



DYNAMIC FINITE ELEMENT MODEL UPDATING USING NEURAL NETWORKS

R. I. LEVIN AND N. A. J. LIEVEN

Department of Aerospace Engineering, University of Bristol, Bristol BS8 1TR, England

(Received 12 January 1996, and in final form 17 September 1997)

In this paper, a new method of finite element model updating using neural networks is presented. Many previous model updating techniques have exhibited inconsistent performance when subjected to noisy experimental data. From this background it is clear that a successful model updating method must be resistant to experimental noise. A well-known property of neural networks is robustness in the presence of noise, and it is hoped to exploit this property for model updating purposes. The proposed updating method is tested on a simple simulated model, both in the absence and presence of noise, with promising results. A further advantage of this updating method is the ability to work with a limited number of experimentally measured degrees of freedom and modes.

© 1998 Academic Press Limited

1. INTRODUCTION

Dynamic Finite Element (FE) models are used extensively in industry to model the behaviour of physical structures. While these models provide a convenient means of understanding the performance of structures under a variety of user defined load conditions, there are often discrepancies between analytical results from an FE model and actual experimental results [1]. To reduce these discrepancies, attempts are made to adjust the FE model using the experimental data. The basis of this approach is founded on the supposition that tests on real structures are more trustworthy than the analytical counterpart.

There have been many previous attempts to update an analytical FE model of a physical structure using experimentally measured data [2–4]. While most of these methods work successfully when the (simulated) experimental data are noise free, the addition of noise to the “experimental” data causes many of the methods to fail. This unfortunate characteristic is inevitably exacerbated by noise on real data, which is consistently more inventive than computer generated simulations. Thus it is known that all practical experimental data will contain noise, so it is desirable to develop an updating method that is resistant to noise.

Neural networks have been applied successfully to many diverse areas of application. They are particularly applicable to problems in which plenty of examples are available but it is difficult to specify an explicit algorithm, such as character recognition, time series prediction, etc. [5, 6]. It is widely known that neural networks tend to be robust in the presence of noise and are able to distinguish between these random errors and the desired systematic outputs. Hence it seems natural and appropriate to apply neural networks to this field.

In this paper an FE model updating technique using neural networks is demonstrated and the technique is applied to a simple model, with and without noise.

2. INTRODUCTION TO NEURAL NETWORKS

Artificial neural networks—usually referred to simply as neural networks—have certain properties that make them attractive to some applications. The two main properties that are of use for this application are the capacity to generalize, i.e., the production of a sensible response from previously unseen data, and robustness in the presence of noise. Other properties include excellent damage tolerance, i.e., the removal of a few neurons from a network does not significantly degrade performance, and very fast running speeds once trained, although training a neural network tends to be a slow process. A more detailed discussion of the properties of neural networks can be found in Bishop [7].

The phrase neural network refers to a computational structure derived from the study of biological neurons. Neural networks consist of a number of simple processing units, *neurons*, linked to each other. All neurons have multiple *inputs* and a single *output* as shown in Figure 1, but there are many different types of neuron. The output of a neuron is related to the inputs in a manner that depends on the type of neuron. The response characteristic of a common neuron called a *perceptron* [8] is shown below (a list of notation is given in the Appendix):

$$f = \phi\left(b + \sum_{i=1}^n W_i h_i\right), \quad (1)$$

where n is the number of inputs to the neuron, h_i is the value of the i th input, f is the output of the neuron, W_i is the value of the i th weight of the neuron, b is the bias of the neuron, and ϕ is the transfer function of the neuron.

Common transfer functions include the log sigmoid function $\phi(v) = 1/(1 + \exp(-v))$ and the pure linear function $\phi(v) = v$. The weights and bias of the neuron are the *parameters* of the neuron. In neural networks it is these parameters that contain information. A single neuron is not very powerful, but many neurons combined may perform complex tasks.

The *architecture* of a neural network is the number and type of neurons present in the network, and how they are connected to each other and to the outside world. Many common networks have a *layered* architecture; that is, the neurons are divided into distinct layers, with every output from one layer connected to every input in the next layer. The

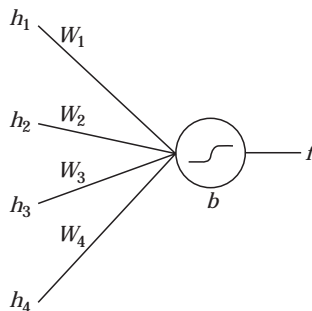



Figure 1. A simple neuron (perceptron): $f = \phi(\sum_{i=1}^n W_i h_i)$. , Log-sigmoid transfer function.

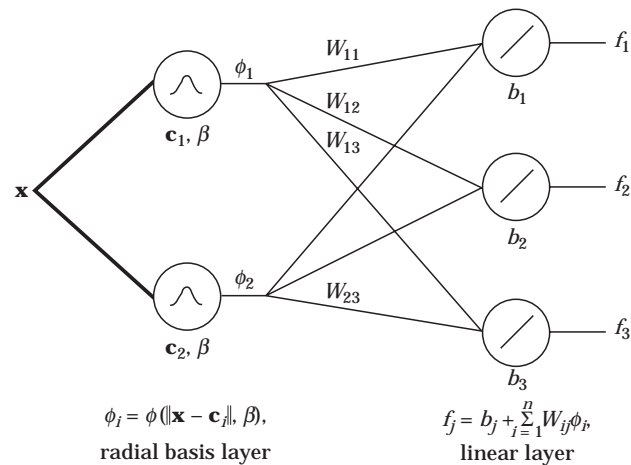


Figure 2. A schematic of a small RBF network.

inputs of one specified layer and the outputs of another specified layer will be connected to the outside world. All other layers are termed hidden layers.

