AIAA JOURNAL
Vol. 35, No. 9, September 1997

Dynamic Learning Rate Neural Network Training and Composite Structural Damage Detection

H. Luo* and S. Hanagud†
Georgia Institute of Technology, Atlanta, Georgia 30332

A new learning procedure, called dynamic learning rate steepest descent (DSD) method, is proposed for training neural networks. Based on the simple steepest descendent method, the proposed method improves the learning convergence speed significantly without increasing the computational effort, the memory cost, the algorithm simplicity, and the computational locality in the standard layered error backpropagating training algorithm. Through numerical experiments, the current method is shown to have much better learning ability than that of the standard constant learning rate steepest descent method and the accelerated steepest descendent method. The numerical experiments also indicate that the current method is robust to the selection of the initial learning rate, which is critical in the standard steepest descent method. It is also shown to be efficient. The CPU time increase, due to extra operations in the DSD algorithm, is negligible. The DSD method is then used to train a neural network for direct identification of composite structural damage through structural dynamic responses. The result indicates that neural network can be used for real-time flaw detections and advanced structural health monitoring.

Nomenclature

c_1	= learning rate increasing controller
c_2	= learning rate decreasing controller
E	= error function
o	= neuron network output
S	= activation function
T	= number of training iteration
t	= designed output of the training data
α	= acceleration coefficient
Δw	= weight modification
$\partial E/\partial w$	= error function gradient vector
ε	= error rate control threshold
η	= learning rate

Introduction

DAMAGE detection based on the structural dynamic response and system identification techniques has been a very active research subject in the past several years. Traditionally in the detection methods, an intact system model is first formulated numerically or experimentally as a reference system. The in-service response of the system is then detected and compared with the response of the intact model. The discrepancies are used to infer the health condition of the system. Inasmuch as the damage information inference is usually a complicated inverse process and the system identification technique is usually time consuming, the conventional damage detection methods cannot be applied to real-time system health monitoring.

The neural network technique offers a potential for damage detection and health monitoring of strucutures.[1–3] Instead of detecting the damage information through a reconstruction of the system on the basis of the dynamic response, the neural network technique can map the dynamic response to the damage information directly. The most important task in the neural network technique is to train the network so that it can perform the mapping correctly.

The error backpropagation algorithm,[4,5] which is based on the steepest gradient descent method, has been widely used in training layered feedforward artificial neural networks. The steepest descent method uses an error-function gradient as the searching direction, but the magnitude scaling needs linear search to get an optimal step. Because the linear search is a computationally expensive technique, a small initial scaling constant (learning rate) is usually used in backpropagation algorithms.

The steepest descent method has several favorable characteristics. It is a globally convergent algorithm, even though it can converge to a local minimum. Moreover, the steepest descent method has a computationally simple structure and can be implemented as a local algorithm, that is, the training of a node at the current layer needs only the previously trained data connected to the current node. The locality characteristic is particularly useful when the algorithm is applied to a distributed signal processing architecture. However, the steepest descent method suffers from slow convergent speeds and from the possibility of being trapped by a local minimum. Much research effort has been spent on improving the performance of the training algorithm while maintaining the desirable characteristics, such as simplicity, locality, and convergence of the steepest descent method.

One of the ideas for improving the training performance is to change the learning rate adaptively during the iteration so that the learning effect at each step is optimized. Most of the adaptive learning rate network training methods, such as Delta-Bar-Delta,[6] SuperSAB,[7] and RPROP,[8] were based on a so-called delta-bar-delta rule, which was introduced into the neural network community by Sutton[9] and was further studied by Jacobs[6] and Silva and Almeida.[10] The basic idea of this approach was to allow each weight to have its own learning rate and to be adapted according to the training information. In the application, a learning rate for each weight at each epoch (iteration) was selected as follows. If the product of the error derivatives of the two consecutive learning epochs is positive, the error surface along the axis is smooth. Then the learning rate related to this component can be increased; otherwise, it is concluded that an oscillation has occurred in this component and a decrease in learning rate is needed.

Obviously, this approach does not follow the steepest direction any more. On the other hand, besides a different learning rate for each weight component, the partial derivative information from the previous training cycle must be stored in the computer memory to get the local information for training. Thus, the memory expense in these methods can be as much as three times that of the conventional steepest descent (SSD) method.

In this paper, we present a dynamic learning rate steepest descent (DSD) method. In the proposed method, the steepest direction is kept as the weight modification direction. Global information is used to dynamically adjust the learning rate. This global information is the global error change rate in the two consecutive trainings. With this training algorithm, the extra memory cost is minimal and the computational locality is kept in the algorithm. Numerical simulations indicate that the DSD method is effective and efficient in training

Received Oct. 11, 1996; revision received May 8, 1997; accepted for publication May 20, 1997.

