The Applicability of Neural Network Systems for Structural Damage Diagnosis

by

Chatmongkol Peetathawatchai

Bachelor of Engineering, Chulalongkorn University, Thailand (1990)

Master of Science in Civil Engineering,
Massachusetts Institute of Technology (1992)

Submitted to the Department of Civil and Environmental Engineering
in partial fulfillment of the requirements for the degree of

Doctor of Science in Civil Engineering

at the

Massachusetts Institute of Technology

June 1996

Signature of Author..... .....
Department of Civil and Environmental Engineering
March 8, 1996

Certified by..... .....
Jerome J. Connor
Professor of Civil and Environmental Engineering
Thesis Supervisor

Accepted by.....
Joseph M. Sussman
Chairman, Departmental Committee on Graduate Studies

The Applicability of Neural Network Systems for Structural Damage Diagnosis

by

Chatmongkol Peetathawatchai

Submitted to the Department of Civil and Environmental Engineering
on March 8, 1996, in partial fulfillment of the
requirements for the degree of
Doctor of Science in Civil Engineering

Abstract

The primary objective is to explore the potential of neural networks for structural damage diagnosis. To achieve this objective, a general neural network architecture for structural damage diagnosis and a methodology for designing the components of this architecture are formulated and evaluated. The main components of the architecture include i) the physical system of interest and its model ii) the data preprocessing units and iii) neural networks that operate on the processed data and produce a prediction of the location and magnitude of damage. Important design issues are the choice of variables to be observed, the methodology for choosing the excitation and type of vibrational signature for the monitored structure, the actual configuration of neural networks, and their training algorithm. These design issues are first examined in detail for the case of single-point damage, and the evaluation is then extended to multiple-point damage. The diagnosis strategy is based on first identifying which substructures are damaged (global diagnosis), and then examining independently each individual damaged substructure to establish the location and extent of damage (local diagnosis). Global diagnosis requires a neural network for predicting which substructures are damaged. Local diagnosis employs two neural networks for predicting the locations and extent of damage at each location within a substructure. The total number of local diagnosis systems is equal to the number of substructures.

The evaluation phase is carried out with beam-type structures. Firstly, a single-point damage diagnosis system for a 2-span bending beam model is developed and evaluated. The second step considers a 4-span model with multiple-point damage. Numerical modeling of the structure and computation of the response are carried out with MATLAB. Damage is introduced by reducing the bending resistance at specific locations. Simulation studies are performed to evaluate the performance for different choices of excitation and types of input. Observation based on simulation studies indicates that the global and local approach considerably improves the practical feasibility from the other existing neural network-based approaches. Difficulties in developing good simulation models of large-scaled civil engineering structures, and the extensive amount of possible damage states that the structures involve, are the major practical problems, and may result in limited applicability of this approach in the field of civil structures.

Thesis Committee:
Prof. Jerome J. Connor Jr.
Prof. Daniele Veneziano
Prof. Eduardo Kausel

To Ahtorn and Phaga, my parents

For their love and support

Acknowledgements

I would like to express my great appreciation to my thesis supervisor, Professor Jerome J. Connor Jr., for his invaluable guidance and continuous encouragement throughout my graduate study at MIT.

I am also grateful to other members of my thesis committee, Professor Daniel Veneziano and Professor Eduardo Kausel, for their helpful advice.

My utmost gratitude are due to my family, whose love and support during these long years mean the most to me.

Table of Contents

Chapter 4

Chapter 5

Chapter 6

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Neural Network-Based Damage Diagnosis Approach

Although substantial research has been carried out on the topic of damage diagnosis of structures, the most reliable diagnostic techniques are still largely based on human expertise (DARPA, 1988). Since human-based structural damage diagnosis requires human experts to look for any signature that could identify damage of the structure, problems arise due to human errors and the scarcity of qualified human diagnosticians (Pham, 1995). There are also situations for which the human-based approach is not practical such as damage diagnosis of structures in space, or structures in hazardous environments. Recently, the use of Knowledge Based Systems (KBS) to support human judgment has been proposed (DARPA, 1988 and Garrett, 1992), but this technology is also limited. Developing a KBS is very time-consuming (DARPA, 1988). Moreover, KBS's are not very adaptable and robust, and are too slow to be operated in real time (Pham, 1995). Therefore, a better non-human based approach is still needed.

By definition, a neural network consists of a number of units called " processing elements" which are connected to and interact with each other. The activation of a network starts when there is input to any unit. The weighted summation of the inputs for a unit is passed through a function called "transfer function" and the output of this function is provided at the output connection, which can be connected to the input connection of

any other unit including itself. The activation process of each unit will continue to execute until there is no more input, or until the output converges to some value. More details about the architecture and operation of neural networks are presented in Chapter 2 and 3.

Considering the ability of artificial neural networks to approximate functions (Cybenko, 1988, 1989; Hornik, 1989; Dyn, 1991; Park, 1991), the pattern mapping aspect of structural damage diagnosis appears to be a promising application area for neural networks. If artificial neural networks can extract knowledge from remotely collected data in an effective way, they can be incorporated in a computer based diagnosis system that can complement existing human diagnosis approaches.

Neural network-based damage diagnosis approach has been studied recently by several researchers. Elkordy (1992, 1993), Rehak (1992), and Liu (1995) determine changes in certain mode shapes of the damaged structure and use this information to estimate the location and level of damage. Use of the frequency-domain properties of the physical structure, referred to as the "vibrational signature" of the structure, transfers the problem to static pattern mapping between changes in mode shapes and types of damage. In this case, any type of network for function approximation (as discussed in Chapter 3) can be applied to solve the mapping problem.

Elkordy (1992, 1993) trained a multilayer network with analytically generated states of damage to diagnose damage states obtained experimentally from a series of shaking-table tests of a five-story frame. The physical and analytical models are defined in Fig 1.1. Damage states are simulated using a variety of smaller areas of bracing members in the first two stories. Tables 1.1 and 1.2 contain the damage states of the testing and training damage cases respectively. The performance of the neural network-based diagnosis system that is optimized for the analytical model was examined and found to be good for detecting a limited number of damage conditions of the real frame (see Table 1.1).

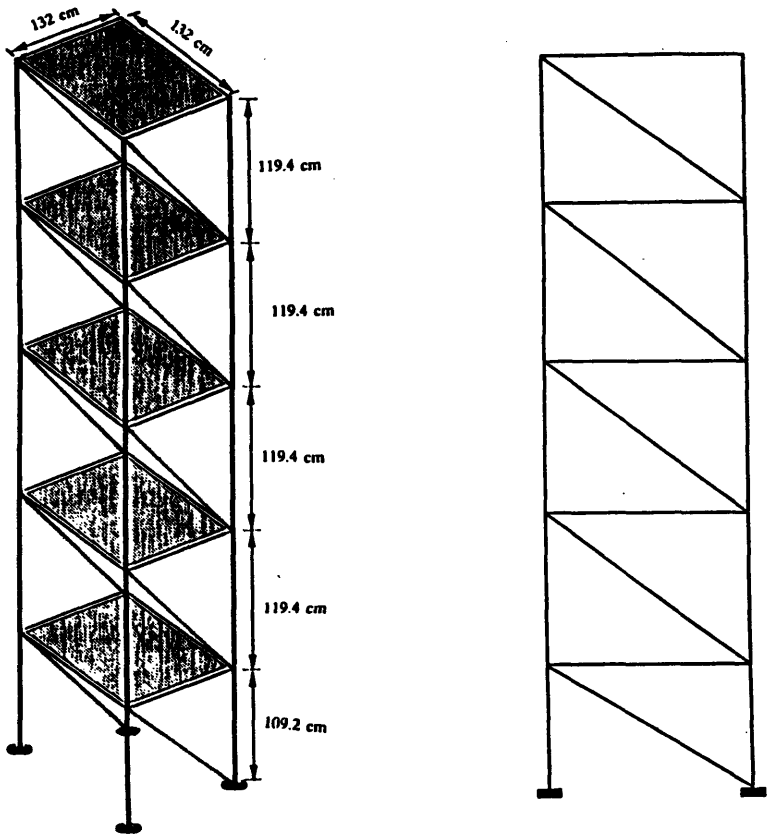


Figure 1.1: The physical and analytical model of the frame (Elkordy, 1992).

Case Number	Effective Bracing Area (cm^2)		Reduction of Bracing Area (%)	
	First Floor	Second Floor	First Floor	Second Floor
(a) First Floor Damage Class				
1	2.28	3.4	33	0
2	1.7	3.4	50	0
3	1.15	3.4	66	0
(b) Second Floor Damage Class				
4	3.4	2.28	0	33
5	3.4	1.7	0	50
6	3.4	1.15	0	66
(c) Combined Damage				
7	2.28	2.28	33	33
8	1.7	1.7	50	50
9	1.15	1.15	66	66
10	0.484	0.960	66	33

Table 1.1: Damage cases of the testing data set (Elkordy, 1992).

Case Number	Reduction of Bracing Area (%)		Target Diagnosis[a]	
	First Floor	Second Floor	First Floor	Second Floor
1	10	0	1	0
2	30	0	1	0
3	50	0	1	0
4	60	0	1	0
5	0	10	0	1
6	0	30	0	1
7	0	50	0	1
8	0	60	0	1
9	30	30	1	1
10	50	50	1	1
11	60	60	1	1

[a]Key: 1 = damage exists; 0 = no damage exists.

Table 1.2: Damage cases of the training data set (Elkordy, 1992).

Using mode shapes to identify damage is only one approach. In the mechanical engineering field, vibrational response spectrums have long been used to detect damage of rotating tools and machines. This approach has been followed by Wu et al (1992) and MacIntyre et al (1994). Wu utilized a multilayer network to detect changes in the response spectrum of the numerical model of a 3-story shear building, and to correlate these changes with corresponding damage states. The model is subjected to earthquake base acceleration, and the Fourier spectra of the computed relative acceleration time histories of the top floor are used in training the neural network. Damage is defined as a reduction of shear stiffness of a specific story. Only one story can be damaged at a time. The illustration of the model and data processing procedure are shown in Fig 1.2. Figure 1.3 shows the neural network employed for damage diagnosis. Tables 1.3 and 1.4 show the damage cases of the testing and training data sets respectively. The results indicate good performance of neural network in detecting damage of the model.

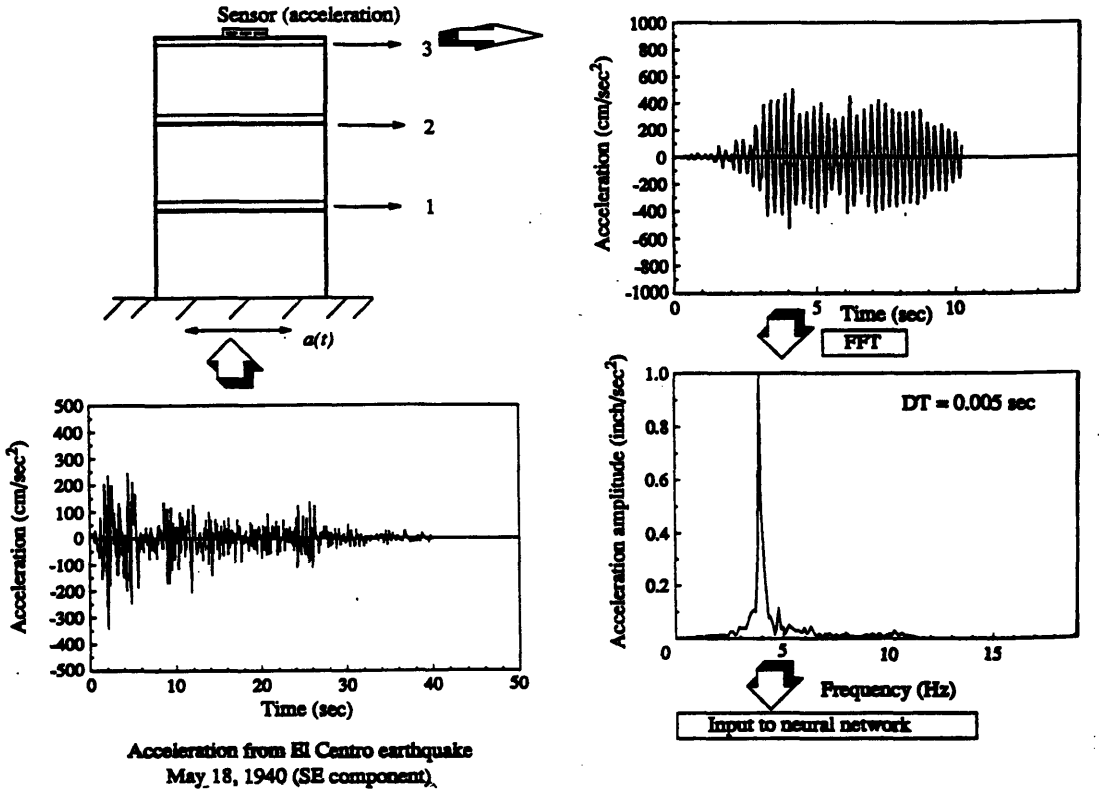


Figure 1.2: The model and data processing procedure (Wu, 1992).

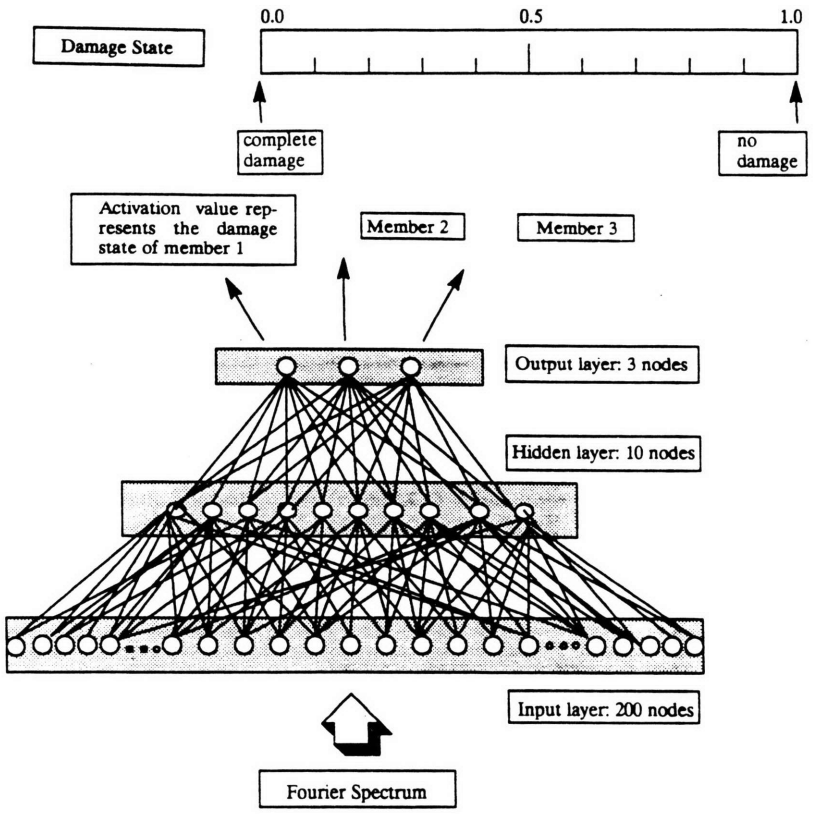


Figure 1.3: The multilayer network for damage diagnosis (Wu, 1992).

Case Number	Reduction of Shear Stiffness (%)		
	1st Story	2nd Story	3rd Story
1	0	0	0
2	50	0	0
3	75	0	0
4	0	50	0
5	0	75	0
6	0	0	50
7	0	0	75

Table 1.3: Damage cases of the testing data set (Wu, 1992).

Case Number	Reduction of Shear Stiffness (%)		
	1st Story	2nd Story	3rd Story
1	0	0	0
2	60	0	0
3	0	60	0
4	0	0	60

Table 1.4: Damage cases of the training data set (Wu, 1992).

There are also other works that involve utilizing other vibrational signatures. Szewczyk and Hajela (1992, 1994) model the damage as a reduction in the stiffness of structural elements that are associated with observed static displacements under prescribed loads. They performed simulation of a frame structure model with nine bending elements and 18 degrees of freedom (*X-Y* displacements and rotation at each node). A variation of multilayer feedforward network was utilized to detect the damaged frame element. Barai and Pandey (1995) carried out a study on damage detection in a bridge truss model using multilayer network with simulated damage states, each represented by one damaged truss member. Discretized time history response at various locations of the truss due to a single-wheel moving load is used as the vibrational signature.

All of the research studies mentioned have been restricted to narrowly defined application areas. Issues related to the architecture and design strategy for a general neural network-based damage diagnosis system have not been addressed. Also, the efforts have focused on very small-scale problems that involve very limited number of possible damage states, and it is not obvious that the methods can be scaled up to deal with larger-scale problems. In most cases, at a particular instant in time, damage is assumed to occur at a single location (single-point damage condition) in order to reduce the complexity of the problem (Wu, 1992; Szewczyk and Hajela, 1994; Barai and Pandey, 1995). The treatment of multiple-point damage has been dealt with in a very limited way (Elkordy, 1992, 1993), and the methodology for dealing with multiple-point damage has not been adequately developed.

Based on our review of the literature, the important research issues concerning the applicability of neural networks for damage diagnosis that need to be addressed are as follows:

- The appropriate general architecture of a neural network-based structural damage diagnosis system.

- A comprehensive design strategy for neural network-based diagnosis system.

- The detection of simultaneous damage sources.

- The scalability of neural network-based diagnosis systems to large structures.

- The error introduced by representing the actual physical structure with an ideological numerical model.

- The choice of data such as mode shapes, response spectrums, and other vibrational signatures as input for damage classification.

- The relationship between the data collection procedure and the performance of the diagnosis system.

1.2 Objective and Scope

The primary objective of this research is to explore the potential of neural networks for structural damage diagnosis. To achieve this objective, a general neural network architecture for structural damage diagnosis and a methodology for designing the components of the architecture are formulated and evaluated.

The main components of the general architecture include i) the physical system of interest and its model ii) the data preprocessing units and iii) a collection of neural networks that operate on the processed data and produce a prediction of the location and magnitude of damage. Important system design issues are the choice of variables to be observed, the methodology for choosing the excitation and type of vibrational signature

for the monitored structure, the actual configuration of the neural networks, and their training algorithm. These design issues are first examined in detail for the case of single-point damage condition, and the evaluation is then extended to the case of multiple-point damage.

The evaluation phase is carried out with beam-type structures. As a first step, a single-point damage diagnosis system for a highly-idealized 2-span bending beam model is developed and evaluated. Two choices of excitation and vibrational signature are considered. The first strategy applies ambient excitation to the numerical model of the structure, and then takes the mode shapes as input patterns for the neural networks which estimate the location and magnitude of damage. The second strategy applies a prespecified excitation to generate the model response, and employs the resulting response spectrum as input for the neural network system. The numerical modeling of the structure and the computation of the response are carried out with MATLAB. Damage is introduced by reducing the bending resistance at a specific location. Simulation studies are performed to evaluate the performance for the different choices of excitation and type of input.

The second step in the evaluation phase considers a 4-span bending beam model with multiple-point damage. The diagnosis strategy is based on first identifying which spans are damaged (global structural diagnosis), and then examining independently each individual damaged span to establish the location and extent of damage (local structural diagnosis). Global structural diagnosis requires at least a set of neural networks for predicting which spans are damaged. Local structural diagnosis of each substructure requires at least two sets of neural networks for predicting the locations and extent of damage. The number of local structural diagnosis systems depends on the number of substructures.

Two choices of excitation and vibrational signature are employed for global structural diagnosis. The first choice uses ambient excitation to generate the response of the model, and takes the corresponding mode shapes as the input pattern for the neural

networks. The second choice employs a prespecified excitation and uses the corresponding response spectrum as the input pattern.

Local structural diagnosis employs a prespecified excitation to create the response of each substructure, and uses the frequency transfer functions of the substructure as input for the neural networks that perform damage diagnosis on the substructure.

Based on the results of the evaluation studies, a design methodology for a comprehensive neural network system for structural damage diagnosis is formulated. Results of additional studies on the performance of various configurations of multilayer feedforward network, with back propagation learning algorithm, are utilized to establish guidelines for the choice of appropriate configurations for the individual neural networks contained within the general architecture. Feasibility tests are also performed to explore the practical applicability of this diagnosis approach.

Two types of neural networks, one based on the multilayer feedforward-back propagation learning model and the other on radial basis functions, are appropriate for pattern classification. This study employed only the multilayer feedforward networks for the simulation studies since the lack of a-priori knowledge of the damage pattern mapping problem do not suit Radial Basis Function Network. Data on the performance of both types of networks is included in this document for convenient reference.

1.3 Organization

In Chapter 2, the basic concepts and definitions of artificial neural networks are described, several well-known architectures and training algorithms are demonstrated, and the neural network applications in civil engineering are briefly reviewed. Neural network and other information processing approaches, such as Knowledge Based Systems, are compared. The relation between neural networks and approximation schemes are also discussed.

Chapter 3 contains a detailed treatment of neural networks for function approximation, an application area of considerable interest for civil engineering. Performance data are presented and discussed for both Multilayer Feedforward Networks and Radial Basis Function Networks. The optimum architectures and training strategies for a range of regression and classification problems are also investigated.

Feedforward neural networks, including Radial Basis Function Networks, are discussed in terms of probability and approximation theory in Chapter 4. Probabilistic models of feedforward networks with back propagation learning algorithms and radial basis function networks are investigated. This knowledge provides engineers with a different perspective of the theory of neural networks, and makes it easier to understand and develop a neural network-based system.

Chapter 5 is concerned with the application of neural networks to structural damage diagnosis. A general architecture of neural network for structural damage diagnosis is presented, and methodologies for designing the individual components of the system for both single-point damage and multiple-point damage are proposed.

An application of a neural network based damage diagnosis system to a 2-span beam with single-point damage is developed and evaluated in Chapter 6. Chapter 7 describes the case of multiple-point damage diagnosis for a 4-span beam. These investigations provide a general understanding of neural network based damage diagnosis and the difficulties involved in applying this approach to real structural problems. Chapter 8 summarizes these findings, and suggests strategies for overcoming some of these difficulties. The practical applicability of this diagnosis approach is also discussed, and further research topics are recommended.

Chapter 2

Foundation of Artificial Neural Networks

2.1 Background History

Work in the neural network field began about 50 years ago. The effort during this time period can be considered to have three distinct phases (DARPA, 1987): an early phase, a transition phase, an a resurgent phase.

The early work, (1940's-1960's), was concerned with fundamental concepts of neural networks such as Boolean logic (McCulloch and Pitts, 1943), synaptic learning rules (Hebb, 1949), single layer Perceptron (Rosenblatt, 1962), and associative memory (Steinbuch et al, 1963). The Perceptron generated immediate attention at that time because of its ability to classify a continuous-valued or binary-valued input vector into one of two classes. However, in the late 1960's, the work by Minsky and Papert pointed out that the Perceptron could not solve the "exclusive OR" class of problems, and this finding resulted in a substantial shift in research interest away from neural networks.

During the transition period (1960's-1980's), a small group of researchers continued to develop a variety of basic theories that strengthened the foundation of the field. Contributions include the Least Mean Square (LMS) algorithm (Widrow and Hoff, 1960), Cerebellum model (Albus, 1971), competitive learning (Von Der Malsburg, 1986), and Adaptive Resonance Theory (Grossberg, 1987).

Starting in early 1980's, there was a resurgence in interest for neural networks. This resurgence was driven by the contributions made during the transition period toward improving the understanding of the deficiencies of single-layer perceptron and extending the theoretical work to multilayer systems. The advances in computer technology in the 1980's also provided the computation power needed to deal with large-scale networks. Notable contributions include feature maps classifier (Kohonen, 1982), associative memory theory (Hopfield, 1982), Boltzman machine (Hinton and Sejnowski, 1986), and Back Propagation learning algorithm (Rumelhart et al, 1986). These topics are described in more detail later in this chapter.

2.2 Topological Classification of Neural Networks

The word "neural" is used because the inspiration for this kind of network came initially from the effort to model the operation of neurons in human brain. Artificial neural networks are classified according to their topology and the algorithm that provides their ability to learn. The topology of an artificial neural network defines the connection between the various processing elements contained in the network. The function of a network is determined by its connection topology and the weighting factors assigned to each connection. These weighting factors are adjusted by the learning algorithm during the training phase. Artificial neural networks are also called "Connectionist Models", or "Parallel Distributed Processing Models." For convenience, the simplified term "Neural Networks" is used throughout the remaining portion of this text.

Figure 2.1 shows a common processing element and its activation. A processing element has many input connections and combines, usually by a simple summation, the weighted values of these inputs. The summed input is processed by a transfer function which usually is a threshold-type function. The output of the transfer function is then passed on to the output connection of the element.

The output of a processing element can be passed on as input to any processing element, even to itself. Weights are used to designate the strength of the corresponding connections and are applied to the input signals prior to the summation process.

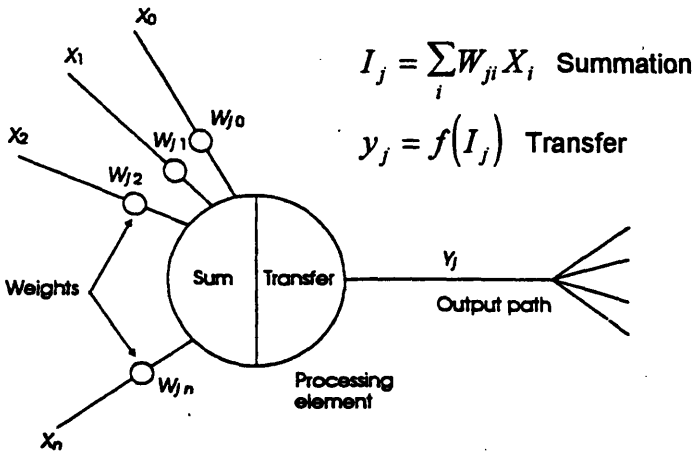
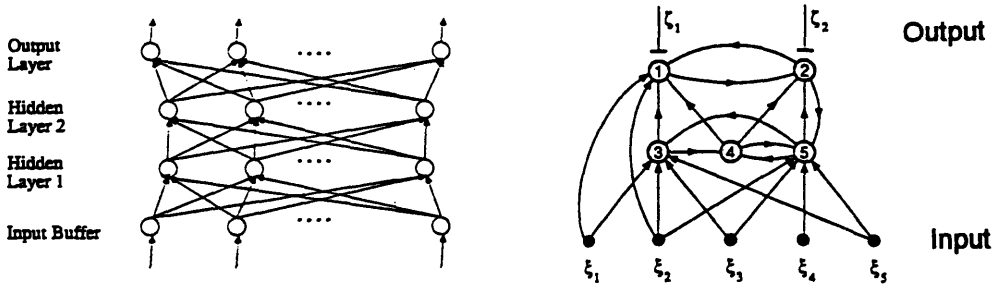


Figure 2.1: An example of a general processing element.