Neural networks need to be trained before they can be used. This requires a *training algorithm* and a set of *training data*. Training data typically consists of a set of input vectors of length n_i and a corresponding set of output vectors of length n_o . Training a network consists of adjusting the network parameters until the output of the network, when presented with the input vectors, matches the target output vectors to within a given tolerance. Often, some of the training data is held back as a *test set* to check that adequate generalization has occurred.

Most layered neural networks used are *feedforward* networks; i.e., there are no loops in the architecture. Consequently, there are only a few architectural decisions to be made; namely, how many layers a network has and how many neurons are presented in each layer. These decisions can be critical as they decide the amount of “freedom” that a network has. A network with too little freedom is not capable of learning the training data. A network with too much freedom will simply remember the training data and not generalize properly, a condition known as *overtraining*.

There are many different training algorithms. Some *constructive* training algorithms automatically decide an appropriate network architecture from the training data. A common training algorithm for multi-layer perceptron networks is back-propagation, which is very slow. Back-propagation uses a gradient descent method on a theoretical error surface. This multi-dimensional error surface gives the sum-square error between the desired network results using the training data and the actual network results for every possible setting of the network parameters. At any given setting of the parameters, i.e., any point on the error surface, the gradient of the error surface is calculated and a hop is taken in the direction of the steepest descent. This then gives a new setting of the network parameters, and the process is repeated until a minimum is reached. Despite many improvements to back-propagation, it still remains slow compared to other more modern network architectures and training algorithms.

The type of network that will be used in this paper is called a radial basis function (RBF), network, which may be trained significantly faster than a multi-layer perceptron with back-propagation. The architecture of a RBF network is shown in Figure 2. RBF networks consist of two layers, the first containing RBF neurons and the second containing linear

perceptrons. The behaviour of RBF neurons is different to that of perceptrons: the response characteristic of an RBF neuron is

$$f = \phi(\|\{\mathbf{x}\} - \{\mathbf{c}\}\|, \beta), \quad (2)$$

where $\{\mathbf{x}\}$ is the value of the input vector, $\{\mathbf{c}\}$ is the *centre* of the neuron, β is the *spread constant* of the neuron, $\|\cdot\|$ represents the Euclidean norm (2-norm), ϕ is the radial basis function of the neuron, and f is the output of the neuron.

Typical radial basis functions include the Gaussian function

$$\phi(v, \beta) = \exp(-v^2/\beta^2) \quad (3)$$

and the thin plate spline function

$$\phi(v, \beta) = v^2 \log(v) \quad (4)$$

(this function does not require a spread constant).

The overall response characteristic of an RBF network is given by

$$f_j = b_j + \sum_{i=1}^n W_{ij} \phi(\|\{\mathbf{x}\} - \{\mathbf{c}_i\}\|, \beta) \quad (5)$$

where $\{\mathbf{x}\}$ is the value of the input vector, \mathbf{c}_i is the centre of the i th neuron, β is the spread constant of the network (the same for every neuron), $\|\cdot\|$ represents the Euclidean norm (2-norm), ϕ is the radial basis function of the network (the same for every neuron), f_j is the output of the j th linear neuron (j th network output), b_j is the bias of the j th linear neuron, and W_{ij} is the weight between the i th RBF neuron and the j th linear neuron.

A radial basis neuron using the Gaussian transfer function can be visualized as having an “active fuzzy hypersphere” surrounding its centre. If an input vector lies within this hypersphere, then the output from the neuron will be high; otherwise, the output will be low. The hypersphere is said to be fuzzy because there is no absolute cut-off radius between high outputs and low outputs.

To train an RBF network, it is necessary to decide how many RBF neurons should be in the first layer. The number of linear neurons in the second layer is set by the length of the output vectors; i.e., there are n_o linear neurons. It is then necessary for the training algorithm to set every network parameter, i.e., every RBF centre, all the weights and biases of the linear neurons and the spread constant.

The centres of an RBF network should be spread out over the range of the input space occupied by the input vectors. A common method of positioning the centres is to choose a subset of the input vectors as the centres. This guarantees that the centres will be in the region of the input data, but care must be taken to ensure that all of the input data is covered. Typically, an adequate RBF network requires many fewer RBF neurons (and hence centres) than there are input vectors available. If too many centres are used, then the ability of the network to generalize is reduced.

The spread constant should give a measure of the average Euclidean distance between any two input vectors. The Gaussian function will be used in this paper, so a spread constant is required. The method of calculating the spread constant used in this paper is first to calculate the distance between every pair of input vectors and then to choose the median distance as the spread constant. Once the centres and spread constants of an RBF network are fixed, the weights and biases can be found by using a simple least squares algorithm [9].

The actual training algorithm used to select which input vectors are used as centres is the Orthogonal Least Squares (OLS) algorithm [10, 11]. This is a constructive algorithm

that generates a very compact subset of the input vectors as the RBF centres. It works by adding an RBF neuron at each iteration and finding the best input vector to use as the new RBF centre. At each step, the error between the network results acting on the input vectors and the target output vectors is calculated. When this error drops below a previously selected threshold, the training is terminated. This error threshold is the only parameter that needs to be set by the user for training an RBF neural network.