*Post Doctoral Fellow, School of Aerospace Engineering. Member AIAA.
†Professor, School of Aerospace Engineering. Member AIAA.

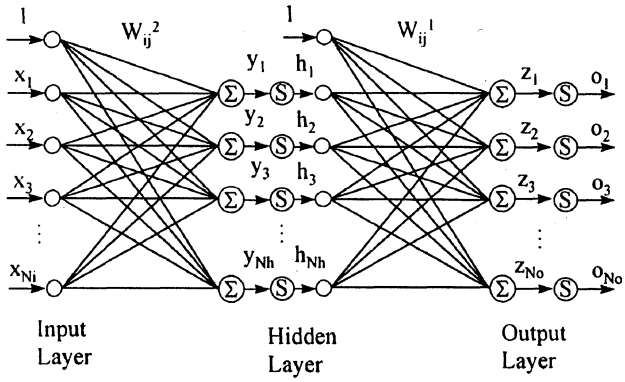

Fig. 1 Typical three-layer feedforward neural network.

and is robust with respect to the initial learning rate selection. The proposed training methods are then used to train a network for delamination and impact damage detection in composite structures.

DSD Method

An artificial multilayer neural network, inspired by the neuronal architecture of the brain, is a computational model composed of simple processors, called neurons or nodes, and connections between them. A classical three-layer feedforward neural network is shown in Fig. 1. The network consists of N_i input nodes (neurons), N_h hidden layer nodes, and N_o output nodes. The notation x_i, h_i, and o_i represents activation levels of input, hidden, and output units, respectively. Weights on connections between the hidden and output layers are denoted by w_{ij}^1, and weights on connections between the input and hidden layers are denoted by w_{ij}^2. The unit activations in the input and hidden layers provide the firing threshold. Each unit in one layer is connected in the forward direction to every unit in the next layer. Activations flow from the input layer through the hidden layer, then on to the output layer. The activation levels of the units in the output layer determine the output of the network. The activation function is denoted by S. In many applications, S has been chosen as the sigmoidal function

$$S(x) = \frac{1}{1+e^{-x}} \tag{1}$$

The purpose of the network training is to adjust network weights w [$w = (w_{ij}^1, w_{ij}^2)$] such that the following error function E is minimized:

$$E(w) = \sum_{i=1}^{N_o} (o_i - t_i)^2 \tag{2}$$

where

$$o_i = S(z_i), \qquad z_i = \sum_{k=0}^{N_h} w_{ki}^1 h_i$$

$$h_i = S(y_i), \qquad y_i = \sum_{k=0}^{N_i} w_{ki}^2 x_i$$

The basic idea behind the SSD method is that the modification of the network weight vector is proportional to the gradient of the network output error function with respect to the weight factor, i.e.,

$$\Delta w = -\eta \frac{\partial E}{\partial w} \tag{3}$$

The backpropagation training continues until the error is within a given tolerance.

The selection of the learning rate is very important in the SSD method to avoid overshooting and to increase the training speed. On the other hand, even though most neural network architectures are parallel in nature, many implementations model the parallelism on a sequential machine. Because of the massive information processing, the training efficiency of the neural network design has been one of the key issues in the neural network research.

Simply increasing the learning rate will not solve the problem. As Hinton[11] pointed out, for example, high-dimensional weight spaces typically contain ravines with steep sides and a shallow gradient along the ravine. In the shallow gradient region, a large learning rate is needed for fast convergence. The resulting small step, on the other hand, is needed near the goal. One way to speed the training is called the accelerated steepest descent (ASD) method,[11] which includes a momentum term in the weight updates,

$$\Delta w(T) = -\eta \frac{\partial E}{\partial w} + \alpha \Delta w(T-1) \tag{4}$$

This method has been proved to be more efficient than the SSD method numerically. But the memory cost is double that of the SSD method. Furthermore, the efficiency still depends on the selection of the learning rate and acceleration coefficient.

In this paper, we used a dynamic learning rate to solve this problem. The major procedures in the DSD method are as follows. Assuming after T times of training iteration, the error function is $E(T)$ and the corresponding error is $E(T+1)$ after $T+1$ times of training iteration, the error rate is defined as

$$\text{ERROR\_RATE} = \frac{E(T) - E(T+1)}{E(T)} \tag{5}$$

The learning rate for the next training iteration is determined as follows: 1) if $0 <$ ERROR_RATE $< \varepsilon$, then $\eta = \eta \times c_1$; 2) else if ERROR_RATE < 0, then $\eta = \eta \times c_2$; and 3) else, do nothing. Here ε is chosen as a small positive number, for example, 0.001. The learning rate increasing controller c_1 is a positive number larger than 1. The learning rate decreasing controller c_2 is a positive number less than 1.

Reasons behind the specified self-adaptive learning rate control are as follows.