A neural network contains many "processing elements," or "units," connected and interacted to each other. Processing elements are usually clustered into groups called layers. Data is presented to the network through an input layer, and the response is stored in an output layer. Layers placed between the input and output layers are called hidden layers. When a network is not organized into layers, a processing element that receives input is considered to be an input unit. Similarly, a processing element that provides some output of the network is called an output unit. It is also possible that a processing element may act simultaneously as both input and output unit.

Based on topology, a neural network is classified as either feedforward or recurrent. Figure 2.2 illustrates these categories. Feedforward networks have their processing elements organized into layers. The first layer contains input units whose task is only to provide input patterns to the network. Next to the input layer are one or more hidden layers, followed by the output layer which displays the result of the computation.

In feedforward networks there is no connection from a unit to the other units in either previous layers or the same layer. Therefore, every unit provides information, or input, only to units in the following layer. Since the input layer has no role other than to provide input data, it is not included in the layer count. It follows that an N-layer network has N-1 hidden layers and an output layer.



a) Feedforward Network b) Recurrent Network

Figure 2.2: Feedforward network and recurrent network (Hertz, 1991).

Recurrent networks are networks that have connections "both ways" between a pair of units, and possibly even from a unit to itself. Many of these models do not include learning, but use a prespecified set of weights to perform some specific function. The network iterates over itself many cycles, until some convergence criterion is met, to produce an output. For example, the simplest recurrent network performs the following computation,

$$\underline{x}(k+1) = \underline{A}\ \underline{x}(k) \quad , \qquad (2.1)$$

where $\underline{x}(k)$ is the output vector at time step k and $\underline{A}$ is a weighting matrix. The convergence of this type of computation depends on certain properties of the weighting matrix.

2.3 Learning Algorithms

Learning, or training, is the process of adapting or modifying the connection weights and other parameters of specific networks. The nature of the learning process is based on how training data is verified. In general, there are 3 training categories; supervised, unsupervised, and self-supervised.

Supervised learning requires the presence of an external teacher and labeling of the data used to train the network. The teacher knows the correct response, and inputs an error signal when the network produces an incorrect response. The error signal then "teaches" the network the correct response by adjusting the network parameters according to the error. After a succession of learning trials, the network response becomes consistently correct. This is also called "reinforcement learning" or "learning with a critic".

Unsupervised Learning uses unlabeled training data and requires no external teacher. Only the information incorporated in the input data is used to adjust the network parameters. Data is presented to the network, which forms internal clusters that compress the input data into classification categories. This process is also called "self-organization."

Self-supervised Learning requires a network to monitor its performance internally, and generates an error signal which is then fed back to the network. The training process involves iterating until the correct response is obtained. Other descriptors such as "learning by doing", "learning by experiment", or "active learning" are also used to denote this approach.

Each learning method has one or more sub-algorithms that are employed to find the optimum set of network parameters required for a specific task. These sub-algorithms generally are traditional parameter optimization procedures such as least square minimization, gradient descent, or simulated annealing. In more complicated networks, several types of learning algorithms may be applied sequentially to improve the learning ability.

Back propagation learning, the most popular for multilayer feedforward network, is a supervised procedure that employs a gradient descent type method to update the weighting parameters. With a gradient descent procedure, one updates parameters as follows:

$$W_{n+1} = W_n - \lambda \ \nabla_w f. \qquad (2.2)$$

where W is the vector of weighting parameters, f is the function to minimize (usually called an "error function"), and λ is a parameter called "step size" or "learning coefficient." The advantages of this method are its simplicity and reliability. However, the method requires more computation than others, and also tends to lock in on local minima of the function surface rather than the global minima. The speed of convergence can be improved by varying the step size and including a momentum term. There are also improved gradient procedures such as Newton's method, Quasi-Newton, Conjugate gradient, and Stochastic Gradient Descent which is the stochastic version of the gradient method. The equation for Stochastic Gradient Descent has the following form

$$W_{n+1} = W_n - \lambda \ \left(\nabla_w f + S_n\right). \qquad (2.3)$$

where S is a sequence of random vectors with zero mean. Under certain assumptions, this sequence converges to a local minimum of f (Girosi, 1993, 1995).

Although the stochastic version of gradient method is more likely to avoid a local minima, convergence to the global minimum is conditioned on certain assumptions. For instance, a particular stochastic method, called "Simulated Annealing" (Kirkpatrick et al, 1983), will converge to the global minimum with infinite updating. Stochastic methods are theoretically interesting, but they require extensive computation power and consequently their training time is too long for large-scale applications on current hardware.

The tasks which neural networks have to perform and the availability of training data generally determine the appropriate type of training algorithm. In practice, there are many criteria that have to be considered. Speed of training and degree of accuracy is always a trade-off. Amount of available memory has to be considered when hardware is restricted. In some case, even the characteristics of the error function have to be taken into account. Having so many alternatives for the type of architecture and training algorithm, it is difficult to find the right combination for a specific application. In most cases, a number of candidate solution need to be evaluated in order to identify the best choice. This research provides guidelines for selecting the proper architecture and training method for function approximation. Details are presented in the next chapter.

2.4 Review of Types of Neural Networks

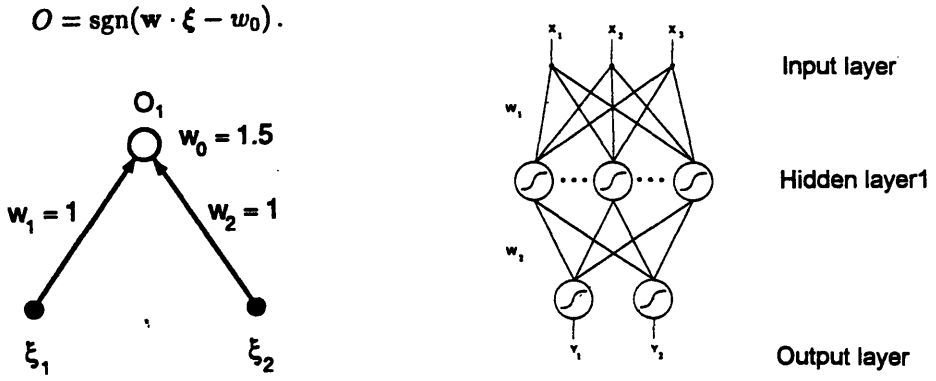
As mentioned earlier, neural networks are categorized by their architecture and learning algorithm. In this section some well-known networks are described in order to provide further background for the neural network field.

Single and multilayer perceptrons are the simplest types of feedforward neural network model. The single-layer perceptron was first introduced by Rosenblatt (1962). An example of the network is shown in Fig 2.3a. A processing element computes a weighted sum of input, subtracts a threshold, and passes the result through a nonlinear threshold function,

$$f(x) = 0\,,\ x \le 0 \quad \text{and}$$
$$f(x) = 1\,,\ x > 0\,. \qquad (2.4)$$

The two possible outputs correspond to the two different classes which can be recognized by the network. The single-layer perceptron can be used to classify a continuous-valued or binary-valued input vector into one of two classes. Training can be done with the Least

Mean Square (LMS) algorithm, which is a linear supervised training approach with guaranteed convergence. Minsky and Papert (1988) analyzed the single-layer perceptron and demonstrated that the network can only solve linearly separable problems like the exclusive AND problem, but cannot handle a nonlinearly separable problem such as the exclusive OR problem. A more detailed review of Rosenblatt's perceptron and Minsky and Papert's analysis can be found in (Minsky and Papert, 1988).



(a) Perceptron that implement AND. (b) One-hidden-layer Perceptron

Figure 2.3: Perceptron.

Multilayer perceptron is a feedforward network with one or more hidden layers. The transfer function of each processing element is the same as that of single-layer perceptron. Although multilayer perceptrons perform better in many aspects compared to single-layer perceptrons, especially in their ability to solve nonlinearly separable problems, they were not very popular prior to the mid 1980's because of the lack of an efficient training algorithm. The development of training algorithm called "back propagation" in the mid 1980's (Parker, 1985; Rumelhart et al, 1986 and Werbos, 1974) resulted in renewed interest in multilayer perceptrons. Back propagation is a supervised training procedure that, although convergence is not guaranteed, has been applied successfully to many problems such as spoken vowels classification (Huang et al, 1987 and Lippmann et al,

1987), speech recognizer (Waibel et al, 1987), and nonlinear signal processing (Arbib, 1987). More details on back propagation are given in Chapter 3.

The Hopfield network is a one-layer unsupervised training recurrent network with fully connected symmetrically weighted elements. Each unit functions as input and output unit. In the initial version, all parameters had to be prespecified. This limitation is removed in later versions where the parameters can be adjusted via gradient method such as Back Propagation Through Time (Rumelhart et al, 1986), or Recurrent Back Propagation. Given the input, the network iterates until it reaches a stable state (output from next iteration does not change from the previous one), and provides the output. This type of network can be used to solve pattern classification, associative memory, and optimization problems.

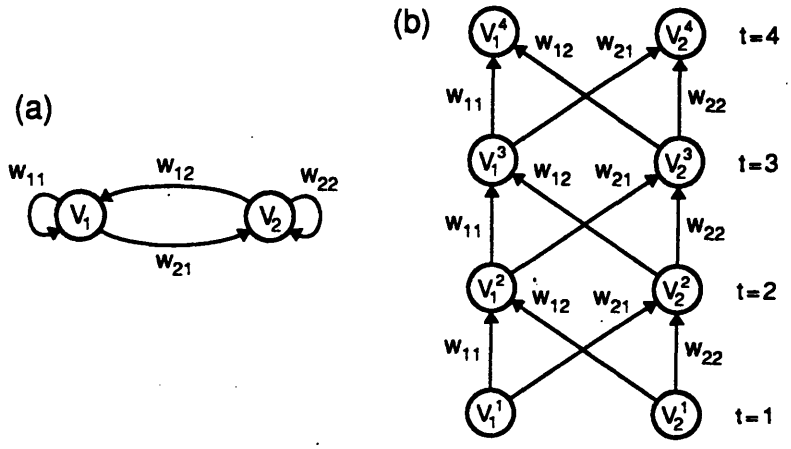


Figure 2.4 Hopfield Network (Hertz, 1991). (a) Network architecture.
(b) The activation of the network for 4 time steps.

The optimization networks developed by Hopfield tend to converge to local minima. The problem can be eliminated by adding a stochastic aspect to the training algorithm during the iteration. This approach led to the development of a type of network called "Boltzmann machine" (Ackley et al, 1985 and Hinton et al, 1986). The Boltzmann

machine training algorithm solved the credit assignment problem for the special case of recurrent networks with symmetrical connections and was demonstrated to be able to learn a number of difficult Boolean mappings. Due to the nature of stochastic parameter optimization, however, the training time of the Boltzmann machine is too long for most practical applications.

The Cerebellar Model Articulated Controller, or CMAC model (Albus, 1981), is an original model of the cerebellum, which is the part of the biological brain that performs motor control. The network adaptively generates complex nonlinear maps and is generally used in motor control problems such as robotic control (Kawato et al, 1987 and Miller et al, 1987). CMAC is actually an adaptive table look-up technique for representing complex, nonlinear functions over multi-dimensional, discrete input spaces. A diagram of the CMAC model is shown in Fig 2.5.

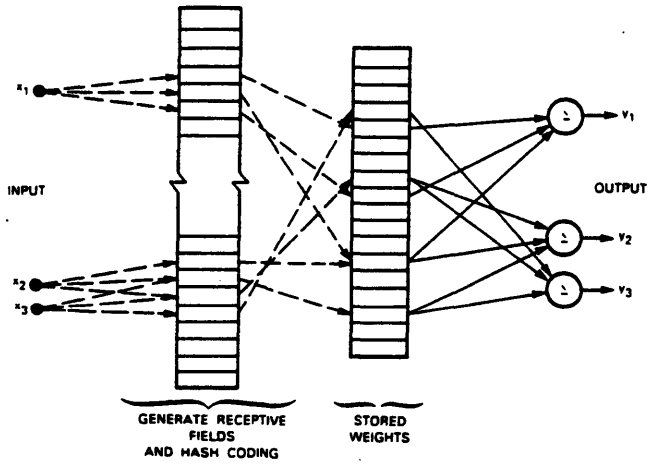


Figure 2.5: The CMAC model (Albus, 1981).

CMAC reduces the size of the look-up table through coding, provides for response generalization and interpolation through a distributed topographic representation of

inputs, and learns the nonlinear function through a supervised training process that keeps adjusting the content or weight of each address in the look-up table (DARPA, 1987).

The feature map classifier (Hampson et al, 1987 and Valiant ,1985), as shown Fig 2.6, is a hierarchical network that utilizes both unsupervised and supervised training. The lower part of the network classifies the input and is trained first using Kohonen's feature map algorithm (DARPA, 1987), a type of learning algorithm that does not require explicit tutoring of input-output correlations and perform unsupervised training based on the input data. The perceptron-like upper part is then trained using a supervised training algorithm. This approach is especially useful when the amount of unsupervised training data available is significantly greater than the quantity of supervised training data.

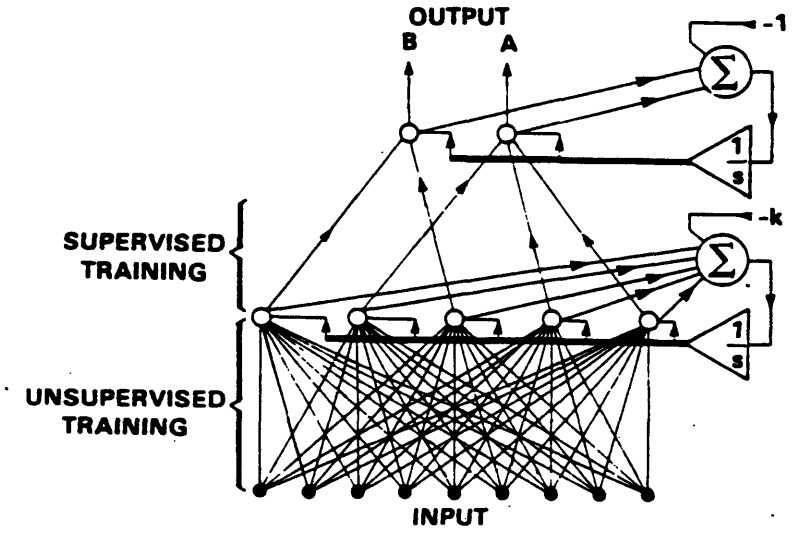


Figure 2.6: A Feature Map Classifier (Hampson, 1987).

Adaptive Resonance Theory (ART) networks (Carpenter and Grossberg, 1987) are complex nonlinear recurrent networks with an unsupervised training algorithm that

creates a new cluster by adjusting weights or adding a new internal node when an input pattern is sufficiently different from the stored patterns. "Sufficient difference" can be adjusted externally by a parameter called the "vigilance parameter". They are used mainly for pattern classification.

The Radial Basis Function Network (Broomhead and Lowe, 1988), or RBFN, is a 1-hidden-layer feedforward network with fixed nonlinear transformations in the hidden layer, and linear transformation in the output layer (see Fig 2.7). Radial basis functions are used as the transfer functions for the interior elements. The most popular choice of radial basis function is the Gaussian function

$$G_i(x) = e^{-\|x-x_i\|^2} . \qquad (2.5)$$

The output of RBFN is given by

$$y = \sum_{i=1}^{n} C_i G_i(\|x - x_i\|) . \qquad (2.6)$$

The type, the center points of the radial basis functions, and the weighting parameters need to be specified. Since the gradient of the output error function is linear to weighting parameters, the error function does not have local minima and the weights can be adjusted by a linear optimization procedure such as a least square approach. These procedures converge rapidly to the global minimum of the error surface. This aspect makes RBFN an attractive alternative to MLN which requires a stochastic optimization procedure since its error function has local minima. More detail of RBFN is presented in Chapter 3.

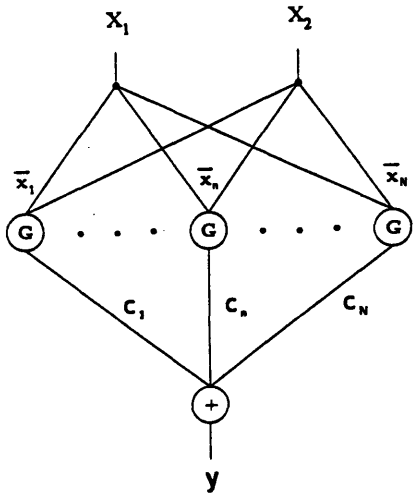


Figure 2.7: A general Radial Basis Function Network.

2.5 Neural Network Applications

Examples of application of neural networks are:

- Language Processing
  - Text-to-Speech Conversion
- Image or Data Compression
- Signal Processing
  - Prediction or forecasting
  - System modeling
  - Noise filtering
  - Risk analysis
- Complex System Control
  - Plant and manufacturing control
  - Robotics control
  - Auto pilot and navigation

- Adaptive Control

- Pattern Recognition and Classification
  - Target classification
  - Defect or fault detection
  - Vision
  - Symptoms-Source diagnosis
- Artificial Intelligence
  - Expert system

The tasks that neural networks are required to perform depend on the application area. Examples of different functions are:

- Prediction

  Use input value to predict output value.

- Classification

  Use input value to predict categorical output.

- Data Association (associative memory)

  Learn associations of error-free or ideal data, then classify or associate data that contains error.

- Data Conceptualization

  Analyze data and determine conceptual relationships.

- Data filtering

  Smooth an input signal, reduce noise.

- Optimization

  Determine optimum value or choice.

Each application usually requires a different topology and learning algorithm. Chapter 3 discusses this aspect, mainly for networks that are employed to carry out

function approximation. A listing of the types of networks that are suitable for the various tasks is given below.

Application	Network Type
Prediction	multilayer network with nonlinear element, radial basis function network.
Classification	multilayer network with nonlinear element, radial basis function network, recurrent network.
Data Association	multilayer network
Data filtering	recurrent network
Optimization	recurrent network

Application	Supervised Training
Prediction	Yes
Classification	Yes
Association	Yes
Conceptualization	No
Data Filtering	No
Optimization	No

There is no exact way to identify the best network or learning algorithm for a particular task. Different alternatives need to be evaluated. It may turn out that the optimum approach is to use a combination of several networks.

When the tasks are complex, it is usually better to divide these tasks into several less complex subtasks and develop separate networks for the subtasks. This approach is followed for the structural diagnosis application, and a detailed discussion is presented in Chapters 5 and 7.

2.6 Neural Networks in Civil and Environmental Engineering

Civil engineering applications of neural networks have become popular only since the late 1980's, after the work of Rumelhart (1986)l revealed the potential of back propagation learning algorithm. Researchers have utilized neural networks mostly for their regression and classification capability. Therefore, feedforward-type networks, mostly multilayer feedforward networks with a back propagation learning algorithm, have been the common choice since they are capable of performing well in both regression and classification. Some of the current applications of feedforward neural networks in Civil Engineering are described below (Garrett, 1992 and Pham , 1995):

- classification of distributed, noisy patterns of on-site information, such as classification of the level of cost for remediating hydraulic conductivity fields (Ranjithan, 1992), vehicle identification and counting applications (Bullock, 1991);

- interpretation of nondestructive evaluation sensory feedback, such as the use of neural networks in detecting the changes of response spectrums due to structural damage (Wu, 1992), detecting flaws in the internal structure of construction components (Flood, 1994), or damage detection from changes in vibrational signatures in a 5-story model steel structure (Elkordy, 1993);

- modeling of complex system behavior, such as modeling complex material constitutive behavior (Wu, 1991), modeling the behavior of large-scale structural systems for the propose of control (Rehak, 1992), modeling concrete material using actual experimental data (Ghaboussi, 1990), or predicting the flow of a river from historical flow data (Karunanithi et al, 1994);

- control of complex engineered facilities, such as the control of deflection of large-scale flexible structures (Rehak, 1992), or the control of an HVAC system for a large structure (Garrett, 1992).

In addition to the above applications, there are other civil engineering applications which employ recurrent or other feedback-type networks. These applications are mainly concerned with large-scale optimization problems such as resource leveling in PERT analysis for construction projects (Shimazaki et al, 1991).

2.7 Comparison of Neural Networks to Other Information Processing Approaches

An expert system is constructed by first acquiring a human expert's way of solving a specific problem via extensive observation. This knowledge, or expertise, is analyzed and represented as rules which are embedded in a computer program. The construction of a neural network, on the other hand, starts by selecting an appropriate architecture and learning algorithm based on a priori knowledge of the problem. The networks are trained, either in supervised or unsupervised mode, with example data. The actual implementation of a neural network then can be done either with a computer program or a specific-purpose hardwired device.

Although neural networks have been highly acclaimed as one of the most versatile approaches, it is unlikely that neural networks will replace either database processing or knowlegebase processing in the near future. Most likely, they will complement existing schemes in areas where their adaptability, ability to learn, and massive parallelism provides them with a significant advantage.

2.8 Relation of Neural Networks to Approximation Schemes

From another point of view, a neural network can be considered as a simple graphic representation of a parametric approximation scheme (Poggio and Girosi, 1991). The network interpretation adds nothing to the theory of the approximation scheme, but it is useful from an implementation perspective. For example, its parallelism can take advantage of parallel computing devices, and its modular form allows for sub-structuring to solve tasks.

Figure 2.8 shows a one-hidden-layer feedforward neural network that is a graphical representation of the following approximation scheme:

$$f^*(x) = \sum_{i=1}^{n} c_i f(X \cdot W_i) \quad , \tag{2.7}$$

which is known as the ridge function approximation scheme.

Similarly, the Radial Basis Function Network (see Fig 2.7) can be interpreted as the representation of the radial basis function approximation scheme,

$$f^*(x) = \sum_{i=1}^{n} c_i G(\|x - x_i\|) \quad . \tag{2.8}$$

Both types of networks are discussed extensively in Chapter 3.

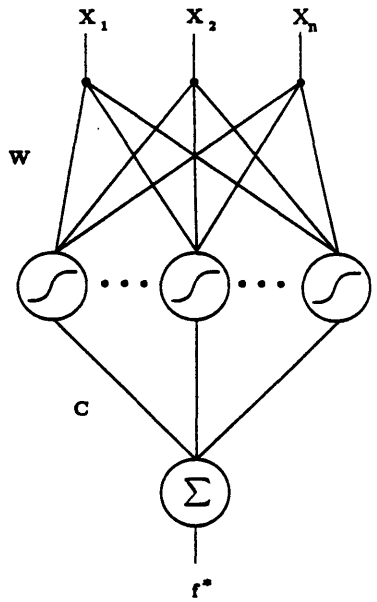


Figure 2.8: A 1-hidden-layer feedforward network.

Approximation schemes have specific algorithms for finding the set of parameters that minimizes a prespecified function. These algorithms correspond to the learning algorithms of neural networks. For example, the back propagation learning algorithm can be considered as a version of the gradient method optimization approach combined with a specific credit assignment method. Similarly, the learning algorithm for RBFN is actually the least-square minimization approach.

Chapter 3

Neural Networks for Function Approximation

3.1 Introduction

Physical system modeling can be divided into two activities: performing regression to predict the system behavior, and performing classification to identify changes of behavior. Regression involves mapping a numerical domain input to another numerical domain output, while classification maps a numerical or categorical domain input to a categorical domain output. Assuming that there is a function F that can perform regression or classification for the purpose of physical system modeling, approximating F can then be considered as a problem of system modeling.

As shown earlier in Chapter 2, networks that perform well in regression and classification are of the feedforward type such as the multilayer feedforward and radial basis function models. Although recurrent networks can also be used for classification, they are not considered here since they are based on unsupervised learning, whereas structural damage diagnosis applications requires supervised learning. In this chapter, the performance of multilayer feedforward network for function approximation, a popular application in civil engineering, is discussed in detail. The performance of radial basis function networks is also investigated for the propose of providing a comparison. Extensive performance tests are presented to show the relative advantages and disadvantages of the multilayer feedforward networks versus radial basis function networks, and to identify their limitations for function approximation.

3.2 The Multilayer Feedforward Network with Back Propagation Learning Algorithm (MLN with BP)

The network consists of an input layer, an output layer, and at least one hidden layer. Each layer is fully connected to its neighboring layers. Figure 3.1 illustrates the connectivity and the notation used to described how a processing element operates (Hertz et al, 1991).

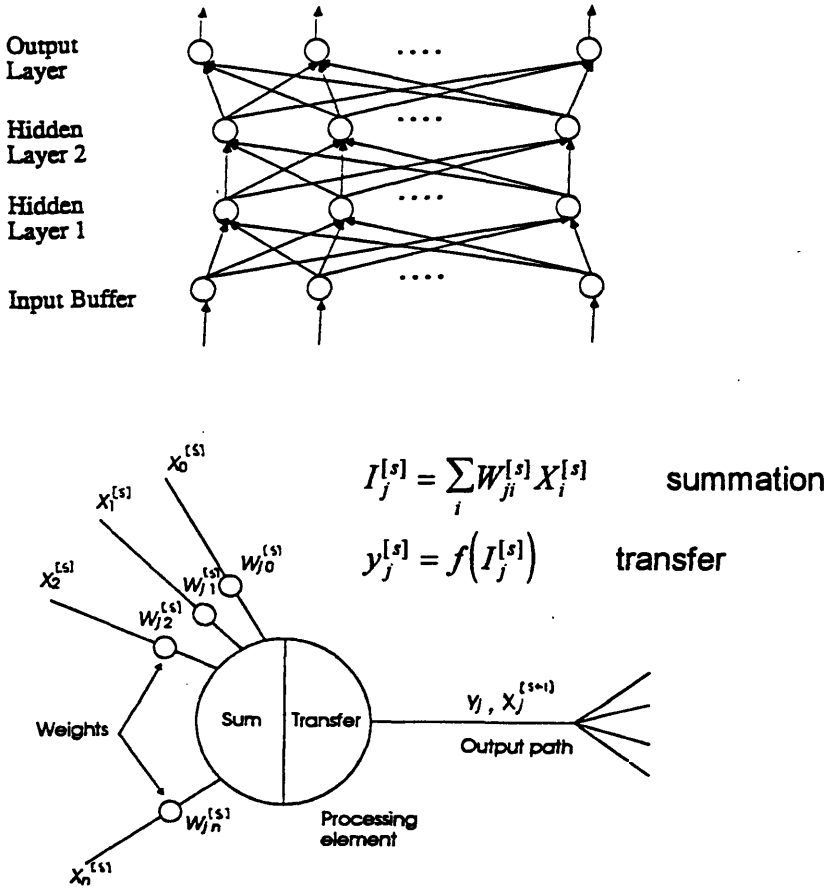


Figure 3.1: Feedforward network.

