EARTHQUAKE ENGINEERING AND STRUCTURAL DYNAMICS
Earthquake Engng Struct. Dyn. 2002; **31**:217–234 (DOI: 10.1002/eqe.106)

Structural damage detection using the optimal weights of the approximating artificial neural networks

Shih-Lin Hung[*,†] and C. Y. Kao

Department of Civil Engineering, National Chiao Tung University, 1001 Ta Hsueh Road, Hsinchu 300, Taiwan

SUMMARY

This work presents a novel neural network-based approach to detect structural damage. The proposed approach comprises two steps. The first step, system identification, involves using neural system identification networks (NSINs) to identify the undamaged and damaged states of a structural system. The partial derivatives of the outputs with respect to the inputs of the NSIN, which identifies the system in a certain undamaged or damaged state, have a negligible variation with different system errors. This loosely defined unique property enables these partial derivatives to quantitatively indicate system damage from the model parameters. The second step, structural damage detection, involves using the neural damage detection network (NDDN) to detect the location and extent of the structural damage. The input to the NDDN is taken as the aforementioned partial derivatives of NSIN, and the output of the NDDN identifies the damage level for each member in the structure. Moreover, SDOF and MDOF examples are presented to demonstrate the feasibility of using the proposed method for damage detection of linear structures. Copyright © 2001 John Wiley & Sons, Ltd.

KEY WORDS: artificial neural network (ANN); partial derivative form of ANN; system identification; structural damage detection

INTRODUCTION

Civil engineering structures are prone to damage and deterioration during their service life. Damage assessment attempts to determine whether structural damage has occurred and, if so, to determine the location and extent of the damage. However, detecting structural damage

[*] Correspondence to: Shih-Lin Hung, Department of Civil Engineering, National Chiao Tung University, 1001 Ta Hsueh Road, Hsinchu 300, Taiwan.
[†] E-mail: slhung@cc.nctu.edu.tw

Contract/grant sponsor: National Science Council of the Republic of China; contract/grant number: NSC 89-2211-E-009-014.

Received 21 November 2000
Revised 22 March 2001
Accepted 16 April 2001

and identifying damaged elements in a large complex structure are challenging tasks since the *in situ* measured data of large civil engineering structures such as bridges and buildings are supposed to be imprecise (because of noise corruption) and often incomplete (for economy consideration).

Conventional damage assessment methods [1, 2] are inherently direct process methods, proceeding linearly from causes to effects. These methods initially involve constructing a mathematical model for the structure and, then, using that model to elucidate the structural behaviour and to establish correlations between specific member damage conditions and changes in the structural response. Despite their many attractive features, conventional damage assessment methods may encounter difficulties, such as measurement noise and modelling errors, when detecting systems that are difficult to model. In addition, the damage assessment algorithm conventionally adopted is generally complex and is inappropriate where measured data are imprecise.

The artificial neural network (ANN) model is robust and fault-tolerant [3–5]. ANNs can also effectively deal with qualitative, uncertain, and incomplete information, thereby making it highly promising for detecting structural damage. The feasibility of applying these networks to detect structural damage has received considerable attention. Wu *et al.* [6] used spectral acceleration, generated from a numerical model of a simple frame, as an input to a neural network. According to their results, ANNs can learn about the behaviour of undamaged and damaged structures to identify the damaged member and the extent of the damage from patterns in the frequency response of the structure. While using ANNs trained with training samples generated from a finite element model, Elkordy *et al.* [7] diagnosed damage states obtained experimentally from a series of shaking-table tests of a five-storey steel frame. Despite their promising results, they pointed towards the need for further study of the relation between the number of damage patterns required for training the network to perform satisfactorily and the degree of simplification of the model. Szewezyk and Hajela [8] used a modified counterpropagation neural network to develop the inverse mapping between a vector of the stiffness of individual structural elements and the vector of the global static displacements under a testing load. Their results indicated that the network functions as an associative memory device capable of achieving satisfactory diagnostics even with noisy or incomplete measurements. Pandey and Barai [9] examined the feasibility of applying multilayer perceptron to detect the structural damage of a steel bridge from static vertical displacements of nodes. Masri *et al.* [10, 11] extended the efforts of previous studies by addressing a class of problems where the failure states are unknown. They presented a neural network-based approach for detecting changes in the characteristics of systems where the structure is unknown. Their approach relies on using vibration measurements from a ‘healthy’ system to train an ANN for identification purposes. Subsequently, comparable vibration measurements from the same structure under different episodes of response are input to the trained network to monitor the health of the structure. By utilizing the predictions of the ANN before and after potential structural changes (damage) in the physical system have occurred, quantifiable measures of the degree of fidelity of the predicted response measurements can be used to assess the extent of changes.

Neural networks have been extensively used to identify dynamic systems. The weights of the approximating neural network store the knowledge of the structural properties. Thus, understanding how the system’s physical properties and the weights of the corresponding ANN are related is a worthwhile task. This work presents a novel neural network-based approach to detect structural damage. The first step, system identification, involves using neural system identification networks (NSINs) to identify the undamaged and damaged states of a structural system. The partial derivatives of the outputs with respect to the inputs of the NSIN (functions of weights and activation function), which identifies the system in a certain undamaged or damaged state, have a negligible variation with different system errors. Comparing the partial derivatives of the NSIN that identify a certain damaged state with those that identify the undamaged state allows us to detect changes to the physical system from its undamaged state. The second step, structural damage detection, involves using neural damage detection network (NDDN) to detect the location and extent of the structural damage. The input to the NDDN is taken as the aforementioned partial derivatives of the NSIN, and the output of the NDDN identifies the damage level (i.e. reduced percentage of respective damping and (or) stiffness) for each member in the structure. Moreover, SDOF and MDOF examples are presented to demonstrate the feasibility of using the proposed method for damage detection of linear structures.

ARTIFICIAL NEURAL NETWORKS

Artificial neural networks (ANNs) form a class of systems that are derived from biological neural networks. The topology of an ANN model consists of a number of simple processing elements, called nodes, that are interconnected to each other. Interconnection weights that represent the information stored in the system are used to quantify the strength of the interconnections; these weights hold the key to the functioning of an ANN. ANNs have been used in a broad range of applications, including classification, pattern recognition, function approximation, optimization, prediction, and automatic control. Among the many different types of ANN, the feedforward, multilayered, supervised neural network with the error backpropagation algorithm, the so-called backpropagation (BP) network [5], is by far the most commonly applied neural network learning model owing to its simplicity.

Before an ANN can be used in the application, it needs to learn or be trained from an existing training set consisting of pairs of input–output elements. The training of a supervised neural network using BP learning algorithm usually involves two stages. The first stage is the data feed forward. The output of each node is defined as follows:

$$\mathrm{net}_j = \sum_{i=1}^{n} W_{ij} O_i + \theta_j \tag{1}$$

$$O_j = f(\mathrm{net}_j) \tag{2}$$

where W_{ij} is the weight associated with the ith node in the preceding layer to the jth node in the current layer, O_i is the output of ith node in the preceding layer, θ_j is the threshold value of node j in the current layer, O_j is the output of node j in the current layer and function f is the activation function, which has to be differentiable. Herein, the hyperbolic tangent function and sigmoid function are used as the activation function in the NSIN and the NDDN, respectively. The hyperbolic tangent function and the sigmoid function are defined, respectively, as

$$f(x) = \frac{\mathrm{e}^{x} - \mathrm{e}^{-x}}{\mathrm{e}^{x} + \mathrm{e}^{-x}} \tag{3}$$

and

$$f(x) = \frac{1}{1 + \mathrm{e}^{-x}} \tag{4}$$

The second stage is error back-propagation and adjustment of the weights through the network. In the training process, the system error function is used to monitor the learning performance of the network. This system error function is defined as follows:

$$E = \frac{1}{2P} \sum_{p=1}^{P} \sum_{k=1}^{K} (d_{pk} - o_{pk})^2 \tag{5}$$

where P is the number of instances in the training set and d_{pk} as well as o_{pk} are the desired and calculated output of the kth output node for the pth instance, respectively. The standard BP algorithm uses a gradient descent approach with a constant step length (learning ratio) to train the network.

$$W_{ij}^{(k+1)} = W_{ij}^{(k)} + \Delta W_{ij} \tag{6}$$

$$\Delta W_{ij} = -\eta \frac{\partial E}{\partial W_{ij}} \tag{7}$$

where η is the learning ratio which is a constant in the range of $[0, 1]$. The superscript index (k) denotes the kth learning iteration. BP supervised neural network learning models, however, always take a long time to learn. Moreover, the convergence of a BP neural network is highly dependent upon the use of a learning rate (η). Thus, several different approaches developed to enhance the learning performance of the BP learning algorithm have been applied [3].

