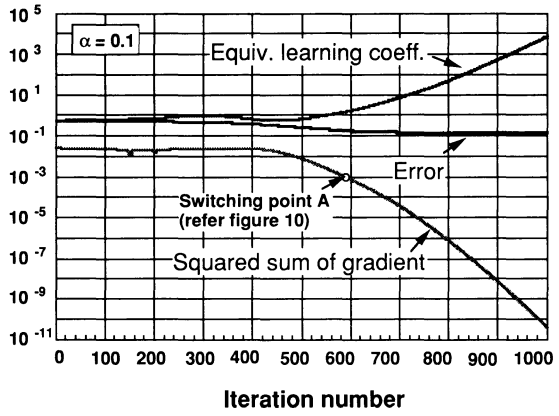

FIGURE 3 History of equivalent learning coefficient, error, and squared sum of gradients.

final error level for the steepest descent method is about 0.02.

Example 2: Time Series Response Identification of a Single Degree of Freedom Mass–Spring Model

A single degree of freedom mass–spring model is shown in Fig. 4. The dynamic equation of the system is written as

$$m\ddot{x} + c\dot{x} + kx = f(t).$$

A single sinusoidal external force is considered with the following system parameters: $m = 1.0$ (kg), $c = 37.95$ (Ns/m), $k = 40000.0$ (N/m), $a = 1540.52$, $\omega = 6.283$ (rad/s). We investigated the identification of this dynamic system. Similar to example 1, a 3-layer neural network with four input units, four hidden units, and three output units is shown in Fig. 5. The outputs are the incremental values of the system response predicted after an incremental time $\Delta t = 0.001$ s.

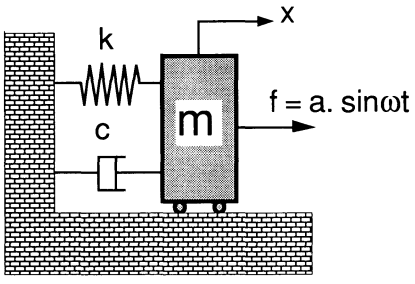

FIGURE 4 Single degree of freedom linear mass–spring model.

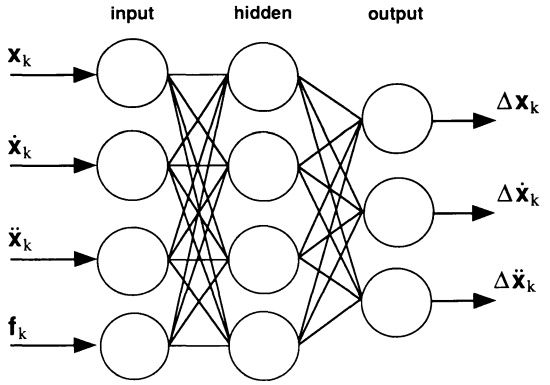

FIGURE 5 Input and output variables of neural network for mass–spring model.

Figure 6(a) and (b) compares the results obtained for the system using the proposed method and the steepest descent method. Similar to the previous XOR problem, it is clear from the results of the steepest descent method, [Fig. 6(a)] that learning performance is highly dependent on the learning coefficient η. On the other hand, with the proposed pseudoinverse learning algorithm, the value of α does not significantly influence learning performance, i.e., the variation in learning speed and accuracy is rather small. Further, to achieve the same level of learning accuracy