A superscript in square brackets is used to indicate the layer being considered. Shown below are the definitions for the various symbols.

$x_j^{[s]}$ = the current output of the *j*th processing element in layer *s*.

$W_{ji}^{[s]}$ = the weight for the connection joining the *i*th processing element in layer (*s*-1) to the *j*th processing element in layer *s*.

$I_j^{[s]}$ = the weighted summation of the inputs to the *j*th processing element in layer *s*.

Each processing element operates on its inputs as follows.

$$x_j^{[s]} = f\left[\sum_i \left[W_{ji}^{[s]} \cdot x_i^{[s-1]}\right]\right]$$

$$= f\left[I_j^{[s]}\right] \quad , \tag{3.1}$$

where f is usually a differentiable monotonic function. For example, f may be defined as a sigmoid function,

$$f(z) = (1+e^{-z})^{-1}. \tag{3.2}$$

The initial step involves defining a global error function for the network, E (Rumelhart et al, 1986). This function is required to be a continuous function of all the connection weights. It is needed to define the local errors at the output layer so that they can be propagated back through the network. The parameter that is passed back through the layers is defined as

$$e_j^{[s]} = -\partial E / \partial I_j^{[s]} \quad . \tag{3.3}$$

It will be shown later that $e_j^{[s]}$ can be considered as a measure of the local error at the processing element j in layer *s*.

Applying the chain rule to evaluate the derivative in Eq. 3.3, one obtains

$$e_j^{[s]} = -\frac{\partial E}{\partial I_k^{[s+1]}} \frac{\partial I_k^{[s+1]}}{\partial f(I_j^{[s]})} \frac{\partial f(I_j^{[s]})}{\partial I_j^{[s]}}$$

$$= e_k^{[s+1]} \frac{\partial \sum_l \left(W_{kl}^{[s+1]} \cdot x_l^{[s]}\right)}{\partial x_j^{[s]}} f'(I_j^{[s]}) \quad ,$$

$$= f'(I_j^{[s]}) \cdot \sum_k \left[e_k^{[s+1]} \cdot W_{kj}^{[s+1]}\right] \quad . \tag{3.4}$$

This equation defines the relationship between the local error at a particular processing element at level s and the local errors at processing elements at level s+1. Note that Eq. 3.4 only applies for the hidden layers.

When f is taken as the sigmoidal function defined by Eq. 3.2, its derivative is given by

$$f'(z) = f(z) \cdot (1 - f(z)) \quad . \tag{3.5}$$

Substituting Eq. 3.5 into Eq. 3.4 gives

$$e_j^{[s]} = x_j^{[s]} \cdot (1 - x_j^{[s]}) \cdot \sum_k \left[e_k^{[s+1]} \cdot W_{kj}^{[s+1]}\right] \tag{3.6}$$

Equations 3.1 and 3.6 are the key equations for describing the mechanics of the back propagation learning algorithm. The process starts with a forward propagation of the input through the layers to the output layer using Eq. 3.1. Next, the error at the output

layer is determined, and the error is then propagated back through the network from the output layer using Eq. 3.6, or more generally Eq. 3.4.

The goal of learning is to minimize the global error, E, of the network by adjusting the weights. Given an initial set of weights $W_{ij}^{[s]}$, a procedure for modifying the weights based on a minimization scheme can be applied. For example, gradient descent leads to the following "correction" equation

$$\Delta W_{ji}^{[s]} = -lcoef \cdot (\partial E / \partial W_{ji}^{[s]}), \tag{3.7}$$

where *lcoef* is called a "learning coefficient." Each weighting parameter is adjusted according to the size and direction of the negative gradient on the error surface. Therefore, successive "updating" will lead to the minimum of error surface.

The partial derivatives in Eq. 3.7 can be calculated directly from the local error values previously discussed (see Eq. 3.3). Using the chain rule and Eq. 3.1 gives

$$\begin{aligned} \partial E / \partial W_{ji}^{[s]} &= \left(\partial E / \partial I_j^{[s]}\right) \cdot \left(\partial I_j^{[s]} / \partial W_{ji}^{[s]}\right) \\ &= -e_j^{[s]} \cdot x_i^{[s-1]} \end{aligned} \tag{3.8}$$

Combining Eqs. 3.7 and 3.8 leads to

$$\Delta W_{ji}^{[s]} = lcoef \cdot e_j^{[s]} \cdot x_i^{[s-1]} \tag{3.9}$$

Assume an input vector $\underline{p}$ is presented at the input layer of the network, and the target output $\underline{t}$ is defined. Given $\underline{o}$ is the actual output vector produced by the network with its current set of weights, a measure of the global error in producing the desired output can be taken as

$$E = 0.5\sum_{k}\left(t_k - o_k\right)^2 \qquad (3.10)$$

where subscript k is the index of the component of $\underline{t}$ and $\underline{o}$. Equation 3.10 defines the sum-squared error of the network in producing the desired output through out the range of the input range. From Eq. 3.3, the local error at each processing element of the output layer can be determined by

$$e_k^{[o]} = -\partial E / \partial I_k^{[o]}$$

$$= -\partial E / \partial o_k \cdot \partial o_k / \partial I_k^{[o]}$$

$$= \left(t_k - o_k\right) \cdot f'(I_k) \qquad (3.11)$$

which can then be propagated back to the inner layer by Eq. 3.6.

In some cases, this type of learning algorithm requires a very long training time in order to reach the minimum. For example, when the surface corresponding to the cost function has the shape of a valley with steep sides, and a shallow slope on the valley floor, there may be oscillation of the adjusted parameters across the valley during the learning process, and consequently very little movement downward the slope toward the minimum. The additional of a momentum term (Plaut et al, 1986) can make gradient descent avoid this problem and improve the speed of convergence. The strategy is to provide the strength of each connection, $W_{ji}^{[s]}$, with some inertia or momentum so that the adjustment is in the direction of the average downward force instead of oscillating back and forth cycle. A larger learning rate can then be used since there is less potential for oscillation. This objective is achieved by including the effect of learning from the previous learning cycle in the present learning cycle. The following equation illustrates this process.

$$\Delta W_{ji}^{[s]}(t+1) = -lcoef * (\partial E / \partial W_{ji}^{[s]}) + a\Delta W_{ji}^{[s]}(t) \quad . \quad (3.12a)$$

The momentum parameter α must be between 0 and 1; a value of 0.9 is usually used.

Identifying the appropriate value for the learning coefficient, lcoef, for a specific application is also a challenge. In most cases, the coefficient is modified as the training progresses. A number of researchers (Cater, 1987; Franzini, 1987; Vogl et al, 1988; Jacobs, 1988; Baldi et al, 1995) have suggested ways to adjust this parameter. The general approach is to observe the direction of the learning path on the error surface and adjust *lcoef* to improve the rate of convergence. For example, when n successive values of $W_{ji}^{[s]}$ are in the same direction, *lcoef* is increased. However, when m successive values of $W_{ji}^{[s]}$ are in the opposite direction, *lcoef* is decreased to dampen the oscillation. This process is defined by:

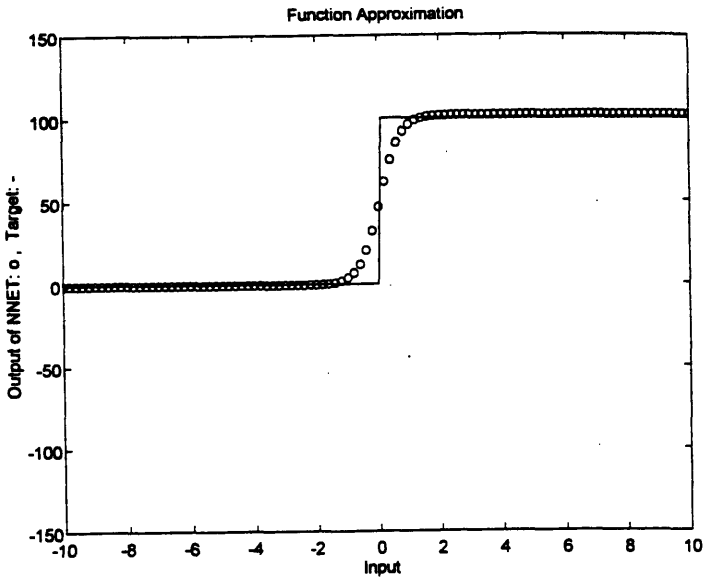
$$\Delta W_{ji}^{[s]} = -\left(lcoef + \Delta\ lcoef\right) \cdot \left(\partial\ E / \partial\ W_{ji}^{[s]}\right),$$

$$\begin{aligned} \Delta\ lcoef &= +a \cdot lcoef \text{ if } \Delta E < 0 \text{ for } n \text{ successive training cycle,} \\ &= -b \cdot lcoef \text{ if } \Delta E > 0 \text{ for } m \text{ successive training cycle,} \\ &= 0 \text{ otherwise,} \end{aligned} \qquad (3.12b)$$

where ΔE is the cost function change, and a and b are appropriate constants.

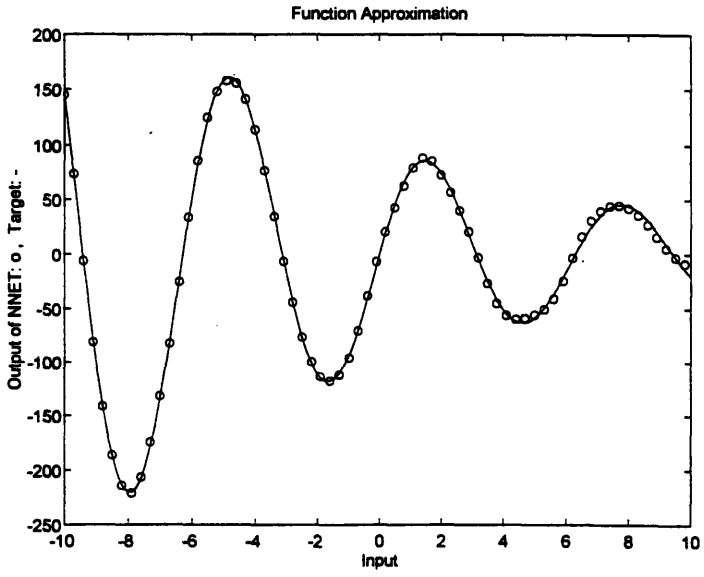
3.2.1 Ability of Multilayer Neural Networks to Approximate Arbitrary Functions

Cybenko (1988) has shown that two hidden layers are adequate for approximating an arbitrary function, provided there are a sufficient number of units per layer. It was also established later that a network with only one hidden layer can represent any continuous function (Cybenko, 1989 and Hornik et al, 1989). However, the relation between the accuracy of the approximation and the number of layers, or the number of units per layer, needs to be established. Figure 3.2 illustrates the approximation (shown by dot line) of a set of functions of the type $y = f(x)$, which are shown by solid line, with various multi-layer architecture. An in-depth analysis of function approximation ability of MLN with BP was discussed by Poggio (1991).



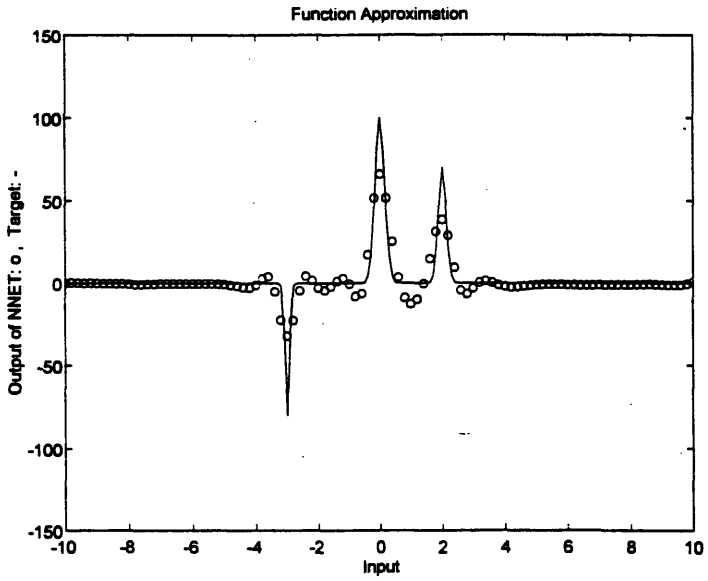
a) $Y = 0 \quad , X < 0$

$Y = c \quad , X \geq 0$

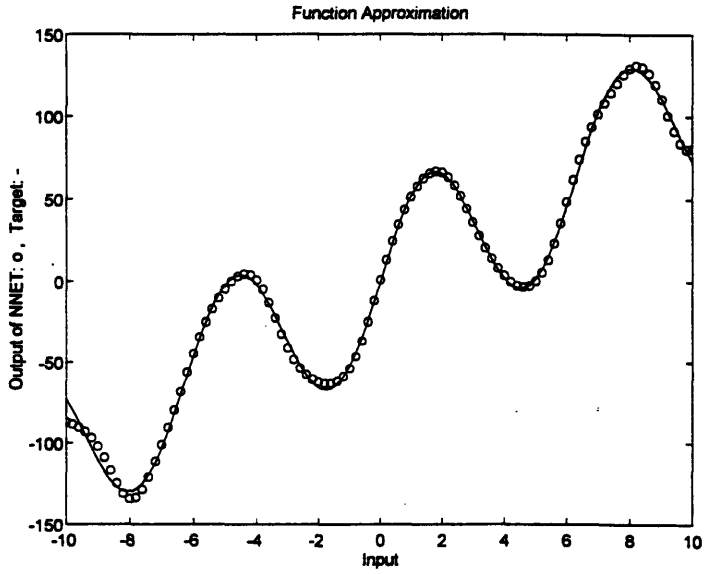


b) $Y = e^{-aX} \sin(bX)$

Figure 3.2: Examples of function approximation by artificial neural networks.



c) $Y = a\, e^{-q(X-k)} + b\, e^{-r(X-m)} + c\, e^{-s(X-n)}$



d) $Y = aX \sin(bX)$

Figure 3.2: Examples of function approximation by artificial neural networks.

3.2.2 One-hidden-layer Network

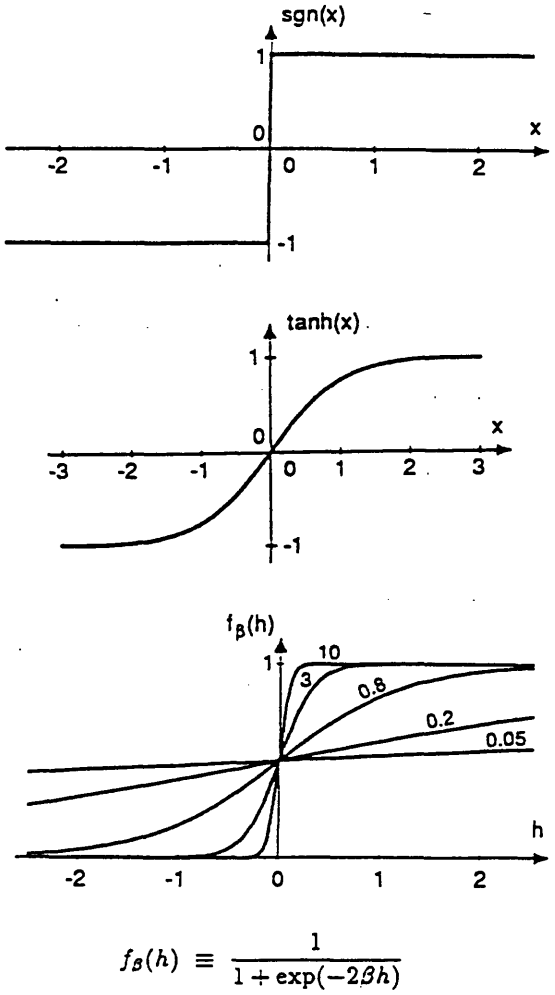
Since it has been established that one hidden layer is sufficient to represent any continuous function, some characteristics of MLN are investigated by conducting numerical experiments on one-hidden-layer network. The issues that are examined are:

- The relationship between the number of processing elements and i) accuracy, ii) training time, iii) local minima, iv) underfitting, and v) the amount of training samples needed.

- The proper network architecture for approximating a given function.

- The effects of using different set of training samples.

The network is taken as fully connected, with single elements in the input and output layers, and multiple elements in the hidden layer. The transfer function for the hidden layer elements is the tangent-sigmoid function shown in Fig 3.3. A linear function is used for the output layer elements. The input and output data of the network are scaled down to the range between -0.8 to 0.8, and the initial values of all weighting parameters are random numbers ranging between -1 to 1. This choice of weights avoids saturating the transfer function in the hidden layer at the beginning of the training process.



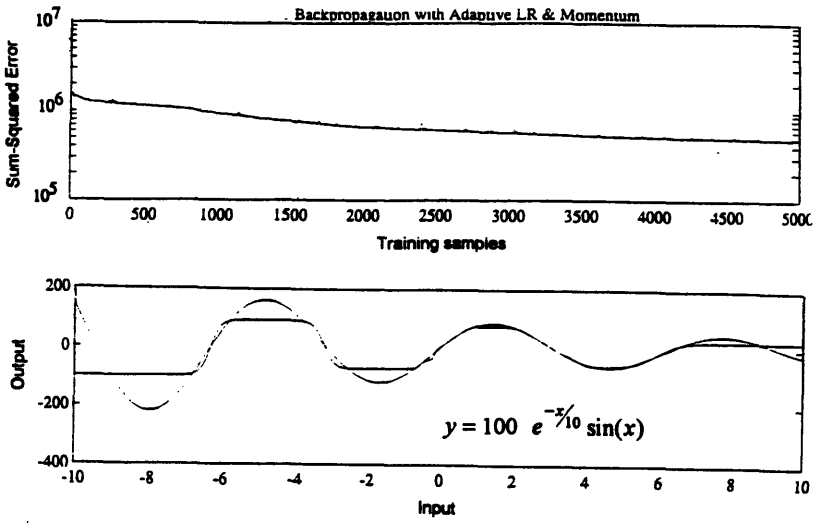
$$f_\beta(h) \equiv \frac{1}{1+\exp(-2\beta h)}$$

Figure 3.3: Example of transfer functions.

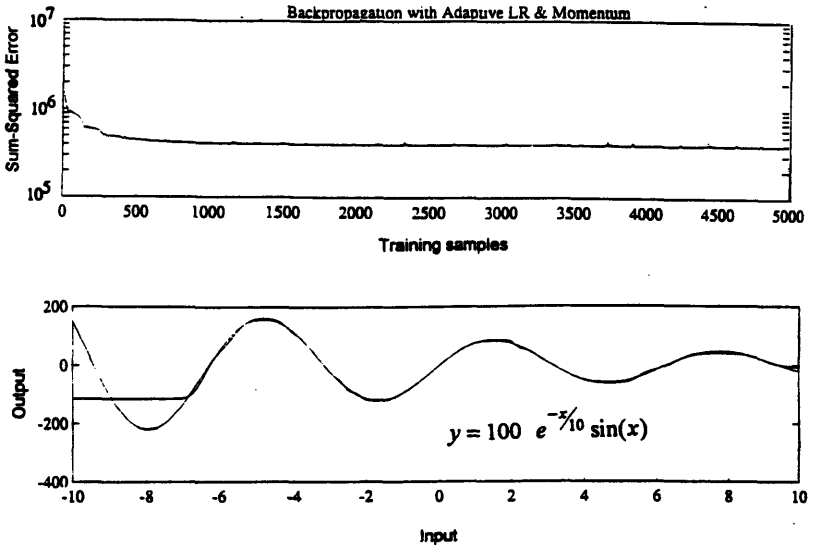
Figure 3.4 contains a sequence of approximations, which are generated by networks having different numbers of interior elements, of the function

$$y = 100\ e^{-x/10} \sin(x) \quad . \tag{3.13}$$

The training and testing data sets contain the values of f for x ranging from -10 to 10 with an interval of 0.1. The results show that, for a given number of training cycles, increasing the number of units improves the accuracy.

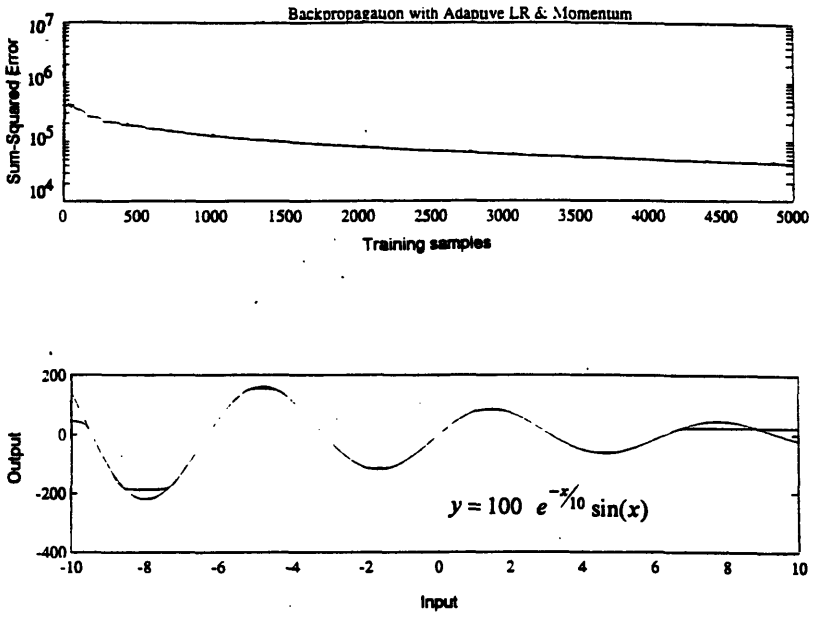


a) Network with 30 processing elements in the hidden layer.

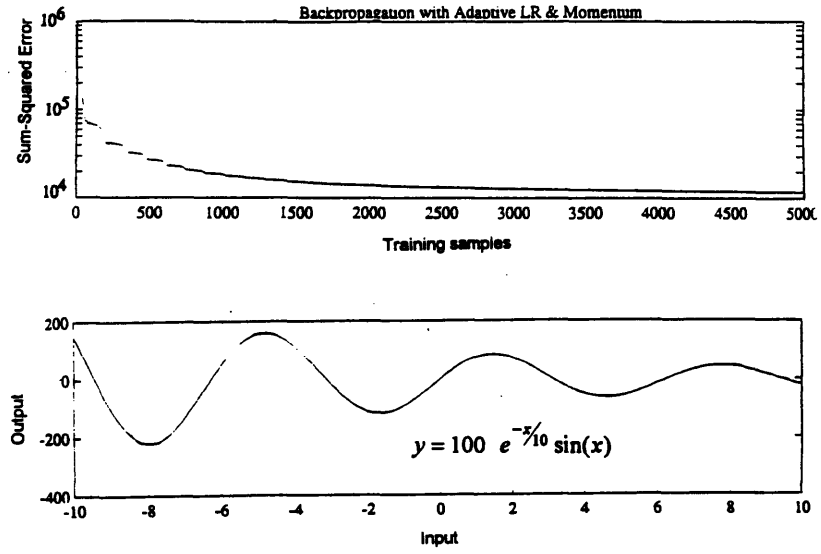


b) Network with 40 processing elements in the hidden layer.

Figure 3.4: Numerical function approximation using one-hidden-layer feedforward networks with different size.

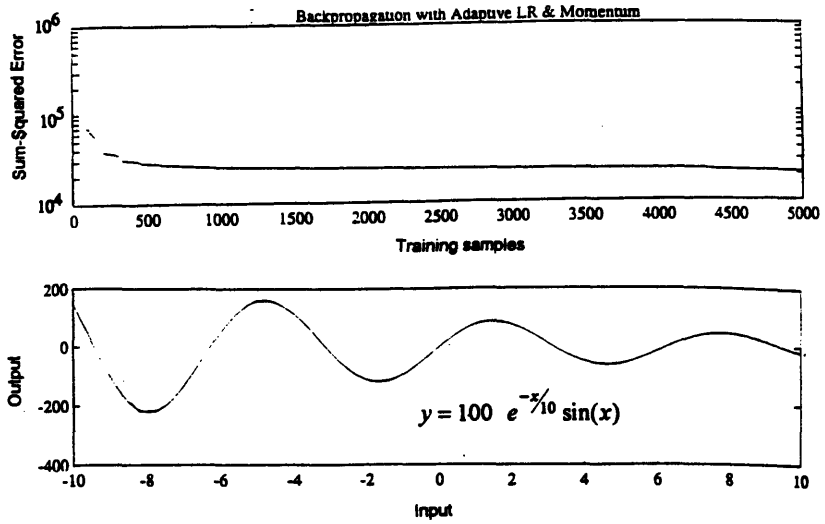


c) Network with 50 processing elements in the hidden layer.



d) Network with 60 processing elements in the hidden layer.

Figure 3.4: Numerical function approximation using one-hidden-layer feedforward networks with different size.



e) Network with 70 processing elements in the hidden layer.

Figure 3.4: Numerical function approximation using one-hidden-layer feedforward networks with different size.

Underfitting is a problem associated with small networks. Figure 3.4a shows the effect of underfitting when a network does not have sufficient parameters to fit a function. Increasing the size of the network lessens this problem. Small networks also tend to converge on local minima of the error function, which is the situation when there is no improvement of accuracy with increasing number of training cycles, even though the network has enough parameters to fit the function. Increasing the number of units decreases the likelihood of converging to local minima, and also improves the accuracy.

Figures 3.5 and 3.6 demonstrate that the appropriate configuration of a network depends on the nature of the function to be approximated. Figure 3.5 shows that a network with 70 units in the hidden layer cannot accurately represent the function

$$y = 100x + x^2 + x^3 + 500\sin(x), \quad -10 \le x \le 10, \qquad (3.14)$$

with 0.1 interval. On the other hand, Fig 3.6 shows that only 10 units are required to represent the function

$$y = 100e^{-20x^2} + 70e^{-30(x-2)^2} + 80e^{-100(x+3)^2} \qquad (3.15)$$

for the same range and interval.