Hung and Lin [12] developed a more effective adaptive limited memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) learning algorithm based on the approach of a L-BFGS quasi-Newton second-order method [13, 14] with an inexact line search algorithm. This algorithm achieved better convergence rate than the BP learning algorithm by using second-order derivatives of the system error function with respect to the network weights. In the conventional BFGS method, the approximation $\mathbf{H}_{k+1}$ to the inverse Hessian matrix of function $E(\mathbf{W})$ is updated by

$$\begin{aligned} \mathbf{H}_{k+1} &= (\mathbf{I} - \rho_k \mathbf{s}_k \mathbf{y}_k^{\mathrm{T}}) \mathbf{H}_k (\mathbf{I} - \rho_k \mathbf{y}_k \mathbf{s}_k^{\mathrm{T}}) + \rho_k \mathbf{s}_k \mathbf{s}_k^{\mathrm{T}} \\ &\equiv \mathbf{V}_k^{\mathrm{T}} \mathbf{H}_k \mathbf{V}_k + \rho_k \mathbf{s}_k \mathbf{s}_k^{\mathrm{T}} \end{aligned} \tag{8}$$

where

$$\rho_k = 1/\mathbf{y}_k^{\mathrm{T}} \mathbf{s}_k \tag{9}$$

$$\mathbf{V}_k = \mathbf{I} - \rho_k \mathbf{y}_k \mathbf{s}_k^{\mathrm{T}} \tag{10}$$

$$\mathbf{s}_k = \mathbf{W}_{k+1} - \mathbf{W}_k \tag{11}$$

$$\mathbf{y}_k = \mathbf{g}_{k+1} - \mathbf{g}_k \tag{12}$$

and

$$\mathbf{g}_k = \frac{\partial E}{\partial \mathbf{W}} \tag{13}$$

Instead of forming the matrix $\mathbf{H}_k$ in BFGS method, we save the vectors $\mathbf{s}_k$ and $\mathbf{y}_k$. These vectors first define and then implicitly and dynamically update the Hessian approximation using information from the last few iterations, say m in the work. Therefore, the final stage of the adjustment of the weights in a BP-based ANN is modified as follows:

$$\mathbf{W}^{(k+1)} = \mathbf{W}^{(k)} + \alpha_k \mathbf{d}_k \tag{14}$$

The search direction is given by

$$\mathbf{d}_k = -\mathbf{H}_k \mathbf{g}_k + \beta_k \mathbf{d}_{k-1} \tag{15}$$

where

$$\beta_k = \frac{\mathbf{y}^{\mathrm{T}}_{(k-1)} \mathbf{H}_{(k-1)} \mathbf{g}_{(k-1)}}{\mathbf{y}^{\mathrm{T}}_{(k-1)} \mathbf{d}_{(k-1)}} \tag{16}$$

The step length, α_k, is adapted during the learning process through a mathematical approach: the inexact line search algorithm. This is used in the L-BFGS learning algorithm instead of a constant learning ratio [12]. The inexact line search algorithm is based on three sequential approaches: bracketing, sectioning, and interpolation. The bracketing approach brackets the potential step length, α, between two points, through a series of function evaluations. The sectioning approach then uses the two points of the bracket as the initial points, reducing the step size piecemeal, and locating the minimum between points, e.g. α_1 and α_2, to a desired degree of accuracy. Finally, the quadratic interpolation approach uses the three points, α_1, α_2, and $(\alpha_1 + \alpha_2)/2$, to fit a parabola to determine the step length, α_k. Consequently, the step length α_k is required to satisfy the following conditions in each iteration [12]:

$$E(\mathbf{W}_k + \alpha_k \mathbf{d}_k) \leqslant E(\mathbf{W}_k) + \beta \alpha_k (\nabla E(\mathbf{W}_k)^{\mathrm{T}} \mathbf{d}_k) \quad \beta \in (0,1) \quad \text{and} \quad \alpha_k > 0 \tag{17}$$

$$\nabla E(\mathbf{W}_k + \alpha_k \mathbf{d}_k)^{\mathrm{T}} \mathbf{d}_k \geqslant \theta (\nabla E(\mathbf{W}_k)^{\mathrm{T}} \mathbf{d}_k \quad \theta \in (\beta, 1) \quad \text{and} \quad \alpha_k > 0 \tag{18}$$

$$\nabla E(\mathbf{W}_k + \alpha_k \mathbf{d}_k)^{\mathrm{T}} \mathbf{d}_{(k+1)} < 0 \tag{19}$$

Hence, the problem of trial and error selection of a learning ratio in the BP algorithm is circumvented in the adaptive L-BFGS learning algorithm.

NEURAL SYSTEM IDENTIFICATION NETWORK (NSIN)

Under some mild assumptions, a discrete-time multivariable linear or non-linear time-invariant structural system with r inputs (external excitations) and m outputs (including relative displacements, velocities and accelerations) can be represented by the following equation:

$$y(k) = g(y(k-1), \ldots, y(k-n_y), p(k-1), \ldots, p(k-n_p)) \tag{20}$$

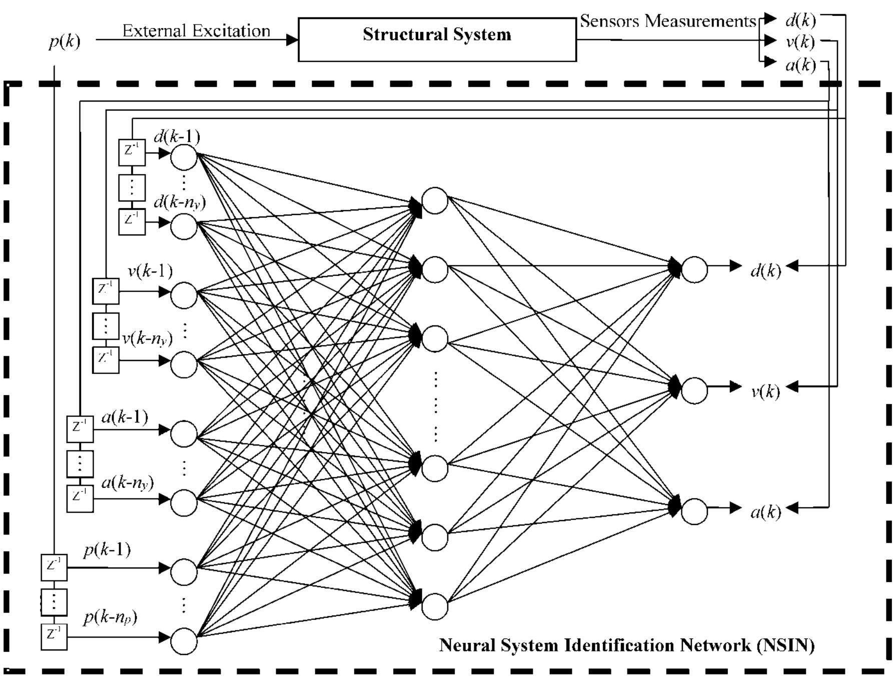


Figure 1. The architecture of the neural structural identification network.

where

$$p(k) = [\,p_1(k) \quad \cdots \quad p_r(k)]^{\mathrm{T}} \tag{21}$$

and

$$y(k) = [\,y_1(k) \quad \cdots \quad y_m(k)\,]^{\mathrm{T}} \tag{22}$$

$$= [d_1(k) \quad v_1(k) \quad a_1(k) \quad \cdots \quad d_m(k) \quad v_m(k) \quad a_m(k)]^{\mathrm{T}} \tag{23}$$

are the system input and output vectors, respectively, n_p and n_y are the maximum lags in the input and output, respectively, index k is an integer number; $k = 0, 1, 2, \ldots, N$, $d(k), v(k)$ and $a(k)$ are, respectively, the relative displacement, velocity and acceleration vectors of the system at time $t = k\Delta t$ and Δt is the time length of the sampling period and g is some vector-valued linear or non-linear function.