1) To guarantee a global convergence in the SSD method, a small learning rate is chosen. In the shallow gradient region, the error rate can be very small using the same learning rate. In these regions, convergence speed can be increased by increasing the value of the learning rate.

2) In the region near the goal state, a large learning rate, which is good for fast learning in the small gradient area, results in an overshooting of the goal (or minimum). The overshooting is reflected in the negative sign of the learning rate. Thus, it can be taken as a signal to decrease the learning rate and fine tune the error function to reach the minimum as closely as possible.

In the preceding analysis, we take advantage of the overshooting and use it as a turning point for fine tuning the neural network weights. To make sure that the overshooting does not miss the minimum (overshooting may be trapped by another local minimum if the overshooting step is too large), the error rate control threshold ε should not be too large. Our numerical experiments showed that numbers between 0.1 and 1% are the optimal choices for ε.

Numerical Experiments

According to the analysis as just stated, a three-layer feedforward signal processing network with error backpropagating training is programmed on a personal computer by using Fortran-77 computer language. For comparison, three training algorithms are coded: the SSD method, the ASD method, and the proposed DSD method. All of the real variables in the programs are assigned with a 64-bit double precision.

Examples

The first example selected is a simple logical XOR function. The needed input and output data for training are listed in Table 1.

The neural network designed for this problem includes an input layer with two nodes, an output layer with one node, and a hidden layer with three nodes. Initial weights are assigned randomly between −0.1 and 0.1. In the DSD method, the error rate controller is

Table 1 XOR function

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

Table 2 Three-digit parity function

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

Table 3 Relative CPU time used after 10,000 epochs

Methods	SSD	ASD	DSD
XOR	1.000	1.002	1.003
Parity	1.000	0.999	1.004

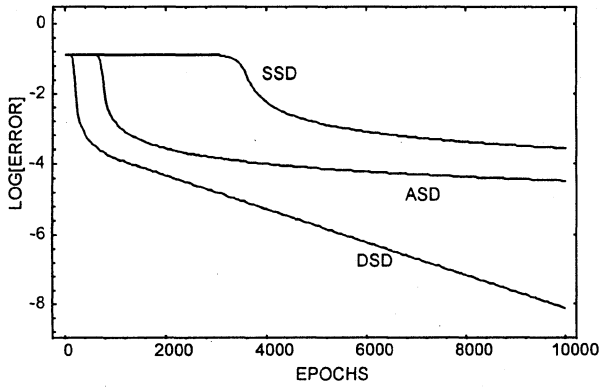

Fig. 2 Training effectiveness, XOR function: initial learning rate = 1.5.

$\varepsilon = 0.001$. The learning rate increasing controller c_1 is selected to be 1.2. The learning rate decreasing controller c_2 is selected to be 0.8. In the ASD method, the acceleration coefficient α is selected to be 0.8.

The second example used here is a three-digit parity function. The input and output data for training are listed in Table 2.

The neural network designed for this parity function problem includes an input layer with three nodes, an output layer with one node, and a hidden layer with five nodes. All of the parameters are the same as in the XOR function.

Numerical Results and Discussion

Training Efficiency

Compared to the SSD method, there is no extra memory cost in the DSD programming because no extra variable needs to be in the memory during the training. The CPU time cost is also negligible. In Table 3, we have shown the relative CPU time used by different methods. The running environment is the Pentium personal computer with a 60-MHz clock under the disk operating system control. From Table 3, we can see that, after 10,000 epochs of iteration, the time increase in the DSD method is less than 0.3% for the XOR function and less than 0.5% for the three-digit parity function.

Training Effectiveness

In Fig. 2, we have shown the results of the XOR function network training with an initial learning rate of 1.5. Results from three different algorithms are shown in this figure. Similarly in Fig. 3, we have displayed the results from the parity function. From Figs. 2 and 3, the superiority of the DSD algorithm in network training over SSD and ASD is evident. Other selections of different initial learning rates have shown similar training effectiveness, which are not shown here.

Training Robustness

Numerical experiments also indicate that the DSD algorithm displays robustness to the selection of the initial learning rate. The

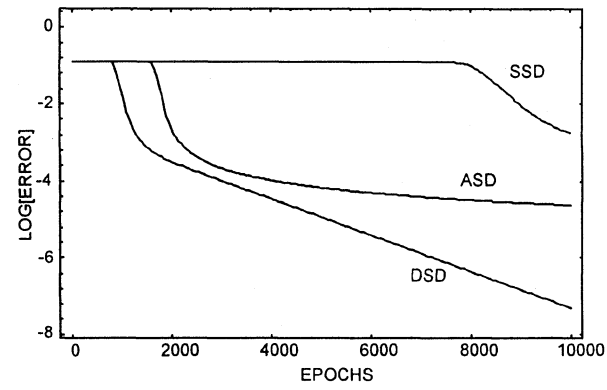

Fig. 3 Training effectiveness, three-digit parity function: initial learning rate = 1.5.