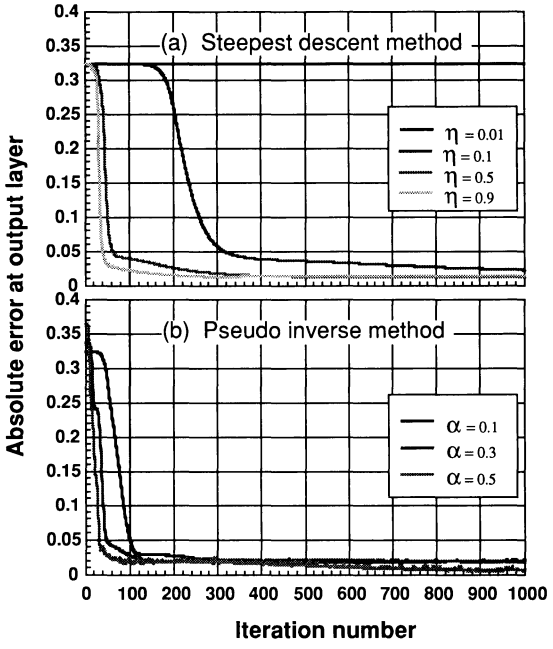

FIGURE 6 History of absolute error for dynamic system identification of linear single degree freedom mass–spring model.

and speed, very high values of $\eta = 0.5$ and $\eta = 0.9$ would have to be chosen. In practice, however, such large values are not generally used for the learning coefficient η because learning performance is very much influenced by the initial parameter values. In extreme cases such high values of η often make the convergence process very difficult, resulting in ultimate failure to train a neural network.

Figures 7 and 8 show the output response of the system corresponding to the values shown in Fig. 6 after 100 learning iterations. For small learning coefficients ($\eta = 0.01$ and 0.1), training of the neural network becomes extremely difficult. Further, compared with the results of $\eta = 0.5$ and 0.9 (Fig. 7) with the steepest descent method, the results in Fig. 8 confirm the efficiency and the accuracy of the proposed algorithm.

The equivalent learning coefficient, the squared sum of the gradients of error with respect to the neural network parameters, and the absolute value of error are plotted in Fig. 9. The overall tendency of the error value to decrease and its sensitivity to the neural network parameters are similar to the results seen in example 1. However, the total error after the convergence is very small compared to that in example 1. Hence, unlike example 1, the equivalent learning coefficient does not increase to such a high value. In the region of very low error values, the fluctuation in sensitivity produces ups and downs in the equivalent learning coefficient.

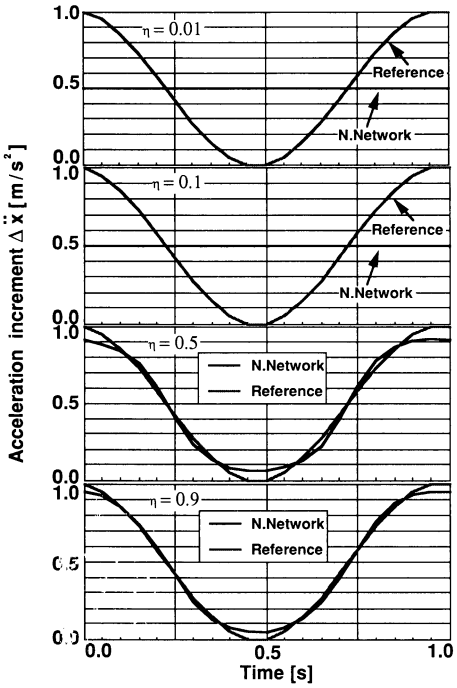

FIGURE 7 Results of dynamic system identification after 100 iterations by stiffest gradient method.

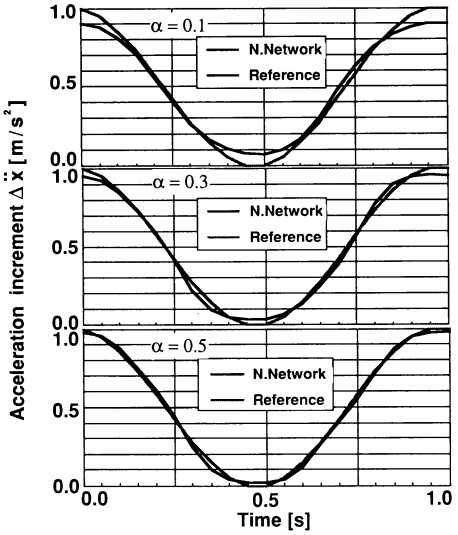

FIGURE 8 Results of dynamic system identification after 100 iterations by pseudoinverse method.