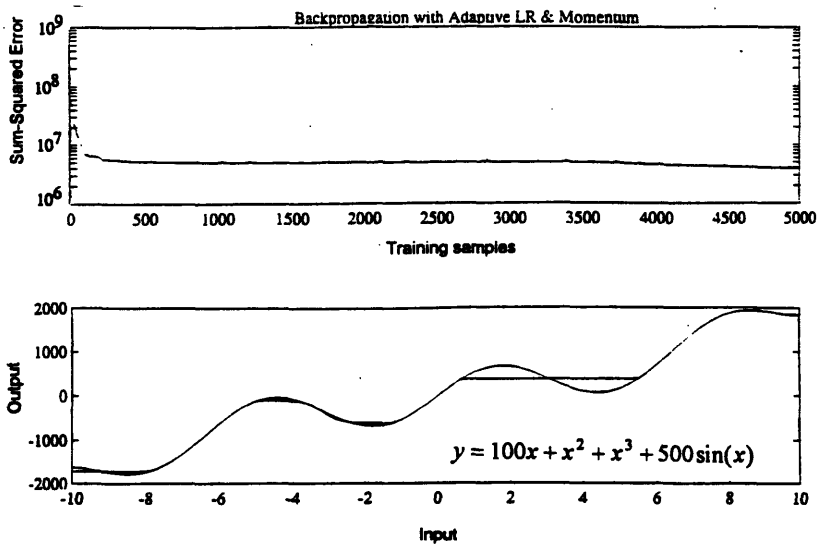


Figure 3.5: Numerical Function approximation using a one-hidden-layer network with 70 processing elements in the hidden layer.

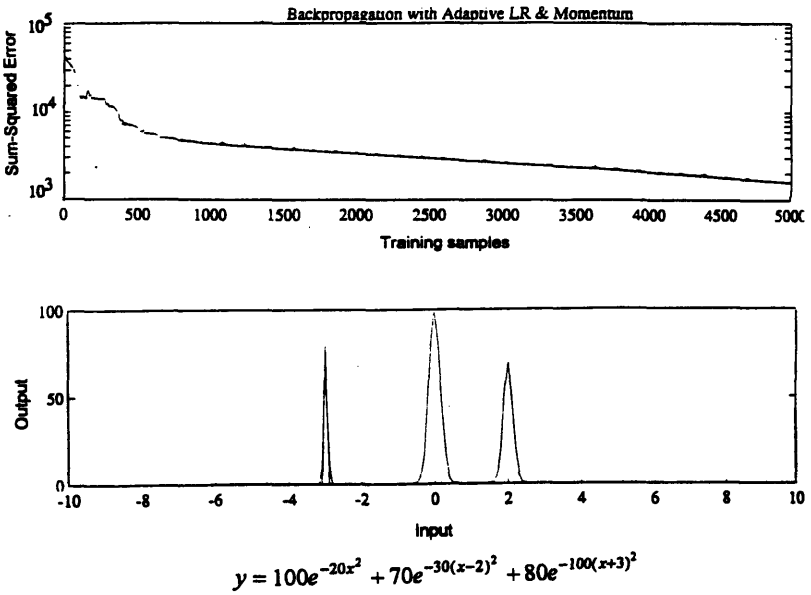


Figure 3.6: Numerical Function approximation using a one-hidden-layer network with 30 processing elements in the hidden layer.

Next important issue is the significance of the representation of training data. Properly prepared training data can improve the convergence and help to avoid local minima. To demonstrate this effect, three different training data sets are used for training three networks, each having 25 units in the hidden layer, to represent the function

$$y = 100\sin(x), \quad -10 \le x \le 10. \tag{3.16}$$

The data interval for each training data set varies from 0.1 to 0.3, while the testing data set interval size is 0.33. As shown in Fig 3.7, the network that is trained with the data having an interval size of 0.1, which corresponds to 200 training data pairs, gives the best accuracy for this choice of test data.

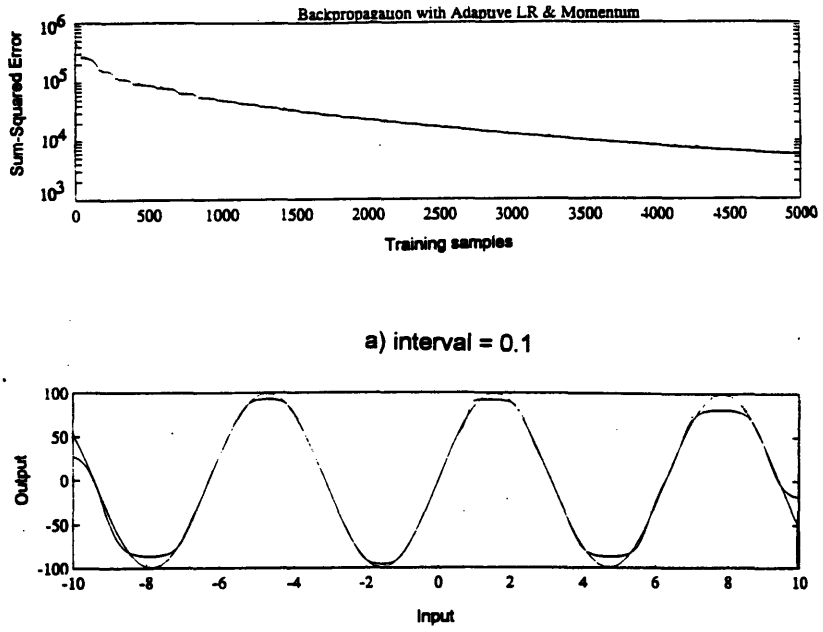


Figure 3.7: Numerical function approximation using a one-hidden-layer network.

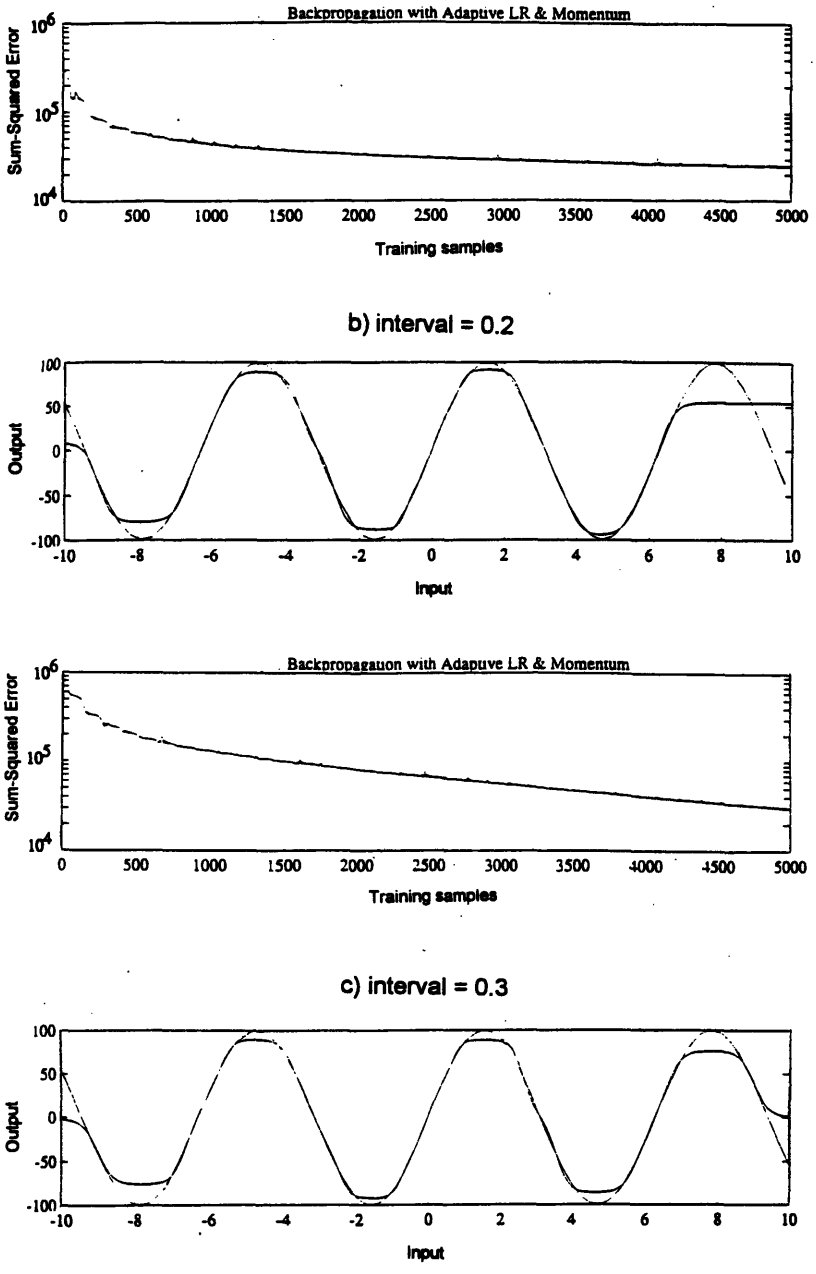


Figure 3.7: Numerical function approximation using a one-hidden-layer network.

Figure 3.8 shows the effect of the amount of training data on the approximation accuracy, and indicates that the performance of networks increases with the amount of training data. However, there is no way of establishing the "a-priori" optimum size of the interval , or optimum size of training data set, for an arbitrary function. The proper size has to be determined either by trial and error or cross-validation method (Wahba, 1980 and Liu, 1995), which is described later in this chapter.

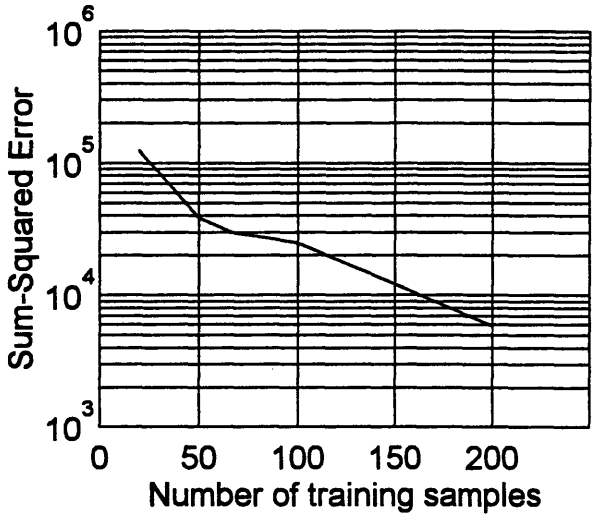


Figure 3.8: Effect of the number of training samples to the accuracy of neural network.

For pattern classification, a Boolean type representation is used to denote the categorical output. This is implemented in a feedforward network by using a sigmoid transfer function in the output layer so that the output of each output unit ranges from 0 to 1. The number of output units is set equal to the number of categories that the network is supposed to differentiate between, and each output unit corresponds to a specific category. The classification of a particular category is considered to occur when the output of the corresponding output unit is sufficiently close to unity, and the other outputs are sufficiently close to zero.

Figure 3.9 shows 4 classes of Gaussian distributed data, with different standard deviations, which are used to evaluate the classification performance of various networks. The symbols x, o, +, and * represent the different data classes.

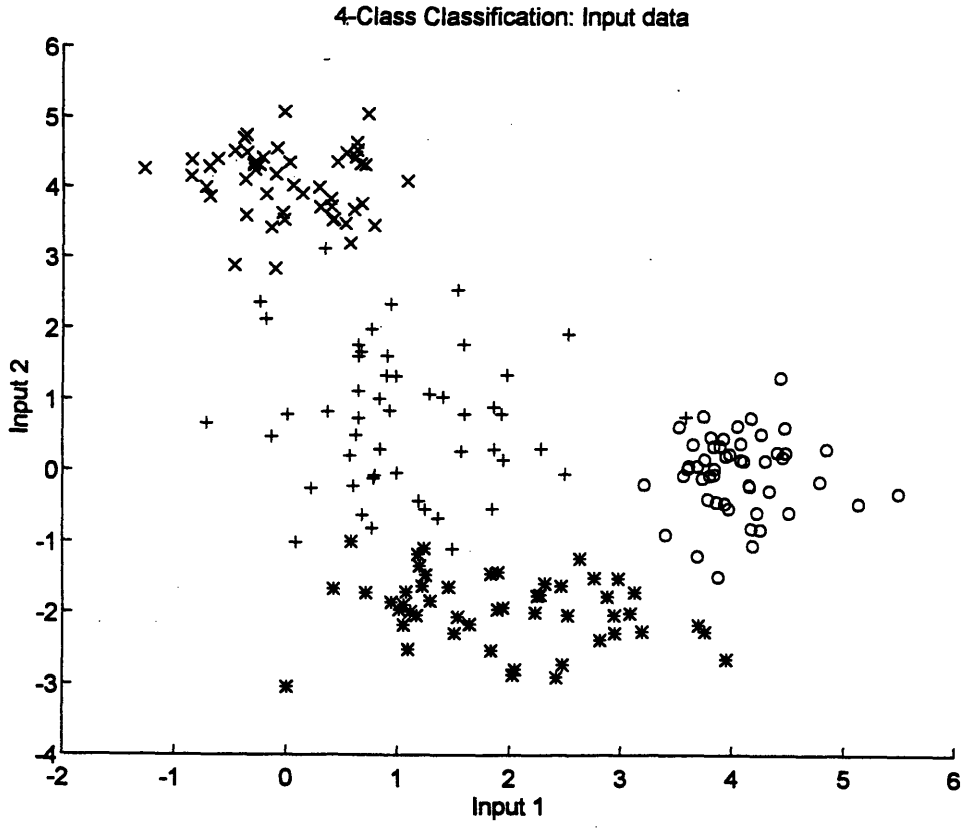
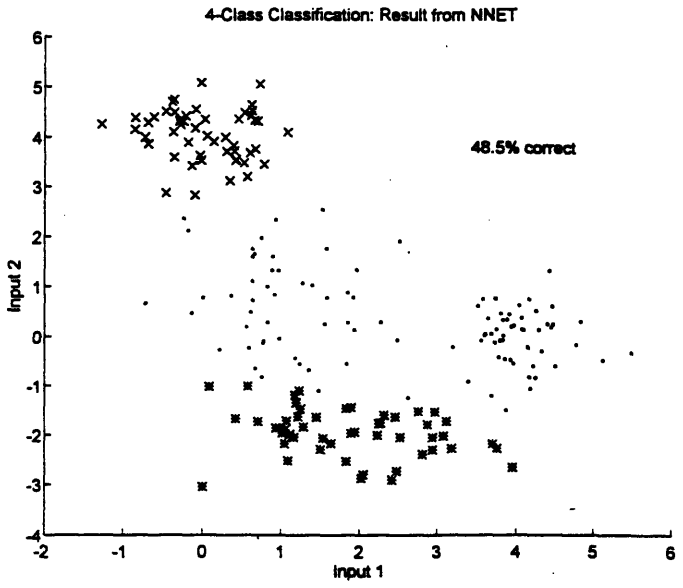
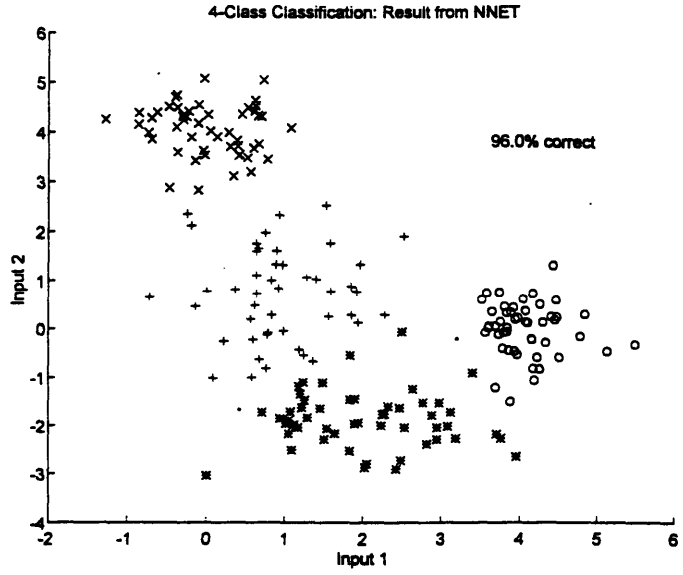


Figure 3.9: Data for a classification problem.

These classes are nonlinearly separated and partially overlap. Each network has 2 input units and 4 output units. Figure 3.10 shows the classifications by the networks of different size. The performance is measured by "the percentage of correct classification" plotted in the Fig 3.11. The symbol ∘ indicates that the output for the particular input did not reach the threshold level, which is set at 0.8 for every output node, and therefore the classification cannot be made. Generally, the threshold is allowed to be lower when the amount of training data is smaller, or when the training data is noisy.

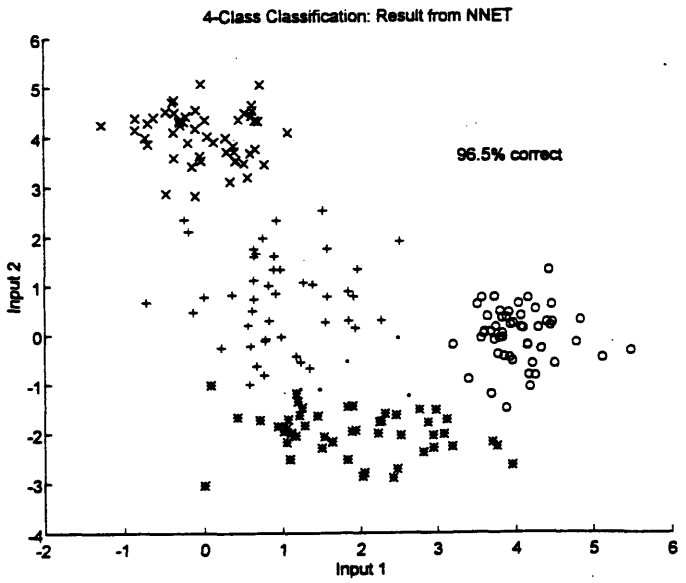


a) Network with 2 processing elements in the hidden layer.

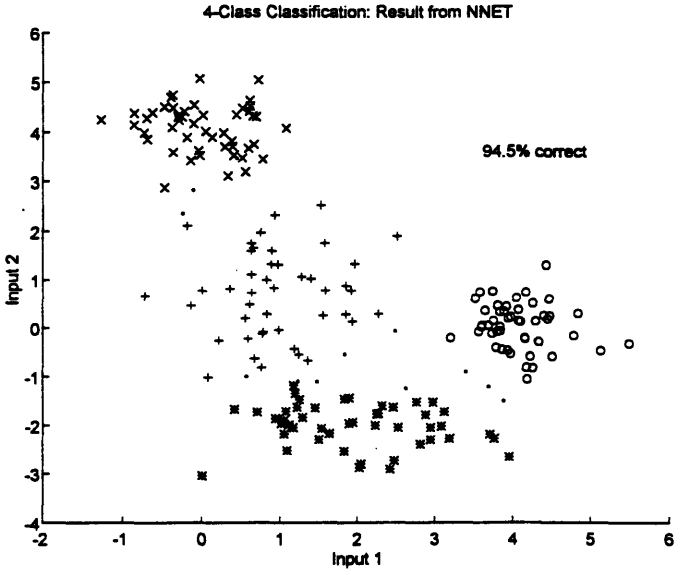


b) Network with 4 processing elements in the hidden layer.

Figure 3.10: Classification results using one-hidden-layer networks.



c) Network with 6 processing elements in the hidden layer.



d) Network with 10 processing elements in the hidden layer.

Figure 3.10: Classification results using one-hidden-layer networks.

The results also show that the classification accuracy increases with the number of processing elements in the hidden layer. However, as shown in Fig 3.11, the classification performance stops improving when the number of processing elements is beyond a certain number.

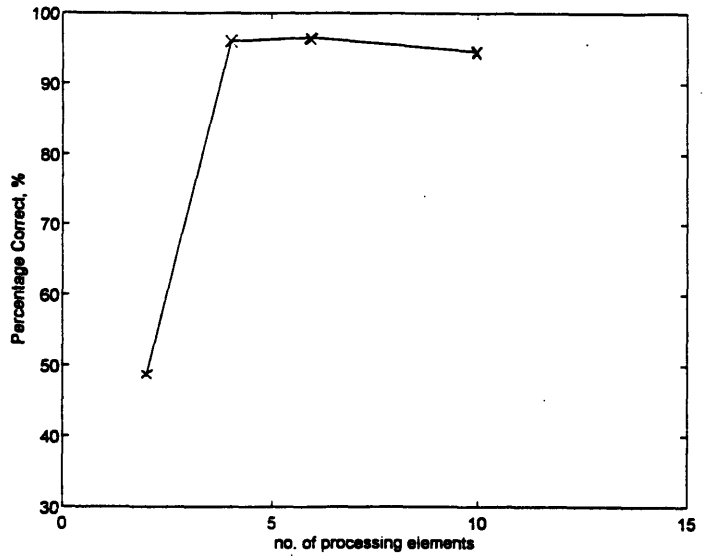


Figure 3.11: The effect of number of processing elements to the classification accuracy.

3.2.3 Two-hidden-layer Network

In order to properly design a MLN for approximating an arbitrary function, the significance of the "number" of hidden layers also has to be investigated. The important issues are:

- For a given number of processing elements, does a 2-hidden-layer model provide better accuracy than a one-hidden-layer model?

- What is the optimum distribution of elements between layers, given the same total amount of units for the whole network ?

- How 2-hidden-layer networks perform on a given task when it is compared to the one-hidden-layer network with the same number of total units ?

A series of numerical simulations were carried out with various 2-hidden-layer configuration trained to approximate the following function,

$$y = 100e^{-20x^2} + 70e^{-30(x-2)^2} + 80e^{-100(x+3)^2} \qquad (3.15)$$

when $-10 \leq x \leq 10$ with an increment of 0.1.

Figures 3.12 to 3.15 show the results for a set of networks with a 2 to 1 distribution of elements between the first and second hidden layers. The total number of units ranges from 40 to 80. For instance, Fig 3.14 demonstrates the performance of the network that has 60 total units, with 40 units in the first hidden layer, and 20 in the second hidden layers. Similar to the findings for the one-hidden-layer case, these results show that the accuracy for a network given the same amount of training samples improves with the number of total elements. The results also show that smaller networks tend to underfit and converge on local minima more frequently (see Figs 3.12 and 3.13). Both circumstances rarely occur when the two-hidden-layer networks with larger number of units are employed.

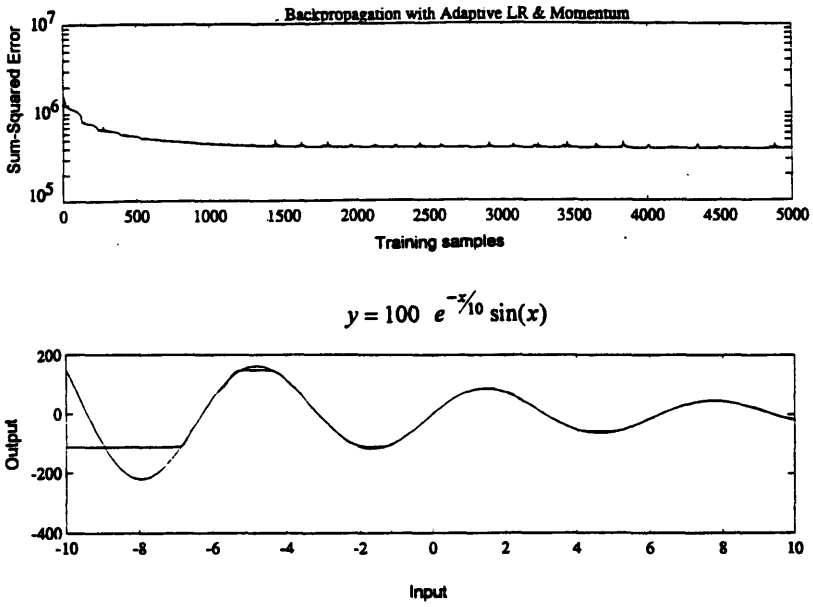


Figure 3.12: Function approximation using a two-hidden-layer network with 26 units in the 1st hidden layer, and 13 units in the 2nd hidden layer.

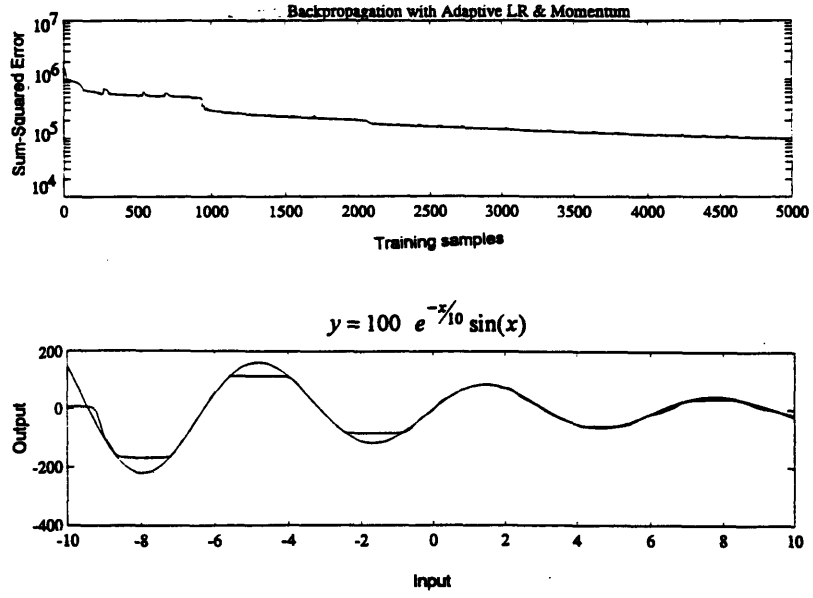


Figure 3.13: Function approximation using a two-hidden-layer network with 40 units in the 1st hidden layer, and 20 units in the 2nd hidden layer.