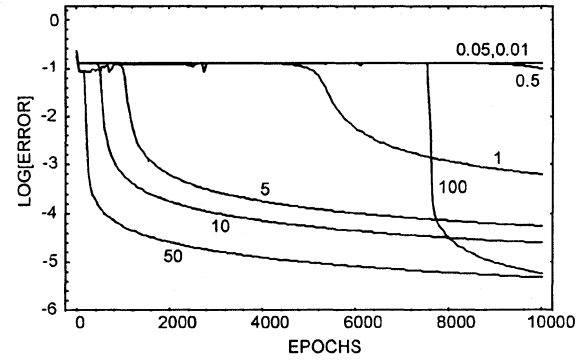

Fig. 4 Training robustness, SSD method, XOR function: different learning rate.

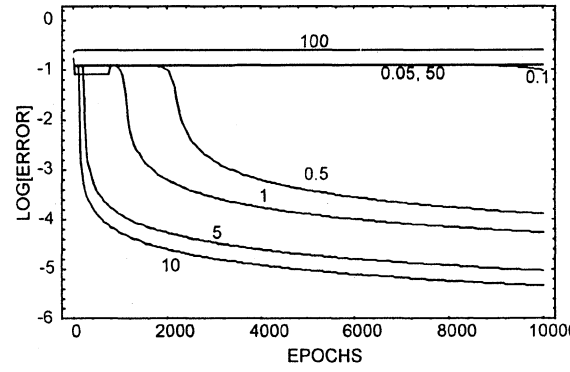

Fig. 5 Training robustness, ASD method, XOR function: different learning rate.

convergence of SSD and ASD algorithms depends on the selection of the learning rate value, especially in the SSD algorithm.

The convergence of the SSD algorithm at different learning rate values is shown in Fig. 4 for the XOR function. As can be seen, the convergence speed of the SSD method is heavily dependent on the value of the learning rate. When the learning rate is small, for example, $\eta = 0.05$ or 0.01, the training does not show any convergence tendency even after 10,000 epochs of iteration. When the learning rate is large, for example, 100, the training procedures experience overshooting. Fortunately, in this example, the overshooting is trapped by the global minimum around 8000 epochs; however, in practice, this kind of overshooting may never be trapped. The selection of the learning rate is critical in the SSD algorithm.

The convergence of the ASD method is also dependent on the selection of the learning rate (Fig. 5). The original purpose of the ASD method is to improve the learning convergence. By adding a momentum term in the weight modifications, the training oscillations caused by a large learning rate in the SSD method can be alleviated in some cases, thus increasing the training efficiency. By comparing Figs. 4 and 5, we can easily see that for the XOR function network

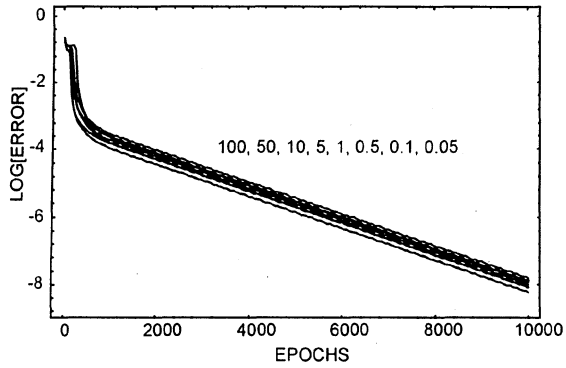

Fig. 6 Training robustness, DSD method, XOR function: different learning rate.

design, the ASD method improved the training convergence for the cases in which the learning rates are relatively small, for example, $\eta = 0.5, 1.0, 5.0$, and 10.0. But for large learning rates, the training could converge to a local minimum. For example, when a learning rate of 50 is chosen, the learning converged to (0, 1, 1, 1). If a learning rate of 100 is chosen, the learning converged to (0, 0, 0, 0). Both (0, 1, 1, 1) and (0, 0, 0, 0) are local minima in the hypersurface of the network weight space. The global minimum in the space should be (0, 1, 1, 0) in the designed XOR function network. It is clearly seen from this example that the improvement in the ASD method is limited.

The robustness of the DSD algorithm at the learning rate selections can be seen in Fig. 6. Under the same learning rate range as used in the SSD and ASD algorithms, the DSD algorithm training results in a consistently high convergence speed. The DSD method is robust to the selection of the initial learning rate. Similar superior characteristics of the DSD algorithm are indicated by the neural network training of the parity function.

Composite Structural Damage Detection

Delamination and impact damage are two major types of damage in composite structures. Modal analysis of a damaged composite structure reveals features such as delamination modes and stiffness loss that are unique to composite structures.[12–14] These features are good candidates for detecting damages, but the modal analysis procedures are usually time consuming and thus suitable only for off-line damage detection. A real-time technique is required in health monitoring of such structures. We use a modified neural network technique and combine it with structural dynamic response for real-time damage detection.