IMPROVEMENT IN VICINITY OF VERY LOW VALUE OF ERROR SURFACE

The amount of modification of the neural network parameters was calculated with the proposed method with the help of the sensitivity and absolute value of the error surface defined by Eq. (14). It was observed that in the vicinity of a very low value of error surface sensitivity, updating the

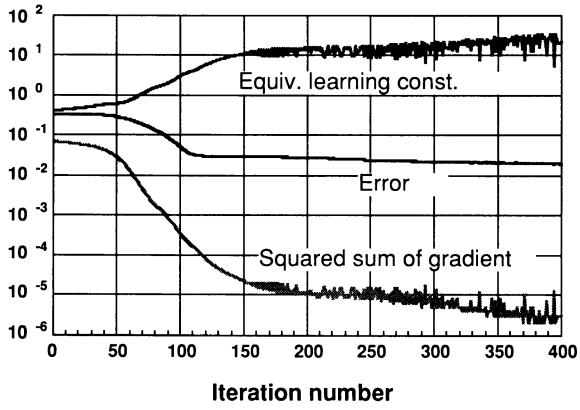

FIGURE 9 History of equivalent learning coefficient, error, and squared sum of gradients of linear single degree of freedom mass–spring model.

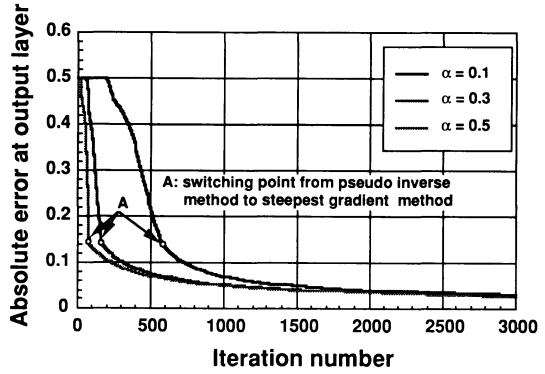

FIGURE 10 History of error for the combination of pseudoinverse and steepest gradient methods.

network parameters did not produce many fruitful effects as training progressed. This suggests that as the sensitivity value becomes considerably smaller, switching from the proposed pseudoinverse method to the existing steepest descent method in the case of the XOR problem, would be an efficient measure to prevent an unnecessary and unwanted increase in the equivalent learning coefficient. Although it was difficult to determine the exact instant of timing for the most effective switching, we selected a point when the squared sum of the gradients became 10^{-3} (point A) as shown in Fig. 3. The corresponding error history is plotted in Fig. 10. Comparing Fig. 2(b) with Fig. 10, fewer iterations are required to bring down the high level of initial absolute error of 0.5 to a moderately low value 0.1 by the pseudoinverse technique. Consequently the final error level after 3,000 training iterations was significantly reduced by switching to the steepest descent method.

CONCLUSIONS

This article proposed a new teaching algorithm for neural networks based on a pseudoinverse method. The following conclusions can be drawn from the results of numerical simulations of an XOR problem and a time-series response identification problem of a single degree of freedom mass–spring model.

In using the pseudoinverse method as a learning algorithm, we introduced a variable equivalent learning coefficient in contrast to the fixed learning coefficient of the steepest descent method, and its relation to the sensitivity and absolute value of the error surface was discussed.

Compared with an arbitrarily chosen fixed learning coefficient parameter η, the equivalent variable learning coefficient of the proposed method leads to a more robust and stable learning procedure with respect to speed and degree of accuracy.

Even when the sensitivity of the error surface is reduced to a very small value, a combination of the pseudoinverse method and the existing steepest descent method can be used effectively to achieve a higher degree accuracy.

REFERENCES

Aleksander, I., and Morton, H., 1990, *An Introduction to Neural Computing,* Chapman & Hall, London.

Anderson, J. A., and Rosenfeld, E., 1988, *Neurocomputing (Foundations of Research)*, MIT Press, Cambridge, MA.