Function g in Equation (20) can be approximated by the ANN, called neural system identification network (NSIN), as shown in Figure 1. The inputs of the NSIN are relative displacements, velocities, and accelerations in $(k-1)$ back-to $(k-n_y)$ time steps, and external excitations in $(k-1)$ back-to $(k-n_p)$ time steps. They are denoted as $d_1(k-1), \ldots, d_m(k-1), \ldots, d_1(k-n_y), \ldots, d_m(k-n_y)$ for relative displacements, $v_1(k-1), \ldots, v_m(k-1), \ldots, v_1(k-n_y), \ldots, v_m(k-n_y)$ for relative velocities, $a_1(k-1), \ldots, a_m(k-1), \ldots, a_1(k-n_y), \ldots, a_m(k-n_y)$ for relative accelerations, and $p_1(k-1), \ldots, p_r(k-1), \ldots, p_1(k-n_p), \ldots, p_r(k-n_p)$ for external excitations. The outputs of the NSIN are relative displacements, velocities, and accelerations in the kth time step and denoted as $d_1(k),\ldots,d_m(k), v_1(k),\ldots,v_m(k)$ and $a_1(k),\ldots,a_m(k)$, respectively. Notably, the approximation by the NSIN in a discrete linear system is analogous to identifying the mass, damping and stiffness coefficients in the equation of motion. Herein, the NSIN is implemented through an adaptive L-BFGS neural network model with the hyperbolic tangent activation function.

ANN models have been extensively used to identify dynamic systems. Cybenko [15] and Funahashi [16] rigorously demonstrated that, even with only one hidden layer, neural networks can uniformly approximate any continuous function. This theoretical basis for modelling linear or non-linear systems by neural networks is therefore sound. The weights of a trained NSIN store the knowledge of the properties of the identified system. Thus, understanding how the system's physical properties and the weights of the trained NSIN are related is a worthwhile task. However, Masri *et al.* [11] indicated that determining the system's dynamic characteristics directly from the optimal weights of the approximating neural network is extremely difficult because of the non-uniqueness of the optimal weights, and also since the values of the network weights are not directly related to the system's physical properties. To overcome this difficulty, the partial derivative form of the ANN is developed and introduced in the next section.

THE PARTIAL DERIVATIVE FORM OF THE ANN

If there is a network with n hidden layers, the output of the kth node in the output layer is defined as follows:

$$O_k = f(Net_{ok}) \tag{24}$$

$$Net_{ok} = \sum_j W_{hnj,ok} H_{nj} + \theta_{ok} \tag{25}$$

where H_{nj} is the output of the jth node in the nth hidden layer, θ_{ok} is the threshold value of the kth node in the output layer, $W_{hnj,ok}$ is the weight associated with the jth node in the nth hidden layer to the kth node in the output layer and function f is the activation function.

If the number of input nodes is I, then the Taylor expansion of Equation (24) can be written as follows:

$$O_k = f(X_1,\ldots,X_I,\boldsymbol{\theta}_{h1},\ldots,\boldsymbol{\theta}_{hn},\theta_{ok}) \tag{26}$$

$$= f(0,\ldots,0,\boldsymbol{\theta}_{h1},\ldots,\boldsymbol{\theta}_{hn},\theta_{ok}) + \sum_{i-1}^{I} \frac{\partial O_k}{\partial X_i} X_i + \frac{1}{2}\sum_{i=1}^{I}\sum_{j=1}^{I} \frac{\partial^2 O_k}{\partial X_i \partial X_j} X_i X_j + \cdots \tag{27}$$

where X_i is the input of ith node in the input layer, $\boldsymbol{\theta}_{hj}$ is the threshold value vector of the jth hidden layer and

$$f(0,\ldots,0,\boldsymbol{\theta}_{h1},\ldots,\boldsymbol{\theta}_{hn},\theta_{ok}) = f\left(\sum_{jn}\left(W_{hnjn,ok}\cdots f\left(\sum_{j1} W_{h1j1,h2j2} f(\theta_{h1j1})\right)\right) + \theta_{ok}\right) \tag{28}$$

where θ_{h1j1} is the threshold value of the $j1$th node in the first hidden layer. The higher order partial derivatives of O_k are negligible if the nonlinearity in the function is not large. By keeping only the linear terms in X and assuming that the number of the output nodes is K, the linear relationship between inputs and outputs of the ANN can be expressed as the following first-order partial derivative form:

$$\left\{\begin{matrix} O_1 \\ \vdots \\ O_K \end{matrix}\right\} \approx \begin{bmatrix} \dfrac{\partial O_1}{\partial X_1} & \cdots & \dfrac{\partial O_1}{\partial X_I} \\ \vdots & \cdots & \vdots \\ \dfrac{\partial O_K}{\partial X_1} & \cdots & \dfrac{\partial O_K}{\partial X_I} \end{bmatrix} \left\{\begin{matrix} X_1 \\ \vdots \\ X_I \end{matrix}\right\}$$
$$+ \left\{\begin{matrix} f\left(\sum_{jn}\left(W_{hnjn,o1}\cdots f\left(\sum_{j1} W_{h1j1,h2j2} f(\theta_{h1j1})\right)\right) + \theta_{o1}\right) \\ \vdots \\ f\left(\sum_{jn}\left(W_{hnjn,oK}\cdots f\left(\sum_{j1} W_{h1j1,h2j2} f(\theta_{h1j1})\right)\right) + \theta_{oK}\right) \end{matrix}\right\} \tag{29}$$

The first-order derivative of the kth output with respect to ith input D_{ki} can be derived as follows:

$$D_{ki}^{1} = \frac{\partial O_k}{\partial X_i} = \sum_{jn}\cdots\sum_{j1} W_{hnjn,ok} f'(Net_{ok}) \cdots W_{xi,h1j1} f'(Net_{h1j1}) \tag{30}$$

where $W_{xi,h1j1}$ is the weight associated with the ith node in the input layer to the $j1$th node in the first hidden layer.

Equation (30) indicates that D_{ki}^{1} is a function of weights and the first-order derivative of the activation function. Actually, D_{ki}^{n} $(n \geqslant 2)$, i.e., the higher-order derivative of the kth output with respect to the ith input, is a function of weights and the derivatives, including first and higher order, of the activation function. Notably, for a trained NSIN, D_{kl}^{l} relates to the linear properties of the structure, while D_{ki}^{n} $(n \geqslant 2)$ relates to the non-linear properties of the structure.