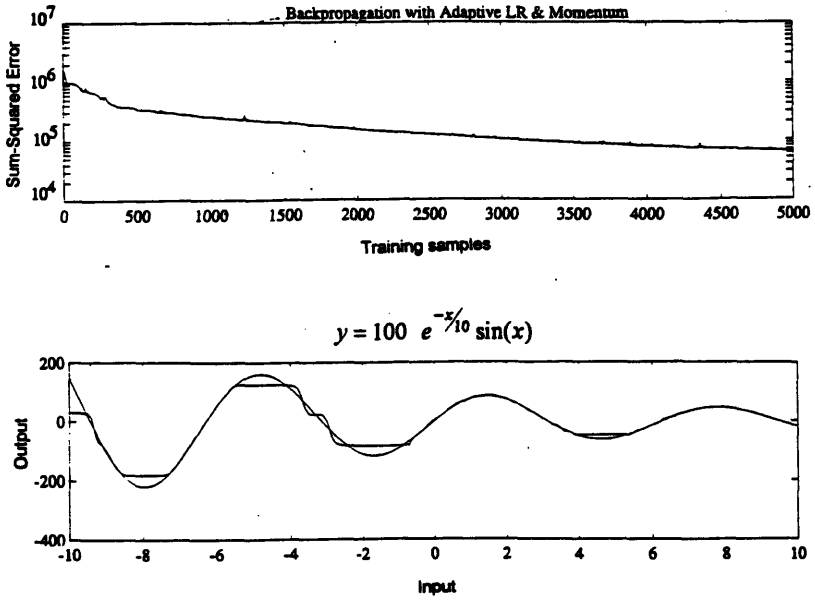


Figure 3.14: Function approximation using a two-hidden-layer network with 46 units in the 1st hidden layer, and 23 units in the 2nd hidden layer.

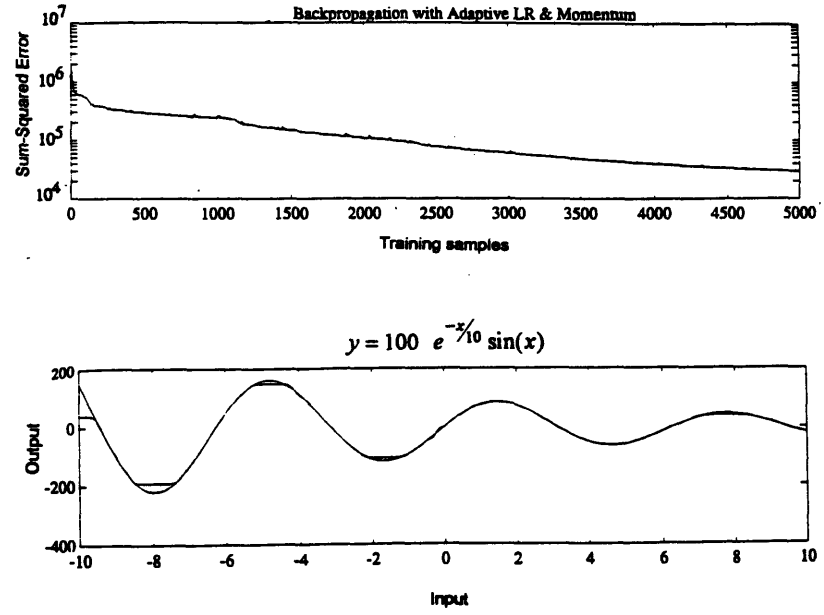


Figure 3.15: Function approximation using a two-hidden-layer network with 53 units in the 1st hidden layer, and 27 units in the 2nd hidden layer.

Figures 3.15 to 3.17 illustrate the effect of varying the proportion of elements between the layers, holding the number of the total elements constant. The ratios here are 2:1, 1:1, and 1:2 respectively. The results indicate that assigning more elements to the second layer tends to degrade the performance and increase the likelihood of convergence to a local minima. This can be observed by comparing Figs 3.16 and 3.17.

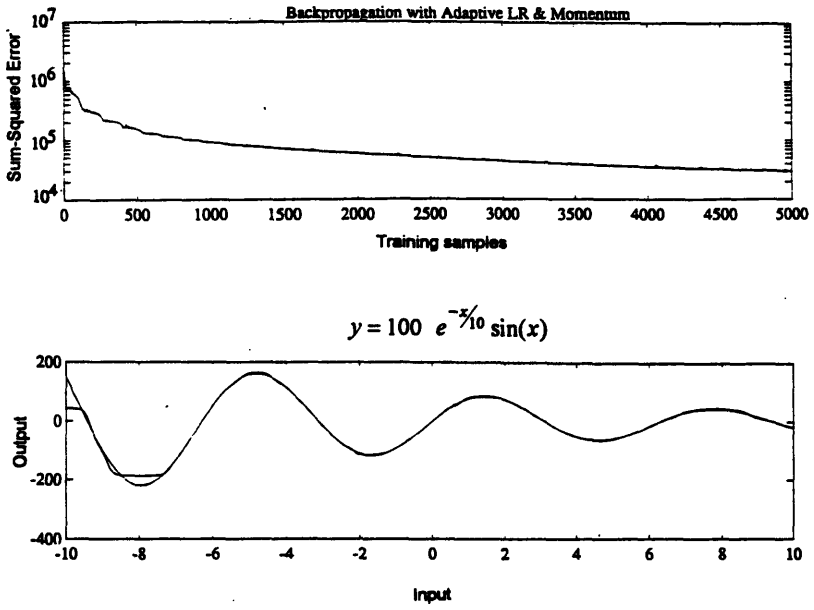


Figure 3.16: Function approximation using a two-hidden-layer network with 40 units in the 1st hidden layer, and 40 units in the 2nd hidden layer.

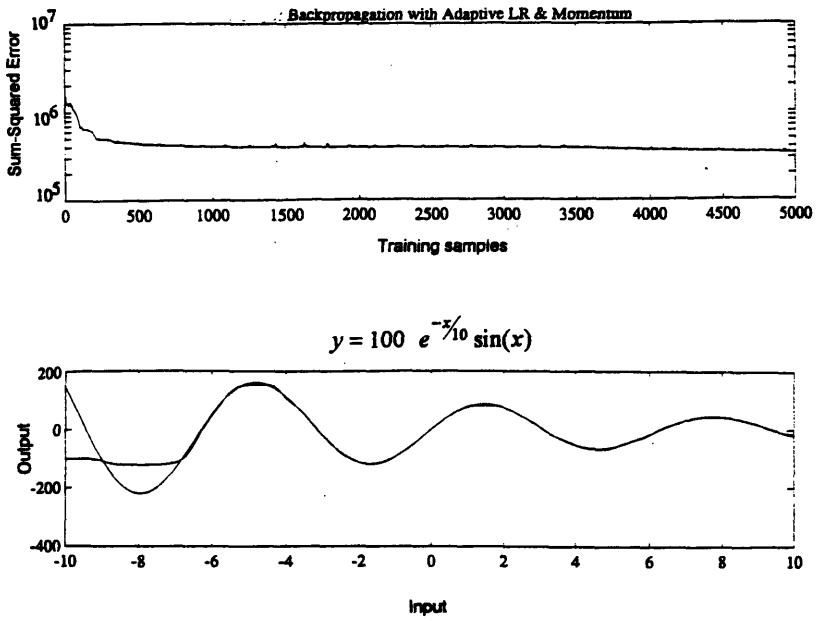


Figure 3.17: Function approximation using a two-hidden-layer network with 27 units in the 1st hidden layer, and 53 units in the 2nd hidden layer.

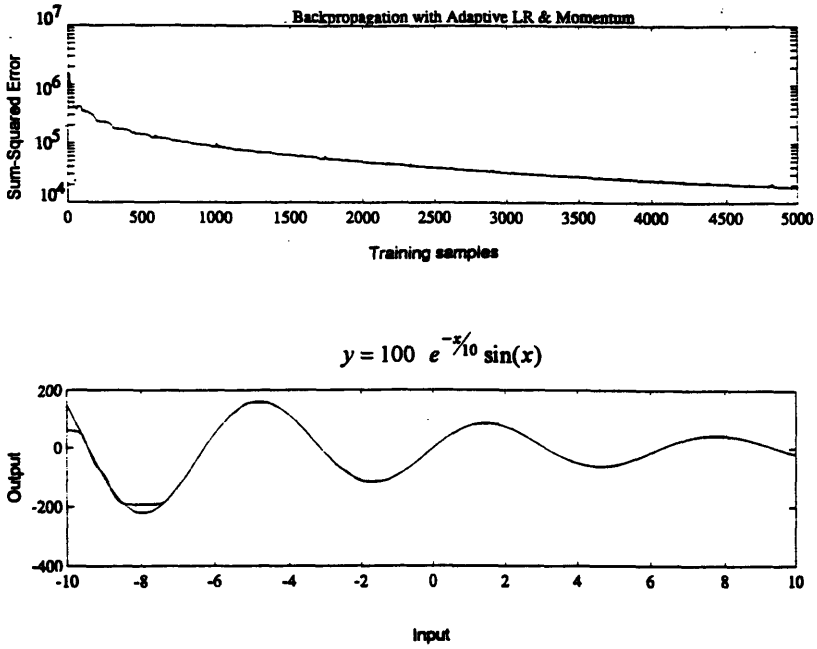


Figure 3.18: Function approximation using a one-hidden-layer network with 80 units in the hidden layer.

Figure 3.18 contains the result for a one-hidden-layer network with 80 total elements. Comparing this performance with the corresponding 2-layer results shown in Fig 3.15 to 3.17, one observes that the one-hidden-layer network performs better, and even converges more rapidly.

In addition to the accuracy and convergence, one has to consider the computation time. Since there are more connections in a two-hidden-layer network than in a one-hidden-layer network with the same number of total units, two-hidden-layer network requires more computation time. The following example illustrates this point.

Consider two networks, a one-hidden-layer network with $m+n$ processing elements in its hidden layer, and a fully connected two-hidden-layer network with m processing elements in the first hidden layer and n processing elements in the second hidden layer. The total numerical operation required to forward propagate the one-hidden-layer network, *TC1*, is given by

$$TC1 = 2(m+n)M + (m+n+1)S + (m+n+1)T \qquad (3.17)$$

where M is a multiplication operation; S is a summation operation; and T is a function transferring operation. The corresponding operation count for the 2-layer network, *TC2*, is

$$TC2 = (m+mn)M + (m+mn+1)S + (m+n+1)T\,. \qquad (3.18)$$

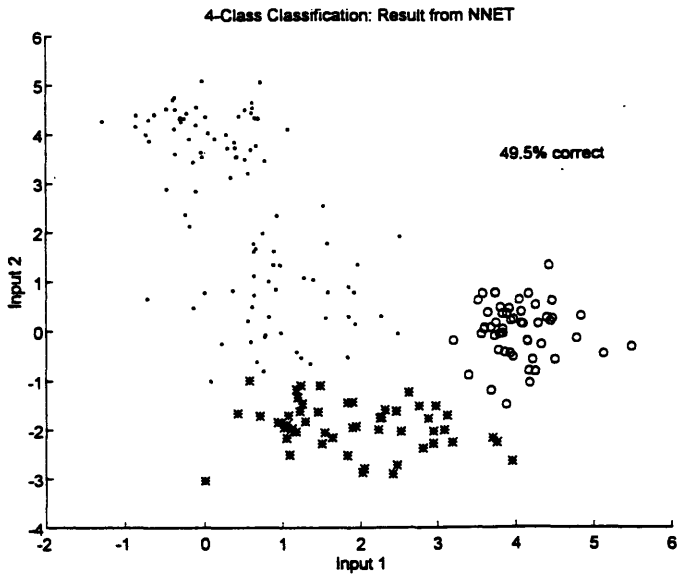
From Eqs. 3.17 and 3.18, it is apparent that more numerical operation is required for the two-hidden-layer network. The difference becomes even greater with increasing m and n.

In addition to the operational cost, the time required to train the network needs to be considered. The computational time required to train two types of networks using the software MATLAB on a 486DX (33MHz) machine is listed in Table 3.1. One network has one-hidden-layer network with 80 total units; the other is a two-hidden-layer network with 40 units in each of its hidden layers. The results indicate that the two-hidden-layer network requires about twice as much computation time for the same amount of training cycles.

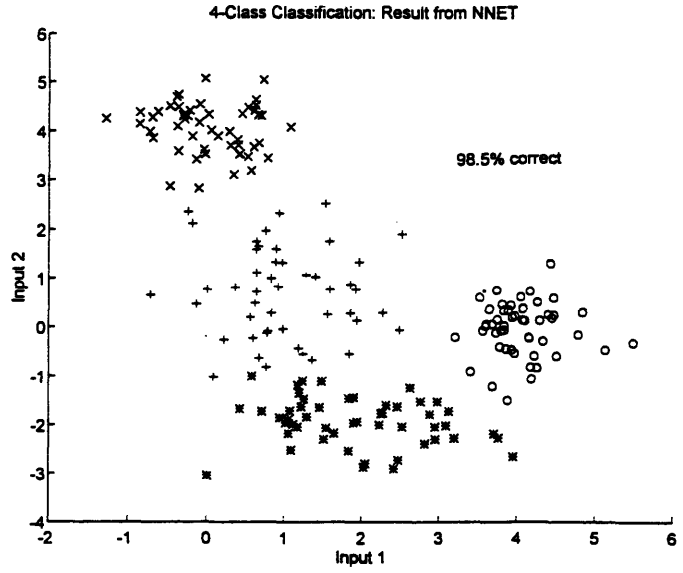
Training cycles	One-hidden-layer network (sec)	Two-hidden-layer network (sec)
500	587	1120
1000	1155	2145

Table 3.1: The training time of feedforward networks.

The previous discussion pertains to regression applications. For classification applications, one-hidden-layer networks do not generally outperform two-hidden-layer networks that have the same number of total units. Figure 3.19 shows the results for 3 different two-hidden-layer networks having a total of 6 units. The corresponding one-layer result is presented in Fig 3.10c. As shown by the figures, two 2-hidden-layer networks perform at essentially the same level as the 1-hidden-layer network, and one cannot say, based on these studies, that one model is "better" than the other.

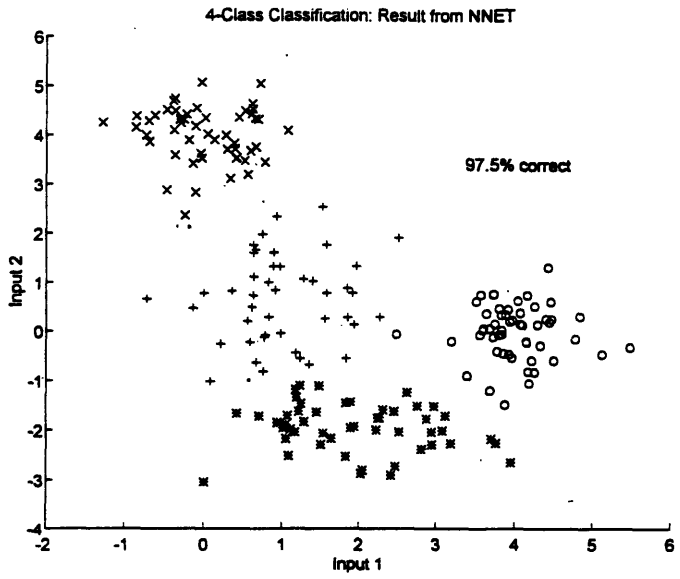


a) Network with 4 units in the 1st hidden layer, and 2 units in the 2nd hidden layer.



b) Network with 3 units in the 1st hidden layer, and 3 units in the 2nd hidden layer.

Figure 3.19: Classification by two-hidden-layer networks.



c) Network with 2 units in the 1st hidden layer, and 4 units in the 2nd hidden layer.

Figure 3.19: Classification by two-hidden-layer networks.

3.2.4 Optimum Network Architecture

The data presented in the previous sections indicates that the performance of the network for a given task is very sensitive to the architecture of the network, and therefore a method for finding the right architecture would be very useful. The Cross-Validation method (Wahba, 1980 and Liu, 1995) is the most popular method. It is also a reliable way to select the right architecture and avoid overtraining, which will be discussed later in this section.

To perform cross-validation, the available training data is divided into two groups, a training set and a cross-validation set. The initial architecture, which usually is the architecture with the smallest number of processing elements possible, is then trained with the training set, and tested with both training and cross-validation set. The error index of

the network based on the training data is monitored while the network is being trained. At the same time, the error index of the network based on the cross-validation data set is also being monitored. According to Fig 3.20, the training should be stopped when the rate of change of the cross-validation error index with the number of training cycles reverses sign, even though the training set error index is still decreasing. More training from this point on produces a network that is more tuned to the training data set instead of the whole data, and hence reduces the ability of the network to deal with a broader range of inputs. This effect is called "overfitting" or "overtraining" (Ling, 1995).

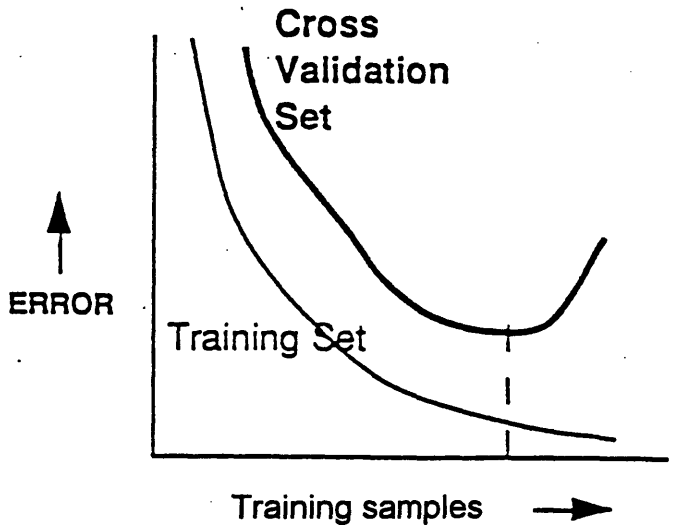


Figure 3.20: Cross-validation method.

The Cross-Validation method should be carried out for every combination of network architecture and training algorithm, and the performance comparison is performed to select the most appropriate network for the particular application (given a set of data). It is also important to note that differently divided data sets (into training set

and cross-validation set) can lead to different performances for the same network. However, the difference is minimal when the size of the training data set is large.

A more realistic application, i.e. modeling the behavior of a bending beam, is considered. Figure 3.21 shows a cantilever bending beam subjected to a point load.

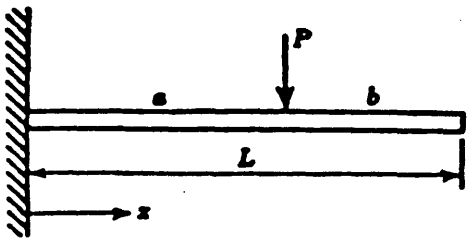


Figure 3.21: A cantilever bending beam.

The beam has length L with a point load of magnitude P placed at the distance a from the fixed support. Considering only the linear planar bending behavior of the beam, the amount of deflection y at the distance x from the support due to the loading can be determined analytically as

$$y = \frac{Px^2}{6EI}(3a - x) \quad , \qquad 0 \le x \le a, \quad \text{and}$$

$$y = \frac{Pa^2}{6EI}(3x - a) \quad , \qquad a \le x \le L \quad , \qquad (3.17)$$

where E is the modulus of elasticity of the beam, and I is the moment of inertia of the bending axis. Assuming the beam length is 500 cm; the modulus of elasticity is $2*10^6$ kg/cm^2; the bending moment of inertia is 20000 cm^4, and the magnitude of point load is 1 ton, Eq. 3.17 takes the form

$$y = \frac{x^2}{2.4 \times 10^8}(3a - x) \quad , \quad 0 \le x \le a$$

$$y = \frac{a^2}{2.4 \times 10^8}(3x - a) \quad , \quad a \le x \le 500 \quad . \qquad (3.18)$$

The training data is created from the analytical model. Variables a and x are considered as the inputs, while y is the output. The inputs, a and x, range from 0 to 500 cm with an interval of 20. Each output y is determined from a particular combination of a and x, hence providing a total of 676 input-output data pairs. Gaussian noise is added to each data pair in order to simulate the noisy signal of real sensors. The plot of noise-free and noisy input-output data pairs are demonstrated in Fig 3.22.

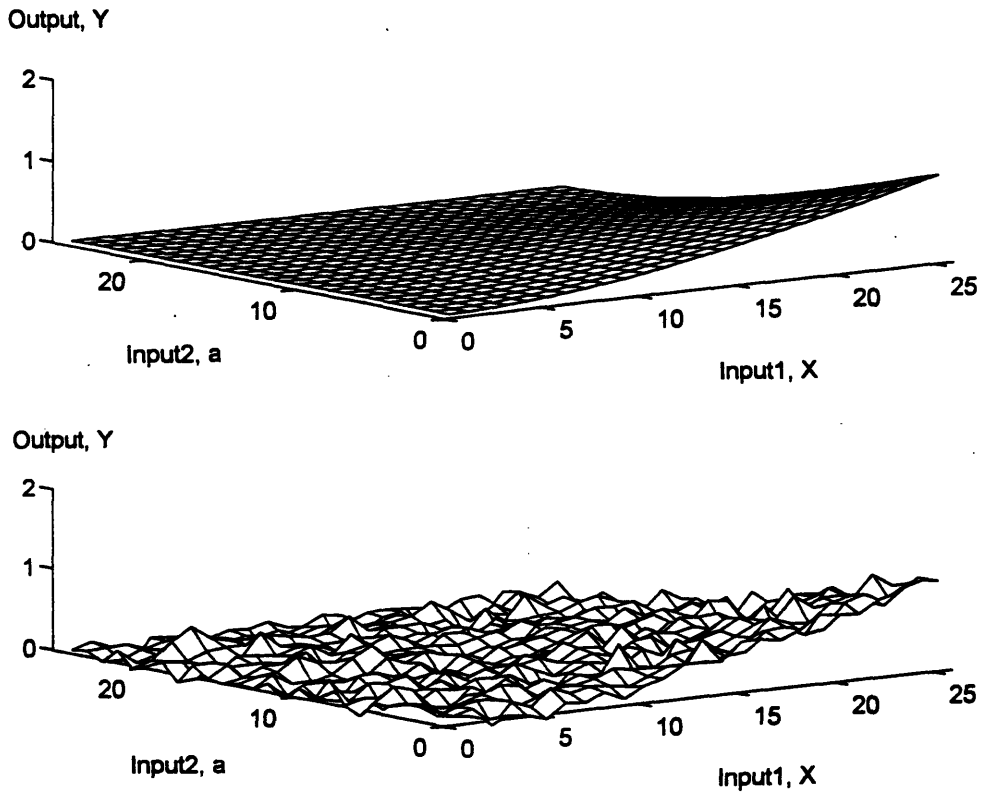


Figure 3.22: Noise-free and noisy input-output data.

The Cross-Validation method is employed with the assumption that the noisy data is obtained from real experiments (Wahba, 1980). Three-fourth of the data is randomly assigned as the training data set, and the rest as the cross-validation set. A 1-hidden-layer back propagation network with 2 processing elements in the hidden layer is used as the initial network. The Cross-Validation method is then utilized for the 1-hidden-layer network with various numbers of processing elements. The plot between the Sum-Squared Error (SSE) after convergence of each architecture on the cross-validation set and the corresponding number of processing elements in the hidden layer is shown in Fig 3.23. The result indicates that the optimum architecture has 12 processing elements in the hidden layer. The Cross-Validation method can also be applied to networks with 2 or more hidden layers, or even with other types of networks (Wahba, 1980).

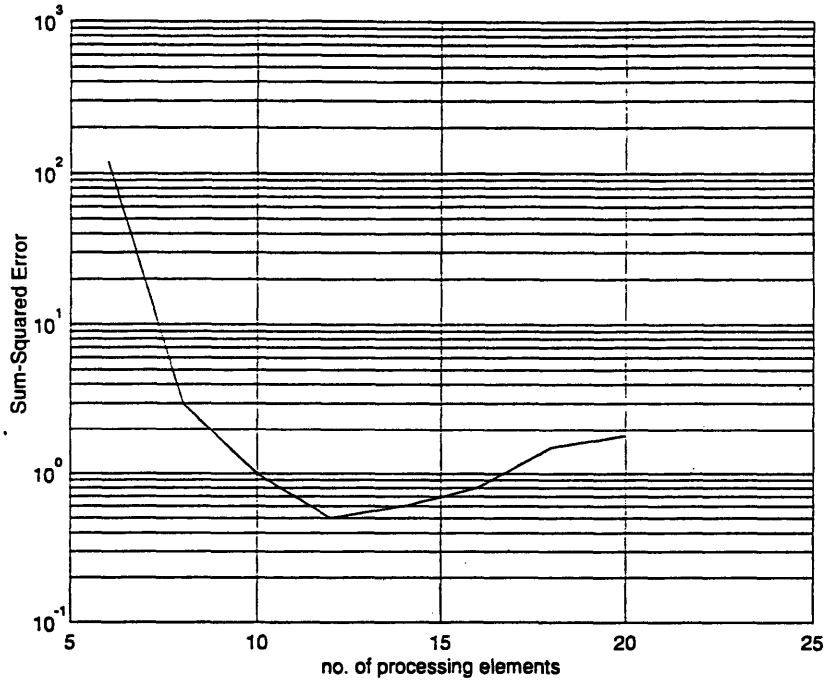
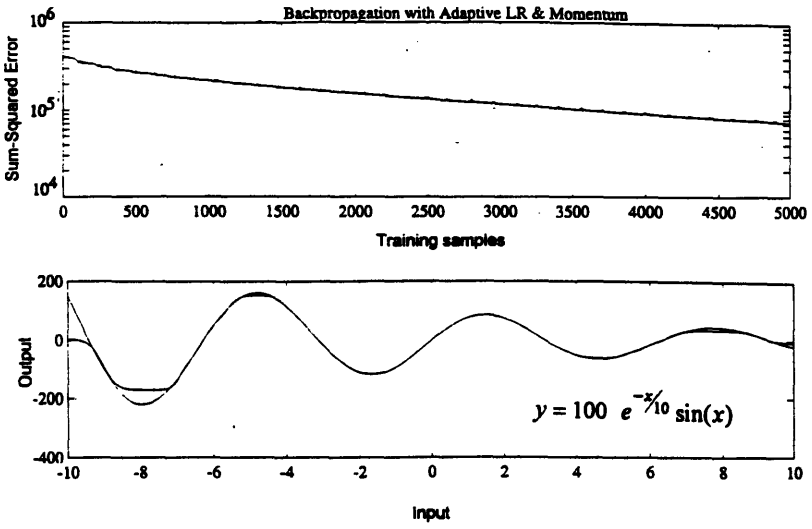


Figure 3.23: Effect of no. of units to approximation error.