3. MODEL UPDATING USING NEURAL NETWORKS

To update a dynamic FE model, it is first necessary to obtain experimental data from the physical model. The experimental data will be in the form of frequency response functions (FRFs). However, even for small models the FRFs contain too many data points to use realistically with neural networks because of “*the curse of dimensionality*”, as discussed by Bishop [7]. As the dimension of the input vector to the neural net increases, the number of training data vectors required for adequate network generalization also increases, often exponentially. The FRFs may contain tens of thousands of data points; adequate network generalization is clearly unlikely in these circumstances.

It is necessary to reduce the number of data points to manageable proportions. There is no clearly defined boundary at which the number of data points becomes manageable—it is perhaps a matter of judgement more than science. Data reduction could be achieved by simply discarding many of the FRF data points. Techniques such as sensitivity analysis [12] could be used to select frequency points that contain the most information. However, any gains in improved network generalization may be offset by the corresponding loss of information in the data.

An alternative method of data reduction is to work in the modal domain, using modal analysis of the FRFs to derive the mode shapes and natural frequencies of the structure [13]. The number of data points is typically reduced by several orders of magnitude by modal analysis. Additionally, no loss of information occurs, although errors may be introduced to the data by the process of modal analysis. For this paper, the experimental data will be prepared by using modal analysis on the FRFs. After modal analysis has been performed on the experimental FRFs, the resulting mode shapes and natural frequencies are assembled into an experimental vector, $\{\mathbf{E}\}$. The particular order of mode shapes and frequencies in $\{\mathbf{E}\}$ is not important, but it must be recorded.

An outline of the model updating strategy considered here will now be given. The key stages will be considered in more detail below this outline.

The initial stage of the model updating procedure consists of generating a set of training data vectors from the original FE model. Each training vector is generated by altering the FE model in a prescribed manner (discussed below). Each input vector consists of the mode shapes and natural frequencies of the altered FE model. Each output vector consists of the changes made to the model to generate the input vector. Thus, the training set shows how the modal properties of the FE model vary as the model varies.

A neural network is then trained using this data. Ideally, when this trained network is shown a set of modal data, it will be capable of deciding what changes to the FE model would produce this modal data. The experimental vector $\{\mathbf{E}\}$ is then shown to the trained neural network. The results from the network will be a set of changes to the original FE model. This gives an updated FE model that can be updated again using the same method; i.e., an iterative technique is used. It is hoped that the FE models converge, giving the final updated FE model.

The iterated model updating procedure is summarized as follows: (1) generate some training data from the current FE model; (2) train an RBF neural network using this

training data and the OLS algorithm; (3) use the trained net with the experimental data to obtain FE model adjustments; (4) apply the adjustments to the FE model, generating a new FE model; and (5) repeat stages (1)–(4) until the FE model converges.

Stage 1 is the most important stage of the updating process, because the success of the networks depends on the quality of the training data. The remainder of this section will consist of a detailed discussion of this stage.

Three key decisions are required before training data can be generated. Two of these decisions relate to the changes that are made to the FE model to generate different modal data sets. The third decision relates to the modal data itself. The training data can be automatically generated from the FE model once these decisions have been made. These decisions are as follows: (i) What updating parameters (or p -values) are to be used? (ii) What system of altering these p -values is to be used? (iii) What modal information is to be shown to the net for each p -value alteration?

(i) *Choice of updating parameters (p -values)*. The p -values are the selected FE model parameters that will be updated. Each p -value has a *location* in the model and a *value* that represents the adjustment to the FE model parameter. For example, the p -value p_4 may have the location “Cross-sectional area of element number 12” and the value 1.2, corresponding to a 20% increase. Before updating can begin, it is necessary to choose the number of p -values (n_p) and the location of each p -value. The p -values can be any combination of numerical parameters of the FE model, such as: (1) individual terms of the elemental mass and stiffness matrices; (2) individual elemental parameters, such as Young’s modulus, density, etc. (and possibly the nodal co-ordinates); and (3) super-elemental parameters, such as the density of a group of elements.

There are two rules that should be taken into consideration when choosing p -values for this updating technique: uniqueness and sufficiency.

Uniqueness. The modal results of a given alteration to the chosen p -values should be unique; i.e., it should not be possible for different settings of the p -values to produce the same modal results. If the p -values are not unique, then the networks are effectively given an impossible task, for which there is more than one “correct” answer. The technique will converge to an answer, but it may not be the desired result.

Sufficiency. The choice of p -values should be sufficiently broad to cover the actual parameter changes that are necessary to update the model successfully. If too few or the wrong p -values are used, then the network iteration results will not converge. This is because no settings of the chosen p -values are capable of matching the experimental results.

Choosing individual terms of the mass and stiffness matrices as p -values may have unforeseen effects on the results and should be avoided if possible. Additionally, the number of p -values should be kept to a minimum to reduce the amount of computational effort required. Each additional p -value will result in a significant number of extra eigensolutions of the FE model to be performed.

A sensible choice of p -values for a small structure is the individual elemental parameters of every element, such as Young’s modulus, the cross-sectional area, etc. Many previous updating methods have used p -values consisting of the elemental Young’s moduli and densities. (Note that an alteration to the Young’s modulus of an element is equivalent to factoring the elemental stiffness matrix, and an alteration to the density is equivalent to factoring the elemental mass matrix.)

Once the p -values have been chosen, it is necessary to create the training data. A single training data set is generated by first applying a given alteration to the p -values. The altered FE model is then constructed and its modal solution generated. Some of the data contained in this modal solution will become the input vector. The alterations that were applied to

the p -values will become the output vector. The aim is to produce a network that is capable of deciding what alterations to the p -values produced the modal data that it was shown.