Davis, L. (Ed.), 1991, *Handbook of Genetic Algorithms in Search Optimization and Machine Learning,* Van Nostrand Reinhold, New York.

Grossberg, S., 1976, "Adaptive Pattern Recognition and Universal Recording: Part I, Parallel Development and Coding of Neural Feature Detectors," *Biological Cybernetics,* Vol. 23, pp. 121–134.

Hebb, D. O., 1949, *The Organization of Behavior,* Wiley, New York.

Hopfield, J. J., 1982, "Neural Networks and Physical Systems with Emergent Collective Properties," *Proceedings of the National Academy of Science USA,* Vol. 79, pp. 2554–2558.

Iiguni, Y., Sakai, H., and Tokumaru, H., 1992, "A Real-Time Larning Algorithm for a Multi-Layered Neural Network Based on the Extended Kalman Filter," *IEEE Transactions on Signal Processing,* Vol. 40, No. 4, pp. 959–966.

Khanna, T., 1990, *Foundations of Neural Networks,* Addison–Wesley, Reading, MA.

Kohonen, T., 1984, *Self Organization and Associative Memory,* Springer–Verlag, Heidelberg, Germany.

McCulloch, W. S., and Pitts, W. H., 1943, "A Logical Calculus of the Ideas Imminent in Nervous Activity," *Bulletin of Mathematical Biophysics,* Vol. 5, pp. 115–133.

McLelland, J. L., and Rumelhart, D. E., 1986, *Parallel Distributed Processing,* Vol. 1, MIT Press, Cambridge, MA.

Mead, C., 1989, *Analog VLSI for Neural Systems,* Addison–Wesley, New York.

Minsky, M. A., and Papert, S. A., 1969b, *Perceptions—Expanded Editions,* MIT Press, Cambridge, MA.

Morishita, S., and Ura, T., 1993, "ER Fluid Applica-

tions to Vibration Control Devices and an Adaptive Neural-Net Controller," *Advances in Intelligent Material Systems and Structures,* Vol. 1, pp. 83–89.

Murase, H., 1991, "Kalman Filter Neuron Training," *Bulletin of the University of Osaka Prefecture, Ser. B,* Vol. 43, pp. 91–101.

Narendra, K. S., and Parthasarathy, K., 1991, "Gradient Methods for the Optimization of Dynamical Systems Containing Neural Networks," *IEEE Transactions on Neural Networks,* Vol. 2, No. 2, pp. 252–262.

Nelson, M. M., and Illingworth, W. T., 1991, *A Practical Guide to Neural Nets,* Addison–Wesley, Reading, MA.

Nguyen, D. H., and Widrow, B., 1990, "Neural Networks for Self Learning Control Systems," *IEEE Transactions on Neural Networks,* Vol. 1, No. 1, pp. 4–27.

Parker, D. B., 1987, "Optimal Algorithms for Adaptive Networks: Second-Order Backpropagation, Second-Order Direct Propagation, and Second-Order Hebbian Learning," *Proceedings of the 1st International Conference on Neural Networks,* Vol. 2, pp. 593–600.

Pal, C., Kayaba, N., Morishita, S., and Hagiwara, I., 1994, "Dynamic System Identification by Neural Networks, A Fast Learning Method Based on Error Backpropagation Method," *Journal of Intelligent Material Systems and Structures,* Vol. 5, No. 1, pp. 127–135.

Rosenblatt, F., 1962, *Principles of Neurodynamics: Perceptrons and the Theory of Brains Mechanisms,* Spartan Books, New York.

Shah, S., Palmeiri, F., and Datum, M., 1992, "Optimal Filtering Algorithms for Fast Learning in Feed-Forward Neural Networks," *Neural Networks,* Vol. 5, pp. 79–87.

Werbos, P., 1974, "Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences," PhD Thesis, Harvard University.

Yamada, T., and Yabuta, T., 1992, "Neural Network Controller Using Autotuning Method for Nonlinear Functions," *IEEE Transactions on Neural Networks,* Vol. 3, No. 4, pp. 595–601.