To avoid the work involved in performing Cross-Validation, an algorithm that helps optimize the architecture of the network could be incorporated into the existing

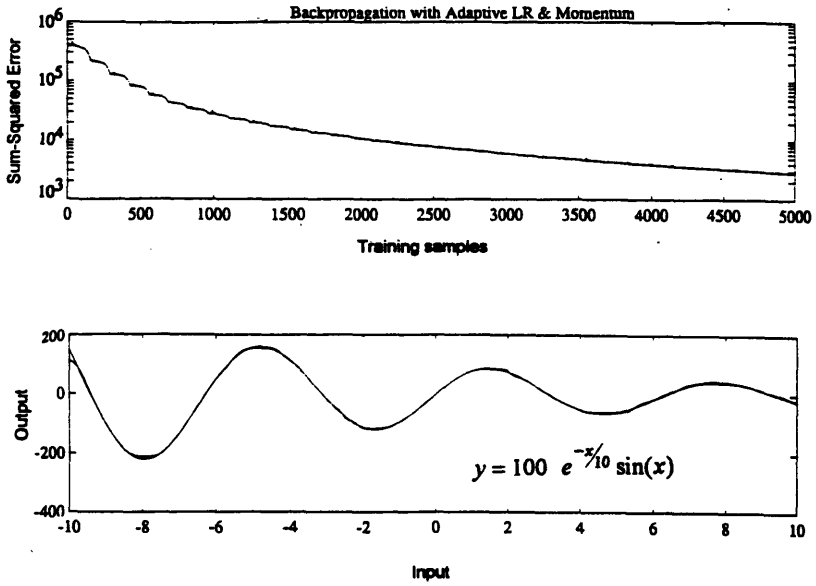
training algorithm. The problem is there are many different criteria for network optimization that have to be considered, such as training time, size and quality of training data set, number of units, number of connections, number of layers, and generalization ability, which is the ability of neural networks to predict given the input that is not included in the training data.

Since the one-hidden-layer network can accurately represent any arbitrary function, and is more computational efficient than any two-or-more-hidden-layer network, it is considered the optimum architecture. The investigation is next focused on how to identify the number of units required to avoid converging on a local minima and achieve a certain level of accuracy. Figure 3.24 demonstrates the performance of three 1-hidden-layer networks, each having 80 total units, in representing a SISO function. A sub-set of the initialized weights of the networks, are obtained from a smaller network trained with 5,000 training samples, while the remaining are random numbers between -1 and 1. Comparing these results with the non-pretrained results (Fig 3.18) indicates that pretraining improves the convergence rate.

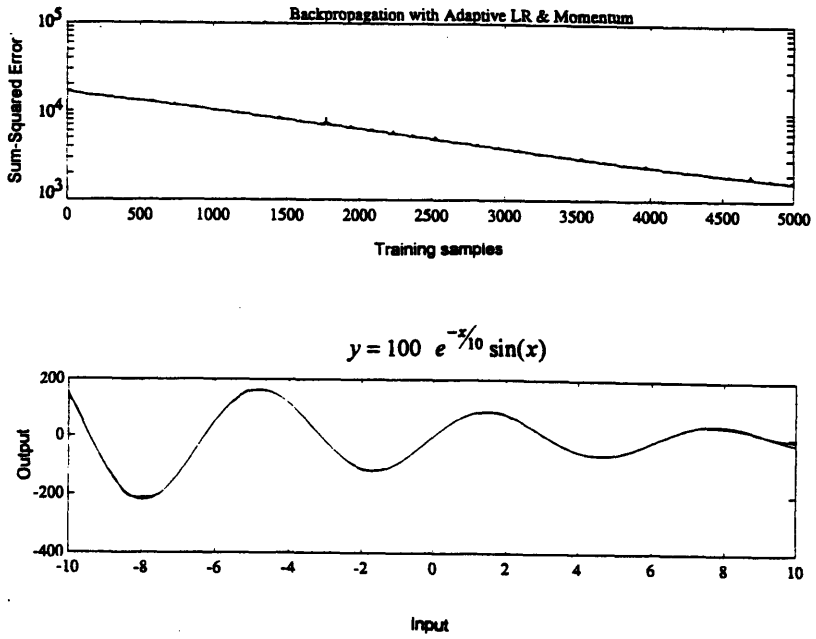


a) with 80 units, which is initialized by a pretained network with 40 units.

Figure 3.24: Function approximation by a one-hidden-layer network.



b) with 80 units, which is initialized by a pretained network with 60 units.



c) with 80 units, which is initialized by a pretained network with 70 units.

Figure 3.24: Function approximation by a one-hidden-layer network.

This finding suggests that one should modify the network architecture continuously during the training process. Figures 3.25 to 3.28 illustrate the strategy. Modification of the architecture is achieved by increasing the number of units in its only hidden layer, 20 units at a time , whenever the gradient between the sum-squared error and amount of training cycles reaches zero. The architecture is modified until a desired sum-squared error is reached. The initialized network has 40 total units and ends up having 80 total units at the end of training process. The modification of architecture is performed twice after being trained through 500 and 1000 training cycles. Its performance can be compared with that of the conventionally trained 80-unit network shown in Fig 3.18.

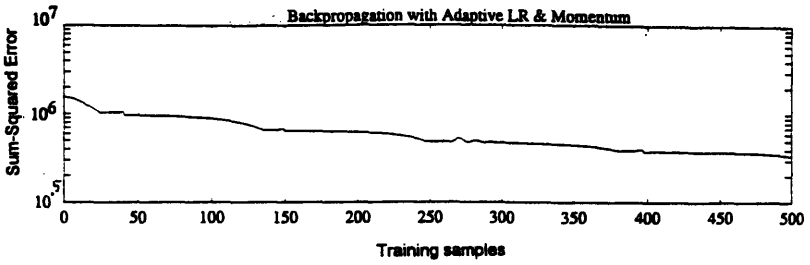


Figure 3.25: Performance of a one-hidden-layer network with 40 units.

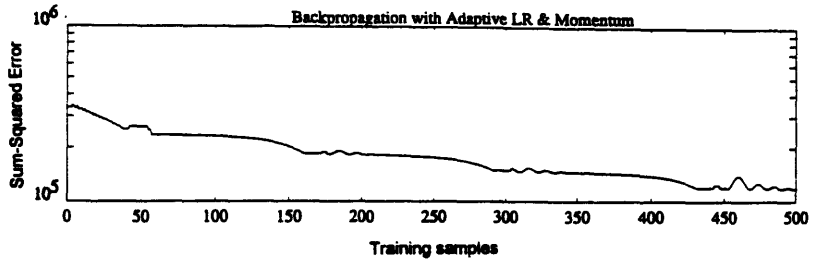


Figure 3.26: Performance of a one-hidden-layer network with 60 units, which is initialized by a pretrained network with 40 units (Fig 3.25).

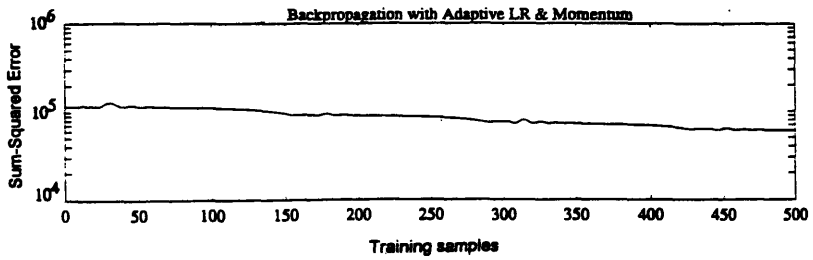


Figure 3.27: Performance of a one-hidden-layer network with 80 units, which is initialized by a pretrained network with 60 units (Fig 3.26).

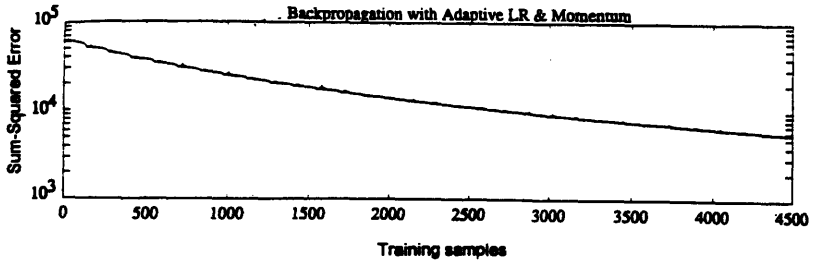


Figure 3.28: Performance of a one-hidden-layer network with 80 units after being pretrained in Fig 3.27.

Although this method may not be as computational efficient as the training that starts with the optimum number of units from the beginning, it still requires less work than the Cross-Validation process. The method automatically helps users avoid using too small or too large networks since it starts with the minimal number of total units, and stops increasing the size when the desired level of accuracy is reached. This method also avoids converging to a local minima during training since the architecture is modified whenever the error gradient reaches zero. However, further testing has to be done on the application of this architecture modification method to other function approximation problems.

Optimizing the network architecture for a classification problem is more complicated since a network with more hidden layers may outperform a one-hidden-layer network. Therefore, more configurations have to be considered. The disadvantage in computational cost of multilayer networks may be factored into the final decision. One has to investigate a number of possible architectures before choosing the final architecture that is most appropriate for a specific application. Given all the difficulties mentioned, Cross-Validation is still the most practical method for optimizing the network architecture for classification problems (Liu, 1995).

3.3 Radial Basis Function Network (RBFN)

The RBFN can be considered as a two-layer feedforward network that has fixed nonlinear transformations with no adjustable parameters in the hidden layer, and linear transformations in the output layer (Broomhead and Lowe, 1988). As shown in Fig 3.29, the network is a fully connected feedforward network with radial basis functions as transfer functions for the interior units, and linear transfer functions at the output units.

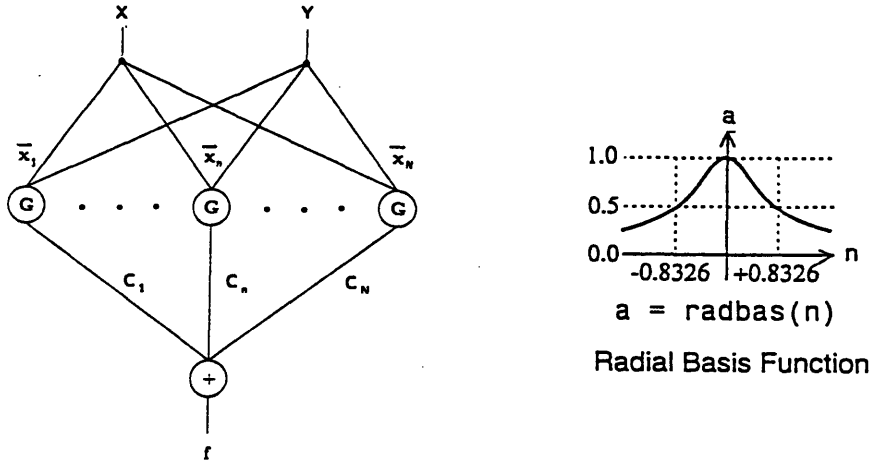


Figure 3.29: An example of a general RBFN.

For a RBFN with an input vector $x \in \Re^n$ and output vector $y \in \Re$, the output y can be described in the simplest form by the equation

$$y = \sum_{i=1}^{n} c_i G_i\left(\|x - x_i\|\right) \quad , \qquad y \in \Re^n, \tag{3.19}$$

where c_i is the weighting parameter of the ith unit; x_i is the center of the radial basis function G_i, and $\|...\|$ is the Euclidean norm on $\Re^n$. The function G_i, or the transfer function of the ith unit, is a continuous function from $\Re^+$ to $\Re$ that has a maximum value at its center and drops off rapidly away from the center. A frequently used class of radial basis functions is the Gaussian function,

$$G_i(x) \quad = \quad e^{-\|x - x_i\|^2}. \tag{3.20}$$

Considering the RBFN in Fig 3.29, the output y can also described by the equation

$$y = \sum_{i=1}^{n} c_i z_i \quad , \text{ where } \; z_i = G_i\left(\|x - x_i\|\right). \tag{3.21}$$

Least Mean Square (LMS) approach employs the sum-squared error of all k input-output pairs as the global error function. The function is described by

$$E = 0.5 \sum_{j=1}^{k} \left(y_j - o_j\right)^2, \tag{3.22}$$

where o_j is the expected output corresponding to y_j, which is the output of RBFN due to input vector x_j. Given the error measure E, the gradient descent algorithm improves c_i by changing c_i by an amount Δc_i proportional to the gradient of E:

$$\Delta c_i = -lcoef \frac{\partial E}{\partial c_i}$$

$$= lcoef \sum_{j=1}^{k} z_i \left(y_j - o_j \right), \tag{3.23}$$

where *lcoef* is a constant called the "learning coefficient."

If the change is made individually for each input vector x_j,

$$\Delta c_i = -lcoef \cdot z_i \left(y_j - o_j \right), \tag{3.24}$$

which is commonly referred to as the Least Mean Square approach, or LMS rule (Rumelhart et al, 1986).

After the radial basis functions and the position of their centers are specified, the only adjustable parameters of the network are the weighting parameter c_i. Since the gradient of the error function is linear to the weighting parameters, the error function of the output does not have local minima, and the parameters can be adjusted by a linear optimization procedure such as the LMS approach. This leads to an optimization procedure that has a very fast convergence rate (Bianchini et al, 1995). This aspect makes RBFN an attractive alternative to the MLN with BP, which requires a time-consuming stochastic optimization procedure.

3.3.1 Ability of RBFN to Approximate Arbitrary Functions

The ability of RBFN to approximate an arbitrary function can be proved by the regularization theory (Girosi et al, 1993, 1995; Bertero et al, 1988; Marroquin et al, 1987; Wahba, 1980, 1990), which relates the radial basis function network to probability and statistics theory. The regularization theory establishes that RBFN can approximate any continuous function within a prespecified error if the network contains all the radial basis functions needed. However, the types of radial basis function required for approximating an arbitrary function cannot be predetermined, and a trial and error method is needed to determine the functions.

3.3.2 Optimum Network Architecture

General clustering algorithms, such as K-Means clustering (Krishnaiah and Kanal, 1982), are usually applied to the input data in order to position the centers of the radial basis functions. The type and number of radial basis functions largely depends on the complexity of the function being approximated. The number of radial basis functions usually increases when the function is more complex, and increases exponentially with the dimension of the input space (Girosi, 1995). Thus RBFN become less practical when the dimension of the input space is high. Since the type and number of radial basis functions that are required to approximate a given function cannot be predetermined, Cross-Validation is usually employed to identify the optimum architecture of the RBFN for a specific task (Liu, 1995).

In case the centers of the radial basis functions are not predetermined and are considered to be adjustable, more parameters have to be considered in the optimization process. This makes the network much more adaptable, but also makes the gradient of error function nonlinear to the network parameters. In this case, a stochastic learning algorithm has to be employed, and the advantage of RBFN's simple training vanishes.

3.4 Performance Comparison Between MLN and RBFN

The performance of RBFN's on the same applications that are applied to MLN's in previous sections is investigated here to provide a comparison between the models. Section 3.4.1 compares results for regression, while that of classification is illustrated in Section 3.4.2.

3.4.1 Comparison of Regression Ability

Figure 3.30 shows the performance of a RBFN, which has 10 Guassian hidden units centered by K-Means clustering algorithm (Krishnaiah and Kanal, 1982), in approximating the function

$$y = 100\ e^{-x/10} \sin(x), \qquad (3.13)$$

where x ranges from -10 to 10 with an interval of 0.1. The result can be compared to that of a MLN with 30 sigmoid units in its hidden-layer that is shown in Fig 3.4.

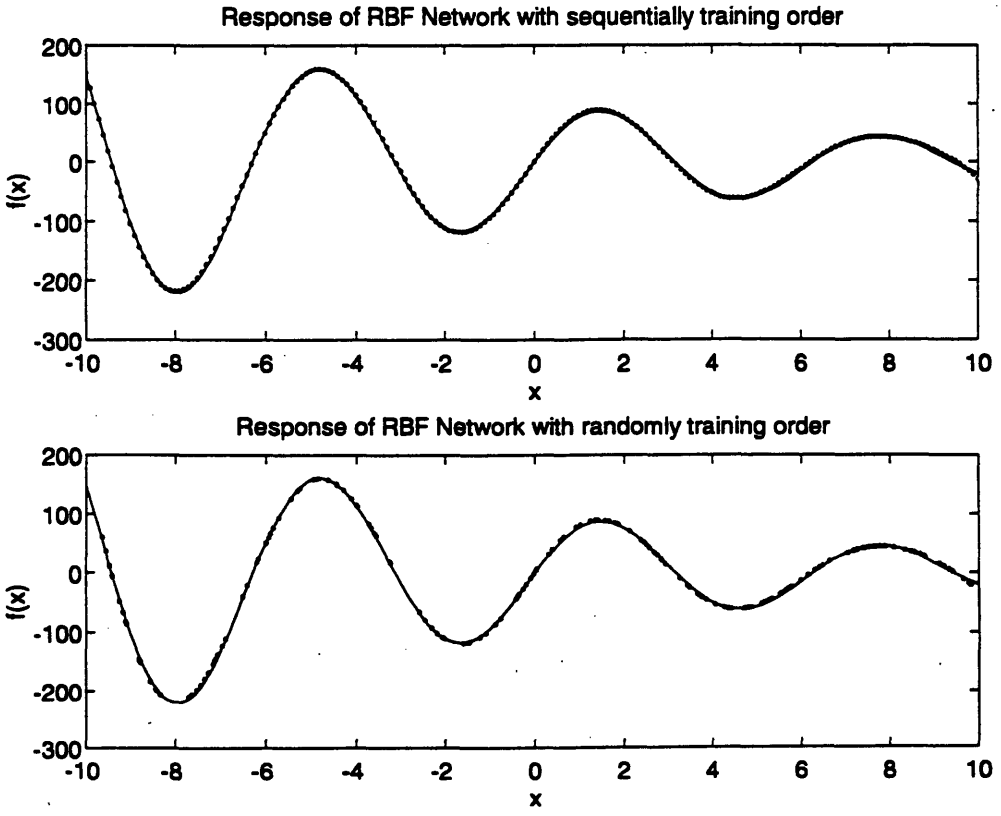


Figure 3.30: Function approximation by RBFN with 10 Gaussian units.

Figure 3.31 compares the performance of a conventional MLN with a RBFN in approximating a two-input-one-output function

$$z = 100\ e^{\frac{-x}{10}\sin(x)} + 100\ e^{\frac{y}{4}\sin(1.5y)}, \qquad (3.25)$$

where x and y range from -5 to 5 with an interval of 0.4. Both networks have one-hidden-layer with 50 processing elements. The RBFN employs Guassian transfer functions, and K-Means clustering algorithm for locating their centers, while the MLN uses a sigmoidal transfer function in the hidden layer, and a linear function in the output layer.

Both comparisons indicate that RBFN, even with fewer processing elements, performs better on these particular functions. RBFN also requires less training time and does not converge to local minima as MLN occasionally does.

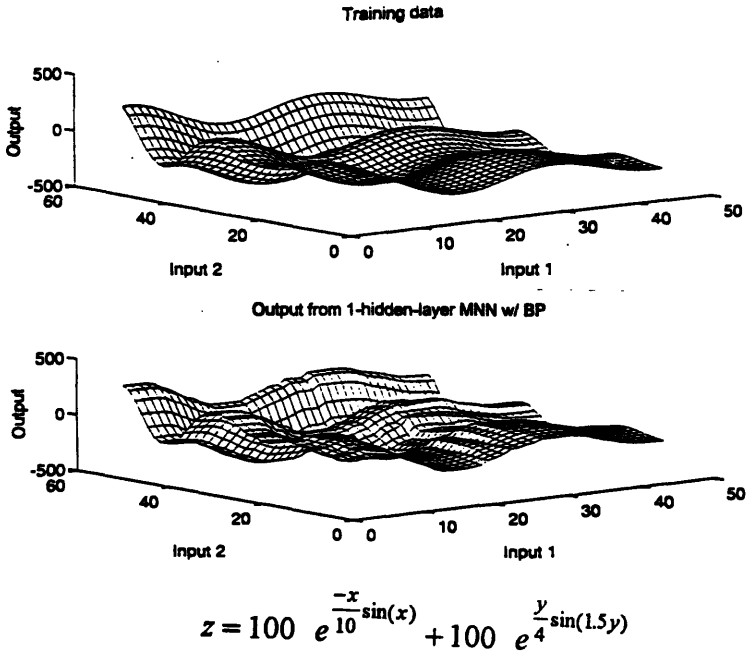
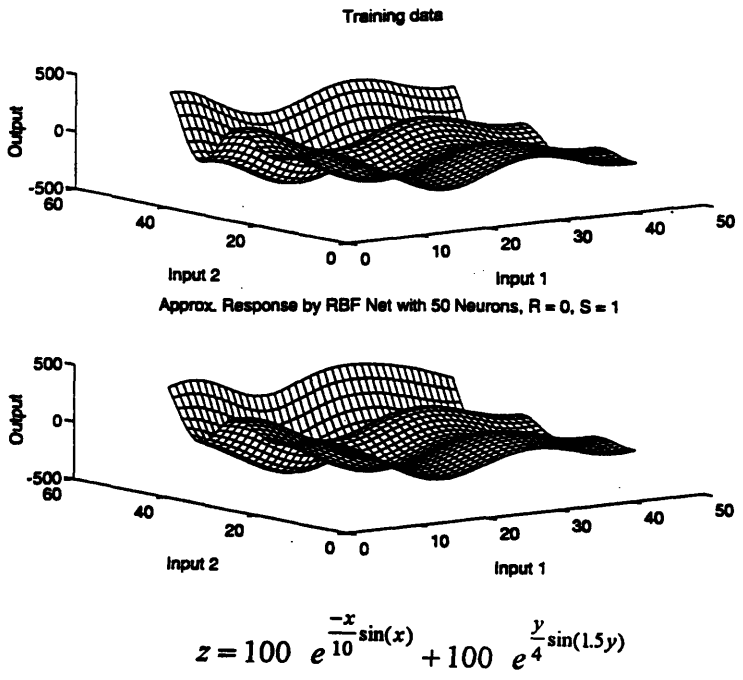


Figure 3.31a: Function approximation by MLN with BP, with total no. of units of 50.



$$z = 100\ e^{\frac{-x}{10}\sin(x)} + 100\ e^{\frac{y}{4}\sin(1.5y)}$$

Figure 3.31b: Function approximation by RBFN with total no. of units of 50.

However, RBFN also has practical difficulties since its performance highly depends on the type and properties of the radial basis functions employed, and it is difficult to find the appropriate type and properties for a specific task (Note that there is no such problem in applying MLN with BP). As demonstrated earlier in Fig 3.6, the function

$$y = 100e^{-20x^2} + 70e^{-30(x-2)^2} + 80e^{-100(x+3)^2} \quad , \qquad (3.15)$$

where x ranges from -10 to 10 with an interval of 0.1, can be very well approximated by a one-hidden-layer network with 10 sigmoidal hidden units. Figure 3.32 shows the performance of 4 different RBFN's, each with 10 Guassian hidden units, in approximating the same function. The Gaussian transfer functions of each RBFN have a specific width ranging from 0.1 to 10.

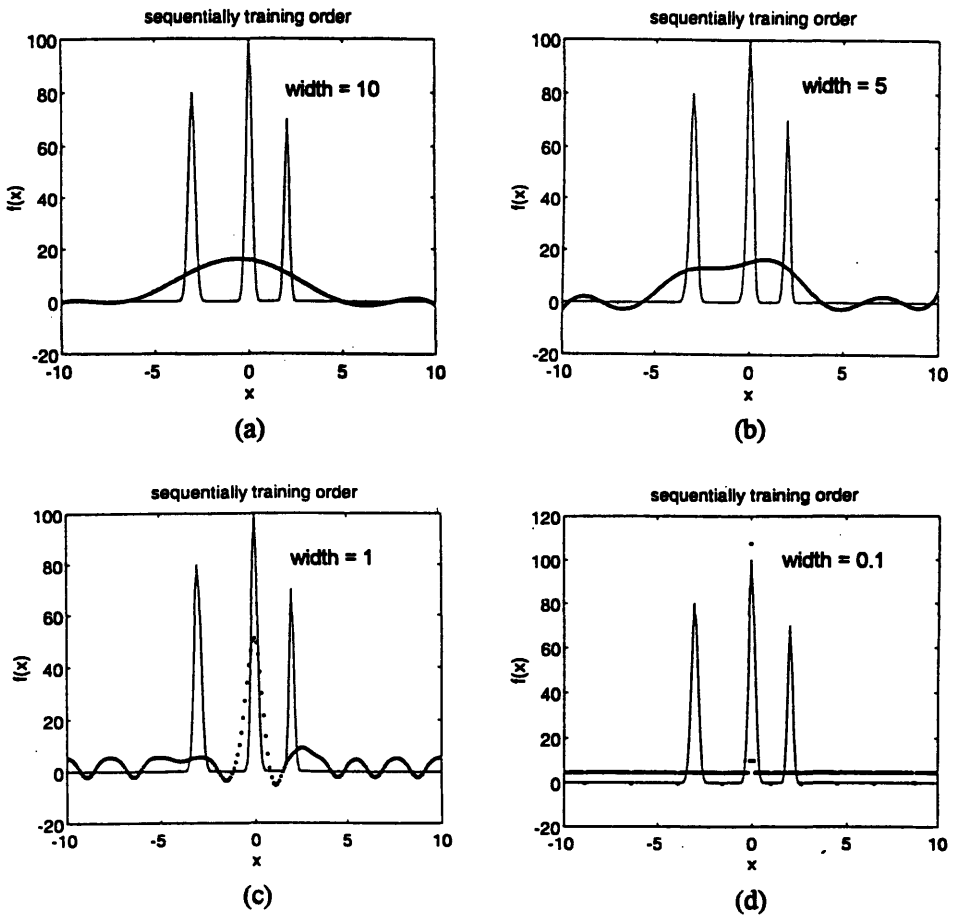


Figure 3.32: Effects of the width of radial basis functions on the approximation ability of RBFN.

The result demonstrates that, despite varying the width of the Guassian transfer function over a broad range, the performance is not satisfactory. The result also shows that a RBFN with a wider Guassian transfer function has better generalization ability, but performs worse in interpolating functions. Figure 3.32d shows the inability of RBFN to interpolate the function's spikes when the centers of the radial basis functions are not properly located. Hence this example also shows how crucial the location of the centers is to the performance of RBFN. Changing the type of radial basis function still would not help since the types of function that improve the interpolating performance would provide worse generalization ability. Detail investigation on the generalization ability of radial basis function networks is discussed by Freeman et al (1995).