A sequence of repeatedly identifying the system in a certain undamaged or damage state by the NSIN reveals that D_{ki}^{n} $(n \geqslant 1)$ of the NSIN has a negligible variation with different system errors. Importantly, the topology of the NSIN during the sequence of training processes must remain the same; otherwise, this loosely defined uniqueness of D_{ki}^{n} $(n \geqslant 1)$ of the NSIN does not exist. Owing to the difficulty of proving the loosely defined uniqueness of D_{ki}^{n} $(n \geqslant 1)$ of the NSIN, a linear SDOF structure example and a linear MDOF one are illustrated in the numerical example section later on. Comparing the partial derivatives of the outputs with respect to the inputs of the NSIN that identify a certain damage state with those that identify the undamaged state allows us to detect changes to the physical system from its undamaged state. The larger changes to the physical system imply a greater difference between these partial derivatives of the two NSINs. Notably, this detection was performed without a priori knowledge of the physical system properties.

NEURAL DAMAGE DETECTION NETWORK (NDDN)

The mathematical function describing a real-world structure can be extremely complex, and its exact form is usually unknown. Thus, the complex system's properties can still not be determined directly from these partial derivatives. As well known, ANNs can model input–output functional relations even when mathematically explicit formulas are unavailable. Therefore, the ANN, called neural damage detection network, can determine the system's properties directly from the optimal weights of a trained NSIN if some priori information about the damaged states are available.

The neural damage detection network attempts to identify the location and extent of the structural damage. The NDDN is trained to recognize the partial derivatives of the outputs with respect to the inputs of NSINs that identify the structure in undamaged as well as various damaged states. According to Figure 2, the input to the NDDN is taken as the partial derivatives of the outputs with respect to the outputs of the NSIN. In addition, the output of the NDDN is an identification of the damage level (reduced percentage of respective stiffness and damping) for each member in the structure. Hence, the output layer contains two nodes per member and the damage states (both damping and stiffness) in that member; an activation value of zero represents no damage and one represents complete damage. Herein, the NDDN is implemented using an adaptive L-BFGS neural network model with the sigmoid activation function.

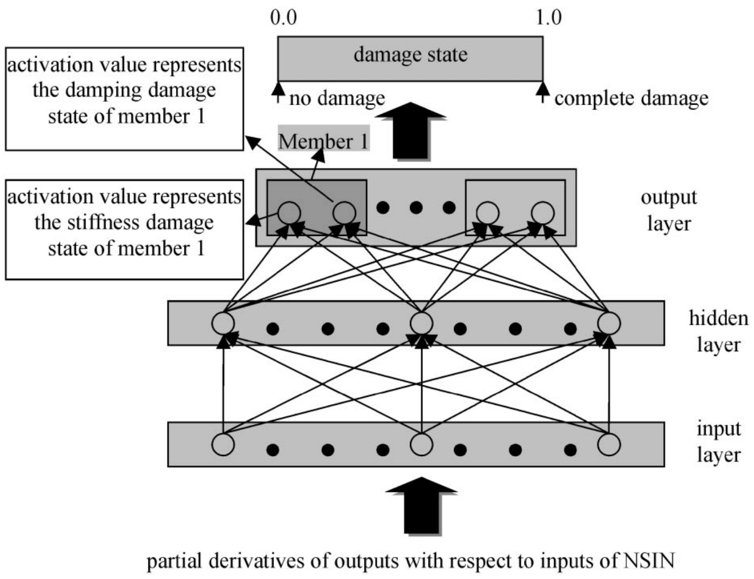

0.0
1.0
damage state
no damage
complete damage
activation value represents the damping damage state of member 1
Member 1
activation value represents the stiffness damage state of member 1
output layer
hidden layer
input layer
partial derivatives of outputs with respect to inputs of NSIN

Figure 2. The architecture of the neural damage detection network.

THE NUMERICAL EXAMPLE

Example 1: SDOF structure system

Herein, a linear SDOF system is selected as the structure to investigate the damage detection capacity of the proposed method. The structural properties are taken to be: mass $m = 2923.38$ kg; stiffness $k = 1391.06$ kN/m; and damping coefficient $c = 6373.74$ N s/m. Response spectra for the 1940 EL-Centro earthquake record are used as the external excitation, but scaled to 25 per cent of the intensity of the original earthquake. The sampling period Δt is 0.01 s. In this example, the relative displacement, relative velocity and relative acceleration time histories, computed by state space procedure (SSP), are used as a measured response of the structure.

NSINs training. In this example, 66 NSINs are used to identify 66 distinct states (including undamaged and damaged states) of the structure. The 66 distinct states (cases) of the structure reveal that the stiffness is reduced from 0 to 50 per cent every 5 per cent and damping from 0 to 25 per cent every 5 per cent. Each NSIN consists of four, four, and three nodes in the input layer, the hidden layer, and the output layer, respectively, and denoted as NSIN_L-BFGS(4-4-3). The four input data are the structural relative displacement d_k, relative velocity v_k, relative acceleration a_k and external excitation p_k. The three outputs are d_{k+1}, v_{k+1} and a_{k+1}.

To verify the loosely defined uniqueness of partial derivatives of the outputs with respect to the inputs of a trained NSIN, the undamaged structure is repeatedly modelled by the NSIN ten times, and each time uses a different set of starting weights and takes 5000 cycles for the complete off-line training process. Since the structure in this example is a linear system, only the first-order partial derivatives of the NSIN need to be computed for damage detection. Results of the ten modellings (Table I) indicate that the NSIN's 12 first-order partial derivatives of the outputs with respect to the inputs (the 2nd to the 13th columns in Table I) have only a slight variance with the different system error (the 14th column in Table I). For any two different modellings, the larger difference between their system errors implies a greater difference between their first partial derivatives of the outputs with respect to the inputs of the NSIN. Nevertheless, the differences of first partial derivatives of the outputs with respect to the inputs of the NSIN between the best and worst modellings are still very small and can even be negligible.

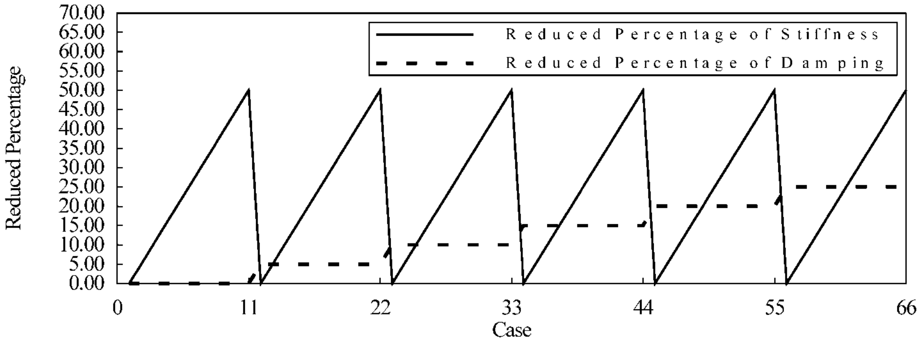
Next, each of the other 65 cases is also repeatedly modelled by a NSIN ten times, and each time uses a different set of starting weights and takes 5000 cycles for the complete off-line training process. For each case, only the best model out of 10 trials is chosen, and a portion of those results are shown in Table II. This table reveals that $\partial v_{i+1}/\partial d_i$ (the 2nd quantitative indicator), $\partial a_{i+1}/\partial d_i$ (the 3rd quantitative indicator), $\partial v_{i+1}/\partial v_i$ (the 5th quantitative indicator) and $\partial a_{i+1}/\partial v_i$ (the 6th quantitative indicator) obviously vary with the stiffness, and only $\partial a_{i+1}/\partial v_i$ with damping, while other eight quantitative indicators vary unobviously with stiffness or damping. Interestingly, in this example, $\partial v_{i+1}/\partial d_i$, $\partial a_{i+1}/\partial d_i$ and $\partial a_{i+1}/\partial v_i$ are nearly proportional to stiffness as shown in Figure 3. Besides, the larger changes to the physical system imply a larger change of the four obviously varying quantitative indicators. Notably, this detection is performed without *a priori* knowledge of the physical system properties.