(ii) *Choice of p -value alterations.* The system of altering p -values should aim to show the network the modal results of adjusting any combination of p -values. Obviously, it is impossible to generate every combination of p -value alterations, so a good cross-section of possible alterations is required. It is hoped that the network will generalize, allowing it to recognize the results of p -value alterations that it has not previously seen.

One strategy is to vary each p -value in turn, with all the other p -values set to one (i.e., unchanged). A typical scheme would be to set each p -value to the values $\{0.4, 0.7, 0.9, 1.1, 1.3, 1.6\}$ in turn, while the others are unchanged. This scheme would generate six training vectors for each p -value, i.e., $6n_p$ training vectors, and ensure that the net can see the effect of adjusting every p -value. However, it does not show the effects of altering multiple p -values, which is likely to happen for real structures. To avoid this problem, the data set can be augmented by a given number (say 50) of training vectors resulting from adjusting a random selection (more than one) of the p -values by a random amount. This combined strategy covers a good cross-section of all possible p -value alterations.

(iii) *Choice of modal information to be shown to the nets.* For each individual alteration to the p -values, an analytical modal solution will be generated from equation (6); i.e., $[\lambda_r]$ and $[\phi]$ are found by using a generalized eigensolver on $[M]$ and $[K]$:

$$[\mathbf{K}]\{\phi\} = \lambda[\mathbf{M}]\{\phi\}. \quad (6)$$

The full modal solution will contain N eigenvalues and N corresponding mass-normalized mode shapes, where N is the number of DOFs of the FE model. It is now necessary to decide what information from the modal solution is to be included in the input training vector. The decision will be influenced by the following four factors: (1) it is desirable to include as much information as possible to allow the nets to generalize successfully; (2) there will, in general, be significantly fewer modes and DOFs present in the experimental modal data than in the analytical modal data; (3) the very highest analytical modes do not contain useful information; and (4) some of the experimental modes may be known to be inaccurate—for example, close modes of highly damped structures are hard to extract.

Hence only those modes and DOFs present in the experimental data should be shown to the net. The eigenvalues of the chosen modes should also be shown to the net to avoid uniqueness problems; e.g., the density of every element being adjusted by the same factor only alters the eigenvalues. Note that the eigenvalues should be suitably scaled down so that they do not dominate the analytical data vector. The chosen analytical data is stored in the same order as the experimental vector $\{E\}$. The remaining analytical information is ignored.

Note that when generating input vectors for the training data it is important that the modes are ordered in the same way as the experimental vector, and that they are normalized in the same fashion. The modal assurance criterion [14] may be used to re-order the analytical modes and the modal scale factor [14] to invert any modes if necessary.

3.1. COMMENTS

While updating, the FE model iterations will either converge on updated p -values or fail to converge (diverge). Note that if noise is present in the experimental data, then the p -values will only converge to a few significant figures. If the experimental data are noise free, then the p -values will eventually converge to the exact answer. If the p -values diverge, then this shows one of four possibilities, namely: (1) the choice of p -values was too narrow,



Figure 3. The direction of DOFs of a beam element.

i.e., the sufficiency rule was broken—if this was the case, then the updating procedure may be repeated with more p -values; (2) the choice of p -value alterations was too narrow—if this was the case, then the updating procedure may be repeated with a greater range of p -value alterations; (3) the experimental results were too noisy, and the useful information has been lost—if this was the case, then the updating technique will always fail; or (4) the FE model was too distant from the actual structure—if this was the case, then the updating technique will always fail.

Therefore, if divergence occurs, then the updating may be repeated with either more p -values or more p -value alterations, or both. If the updating repeatedly diverges, then it may not be possible to update the model with the experimental data.

4. SIMULATED UPDATING TESTS

A simulated test of the updating technique was performed using a small model; namely, a ten-element two-dimensional cantilevered beam as shown in Figures 3 and 4. The experimental data were generated by altering the chosen p -values of the FE model and deriving the modal solution. Some of the resulting modal data were entered into the simulated experimental vector $\{\mathbf{E}\}$ (see below). The original FE model was updated using this noise-free simulated experimental data.

The noise resistant capability of the updating technique was then tested by adding Gaussian noise to the previous simulated experimental data, and updating the original FE model using this noise data. However, it was felt that the Gaussian noise model does not adequately represent the type of errors that may be present in measured modal data. Consequently, a third set of updates were performed using a more realistic noise model. This noise model consisted of generating analytical FRFs from the FE model, applying noise to these FRFs and then using experimental modal analysis (EMA) techniques to generate the noisy modal data. Thus, the effects of both noise on the FRFs and errors introduced by the EMA are considered.

The p -value locations chosen for this model were the elemental Young's moduli and densities. Thus there were 20 p -values, p_1 to p_{20} . The first ten p -value locations were the elemental Young's moduli and the next ten were the elemental densities; so that p_{13} , for example, refers to the density of the third beam element.

The experimental data were generated by applying the p -values shown in Table 1 and deriving the modal solution. Taking account of experimental limitations, it was assumed

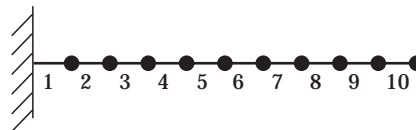


Figure 4. The ten-element two-dimensional cantilevered beam.