3.4.2 Comparison of Classification Ability

The outcome from the classification studies also agrees with that of the regression studies. Figures 3.33 and 3.34 demonstrate the influence of the number of total units to the performance of RBFN in the 4-class classification problem mentioned in Section 3.1.3. RBFN achieves the same performance while requiring less training than the MLN with the same number of total units. Figures 3.35 and 3.36 also show the influence of the width of the radial basis functions to the classification performance, and assure the importance of using radial basis functions with appropriate properties.

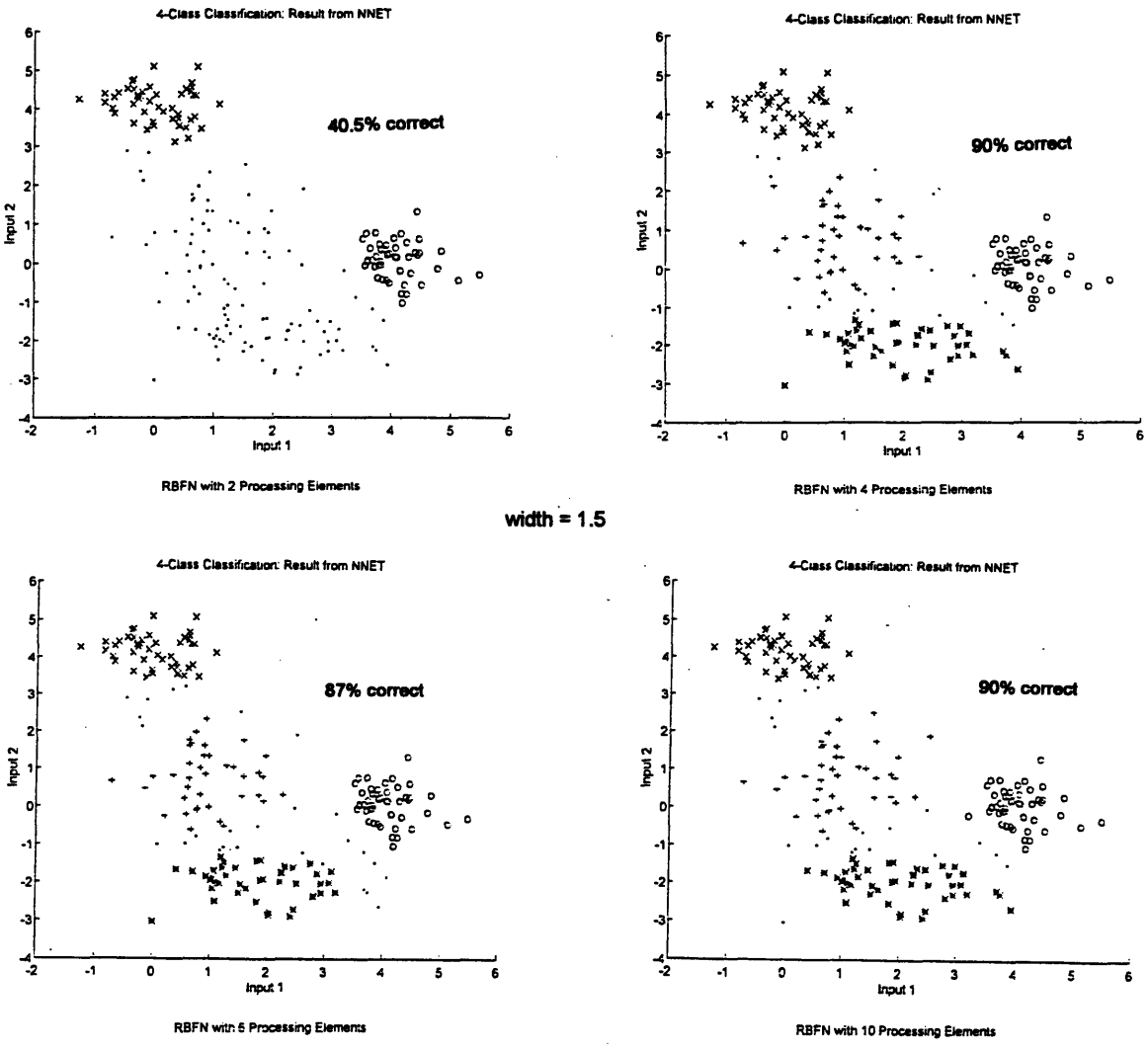


Figure 3.33: Effect of the no. of units on the classification ability of RBFN.

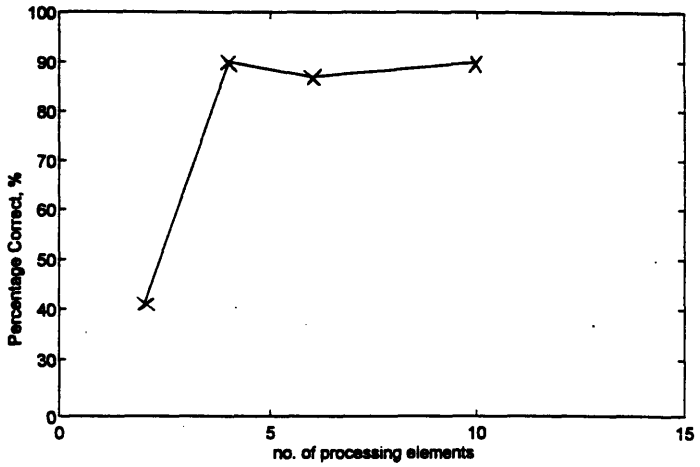


Figure 3.34: Effect of the no. of units on the classification ability of RBFN.

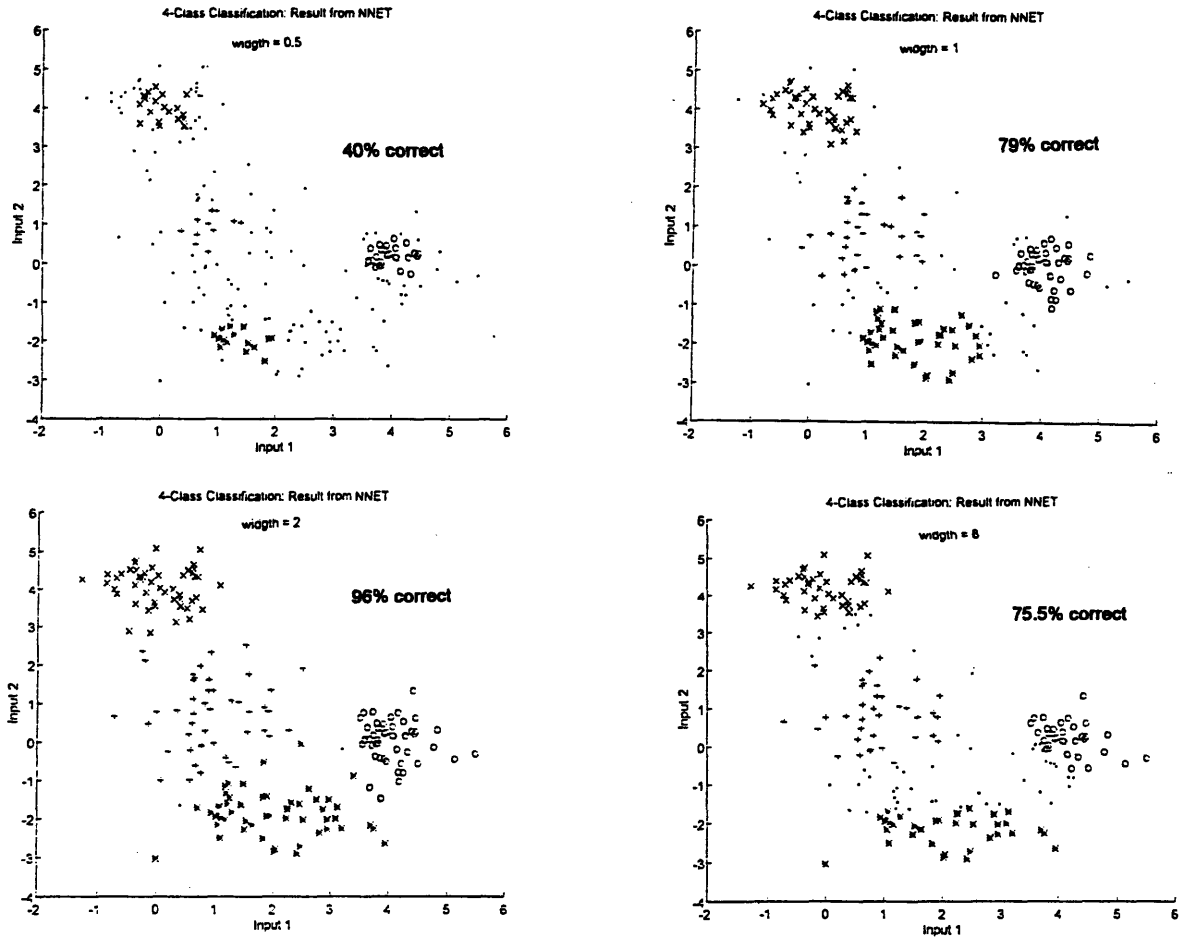


Figure 3.35: Effect of the width of radial basis functions to the classification performance.

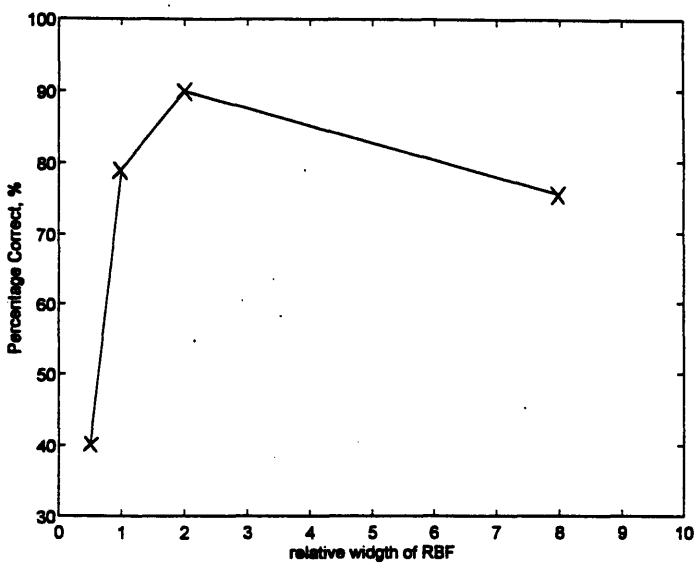


Figure 3.36: Effect of the width of radial basis functions to the classification performance.

3.5 Discussion and Summary

According to the performance of both types of network on these investigations, RBFN clearly requires less training and is more accurate when the appropriate transfer functions and locations of the centers are employed. This can be explained by considering that, since the output of a RBFN is a linear combination of many narrow receptive fields of basis functions, only the parameters that corresponds to the output error are adjusted during a training cycle. This is called "local training effect." On the other hand, MLN adjusts all parameters due to an output error (global training effect), and hence reduce the effect of the previous training cycles in the process (Narrendra, 1992). Moreover, all adjustable parameters of RBFN are linear to the gradient of error function and can be optimized by standard least square techniques, while MLN requires stochastic techniques due to its nonlinearity. Therefore, RBFN is much easier and faster to train.

However, the global effect of parameters in MLN leads to good generalization ability of the network, which is always required as a trade off with the accuracy (Liu, 1995; Musavi et al, 1994; Ling, 1995). RBFN may have problem with generalization when

the function they approximate is highly discontinuous (Girosi, 1993, 1995). MLN also performs better when the function is associated with high dimensional input (Narendra, 1992). Having enough a-priori knowledge about input data is crucial for selecting the configuration of RBFN, which directly reflects the performance of the network in approximating arbitrary functions. MLN does not require such knowledge, and hence is preferred when not much about the function is known. If the fixed parameters of RBFN, such as the centers of the radial basis functions, become adjustable by the network's learning algorithm, the network will be less dependent on a-priori knowledge. However, the network will require a stochastic learning algorithm, which makes the training characteristics and performance of the network similar to MLN (Girosi, 1993, 1995).

The results of these investigations also confirm that the Cross-Validation method can be employed as a general procedure for configuring both types of network. The method should be applied especially when there are factors other than the network architecture that affect the network optimization such as the training time, size and quality of training data set, and generalization ability of the trained network.

Chapter 4

Probability Framework of Neural Networks

4.1 Introduction

In the past, neural networks were considered to be mysterious, and lacking a theoretical foundation. However, since the late 1980's research has established relationship between neural network theory and other fields such as approximation theory, and probability and statistics. In this chapter, feedforward neural networks with back propagation training algorithm and radial basis function networks are discussed from the perspective of these fields. The objective is to provide a better understanding of these neural networks, and thus simplify the development process.

4.2 Probabilistic Model of Feedforward Networks

In this section, a probabilistic model of a simplified feedforward neural network is described. The model demonstrates that the maximum likelihood estimation of the parameters of the probabilistic regression model of a function is equivalent to using a one-layer feedforward network with linear transfer function to approximate the function (Watanabe et al, 1995).

Firstly, the basis of the gradient method used in the back propagation learning algorithm, which is normally used for optimizing a multilayer feedforward network, is described (see Chapter 3 for more detail). Figure 4.1 illustrates a processing element of a neural network.

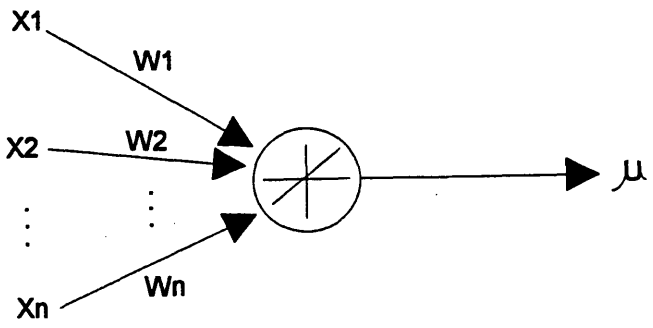


Figure 4.1: A processing element.

The activation of the processing element with a linear transfer function is:

$$\mu = W^T x \ , \tag{4.1}$$

where W is the vector of the weighting parameters of the input connections; x is the input vector; and μ is the output. Suppose that the set

$$\chi = \{(x_i, y_i) \in R^d \times R\}_{i=1}^N \tag{4.2}$$

is the data obtained by random sampling a function f, which belongs to some space of function X defined on R^d, in the presence of noise. The objective of function approximation is to recover the function f, or an estimate of f, from the set of data χ.

Suppose that a processing element is used to approximate the function f, and the error function of the output is the sum-squared error of the output y,

$$J(W) = \sum_{i=1}^{N} \frac{1}{2}(y_i - W^T x_i)^2. \tag{4.3}$$

The gradient method is applied to minimize the error function *J(W)* by finding the gradient of the function

$$\nabla_w J(W) = -\sum_{i=1}^{N} \left(y_i - W^T x_i\right) x_i \quad , \tag{4.4}$$

and then adjusting the parameter $W \in R^d$ in the direction opposite to the gradient,

$$\Delta W_i = \rho \left(y_i - W^T x_i\right) x_i \quad , \tag{4.5}$$

for the *i*th training data pair $\left(x_i, y_i\right)$ given an appropriate learning rate ρ. Successive adjusting corresponding to every data pair will provide a set of parameter *W* that minimizes *J(W)* and estimates *f*.

4.2.1 Maximum Likelihood Estimation Model

Consider a Gaussian density function,

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2\sigma^2}(x-\mu)^2} \quad , \tag{4.6}$$

where μ is the mean, and σ^2 is the variance, of data set χ. This function can be viewed different ways, depending on which parameters are considered known or unknown. By assuming μ and σ^2 known, Eq. 4.6 can be considered as

$$f(x; \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2\sigma^2}(x-\mu)^2} \quad , \tag{4.7}$$

which is the probability of x, given μ and σ^2. Similarly, if the data set χ is known, the same equation is now considered as

$$L(\mu,\sigma;x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2\sigma^2}(x-\mu)^2} , \qquad (4.8)$$

which is the likelihood of μ and σ^2, given data x.

The maximum likelihood estimate of μ and σ is, by definition, the estimated value $\hat{\mu}$ and $\hat{\sigma}$ that maximize $L(\mu,\sigma\ ;x)$. Intuitively, it corresponds to the value of μ and σ that best agrees with the actually observed samples.

Figure 4.2 illustrates a system that has input $X \in R^d$, and generate output μ. Only the real output y, with noise E, can be observed. Suppose that μ can be described by a set of parameter $W = \{w_i\}_{i=1}^{d}$, or

$$\mu = W^T X \ . \qquad (4.1)$$

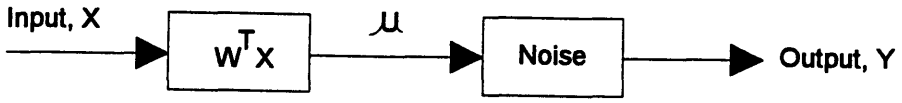


Figure 4.2: A simulated system.

The sensory output of the system can be now considered as

$$y = W^T X + E. \qquad (4.9)$$

Assuming that there is Gaussian noise,

$$f(E;\ \mu_E,\ \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2\sigma^2}E^2} \tag{4.10}$$

with zero mean, the density function of the output y can be described by

$$f(y;\ \mu\ ,\ \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2\sigma^2}(y-\mu)^2}$$

$$= \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2\sigma^2}(y-W^T x)^2} \tag{4.11}$$

The objective of function approximation is to estimate the output of the system by performing the regression of the output given the data set

$$\chi = \left\{\left(x_i,\ y_i\right)\right\}_{i=1}^{N} \quad , \tag{4.12}$$

which is generated by the model.

If σ is assumed as a known variable, Eq. 4.11 becomes the likelihood of μ and σ, given x_i ,

$$L(\mu,\sigma\ ;x_i) = L(W;x_i) \qquad ; \qquad \mu = W^T x$$

$$= \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2\sigma^2}(y_i-W^T x_i)^2} \tag{4.13}$$

Given that χ is an independent, identically distributed data set, the likelihood of W given χ is

$$L(W;\chi) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2\sigma^2}(y_i - W^T x_i)} \tag{4.14}$$

To simplify the maximization of Eq. 4.14, the log likelihood,

$$\log[L(W;\chi)] = l(W;\chi)$$

$$= \left[\log\left(\frac{1}{\sqrt{2\pi}\sigma}\right)\left(\frac{-1}{2\sigma^2}\right)\right] \sum_{i=1}^{N}\left(y_i - W^T x_i\right)^2, \tag{4.15}$$

is maximized instead. Since log function is monotonic, maximizing log of a function still maximize the function. It is interesting that maximizing the log likelihood of W given χ is exactly the same as minimizing the sum-squared error of the output,

$$J(W) = \sum_{i=1}^{N} \frac{1}{2}\left(y_i - W^T x_i\right)^2, \tag{4.3}$$

of a processing element previously described in Section 4.2. Applying the gradient method to maximize Eq. 4.15 gives

$$\nabla_w\left(\sum_{i=1}^{N}\left(y_i - W^T x_i\right)^2\right) = -\sum_{i=1}^{N}\left(y_i - W^T x_i\right)x_i$$

$$\Delta W_i = \rho\left(y_i - W^T x_i\right)x_i \ , \tag{4.16}$$

where ρ is the learning coefficient. By comparing Eqs. 4.5 to 4.16, it is now obvious that the back propagation learning algorithm of a processing element that approximates a

function can be viewed as the effort to find the maximum likelihood estimation of the parameters that associated to the probabilistic model of the function (Ney, 1995).

4.2.2 Choice of Transfer Function: A Probabilistic View

Based on the probabilistic approach used in the previous section, this section demonstrates a technique for selecting the type of transfer function of the feedforward networks for different tasks (Watanabe et al, 1995).

For classification problems, which the output tends to be yes or no, true or false, rather than the real number as in regression problems, it is more rational to model the problem using Bernoulli density function instead of Gaussian. Suppose that there is a system that has input $X \in R^d$, and generates output μ. Only the real output y, with some uncertainty, can be observed (see Fig 4.3).

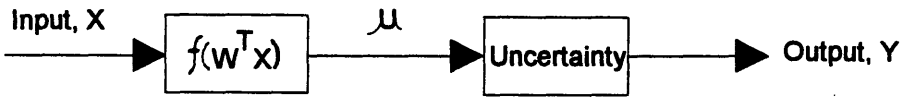


Figure 4.3: A simulated system.

Suppose that μ can be described by a set of parameter $W = \{w_i\}_{i=1}^{d}$ as follows.

$$\mu = f(W^T X) \quad ; \quad X, W \in R^d, \tag{4.17}$$

where f is an arbitrary function.

Using Bernoulli's probability model, $\mu \in [0,1]$ can be considered as the probability of success, and hence the probability of the output y given the probability of success μ can be demonstrated as

$$P(y;\mu) = \mu^{y}(1-\mu)^{1-y} \quad ; \quad y = 0 \text{ or } 1 \quad . \tag{4.18}$$

Given an independent, identically distributed data set

$$\chi = \{(x_i, y_i)\}_{i=1}^{N} \quad ; \quad x_i \in R^d ; \; y_i = 0 \text{ or } 1; \tag{4.19}$$

which is generated by the model, the likelihood of W given x_i is

$$L(W \; ; \; x_i) = \mu_i^{y_i}(1-\mu_i)^{1-y_i} \quad , \tag{4.20}$$

where

$$\mu_i = f(W^T x_i). \tag{4.21}$$

Therefore, the likelihood of W given χ is

$$L(W;\chi) = \prod_{i=1}^{N} \mu_i^{y_i}(1-\mu_i)^{1-y_i} \quad . \tag{4.22}$$

Consider the loglikelihood,

$$l(W \; ; \; \chi) = \sum_{i=1}^{N} \left(y_i \log(\mu_i) + (1-y_i)\log(1-\mu_i)\right) \quad , \tag{4.23}$$

the term in the summation, which is called "cross entropy," can be perceived as a measure of closeness between y and μ.

A processing element (shown in Fig 4.1) with an arbitrary transfer function f can be used to approximate the output of the system. The cross entropy term can be employed as the error function of the feedforward network for classification problems,

$$J(W) = -\sum_{i=1}^{N} \left(y_i \log(\mu_i) + (1 - y_i) \log(1 - \mu_i) \right). \tag{4.24}$$

Applying the gradient method to minimize the error function $J(W)$ gives

$$\frac{\partial J}{\partial w_j} = -\left(\sum_{i=1}^{N} \left(\frac{y_i}{\mu_i} \frac{\partial \mu_i}{\partial w_j} - \frac{1 - y_i}{1 - \mu_i} \frac{\partial \mu_i}{\partial w_j} \right) \right)$$

$$= -\left(\sum_{i=1}^{N} \left(\frac{y_i}{\mu_i} - \frac{1 - y_i}{1 - \mu_i} \right) \frac{\partial \mu_i}{\partial w_j} \right)$$

$$= -\left(\sum_{i=1}^{N} \left(\frac{y_i - \mu_i y_i - \mu_i + \mu_i y_i}{\mu_i (1 - \mu_i)} \right) \frac{\partial \mu_i}{\partial W_j} \right) \quad , \quad \mu_i = f(W^T x_i) = f(z_i)$$

$$= -\left(\sum_{i=1}^{N} \left(\frac{y_i - \mu_i}{\mu_i (1 - \mu_i)} \right) f'(z_i) x_{ij} \right) \quad . \tag{4.25}$$

Hence, the learning rule of the classification network is

$$\Delta W_i = \rho \left(\frac{y_i - \mu_i}{\mu_i (1 - \mu_i)} \right) f'(z_i) x_{ij} \quad , \tag{4.26}$$

which is the adjustment of the parameters in the direction opposite to the gradient. The choice of transfer function $f(z)$ should now be the one that its derivative, $f'(z)$, can cancel the variance term of the Bernoulli density function, $\mu(1 - \mu)$, from the learning rule.

If the network employs the sigmoidal transfer function ,

$$\mu = f(z) = \frac{1}{1 + e^{-z}} \quad , \tag{4.27}$$

the derivative of the function is

$$f'(z) = \mu\ (1-\mu) \quad . \tag{4.28}$$

Therefore the learning rule can be reduced to

$$\Delta W_i = \rho\ (y-\mu)x_i \quad , \tag{4.29}$$

which is similar to the learning rule of the feedforward network for regression problem (see Eq. 4.5). This result suggests that the type of transfer function used in the output layer of the networks for different tasks should be carefully selected in order to increase the reliability of approximation.

The same probabilistic approach can also be employed to find the proper transfer function for other applications. The step by step procedure of the approach for is shown below.

- Pose the problem.
- Develop a probabilistic model $P(y \mid x;\theta)$, where y is the output; x is the input, and θ is the parameter.
- Form the log likelihood, or the error function,

$$l(\theta\ ;\ \chi) = \sum_{i=1}^{N} \log\left(P(y_i \mid x_i;\theta)\right) \quad .$$

- Apply the estimation principle

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}}\, l(\theta ; \chi) \ .$$

- Choose a learning algorithm

  Select an optimization procedure for finding the maximum likelihood estimation of parameters θ.

By following the procedure, choices of transfer function for other types of problem can be determined as shown in the following table.

Problems	**Probability Model**	**Error function**	**Transfer function of the output layer**
Regression	Gaussian	Sum-squared Error	Linear
Classification	Bernoulli (2-way)	Cross Entropy	Logistic
	Multinomial (multi-way)	Cross Entropy	Softmax
Counting	Poisson	Cross Entropy	Exponential
Time to failure	Gamma	Cross Entropy	Exponential
	Weibull	Cross Entropy	Exponential

Table 4.1: Types of transfer function for different neural network application.

In order to properly design feedforward neural networks with back propagation learning algorithm, thoroughly understanding of how the networks work, and the nature of the problem that the networks have to solve, are required. Selecting the proper type of

transfer function of the output layer is one important decision that can be made given the knowledge. Although some improperly designed networks work effectively for some problems, properly designed networks always perform better. Therefore, it is essential that neural networks' users also understand how they works and apply them properly.

4.3 Probabilistic Model of Radial Basis Function Networks

In this section, the relationship between the Radial Basis Function Network (RBFN) and the probability and statistics theory is described through the regularization theory (Girosi et al, 1993, 1995; Bertero et al, 1988; Marroquin et al, 1987; Wahba, 1990). The relationship is informally shown, without demonstrating the related mathematical issue.