Table I. Results of ten modelings of structural undamaged state (SDOF example).

No. of modeling	First-order partial derivatives of outputs with respect to inputs												System error
	$\frac{\partial d_{i+1}}{\partial d_i}$	$\frac{\partial v_{i+1}}{\partial d_i}$	$\frac{\partial a_{i+1}}{\partial d_i}$	$\frac{\partial d_{i+1}}{\partial v_i}$	$\frac{\partial v_{i+1}}{\partial v_i}$	$\frac{\partial a_{i+1}}{\partial v_i}$	$\frac{\partial d_{i+1}}{\partial a_i}$	$\frac{\partial v_{i+1}}{\partial a_i}$	$\frac{\partial a_{i+1}}{\partial a_i}$	$\frac{\partial d_{i+1}}{\partial p_i}$	$\frac{\partial v_{i+1}}{\partial p_i}$	$\frac{\partial a_{i+1}}{\partial p_i}$	
1	0.9975	−0.3040	−1.1599	0.1531	0.9532	−0.2619	0.0074	−0.0006	−0.0176	−0.0048	−0.0728	−0.2794	0.00000347
2	0.9976	−0.3009	−1.1623	0.1527	0.9723	−0.2606	0.0081	0.0082	−0.0197	−0.0064	−0.0706	−0.2784	0.00000594
3	0.9969	−0.3022	−1.1579	0.1525	0.9529	−0.2610	0.0067	0.0013	−0.0155	−0.0050	−0.0745	−0.2786	0.00000338
4	0.9978	−0.3024	−1.1592	0.1514	0.9527	−0.2620	0.0072	0.0028	−0.0172	−0.0045	−0.0719	−0.2790	0.00000356
5	0.9801	−0.2980	−1.1175	0.1503	0.9771	−0.2554	−0.0006	0.0111	0.0010	−0.0094	−0.0707	−0.2688	0.00000445
6	0.9895	−0.3043	−1.1627	0.1485	0.9537	−0.2641	−0.0008	−0.0004	−0.0206	−0.0074	−0.0748	−0.2803	0.00000372
7	0.9949	−0.3052	−1.1595	0.1519	0.9550	−0.2604	0.0050	0.0001	−0.0168	−0.0057	−0.0735	−0.2791	0.00000395
8	0.9924	−0.3043	−1.1562	0.1518	0.9541	−0.2603	0.0032	−0.0002	−0.0142	−0.0065	−0.0718	−0.2778	0.00000320
9	0.9933	−0.3072	−1.1579	0.1524	0.9763	−0.2607	0.0048	0.0041	−0.0175	−0.0063	−0.0725	−0.2779	0.00000708
10	0.9926	−0.3060	−1.1320	0.1580	0.9689	−0.2538	0.0081	0.0026	0.0016	−0.0102	−0.0664	−0.2741	0.00000390

Table II. A portion of the first-order partial derivatives of outputs with respect to inputs of the 66 NSINs (SDOF example).

Case	First-order partial derivatives of outputs with respect to inputs												Reduced percentage of stiffness or damping coefficient	
	$\frac{\partial d_{i+1}}{\partial d_i}$	$\frac{\partial v_{i+1}}{\partial d_i}$	$\frac{\partial a_{i+1}}{\partial d_i}$	$\frac{\partial d_{i+1}}{\partial v_i}$	$\frac{\partial v_{i+1}}{\partial v_i}$	$\frac{\partial a_{i+1}}{\partial v_i}$	$\frac{\partial d_{i+1}}{\partial a_i}$	$\frac{\partial v_{i+1}}{\partial a_i}$	$\frac{\partial a_{i+1}}{\partial a_i}$	$\frac{\partial d_{i+1}}{\partial p_i}$	$\frac{\partial v_{i+1}}{\partial p_i}$	$\frac{\partial a_{i+1}}{\partial p_i}$	k	c
1	0.9924	−0.3043	−1.1562	0.1518	0.9541	−0.2603	0.0032	−0.0002	−0.0142	−0.0065	−0.0718	−0.2778	0	0
2	0.9962	−0.2889	−1.0973	0.1530	0.9550	−0.2492	0.0053	−0.0007	−0.0119	−0.0056	−0.0735	−0.2767	5	0
3	1.0001	−0.2691	−1.0160	0.1529	0.9563	−0.2366	0.0068	0.0022	−0.0058	−0.0026	−0.0731	−0.2769	10	0
4	0.9912	−0.2592	−0.9694	0.1527	0.9595	−0.2262	−0.0008	0.0005	−0.0002	−0.0057	−0.0727	−0.2736	15	0
5	0.9925	−0.2449	−0.9178	0.1528	0.9606	−0.2164	−0.0001	−0.0011	−0.0049	−0.0050	−0.0737	−0.2765	20	0
6	0.9904	−0.2360	−0.8597	0.1515	0.9704	−0.2036	−0.0017	−0.0043	−0.0049	−0.0054	−0.0734	−0.2775	25	0
7	0.9924	−0.2097	−0.8008	0.1556	0.9702	−0.1904	−0.0042	0.0075	−0.0008	−0.0111	−0.0681	−0.2769	30	0
8	0.9826	−0.1975	−0.7384	0.1506	0.9829	−0.1808	−0.0024	0.0057	−0.0016	−0.0073	−0.0741	−0.2752	35	0
9	0.9992	−0.1873	−0.6884	0.1548	0.9723	−0.1670	−0.0009	−0.0053	−0.0001	−0.0078	−0.0763	−0.2752	40	0
10	0.9850	−0.1721	−0.6274	0.1508	0.9874	−0.1580	−0.0031	−0.0019	−0.0006	−0.0079	−0.0759	−0.2755	45	0
11	0.9861	−0.1574	−0.5733	0.1513	0.9945	−0.1467	0.0001	−0.0020	−0.0029	−0.0056	−0.0766	−0.2772	50	0
12	0.9970	−0.3037	−1.1575	0.1524	0.9539	−0.2563	0.0065	0.0001	−0.0140	−0.0051	−0.0731	−0.2781	0	5
13	0.9961	−0.2893	−1.1011	0.1532	0.9564	−0.2453	0.0051	−0.0005	−0.0139	−0.0059	−0.0731	−0.2779	5	5
14	0.9946	−0.2731	−1.0361	0.1533	0.9577	−0.2346	0.0027	0.0014	−0.0066	−0.0051	−0.0723	−0.2759	10	5
15	0.9947	−0.2565	−0.9794	0.1534	0.9594	−0.2241	0.0015	0.0024	−0.0064	−0.0046	−0.0721	−0.2767	15	5
16	0.9884	−0.2434	−0.9125	0.1524	0.9615	−0.2123	−0.0030	0.0011	0.0004	−0.0066	−0.0724	−0.2750	20	5
17	0.9902	−0.2277	−0.8586	0.1534	0.9639	−0.2009	−0.0025	0.0016	−0.0021	−0.0067	−0.0717	−0.2772	25	5
18	0.9919	−0.2129	−0.7985	0.1522	0.9643	−0.1895	−0.0048	0.0009	0.0020	−0.0097	−0.0729	−0.2742	30	5
19	1.0001	−0.1985	−0.7473	0.1530	0.9666	−0.1782	0.0023	−0.0001	−0.0036	−0.0082	−0.0737	−0.2755	35	5
20	1.0008	−0.1869	−0.6890	0.1518	0.9726	−0.1669	0.0016	−0.0036	−0.0014	−0.0080	−0.0748	−0.2752	40	5
21	0.9854	−0.1724	−0.6283	0.1511	0.9881	−0.1552	−0.0008	−0.0022	−0.0017	−0.0062	−0.0760	−0.2763	45	5
22	0.9859	−0.1579	−0.5727	0.1513	0.9958	−0.1436	0.0001	−0.0024	−0.0017	−0.0059	−0.0767	−0.2770	50	5
23	0.9915	−0.3035	−1.1592	0.1538	0.9552	−0.2517	0.0032	0.0004	−0.0153	−0.0069	−0.0732	−0.2782	0	10