TABLE 1
The p -values for the simulated experimental data

p -value number	Value	p -value number	Value
1	1.00	11	1.23
2	1.04	12	1.23
3	1.06	13	1.23
4	1.07	14	1.23
5	1.10	15	1.23
6	1.18	16	1.23
7	1.00	17	1.10
8	0.94	18	1.00
9	0.80	19	0.89
10	0.75	20	0.87

that only the translational DOFs could be measured, so the rotational DOFs of the modal solution were ignored. This FE model has distinct extensional and bending modes. It was assumed that only the first three bending and first two extensional modes could be accurately measured, so all other modes were ignored. The remaining information (five modes, each consisting of ten DOFs and one natural frequency) was placed in a vector (of length 55). This was the simulated experimental vector $\{\mathbf{E}\}$.

The p -value alteration scheme for the generation of training data for each iteration was a two-part scheme. The first part consisted of setting each p -value in turn to the values $\{1.6, 1.3, 1.12, 0.88, 0.7, 0.4\}$ and setting all the other p -values to 1. This generated 120 training data vectors. The second part consisted of generating 50 training vectors by altering an arbitrary “line” of p -values by a random bounded factor; e.g., by setting the p -values p_4, p_5, \dots, p_9 all to 1.23 and keeping the remaining p -values set to 1. These additional training vectors show the effects of altering multiple p -values. Note that in all of the updating cases, the neural network has not “seen” the exact solution during training.

The choice of modal information to be shown to the nets was the same as the experimental data available; i.e., show the translational DOFs of the first three bending and first two extensional modes.

Three different updating cases were considered, as mentioned previously. The first update (case (1)) was performed with noise free data. The second update (case (2)) consisted of generating a noisy vector $\{\mathbf{E}'\}$ from the simulated experimental vector $\{\mathbf{E}\}$. The noise model applied to generate $\{\mathbf{E}'\}$ was 1% proportional Gaussian noise.

The third updating case consisted of generating FRFs from the analytical FE model, applying proportional Gaussian noise to the FRFs and then using experimental modal analysis (EMA) to recover the mode shapes and natural frequencies. Note that an undamped FE model has real FRFs, and EMA techniques can only be applied to complex FRFs. Thus, it was necessary to include a 1% proportional hysteretic damping model in the FE model for the purposes of generating the FRFs.

Four different levels of proportional Gaussian noise were applied to the real and complex parts of the generated FRFs. These levels were 0%, 1%, 5% and 10%. The 0% level shows the effect of the EMA on the update. The other levels show the additional effects of noise contamination. The modal analysis was performed using the GLOBAL-M method from the ICATS package [15]. The effect of the 5% noise model on a few entries of the simulated experimental vector is shown in Table 2.

TABLE 2

Part of the simulated experimental vectors (first extensional mode), case (3)—5% noise

Clean experimental vector, $\{\mathbf{E}\}$	Noisy experimental vector, $\{\mathbf{E}'\}$
0.9761	0.9783
1.8862	1.8449
2.7250	2.6441
3.4781	3.5256
4.1143	3.9108
4.6011	4.5309
5.0422	5.0026
5.3718	5.3238
5.6015	5.7532
5.6833	5.7441

5. DISCUSSION

5.1. RESULTS

For the noise-free case, the p -values quickly converged to near to the correct values (see Figure 5). The results are remarkably good, especially for the p -values that represent changes to the stiffness at the tip of the beam (p -values 8–10), which is a region of low strain energy, and is thus hard to update. Since the p -values represent a correction to the

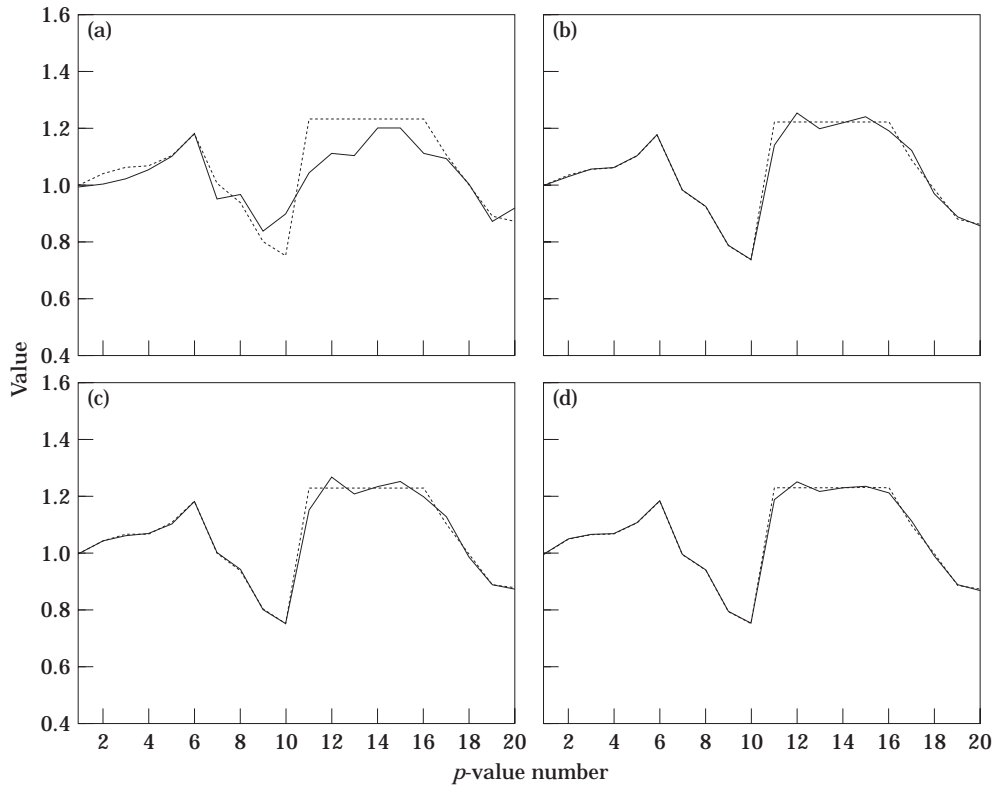


Figure 5. Updating results using noise-free simulated data—case (1). —, Results; - - -, target p -values. (a) After one iteration; (b) after three iterations; (c) after five iterations; (d) after eight iterations.