Traditionally, the damage information in a structure is deduced through its mechanical model. When the damage occurs in the structure, subtle changes take place in the structural dynamic model. The differences between the intact model and the damaged model are used to infer the damage information, usually through an identification procedure or by solving the inverse problem. As a result, many research efforts are directed to the formulation of the structural model and the interpretation of the mathematical results of the inverse problem solutions. Typical detection procedures include optimal matrix modifications,[15–18] sensitivity-based updates,[19–22] and eigenstructure assignment techniques.[23–26]

We present a delamination and stiffness loss detection neural network design that bypasses the system modeling. Instead, we mapped the structural dynamic response directly to the damage pattern. The direct mapping eliminates the model construction, inverse manipulation, and interpretation of results, thus making the damage detection process less time consuming than conventional methods.

Delamination Detection Using Experimental Data

To demonstrate the neural network damage detection concept, a delamination detection experiment for composite beams was designed. Specimens used to train the networks were fabricated by 0/90 woven glass fiber cloth with epoxy as the matrix. Delaminations were simulated by placing very thin Teflon® sheets of film between layers during the layup. All of the specimens consist of 12 layers of fiberglass cloth and were prepared in strips of 30 × 2.5 cm.

Table 4 Delamination information of the training specimens

Sample no.	Length, mm	Thickness location	Thickness Code1	Thickness Code2
1	0	N/A	0	0
2	3	6/6	0	1
3	10	6/6	0	1
4	18	6/6	0	1
5	3	4/8	1	0
6	10	4/8	1	0
7	18	4/8	1	0
8	3	2/10	1	1
9	10	2/10	1	1
10	18	2/10	1	1

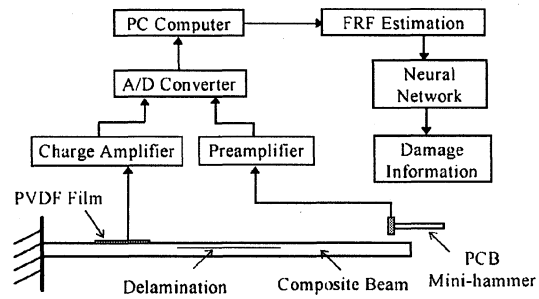

Fig. 7 Experimental configuration setup.

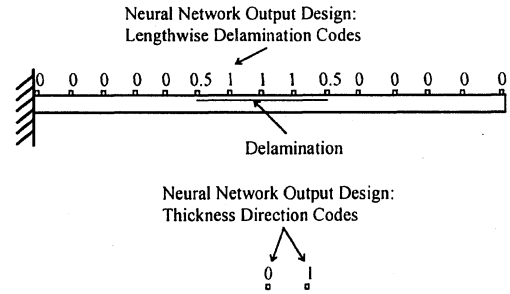

Fig. 8 Delamination code design.

Delaminations were centered at the specimen length. The delamination information of the training set is listed in Table 4.

The specimens were then mounted to simulate cantilever boundary conditions by planting 2.5 cm of the length of the specimens into a vice. In our work, we used the frequency response function (FRF) as network input because it contains the information on the natural frequencies and mode shape components. A polyvinylidene fluoride (PVDF) film, which was fixed near the root, was used as a sensor to pick up the dynamic response. A PCB Piezotronics, Inc., minihammer impulse at the tip of the cantilever beam was used as the excitation. The experimental setup and instrumentation are as shown in Fig. 7.

The network outputs were designed as the delamination information of the structure. At the current stage, we designed the neural network as a pattern classifier. The network output design includes lengthwise codes and thickness direction codes. In the lengthwise code design, we defined the outputs at measuring locations without delamination to be zeros. The outputs at measuring locations with delamination were defined to be ones. The outputs at the measuring locations close to the delamination boundary were defined to be 0.5. The thickness direction delamination codes are designed to indicate thickness direction of the delamination. The thickness direction codes are listed in Table 4 for different delamination cases. To indicate the lengthwise delamination, 16 nodes were designed, and two extra nodes were used as thickness direction indicators (see Fig. 8). As neural network input 128 input nodes were designed. The input data were taken from the FRF of a specimen. The FRF was evenly discretized in the interested frequency range (800 Hz in the delamination detection case). A hidden layer with 30 nodes was used in the network. Thus, the designed network was a three-layer

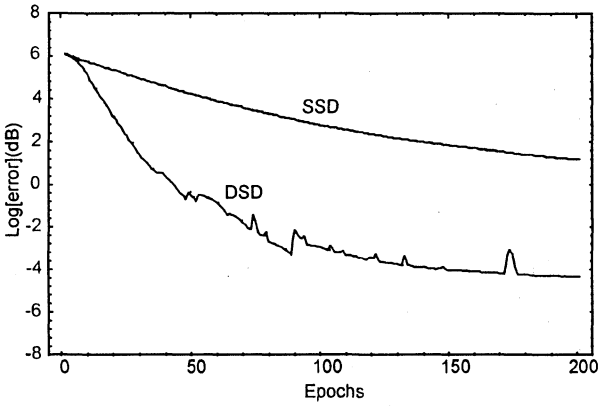

Fig. 9 Training performance.