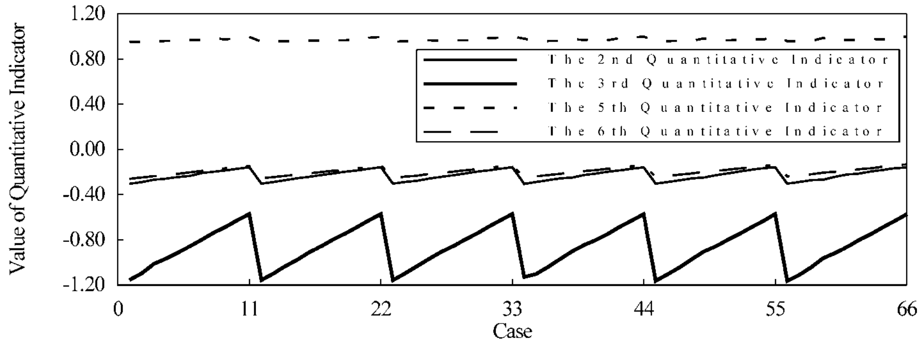


Figure 3. Values of the 2nd, 3rd, 5th and 6th quantitative indicator vary with stiffness and damping in the 66 cases (SDOF example).

The NDDN training and testing. Herein, the NDDN is constructed with 12, five and two nodes in input layer, hidden layer, and output layer, respectively, and denoted as NDDN_L-BFGS(12-5-2). The 12 input data are the 12 first-order partial derivatives of the outputs with respect to the inputs of the NSIN. The two outputs are the reduced percentage of stiffness and damping of the structure.

Among the 66 cases, 54 cases are used to train the NDDN, and other cases (case 2, 11, 12, 21, 26, 31, 36, 41, 50, 51, 60, 61) are used to test the NDDN. The 12 testing cases are uniformly selected in the 66 cases to verify the damage detection ability of the NDDN. The complete off-line training process takes 2000 cycles, and the system error converges to 0.00002305. After the NDDN is trained on the 54 training cases, it is tested to observe how accurately it would recognize other states of damage. Table III summarizes the results of these tests, indicating that the NDDN can satisfactorily diagnose the damaged state (both damping and stiffness) in all 12 testing cases.

Example 2: MDOF structure system

In this example, a 5-storey shear building is chosen to demonstrate the feasibility of using the proposed method to detect the damage of linear MDOF structure systems. The structural properties of the building are assumed to be as follows: floor mass $m = 8 \times 10^4$ kg, floor

Table III. Damage detection results (SDOF example).

Testing case	Real (predicted) reduced percentage of stiffness and damping	
	k	c
1	5 (4.3779)	0 (0.8359)
2	50 (49.9299)	0 (1.4386)
3	0 (0.7469)	5 (4.7800)
4	45 (44.9394)	5 (4.6387)
5	15 (15.5242)	10 (13.9134)
6	40 (39.9733)	10 (12.1338)
7	10 (9.9342)	15 (13.9702)
8	35 (34.8921)	15 (14.2284)
9	25 (25.0451)	20 (20.6451)
10	30 (29.9777)	20 (21.0848)
11	20 (19.8623)	25 (26.1801)
12	25 (24.8726)	25 (25.9880)

stiffness $k = 4 \times 10^7$N/m, and floor damping $c = 1.5 \times 10^6$Ns/m for all floors. Response spectra for the 1940 EL-Centro earthquake record is used as the external excitation. The sampling period Δt is 0.01 s. In this example, only the relative acceleration time histories of the first, third, and fifth floor, computed by SSP, are used as measured responses of the structure.

NSINs training. In this example, 243 NSINs are used to identify the following 243 distinct states (including undamaged and damaged states) of the structure. The 243 distinct states (cases) of the structure stipulate the arrangement of the five floor stiffness reductions, and each floor stiffness reduction varies from 0 to 20 per cent every 10 per cent. Herein, each NSIN consists of five, four and three nodes in input layer, hidden layer, and output layer, respectively, and denoted as NSIN_L-BFGS(5-4-3). The five input data are the relative acceleration of the first floor a_k^1, the third floor a_k^3, the fifth floor a_k^5 and external excitations p_k and p_{k-1}. The three outputs are the relative acceleration of the first floor a_{k+1}^1, the third floor a_{k+1}^3 and the fifth floor a_{k+1}^5.

The undamaged structure is repeatedly modelled by the NSIN ten times, with each time using a different set of starting weights and taking 5000 cycles for the complete off-line training process. Results of the ten modelling (Table IV) indicate that the NSIN's 15 first-order partial derivatives of the outputs with respect to the inputs (the 2nd to the 16th columns in Table IV) have only a slight variance with the different system error (the 17th column in Table IV). According to this table, the loosely defined uniqueness of partial derivatives of the outputs with respect to the inputs of a trained NSIN still holds in this example.

Next, each of other 242 cases is also repeatedly modelled by a NSIN ten times, each time using a different set of starting weights and taking 5000 cycles for the complete off-line training process. In each case, only the best model out of 10 trials is chosen, and a portion of the results are shown in Table V. In Table V, k_i is the stiffness of the ith floor.

The NDDN training and testing. Herein, the NDDN is constructed with 15, nine and five nodes in input layer, hidden layer, and output layer, respectively, and denoted as NDDN_L-BFGS(15-9-5). The fifteen input data are the 15 first-order partial derivatives of the outputs

Table IV. Results of ten modelings of structural undamaged state (MDOF example).

No. of modeling	First-order partial derivatives of outputs with respect to inputs															System error
	$\frac{\partial a^1_{i+1}}{\partial a^1_i}$	$\frac{\partial a^3_{i+1}}{\partial a^1_i}$	$\frac{\partial a^5_{i+1}}{\partial a^1_i}$	$\frac{\partial a^1_{i+1}}{\partial a^3_i}$	$\frac{\partial a^3_{i+1}}{\partial a^3_i}$	$\frac{\partial a^5_{i+1}}{\partial a^3_i}$	$\frac{\partial a^1_{i+1}}{\partial a^5_i}$	$\frac{\partial a^3_{i+1}}{\partial a^5_i}$	$\frac{\partial a^5_{i+1}}{\partial a^5_i}$	$\frac{\partial a^1_{i+1}}{\partial p_i}$	$\frac{\partial a^3_{i+1}}{\partial p_i}$	$\frac{\partial a^5_{i+1}}{\partial p_i}$	$\frac{\partial a^1_{i+1}}{\partial p_{i-1}}$	$\frac{\partial a^3_{i+1}}{\partial p_{i-1}}$	$\frac{\partial a^5_{i+1}}{\partial p_{i-1}}$	
1	0.8550	0.1159	−0.0646	−0.0405	0.9479	0.1813	0.0580	0.0013	0.8836	−0.6802	−0.8233	−0.8211	0.7214	0.8563	0.8393	0.00001166
2	0.8535	0.1162	−0.0695	−0.0379	0.9421	0.1834	0.0564	0.0101	0.8883	−0.6729	−0.8147	−0.8333	0.7155	0.8463	0.8494	0.00001293
3	0.8556	0.1173	−0.0631	−0.0408	0.9434	0.1806	0.0579	0.0045	0.8833	−0.6800	−0.8189	−0.8236	0.7212	0.8526	0.8413	0.00001162
4	0.8556	0.1166	−0.0651	−0.0410	0.9477	0.1822	0.0583	0.0015	0.8827	−0.6807	−0.8236	−0.8213	0.7218	0.8565	0.8394	0.00001170
5	0.8563	0.1205	−0.0681	−0.0425	0.9428	0.1865	0.0590	0.0042	0.8799	−0.6782	−0.8219	−0.8219	0.7198	0.8551	0.8398	0.00001178
6	0.8539	0.1153	−0.0637	−0.0391	0.9479	0.1789	0.0572	0.0014	0.8855	−0.6805	−0.8229	−0.8210	0.7215	0.8558	0.8392	0.00001166
7	0.8551	0.1184	−0.0662	−0.0416	0.9448	0.1839	0.0586	0.0031	0.8820	−0.6790	−0.8230	−0.8215	0.7204	0.8561	0.8395	0.00001169
8	0.8548	0.1206	−0.0682	−0.0410	0.9426	0.1867	0.0583	0.0045	0.8802	−0.6797	−0.8216	−0.8223	0.7209	0.8549	0.8403	0.00001173
9	0.8615	0.1246	−0.0686	−0.0420	0.9414	0.1845	0.0571	0.0043	0.8819	−0.6859	−0.8206	−0.8233	0.7275	0.8543	0.8413	0.00001215
10	0.8549	0.1211	−0.0679	−0.0404	0.9405	0.1854	0.0578	0.0059	0.8814	−0.6806	−0.8201	−0.8229	0.7218	0.8534	0.8408	0.00001175