TABLE 3

Comparison of the modal properties of the models, no noise (case (1))

	Natural frequencies (Hz)			MAC values	
	Original FE	Updated FE	Experimental	Original FE	Updated FE
First bending	64.3	66.7	66.7	0.9999	1.0000
Second bending	402.7	393.6	393.6	0.9852	1.0000
Third bending	1126.9	1070.3	1070.5	0.9895	1.0000
Fourth bending	2206.7	2090.8	2089.8	0.9903	1.0000
First extension	4314.5	4333.1	4333.1	0.9996	1.0000
Second extension	13050.4	12484.0	12484.2	0.9771	1.0000
Third extension	22108.4	20669.6	20665.9	0.9915	1.0000

spatial FE model, this update has actually improved modes and natural frequencies that the networks did not use. This is demonstrated by the improvement in the fourth bending and third extensional modes, as shown in Table 3.

For case 2, with noise applied directly to the modal data, convergence occurred after six iterations. The mean percentage error for the p -values associated with the Young's moduli was 3.45% (see Figure 6). The density p -values were not as accurate; the mean percentage error was 7.32%. However, the density p -values tended to oscillate around the

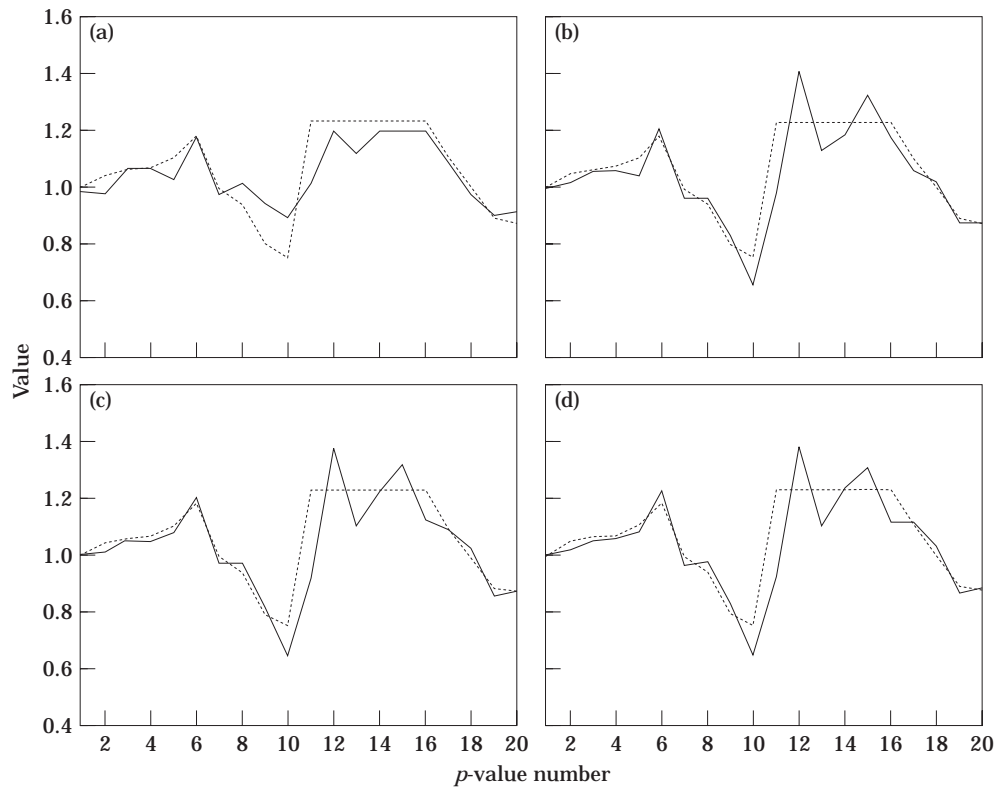


Figure 6. Updating results using simulated data with biased noise—case (2). —, Results; ----, target p -values. (a) After one iteration; (b) after three iterations; (c) after five iterations; (d) after six iterations.

TABLE 4

Comparison of the modal properties of the models, biased noise (case (2))

	Natural frequencies (Hz)			MAC values	
	Original FE	Updated FE	Experimental	Original FE	Updated FE
First bending	64.3	66.5	66.7	0.9999	1.0000
Second bending	402.7	394.1	393.6	0.9852	1.0000
Third bending	1126.9	1073.4	1070.5	0.9895	1.0000
Fourth bending	2206.7	2094.1	2089.8	0.9903	0.9990
First extension	4314.5	4324.1	4333.1	0.9996	1.0000
Second extension	13050.4	12513.4	12484.2	0.9771	0.9999
Third extension	22108.4	20805.1	20665.9	0.9915	0.9994

correct results, so the underlying trend of the graph seems better than the figure of 7.32% may suggest. The modal properties of the model were significantly improved, and the unseen modes were again improved (see Table 4).