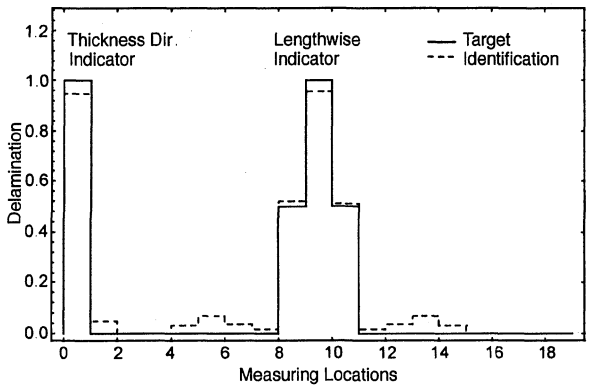

Fig. 10 Selected detection result, 3-cm 4/8 delamination case.

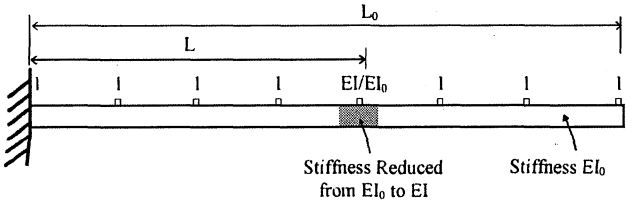

Fig. 11 Stiffness loss damage detection network output design.

feedforward network with 128 input nodes, 30 hidden nodes, and 18 output nodes.

In Fig. 9, we show the training iteration results of the SSD method and the DSD method. The superiority of the training performance of the DSD method over that of the SSD method is seen in this practical problem. The oscillating phenomenon in the DSD training indicates the training overshooting and correction procedures. After the network has been trained, another set of structural dynamic response from delamination specimens was fed into the network; the selected detection result is shown in Fig. 10. Because the network has been trained, the optimal weights are stored in the network memory. A feedforward pass of a new data set through the trained network is nearly in real time.

Stiffness Loss Detection by the Use of Numerical Simulation Data

In the case of stiffness loss damage detection, the FRFs are used as the damage information carriers. The network outputs are designed as the relative stiffness ratio, that is, the ratio of the stiffness of the damaged structure with respect to the stiffness at the intact state at the network output locations (Fig. 11). For the purpose of design verification, only one stiffness loss location was used in the example. Numerical simulations were used to train the neural network. The intact beam simulated in this example has the same equivalent stiffness and geometry as that of the intact specimen used in the delamination damage detection. Specimen boundary conditions were simulated as cantilever. As the training data set, 16 different stiffness loss cases were calculated. The descriptions of the stiffness loss cases are listed in Table 5.

Table 5 Stiffness loss damage cases

Sample no.	Stiffness loss location, L/L_0	Stiffness loss ratio, EI/EI_0
1	Intact	1
2	0.027	0.729
3	0.245	0.729
4	0.464	0.729
5	0.682	0.729
6	0.900	0.729
7	0.027	0.343
8	0.245	0.343
9	0.464	0.343
10	0.682	0.343
11	0.900	0.343
12	0.027	0.125
13	0.245	0.125
14	0.464	0.125
15	0.682	0.125
16	0.900	0.125

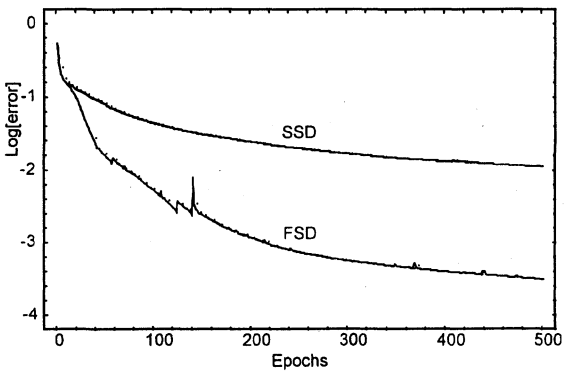

Fig. 12 Training performance of the stiffness loss damage detection network.