Table V. A portion of the first-order partial derivatives of outputs with respect to inputs of the 243 NSINs (MDOF Example).

Case	First-order partial derivatives of outputs with respect to inputs															Reduced percentage of stiffness				
	$\frac{\partial a_{i+1}^1}{\partial a_i^1}$	$\frac{\partial a_{i+1}^3}{\partial a_i^1}$	$\frac{\partial a_{i+1}^5}{\partial a_i^1}$	$\frac{\partial a_{i+1}^1}{\partial a_i^3}$	$\frac{\partial a_{i+1}^3}{\partial a_i^3}$	$\frac{\partial a_{i+1}^5}{\partial a_i^3}$	$\frac{\partial a_{i+1}^1}{\partial a_i^5}$	$\frac{\partial a_{i+1}^3}{\partial a_i^5}$	$\frac{\partial a_{i+1}^5}{\partial a_i^5}$	$\frac{\partial a_{i+1}^1}{\partial p_i}$	$\frac{\partial a_{i+1}^3}{\partial p_i}$	$\frac{\partial a_{i+1}^5}{\partial p_i}$	$\frac{\partial a_{i+1}^1}{\partial p_{i-1}}$	$\frac{\partial a_{i+1}^3}{\partial p_{i-1}}$	$\frac{\partial a_{i+1}^5}{\partial p_{i-1}}$	k_1	k_2	k_3	k_4	k_5
1	0.8556	0.1173	−0.0631	−0.0408	0.9434	0.1806	0.0579	0.0045	0.8833	−0.6800	−0.8189	−0.8236	0.7212	0.8526	0.8413	0	0	0	0	0
2	0.8483	0.1141	−0.0663	−0.0232	0.9481	0.1851	0.0509	0.0002	0.8815	−0.6818	−0.8237	−0.8215	0.7221	0.8555	0.8395	10	0	0	0	0
3	0.8399	0.1134	−0.0706	−0.0059	0.9467	0.1917	0.0446	−0.0002	0.8782	−0.6829	−0.8238	−0.8212	0.7224	0.8545	0.8390	20	0	0	0	0
4	0.8613	0.1105	−0.0600	−0.0543	0.9499	0.1783	0.0656	0.0028	0.8836	−0.6799	−0.8235	−0.8209	0.7208	0.8560	0.8389	0	10	0	0	0
5	0.8680	0.1074	−0.0566	−0.0693	0.9487	0.1785	0.0740	0.0061	0.8812	−0.6796	−0.8237	−0.8210	0.7202	0.8560	0.8388	0	20	0	0	0
6	0.8538	0.1141	−0.0595	−0.0356	0.9445	0.1769	0.0533	0.0056	0.8852	−0.6800	−0.8225	−0.8204	0.7209	0.8552	0.8386	0	0	10	0	0
7	0.8535	0.1130	−0.0559	−0.0316	0.9417	0.1772	0.0487	0.0093	0.8831	−0.6792	−0.8228	−0.8197	0.7199	0.8555	0.8379	0	0	20	0	0
8	0.8539	0.1231	−0.0658	−0.0386	0.9440	0.1808	0.0555	0.0016	0.8864	−0.6797	−0.8201	−0.8224	0.7200	0.8539	0.8421	0	0	0	10	0
9	0.8507	0.1264	−0.0652	−0.0348	0.9471	0.1751	0.0519	−0.0029	0.8933	−0.6807	−0.8190	−0.8217	0.7195	0.8534	0.8431	0	0	0	20	0
10	0.8550	0.1200	−0.0681	−0.0403	0.9442	0.1848	0.0574	0.0025	0.8825	−0.6817	−0.8191	−0.8231	0.7223	0.8525	0.8416	0	0	0	0	10
11	0.8545	0.1204	−0.0681	−0.0417	0.9466	0.1819	0.0579	0.0005	0.8866	−0.6812	−0.8197	−0.8229	0.7209	0.8534	0.8423	0	0	0	0	20
12	0.8568	0.1155	−0.0651	−0.0395	0.9413	0.1865	0.0595	0.0067	0.8788	−0.6805	−0.8221	−0.8217	0.7209	0.8539	0.8394	10	10	0	0	0
13	0.8481	0.1143	−0.0629	−0.0199	0.9434	0.1845	0.0470	0.0049	0.8801	−0.6807	−0.8238	−0.8203	0.7209	0.8557	0.8383	10	0	10	0	0
14	0.8493	0.1200	−0.0676	−0.0246	0.9470	0.1846	0.0502	−0.0015	0.8843	−0.6805	−0.8226	−0.8215	0.7201	0.8552	0.8410	10	0	0	10	0
15	0.8470	0.1207	−0.0709	−0.0209	0.9400	0.1894	0.0492	0.0040	0.8807	−0.6829	−0.8197	−0.8234	0.7227	0.8521	0.8418	10	0	0	0	10
16	0.8623	0.1131	−0.0582	−0.0525	0.9426	0.1800	0.0628	0.0088	0.8806	−0.6795	−0.8219	−0.8211	0.7204	0.8545	0.8390	0	10	10	0	0
17	0.8607	0.1146	−0.0602	−0.0531	0.9513	0.1768	0.0634	−0.0002	0.8868	−0.6795	−0.8232	−0.8208	0.7194	0.8564	0.8404	0	10	0	10	0
18	0.8623	0.1178	−0.0635	−0.0559	0.9437	0.1801	0.0662	0.0055	0.8841	−0.6803	−0.8206	−0.8224	0.7206	0.8537	0.8409	0	10	0	0	10
19	0.8533	0.1162	−0.0608	−0.0354	0.9496	0.1792	0.0519	0.0003	0.8855	−0.6791	−0.8232	−0.8209	0.7191	0.8566	0.8406	0	0	10	10	0
20	0.8546	0.1120	−0.0610	−0.0375	0.9513	0.1785	0.0541	0.0006	0.8850	−0.6801	−0.8236	−0.8206	0.7204	0.8564	0.8393	0	0	10	0	10
21	0.8532	0.1233	−0.0657	−0.0385	0.9457	0.1763	0.0552	−0.0002	0.8913	−0.6797	−0.8202	−0.8218	0.7194	0.8541	0.8419	0	0	0	10	10
22	0.8541	0.1095	−0.0576	−0.0323	0.9437	0.1786	0.0535	0.0074	0.8824	−0.6812	−0.8223	−0.8204	0.7211	0.8538	0.8382	10	10	10	0	0
23	0.8551	0.1144	−0.0625	−0.0370	0.9496	0.1822	0.0567	−0.0004	0.8835	−0.6809	−0.8231	−0.8213	0.7201	0.8554	0.8407	10	10	0	10	0

Table VI. Damage detection results (MDOF example).