In Figure 7 are shown the p -value results after six iterations of the updating procedure for the four different noise values considered in case (3). In each case, the results for the stiffness p -values (p_1 to p_{10}) are more accurate than the results for the mass p -values (p_{11}

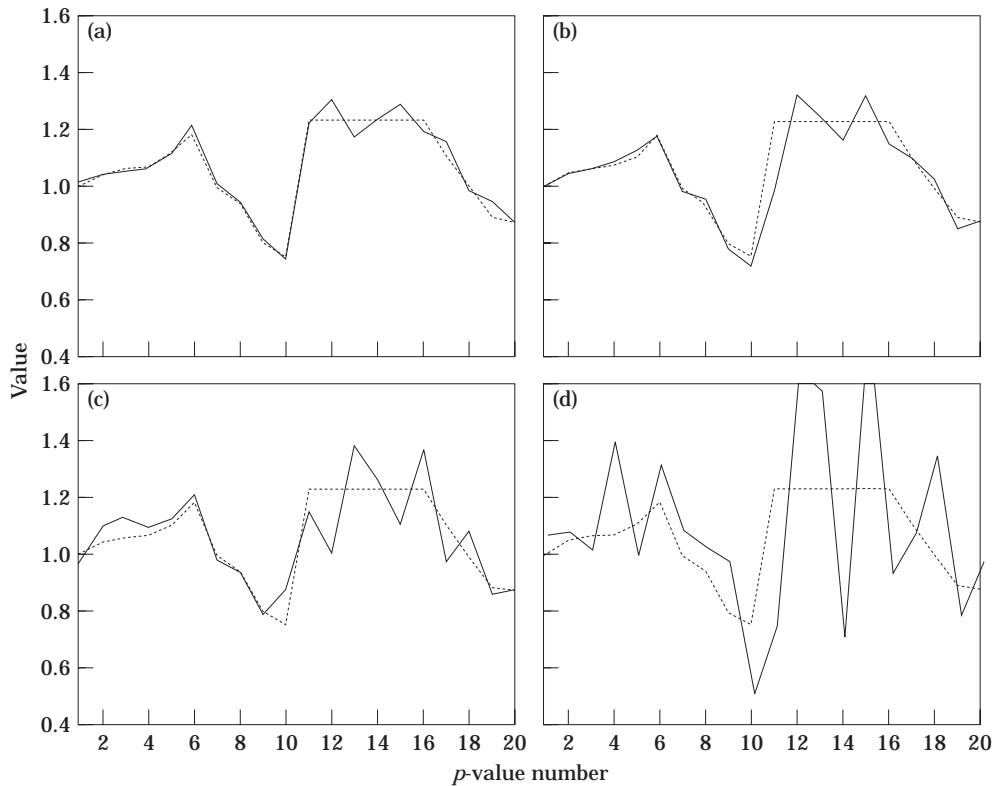


Figure 7. Updating results after six iterations using simulated data with Gaussian noise applied to FRFs and experimental modal analysis—case (3). —, Results; - - -, target p -values. (a) 0% noise; (b) 1% noise; (c) 5% noise; (d) 10% noise.

TABLE 5

Comparison of the modal properties of the models, 1% Gaussian noise applied to FRFs and experimental modal analysis (case (3))

	Natural frequencies (Hz)			MAC values	
	Original FE	Updated FE	Experimental	Original FE	Updated FE
First bending	64.3	66.9	66.7	0.9999	1.0000
Second bending	402.7	394.3	393.6	0.9851	1.0000
Third bending	1126.9	1068.5	1070.5	0.9891	1.0000
Fourth bending	2206.7	2074.6	2089.7	0.9899	0.9998
First extension	4314.5	4343.5	4333.3	0.9996	1.0000
Second extension	13050.4	12470.7	12484.3	0.9768	1.0000
Third extension	22108.4	20701.4	20666.8	0.9916	0.9997

to p_{20}). The 0% noise case shows that the errors introduced by experimental modal analysis on noise-free FRFs did not significantly degrade the updating procedure. This is encouraging, since the value of the modal domain for model updating purposes would have been called into question if EMA on noise-free data significantly degraded the results.

The results show clear degradation in quality as the amount of Gaussian noise applied to the FRFs increases. The update failed at the 10% noise level, since the required information has been lost from the simulated experimental data results. The 5% noise level shows oscillations around the correct values for the mass p -values. This is a phenomenon that has been reported in many other updating methods [16]. The modal properties of the original and updated models for the 1% noise case are shown in Table 5. The modal properties of the model were again improved.

These results show that the updating technique has worked well, even in the presence of a realistic noise model.

5.2. ADVANTAGES

This updating method has several advantages over previous updating methods. The most significant advantage is resistance to noise, which is an essential requirement for any successful updating method.

In many other updating methods, a key problem is that of co-ordinate incompatibility, namely, it is not possible to measure experimentally as many degrees of freedom and modes as the FE model contains. Consequently, it is necessary either to expand the measured data [17] or to reduce the FE model, usually by using Guyan reduction [18]. Both of these approaches introduce errors and are unsatisfactory. However, with this neural network updating method, the problem of co-ordinate incompatibility is neatly side-stepped. The neural networks implicitly reconcile the analytical and experimental models, instead of the user explicitly adjusting the data.

Another advantage of this method is that there is no prior limit on what the updating parameters should be. Any set of parameters may be chosen, from very low level to very high level parameters, and an appropriate training data set generated. As demonstrated, an improvement to the spatial model may also improve unseen modes.