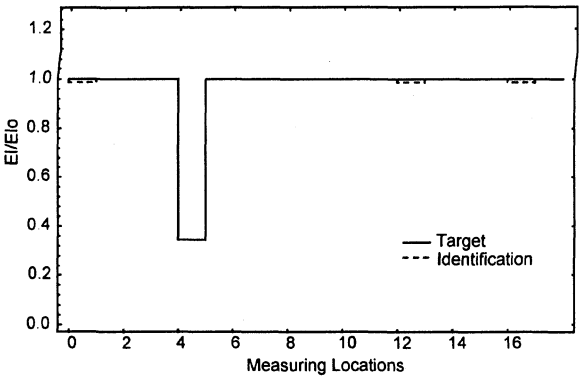

Fig. 13 Network stiffness loss detection, sample 8.

To numerically simulate the dynamic response, Galerkin's method was used to solve the damaged structure approximate natural frequencies and mode shapes. The natural frequencies and mode shapes were then used to calculate the dynamic response by the modal expansion method. The FRFs for network training were obtained by the fast Fourier transformation algorithm on the input signal and the synthesized response signal. In this detection example, an impulse excitation was applied at the tip of the beam. The response signal was the displacement response at the tip of the beam.

The network input includes 128 points from the FRF data. The network output includes five points indicating the relative stiffness ratios at these locations. A hidden layer with 30 nodes was used in the network. Thus, the designed network was a three-layer feedforward network with 128 input nodes, 30 hidden nodes, and 5 output nodes. In Fig. 12, we show the training iteration results of the SSD method and the FSD method for this case.

After the network has been trained, it is able to detect the damage locations correctly within the trained domain. In Fig. 13, the

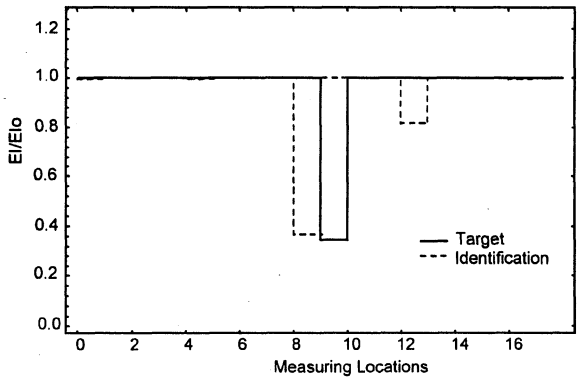

Fig. 14 Network expansion ability, case 2.

detection result of sample 8 is shown. The solid lines stand for the actual stiffness loss damage, whereas the dashed lines represent the network identification. The trained network is able to detect the damages similar to those which have been seen during the training. It can also expand the domain of identification in some cases. For example (as shown in Fig. 14), the actual stiffness loss damage is located between two geometric output nodes of the trained network. Though the damage case has not been seen by the network during the training, the network can successfully reflect the damage information at neighboring nodes.

Conclusions

An improved neural network training method is presented. Numerical experiments on standard functions indicate that the DSD method has an improved performance in comparison to other conventional methods. Based on the DSD training method, neural networks are designed for detection of delaminations and impact damage in composite structures.

Real-time health monitoring techniques that are based on the modal analysis or finite element analysis are not always practical at present due to the time consuming signal processing. Real-time signal processing techniques are required to achieve health monitoring in composite structures. Our experiments with the structural dynamic response and a DSD-based neural network signal processing has clearly shown the feasibility of real-time health monitoring of composite structures. The dynamic learning rate training algorithm provides the potential tool to train large networks, which are usually required in the cases of composite structure health monitoring, in a reasonable time.

The improved performance of the DSD training method was the result of the selection of the training control parameters c_1, c_2, and ε. Further improvement of the training performance is possible through an automatic optimization of those control parameters.

References

[1]Manning, R. A., "Structural Damage Detection Using Active Members and Neural Networks," *AIAA Journal*, Vol. 32, No. 6, 1994, pp. 1331–1333.

[2]Yen, G. G., and Kwak, M. K., "Neural Network Approach for the Damage Detection of Structures," *Proceedings of the AIAA/ASME/ASCE/AHS/ASC, 34th Structures, Structural Dynamics, and Material Conference*, AIAA, Washington, DC, 1993, pp. 1549–1555.

[3]Tsou, P., and Shen M.-H. H., "Structural Damage Detection and Identification Using Neural Network," *Proceedings of the AIAA/ASME/ASCE/AHS/ASC 34th Structures, Structural Dynamics, and Material Conference*, AIAA, Washington, DC, 1993, pp. 3551–3560.

[4]Rumelhart, D. E., Hinton, G. E., and Williams, R. J., "Learning Internal Representations by Error Propagation," *Parallel Distributed Processing*, Vol. 1, edited by D. E. Rumelhart and J. L. McClelland, MIT Press, Cambridge, MA, 1986, pp. 318–362.

[5]Rumelhart, D. E., Hinton, G. E., and Williams, R. J., "Learning Representations by Back-Propagating Errors," *Nature*, Vol. 323, No. 9, 1986, pp. 533–536.