Testing case	Real (predicted) reduced percentage of stiffness				
	k_1	k_2	k_3	k_4	k_5
1	0 (0.00)	0 (0.31)	10 (10.48)	0 (0.32)	0 (0.29)
2	10 (9.78)	0 (0.15)	0 (0.25)	10 (8.26)	0 (0.50)
3	0 (0.00)	10 (9.26)	0 (0.31)	10 (9.83)	10 (10.61)
4	10 (8.65)	10 (7.72)	10 (11.48)	10 (7.37)	0 (3.60)
5	0 (0.03)	20 (20.75)	0 (0.22)	0 (1.73)	20 (20.89)
6	20 (19.82)	0 (0.22)	20 (20.08)	0 (0.48)	20 (24.20)
7	0 (0.00)	20 (18.97)	20 (21.83)	20 (17.21)	20 (17.38)
8	10 (9.70)	0 (0.25)	0 (0.15)	20 (19.29)	0 (2.39)
9	0 (0.00)	20 (19.95)	0 (0.55)	0 (0.27)	10 (7.90)
10	20 (20.16)	10 (8.23)	10 (12.50)	0 (0.24)	0 (0.75)
11	0 (0.00)	10 (11.80)	20 (20.54)	10 (6.29)	0 (2.87)
12	20 (19.89)	0 (0.40)	0 (0.29)	10 (7.09)	10 (13.85)
13	20 (20.03)	10 (11.72)	0 (0.07)	20 (20.11)	0 (0.74)
14	10 (11.07)	0 (0.44)	20 (19.38)	0 (2.28)	20 (17.19)
15	0 (0.00)	20 (19.56)	0 (1.39)	20 (20.47)	10 (10.49)
16	10 (10.26)	10 (10.48)	20 (19.89)	0 (0.54)	10 (8.06)
17	0 (0.00)	10 (9.40)	10 (11.74)	20 (20.38)	10 (12.77)
18	10 (10.43)	20 (20.50)	10 (11.22)	20 (20.00)	0 (0.00)
19	20 (19.85)	0 (0.34)	20 (19.66)	10 (11.98)	10 (10.50)
20	20 (19.79)	20 (20.11)	10 (10.73)	0 (0.60)	20 (22.50)
21	20 (19.99)	10 (9.70)	20 (19.16)	20 (19.45)	20 (19.59)
22	10 (9.86)	20 (19.99)	10 (10.39)	10 (10.69)	20 (19.76)
23	20 (19.80)	10 (8.82)	20 (19.79)	20 (19.06)	10 (14.18)

with respect to the inputs of the NSIN. The five outputs are the reduced percentage of stiffness of each floor.

Among the 243 cases, 220 cases are used for training of the NDDN, and other 23 cases are used to test the NDDN. The 23 testing cases are uniformly selected in the 243 cases to verify the damage detection ability of the NDDN. The complete off-line training process takes 10 000 cycles and the system error converges to 0.000104. Table VI summarizes the testing results, indicating that NDDN can satisfactorily diagnose the damaged state in all of the 23 testing cases.

Although the two examples demonstrate only the feasibility of using the proposed method for damage detection of linear structures, we believe that this method can be applied to non-linear structures as well.

CONCLUSIONS

Neural networks are a highly effective means of identifying systems that are typically encountered in structural dynamics. The weights of the approximating neural network store the knowledge of the structural properties. The partial derivatives of the outputs with respect to the inputs (functions of weights and the activation function) of the approximating neural network have the loosely defined uniqueness property that enables these partial derivatives to quantitatively indicate system damage from the model parameters. This work presents a novel neural network-based approach to detect structural damage. First, NSINs are used to identify the undamaged and damaged states of the structural system. Comparing the quantitative indicators of the NSIN that identify a certain damaged state with those that identify the undamaged state allows us to detect changes to the physical system from its undamaged state. This detection is performed without *a priori* knowledge of the physical system properties. Second, by assuming that some *a priori* information about the system are available, then this knowledge can be used to train the NDDN in order to identify the location and extent of the damage which is never used in training the NDDN. Illustrative examples demonstrate that the proposed structural damage detection method is not only feasible for linear SDOF systems with complete measured data, but also feasible for linear MDOF systems with incomplete measured data. Furthermore, this method is highly promising for applications to non-linear structures since ANNs can treat both linear and non-linear systems with the same formulation.

ACKNOWLEDGEMENTS

The authors would like to thank the National Science Council of the Republic of China for financially supporting this research under Contract No. NSC 89-2211-E-009-014.

REFERENCES

1. Rubin S. Ambient vibration survey of offshore platform. *Journal of Engineering Mechanics*, ASCE 1980; **106**:425–441.
2. Shahrivar F, Bouwkamp JG. Signal separation method for tower mode shape measurement. *Journal of Structural Engineering*, ASCE, 1989; **115**:707–723.
3. Adeli H, Hung SL. *Machine Learning—Neural Networks*, *Genetic Algorithms*, *and Fuzzy Systems*. Wiley: New York, 1995.
4. Johan AK, Joos PL, Bart LR. *Artificial Neural Networks for Modeling and Control of Non-Linear Systems*. Kluwer Academic Publishers: Boston, 1996.
5. Rumelhart DE, Hinton GE, Williams RJ. Learning international representation by error propagation. In *Parallel Distributed Processing*, Rumelhart DE, McClelland JL and the PDP Research Group (eds). The MIT Press: Cambridge, MA, 1986; 318–362.
6. Wu X, Ghaboussi J, Garrett Jr. JH. Use of neural networks in detection of structural damage *Computers & Structures* 1992; **42**(4):649–659.
7. Elkordy MF, Chang KC, Lee GC. Neural networks trained by analytically simulated damage states. *Journal of Computers in Civil Engineering*, ASCE 1993; **7**(2):130–145.
8. Szewezyk P, Hajela P. Damage detection in structures based on feature-sensitivity neural networks. *Journal of Computers in Civil Engineering*, ASCE 1994; **8**(2):163–179.
9. Pandy PC, Barai SV. Multilayer perception in damage detection of bridge structures *Computers & Structures* 1995; **54**(4):597–608.
10. Masi SF, Nakamura M, Chassiakos AG, Caughey TK. Neural network approach to detection of changes in structural parameters. *Journal of Engineering Mechanics*, ASCE 1996; **122**(5):350–360.
11. Masri SF, Smyth AW, Chassiakos AG, Caughey TK, Hunter NF. Application of neural networks for detection of changes in nonlinear systems: *Journal of Engineering Mechanics*, ASCE 2000; **126**(7):666–676.
12. Hung SL, Lin YL. Application of an L-BFGS neural network learning algorithm in engineering analysis and design. Chinese Society of Structural Engineering, Nantou, Taiwan, R.O.C., 1994; 221–230.
13. Nocedal J. Updating quasi-newton matrix with limited storage. *Mathematics of Compuation* 1980; **35**:20–33.
14. Coleman TF, Li Y. *Large-Scale Numerical Optimization*. Society for Industrial and Applied Mathematics: Philadelphia, 1990.
15. Cybenko G. Approximations by superpositions of a sigmoidal function. *Mathematics of Control*, *Signals and Systems* 1989; **2**:303–314.
16. Funahashai K. On the approximate realization of continuous mappings by neural networks. *Neural Networks* 1989; **2**:183–192.