Since the data shown to the nets are manually chosen, spurious modes can be ignored. Possible spurious modes include close modes of highly damped structures, which are difficult to extract, or modes of the support structure holding the physical model.

5.3. DISADVANTAGES

The main disadvantage of this method is that it is computationally expensive, because an eigensolution is required to generate every training vector. The other disadvantage is that extracting the modes from the FRFs introduces errors into the data [13], which ideally should be avoided. Using FRF data directly would avoid this problem but at the expense of introducing new problems, the main one being how to deal with the very large number of data points.

This method will fail if an FE model has repeated modes, because the analytical mode shapes will not be unique to within a scale factor, and experimental modal analysis techniques may fail. However, this situation can usually be avoided by making small adjustments to the FE model to remove symmetries.

6. CONCLUDING REMARKS

A new method of FE model updating using neural networks has been successfully demonstrated using simulated data. A strong advantage of this technique is its ability to withstand the presence of noise in experimental data. A secondary advantage is the avoidance of the common problem of co-ordinate incompleteness; i.e., the neural network updating method is capable of working with a limited number of experimentally measured DOFs and modes. The main drawback is that this method is computationally expensive. However, it would seem that there is significant potential for this model updating method to work with practical structures.

ACKNOWLEDGMENTS

The authors wish to thank the Engineering and Physical Sciences Research Council for financial support and assistance.

REFERENCES

1. D. J. EWINS and M. IMREGUN 1986 *Shock and Vibration Bulletin* **56**, 59–90. The reliability of computational dynamic response prediction (DYNAS).
2. M. BARUCH 1978 *American Institute of Aeronautics and Astronautics Journal* **16**, 1208–1210. Optimization procedure to correct stiffness and flexibility matrices using vibration tests.
3. A. BERMAN and E. J. NAGY 1983 *American Institute of Aeronautics and Astronautics Journal* **21**, 1168–1173. Improvement of a large analytical model using test data.
4. R. M. LIN and D. J. EWINS 1990 *Proceedings of 15th International Seminar on Modal Analysis*. Model updating using FRF data.
5. M. SABOURIN and A. MITICHE 1992 *Neural Networks* **5**, 843–852. Optical character recognition by a neural network.
5. J. T. CONNOR, R. D. MARTIN and L. E. ATLAS 1994 *IEEE Transactions on Neural Networks* **5**, 240–254. Recurrent neural networks and robust time-series prediction.
7. C. M. BISHOP 1995 *Neural Networks for Pattern Recognition*. Oxford: Oxford University Press.
8. F. ROSENBLATT 1962 *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Washington, D.C.: Spartan.
9. J. MOODY and C. J. DARKEN 1989 *Neural Computation* **1**, 281–294. Fast learning in networks of locally-tuned processing units.
10. S. CHEN, C. F. N. COWAN and P. M. GRANT 1991 *IEEE Transactions on Neural Networks* **2**, 302–309. Orthogonal least squares learning algorithm for radial basis function networks.
11. S. CHEN and S. A. BILLINGS 1992 *International Journal of Control* **56**, 319–346. Neural networks for nonlinear dynamic system modelling and identification.
12. C. FRITZEN and T. KIEFER 1992 *Proceedings of the 17th International Seminar on Modal Analysis*,

- 1581–1596. Localization and correction of errors in finite element models based on experimental data.
13. D. J. EWINS 1984 *Modal Testing: Theory and Practice*. Letchworth, Hertfordshire: Research Studies Press.
 14. R. J. ALLEMANG and D. L. BROWN 1982 *International Modal Analysis Conference I*, 110–116. A correlation coefficient for modal vector analysis.
 15. "ANON." 1982 *ICATS User Guide and Reference Manual*. London: Imperial College of Science, Technology and Medicine, Department of Mechanical Engineering.
 16. N. M. M. MAIA and J. M. M. SILVA 1997 *Theoretical and Experimental Modal Analysis*. Letchworth, Hertfordshire: Research Studies Press.
 17. H.-P. GYSIN 1990 *International Modal Analysis Conference VIII*, 195–204. Comparison of expansion methods for FE modeling error localization.
 18. R. J. GUYAN 1965 *American Institute of Aeronautics and Astronautics Journal* **3**, 380. Reduction of mass and stiffness matrices.

APPENDIX: NOTATION

DOF(s)	degree(s) of freedom
EMA	experimental modal analysis
FE	finite element
FRF(s)	frequency response function(s)
MAC	modal assurance criterion
MSF	modal scale factor
OLS	orthogonal least squares
RBF	radial basis function
$\ \cdot\ $	Euclidean norm (2-norm)
$\{\mathbf{x}\}$	network input vector
$\{\mathbf{c}_i\}$	centre of the i th radial basis function neuron
β	network spread constant
ϕ	transfer function/radial basis function
f_j	output of the j th linear neuron (j th network output)
b_j	bias of the j th linear neuron
W_{ij}	weight between the i th RBF neuron and the j th linear neuron
N	number of DOFs of FE model
n_T	number of training vectors
n_I	length of input vectors
n_O	length of output vectors (=number of linear neurons)
n_P	number of p -values
p_i	i th p -value
$\{\mathbf{E}\}$	simulated experimental vector
$\{\mathbf{E}'\}$	noisy simulated experimental vector
$[\mathbf{M}]$	global mass matrix
$[\mathbf{K}]$	global stiffness matrix
$[\mathbf{\Phi}]$	mass-normalised mode shape matrix
$[\lambda_r]$	eigenvalue matrix (diagonal)