[6]Jacobs, R. A., "Increased Rates of Convergence Through Learning Rate Adaptation," *Neural Networks*, Vol. 1, No. 4, 1988, pp. 295–307.

[7]Tollenaere, T., "Supersab: Fast Adaptive Backpropagation Algorithm with Good Scaling Properties," *Neural Networks*, Vol. 3, No. 5, 1990, pp. 561–573.

[8]Riedmiller, M., and Braun, H., "A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm," *Proceedings of the 1993 IEEE International Conference on Neural Networks*, Vol. 1, Inst. of Electrical and Electronics Engineers, New York, pp. 586–591.

[9]Sutton, R. S., "Two Problems with Backpropagation and Other Steepest Descent Learning Procedures for Networks," *Program of the 8th Annual Conference of the Cognitive Science Society*, Lawrence Erlbaum Associates, Hillside, NJ, 1986, pp. 823–831.

[10]Silva, F. M., and Almeida, L. B., "Acceleration Techniques for the Backpropagation Algorithm," *Lecture Notes in Computer Science*, Vol. 412, Springer–Verlag, Berlin, 1990, pp. 110–119.

[11]Hinton, G. E., "Learning Translation Invariant Recognition in a Massively Parallel Networks," *Lecture Notes in Computer Science*, Vol. 258, Springer–Verlag, Berlin, 1987, pp. 1–13.

[12]Hanagud, S., and Luo, H., "Modal Analysis of a Delaminated Beam," *Proceedings of SEM Spring Conference on Experimental Mechanics*, Society for Experimental Mechanics, 1994, pp. 880–887.

[13]Luo, H., and Hanagud, S., "Delamination Detection Using Dynamic Characteristics of Composite Plates," *Proceedings of the AIAA/ASME/ASCE/AHS 36th Structures, Structural Dynamics, and Materials Conference*, AIAA, Washington, DC, 1995, pp. 129–139.

[14]Luo, H., and Hanagud, S., "An Integral Equation for Changes in the Structural Dynamics Characteristics of Damaged Structures," *International Journal of Solids and Structures* (to be published).

[15]Baruch, M., "Optimal Correction of Mass and Stiffness Matrices Using Measured Modes," *AIAA Journal*, Vol. 20, No. 11, 1982, pp. 1623–1626.

[16]Kabe, A. M., "Stiffness Matrix Adjustment Using Mode Data," *AIAA Journal*, Vol. 23, No. 9, 1985, pp. 1431–1436.

[17]Kammer, D. C., "Optimum Approximation for Residual Stiffness in Linear System Identification," *AIAA Journal*, Vol. 26, No. 1, 1988, pp. 104–112.

[18]Smith, S. W., and Beattie, C. A., "Secant-Method Matrix Adjustment for Structural Modes," *AIAA Journal*, Vol. 29, No. 1, 1991, pp. 119–126.

[19]Collins, J. D., Hart, G. C., Hasselman, T. K., and Kennedy, B., "Statistical Identification of Structures," *AIAA Journal*, Vol. 12, No. 2, 1974, pp. 185–190.

[20]Hendricks, S. L., Hayes, S. M., and Junkins, J. L., "Structural Parameter Identification for Flexible Spacecraft," *Proceedings of the AIAA 22nd Aerospace Sciences Meeting* (Reno, NV), AIAA, New York, 1984 (AIAA Paper 84-0060).

[21]Flanigan, C., "Correction of Finite Element Models Using Mode Shape Design Sensitivity," *Proceedings of the Ninth International Modal Analysis Conference*, Union College, Schenectady, NY, 1991, pp. 84–88.

[22]Lin, R. M., "Analytical Model Improvement Using Modified IEM," *Proceedings of the International Conference on Structural Dynamics Modeling*, National Agency for Finite Element Methods and Standards, Glasgow, Scotland, UK, 1993, pp. 181–194.

[23]Zimmerman, D. C., and Kaouk, M., "Eigenstructure Assignment Approach for Structural Damage Detection," *AIAA Journal*, Vol. 30, No. 7, 1992, pp. 1848–1855.

[24]Lim, T. W., and Kashangaki, T. A. L., "Structural Damage Detection of Space Truss Structures Using Best Achievable Eigenvectors," *AIAA Journal*, Vol. 32, No. 5, 1994, pp. 1049–1057.

[25]Kaouk, M., and Zimmerman, D. C., "Structural Damage Assessment Using a Generalized Minimum Rank Perturbation Theory," *AIAA Journal*, Vol. 32, No. 4, 1994, pp. 836–842.

[26]Lim, T. W., "Structural Damage Detection Using Constrained Eigenstructure Assignment," *Journal of Guidance, Control, and Dynamics*, Vol. 18, No. 3, 1995, pp. 411–418.

A. Berman
Associate Editor