Suppose that the set

$$\chi = \left\{\left(x_i,\ y_i\right) \in R^d \times R\right\}_{i=1}^{N} \tag{4.30}$$

is the data obtained by random sampling a function f, which is defined on R^d. In case of noisy data, the function f can be represented as

$$f(x_i) = y_i + E_i, \quad i = 1,\ldots,N \tag{4.31}$$

where the noisy term E_i is a random independent variable with a given distribution.

A probabilistic approach is applied in order to recover the function f. The function is considered as a random field with a known a-priori probability distribution. Let's define:

- $P[f \mid X]$ is the conditional probability of the function f given the examples X.

- $P[X|f]$ is the conditional probability of X given f. In other word, if the function f is corresponded to the data, this is the probability that the set of output $\{y_i\}_{i=1}^{N}$ is obtained by random sampling the function f at the data point $\{x_i\}_{i=1}^{N}$.

- $P[f]$ is the a-priori probability of the random field f. $P[f]$ covers a-priori knowledge of the function, and can be used to apply constraints on the model by assigning significant probability only to those functions that satisfy those constraints.

Assuming that the probability distributions $P[f|X]$ and $P[f]$ are known, the posterior distribution $P[f|X]$ can be determined by applying Bayes rule,

$$P[f|X] \propto P[X|f]\ P[f]. \tag{4.32}$$

Assuming that the noise E is normally distributed with variance σ, the probability of X given f is

$$P[X \mid f] \propto e^{\frac{-1}{2\sigma^2}\sum_{i=1}^{N}(y_i - f(x_i))^2} \tag{4.33}$$

where σ is the variance of the noise.

The model for the a-priori probability distribution $P[f]$ is chosen in correspondence with the discrete case (when the function f is defined on a finite subset of an n-dimensional lattice) for which the problem can be formalized (Marroquin et al, 1987). The a-priori probability can be written as

$$P[f] \propto e^{-\alpha\phi[f]} \tag{4.34}$$

where $\phi(f)$ is the smoothness functional which will be explained in detail later, and α is a positive real number. This type of probability distribution provides high probability to those functions which has the term $\phi(f)$ small, and hence gives a-priori knowledge of the system.

According to the Bayes rule, the posterior probability of function f can be written as

$$P[f|X] \;\alpha\; e^{\frac{-1}{2\sigma^2}\left[\sum_{i=1}^{N}(y_i-f(x_i))^2+2\alpha\sigma^2\phi[f]\right]} \qquad (4.35)$$

The function f can be estimated from this probability distribution by finding Maximum A Posteriori (MAP) estimate, which considers the function that maximizes the posterior probability $P[f|X]$, or minimizes the exponent in Eq. 4.35. The MAP estimate of f is actually the function that minimize the functional

$$H(f) = \sum_{i=1}^{N}\left(y_i - f(x_i)\right)^2 + \lambda\,\phi\,[f]\;, \qquad (4.36)$$

where $\lambda = 2\sigma^2\alpha$ (Girosi, 1993, 1995). The first term enforces closeness to the data, while the second term enforces the smoothness. The smoothness is defined by a smoothness functional $\phi(f)$, which its lower value corresponds to smoother functions. The parameter λ is called "regularization parameter", which is used to control the trade-off between the level of noise and the strength of the a-priori assumptions about the solution. In another perspective, the parameter also controls the compromise between the degree of smoothness and the closeness of the solution to the observed data.

The smoothness is actually a measure of the oscillatory behavior of the function. Therefore, within a class of differentiable functions, one function is defined to be smoother

than another if it has less oscillation. On the other hand, if the functions are considered in the frequency domain, one is smoother than another if it has less energy at high frequency (smaller bandwidth). The high frequency content of a function can be measured by high-pass filtering the function, and then measuring the power (L_2 norm) of the result (Girosi, 1993, 1995). This leads to defining the smoothness functionals of the form

$$\phi(f) = \int_{R^d} ds \frac{|\tilde{f}(s)|^2}{\tilde{G}(s)} \quad , \tag{4.37}$$

where ~ verifies the Fourier transform. $\tilde{G}$ is defined as a positive function that drops to zero as $\|s\| \to \alpha$ so that $1/\tilde{G}$ is a high-pass filter. For a well-defined class of function G, the function $\phi(f)$ is semi-norm with a finite dimensional null space N (Madych and Nelson, 1990; Dyn, 1991). There are several possible choices for the smoothness functional $\phi(f)$ that can be written in the form of the above equation. If $\tilde{G}$ is also assumed to be symmetric, so that its Fourier transform G is real and symmetric (e.g. Radial Basis Functions), it can be proved that the function that minimizes the functional $H(f)$ has the form

$$f(x) = \sum_{i=1}^{N} c_i G(x - x_i) + \sum_{a=1}^{k} d_a \varphi_a(x) \tag{4.38}$$

where $\{\varphi_\alpha\}_{a=1}^{k}$ is a basis in the k-dimensional null space N, and the coefficients d_a and c_i satisfy the linear system

$$(G + \lambda I)c + \psi^T d = y \quad , \quad \psi c = 0, \tag{4.39}$$

where I is the identity matrix (Girosi, 1993, 1995). The existence of the solution of this linear system is guaranteed by the existence of the variational problem (Girosi, 1990).

The approximation procedure of this type can also be shown as a network with one hidden layer, and this type of network is called "Regularization Network" (Girosi, 1993, 1995). Radial Basis Function network (see Chapter 3), which its smoothness functions satisfy the condition

$$\phi\ [f(x)] = \phi\ [f(Rx)] \tag{4.40}$$

for any rotation matrix R, is also classified in this category. This choice of smoothness function indicates that the a-priori assumption assumes equal relevancy of all variables, and no privilege directions. There are many radial basis functions that satisfy these conditions. For example, if the approximation scheme employs the smoothness function of the form

$$\phi\ [f] = \int_{R^d} ds\ \ e^{\frac{\|s^2\|}{\beta}} \left|\tilde{f}(s)\right|^2 \tag{4.41}$$

where β is a fixed positive number, a Guassian function

$$\tilde{G}(s) = e^{\frac{-\|s\|^2}{\beta}} \tag{4.42}$$

is considered as the basis function of the approximation scheme,

$$f(x) = \sum_{i=1}^{N} c_i G(x - x_i) + \sum_{a=1}^{k} d_a \varphi_a(x) \quad . \tag{4.43}$$

Since the Gaussian function is positive definite, Eq. 4.43 can be reduced to

$$f(x) = \sum_{i=1}^{N} c_i G(x - x_i) \quad , \tag{4.44}$$

which is the mathematical form of the radial basis function network (Poggio and Girosi, 1989; Yuille and Grzywacz, 1988).

Chapter 5

Candidate Neural Network Systems for Structural Damage Diagnosis

5.1 Introduction

Considering the ability of artificial neural networks to approximate functions, one promising application area is pattern classification. Our particular interest is in evaluating the potential applicability of the pattern mapping ability of neural networks for remote sensing and damage diagnosis of engineering structures.

To carry out this assessment, a basic neural network-based diagnosis system is developed and applied to the single-point damage cases. The basic system is then extended to a general architecture for neural network-based diagnosis system, which is applicable to multiple-point damage. This chapter is concerned with the overall design approach. Details of application are presented in Chapter 6 and 7.

5.2 Basic Neural Network-Based Diagnosis System

Figure 5.1 shows the architecture of a basic neural network-based diagnosis system. There are 4 major components; the structure and its numerical model; a data preprocessing unit; a neural network for detecting the location of damage (NNET1), and a neural network for detecting the extent of damage (NNET2).

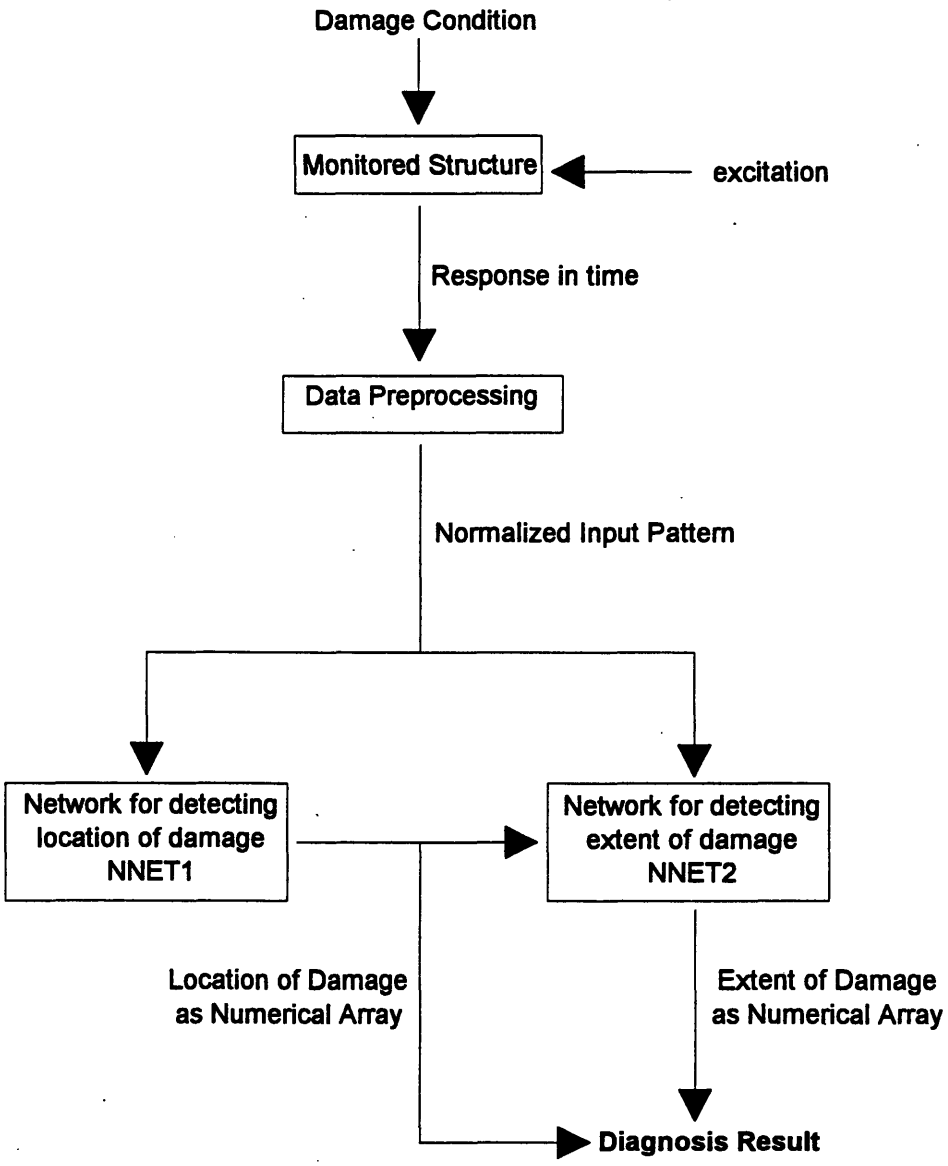


Figure 5.1 A basic neural network-based diagnosis system.

The diagnosis system operates as follows. A specified excitation is introduced in the monitored structure, with or without damage. The resulting structural response is passed to the "Data Preprocessing Unit." This data is transformed into a numerical vector pattern called the "Normalized Input Pattern," which is used as the input for the neural networks.

Network NNET1 converts each normalized input pattern to a numerical vector that can be interpreted as defining the location of damage of the monitored structure. A similar operation is carried out by the second network NNET2. The input in this case is a combination of the normalized input pattern and the output of NNET1 (the predicted location of damage by NNET1 in the form of binary vector). The output of NNET2 is a numerical vector that defines the extent of damage at the given damage location. Once the networks are trained, the diagnosis system should be able to predict the location and extent of damage of the structure from the time history response data given an excitation.

Hence the issue is how to train NNET1 and NNET2 to predict with an acceptable level of accuracy. For the best prediction performance, the training data should cover all the possible damage conditions that may be experienced by the monitored structure. Since this data cannot be obtained from the monitored structure, a simulation model of the real structure is required to generate the training data for both NNET1 and NNET2. If the model adequately represents the real structure, the neural network-based diagnosis system trained by simulation data should be able to effectively predict the location and extent of the damage of the real structure, given the time response of the excited structure. This idea of training with simulation data is called "Simulated training approach" (Elkordy, 1993). Selection of an appropriate simulation model, the feasibility of a diagnostic system trained by simulation data, and the practical applicability of the simulation training approach, are discussed in detail in Chapter 8.

Figure 5.2 shows the training procedure for NNET1. For each training cycle, the time response of the model corresponding to each damage condition is determined via

simulation and is passed to the data preprocessing unit, which transforms the time response data into a normalized input pattern. The normalized input pattern is then passed to NNET1. Given its weighting parameters, NNET1 predicts the location of damage in the form of a numerical vector. The difference between this vector and the vector that indicates the real location of damage is then fed back, and the supervised training algorithm adjusts the parameters according to the error. This process is continued, using different damage cases for each training cycle, until the prediction accuracy of NNET1 is within the desired limit.

The training procedure for NNET2, shown in Fig 5.3, is similar to that of NNET1 except that it uses the binary vector that indicates the location of damage, in addition to the normalized input pattern, as the input. The output of NNET2, which is a numerical vector, is then used to predict the extent of damage at the given location. The difference between the output of NNET2 and the binary vector that indicates the real extent of damage of the damage case, is then employed to adjust the parameters via a supervised training algorithm. The same process is carried on for different damage cases until the desired prediction accuracy of NNET2 is obtained.

If the structure of interest involves a small number of possible damage conditions, application of one basic diagnosis system to monitor the whole structure is feasible. However, due to the limited ability of the neural networks used, a more elaborate system is needed to deal with complex structural damage situations. The following two sections describe how the basic neural network-based diagnosis system can be applied to deal with single-point damage, and then extended to handle multiple-point damage.

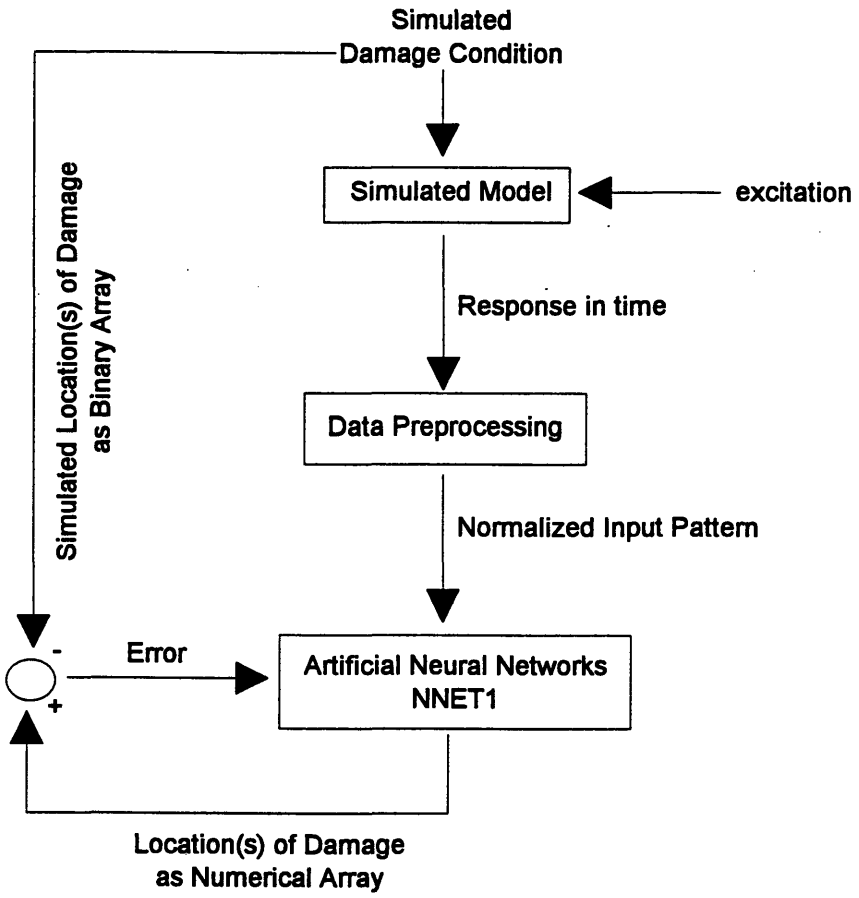


Figure 5.2 The training process of neural network for detecting location of damage, NNET1.

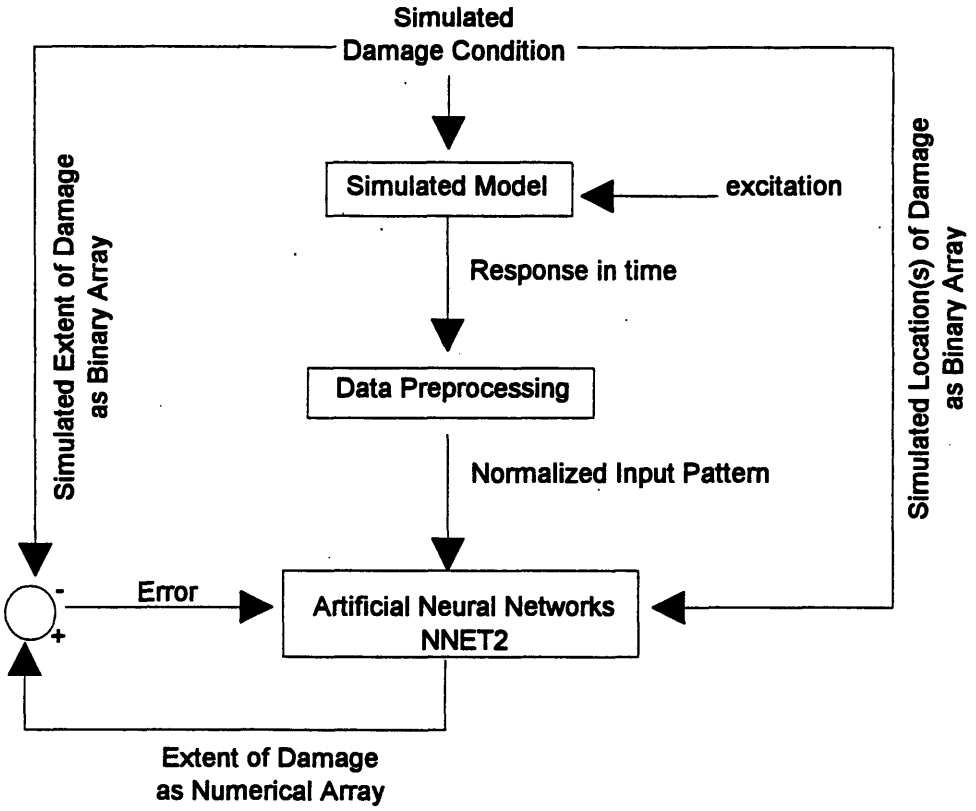


Figure 5.3 The training process of the neural netwok for recognizing the extent of damage.

5.3 Single-Point Damage Diagnosis

5.3.1 Definition of Single-Point Damage

Assuming that the monitored structure is modeled as an assemblage of elements, and damage is introduced by changing the properties of specific elements, single-point damage condition is defined as the case where the properties of a single element are modified at a time. Figure 5.4 illustrates the case of a 2-span beam composed of 22 beam elements, each of which has the potential to be damaged. It follows that there are 22 possible damage states for NNET1 to define when the single-point damage condition is assumed. In this case, the amount of the training data for the neural network is also limited to the same order. The low number of damage cases makes it feasible to use one basic diagnosis system to perform damage diagnosis of the whole structure.

5.3.2 System Design

In performing single-point damage diagnosis, the basic neural network-based diagnosis system (see Section 5.2) can be directly applied. A simulation model of the structure is required for creating the training and testing data, and only two neural networks are used. The change of the vibrational signature of the structure is employed to identify the change in the structure's condition which is then related to damage.

The operation of the diagnosis system follows the flow chart shown in Fig 5.1. The time history response due to the specified excitation is transformed to a "vibrational signature" corresponding to the damage condition of the structure. Two types of vibrational signature are proposed: mode shapes, and spectrums of the response at various locations. The vibrational signature is then further processed into the numerical vector called the "normalized input pattern" that NNET1 uses as input to predict the location of damage on the structure. NNET1's output is also used as input to NNET2, in addition to

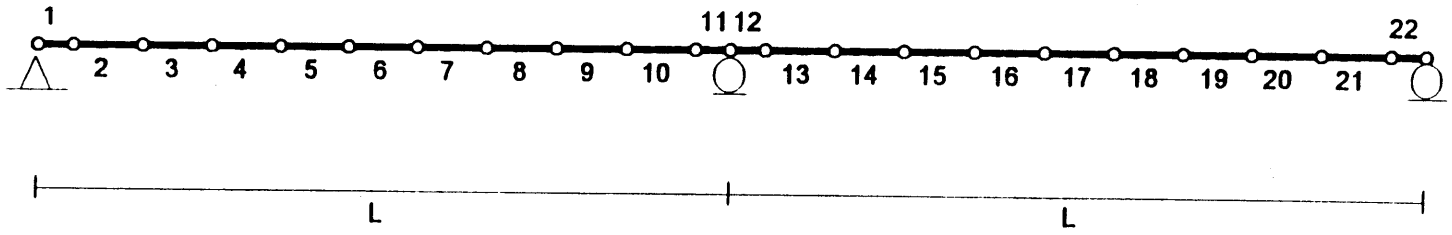


Figure 5.4:

the normalized input pattern, for predicting the extent of damage. The training procedure of NNET1 and NNET2 also follows the flow chart shown in Figs 5.2 and 5.3.

More details concerning the design strategy for a neural network system for single-point damage diagnosis of an idealized 2-span beam model, and results of the numerical simulation studies, are presented in Chapter 6. Two choices of excitation and vibrational signature, ambient excitation & mode shape approach and prespecified impulse excitation & response spectrum approach, are investigated and evaluated.

5.4 Multiple-Point Damage Diagnosis

5.4.1 Definition of Multiple-Point Damage

For most applications, the damage condition is such that several locations of the structure of interest can be damaged at any specific time. In this case, a diagnosis system that is designed based on the assumption of single-point damage is not applicable, and a new diagnosis system that can deal with multiple-point damage is required. By definition, multiple-point damage refers to the case where damage occurs at more than one location. If the simulation model of the monitored structure is made of several elements, and the damage is simulated by the change of the properties of specific elements, the number of possible different damage states is now in the order of the factorial of the total number of the elements. For example, the 2-span beam shown in Fig 5.4 has 22 beam elements, and involves the factorial 22 ($1.124*10^{21}$) different damage states. For single-point damage, only 22 different states are involved.

Multiple-point damage condition makes the number of possible damage cases approximately in the order of the factorial of the number of elements. The number of training samples required for a diagnosis system is also in the same order, which is too large to be effectively used to train a basic diagnosis system. In order to deal with the increased number of possible damage cases associated with multiple-point damage, the

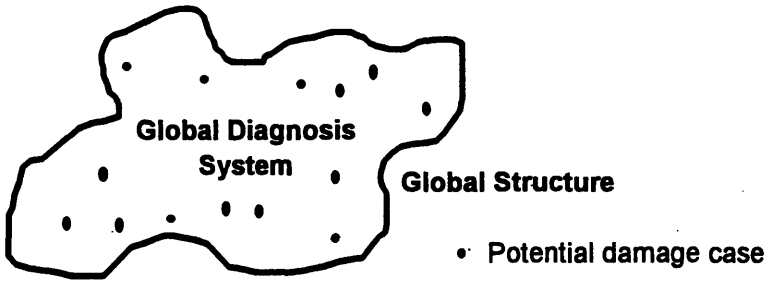
basic diagnosis system is expanded by incorporating a number of neural network-based diagnosis systems operating in a hierarchical way. Details of this approach are provided in the following section.

5.4.2 General Architecture of Neural Network-Based Diagnosis System: Global-Local Structural Diagnosis

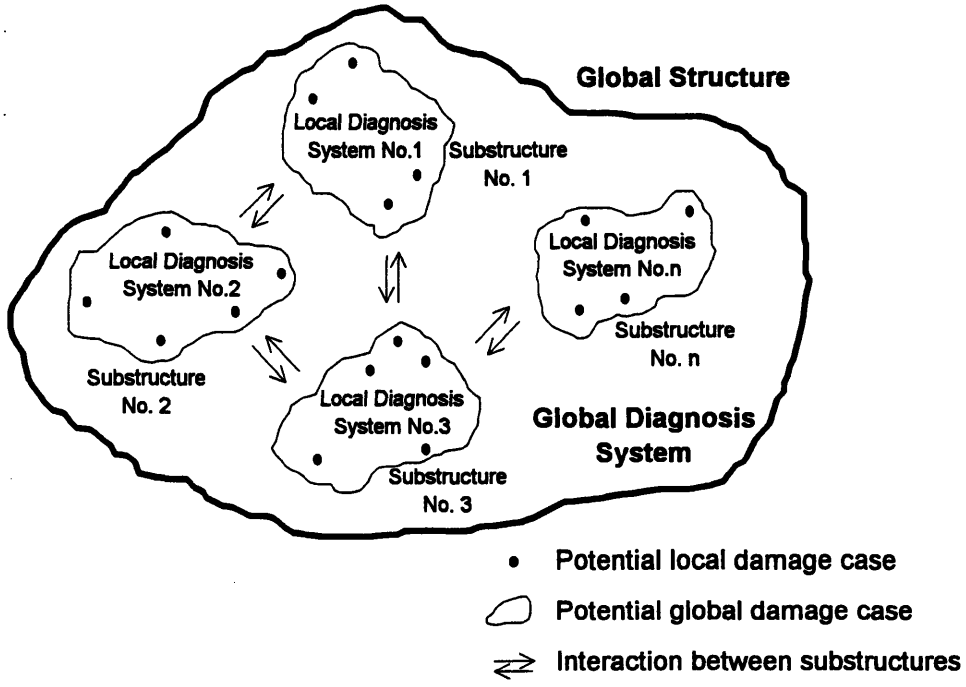
The diagnosis system described in Section 5.3.2 can be categorized as a "global" structural diagnosis system since it employs only one basic neural network system for the whole structure. As mentioned earlier, the applicability of this approach is limited since most structures involve an excessive number of different damage cases.

The global-local structural diagnosis approach transforms a structural damage diagnosis problem to several less complex problems that are handled individually by separate diagnosis systems. As shown in Fig 5.5, the approach is based on considering the whole structure to consist of a set of interacting substructures, and using a "global structural diagnosis system" involving a single neural network to identify which substructures are damaged. Each individual damaged substructure is then independently examined to establish the locations and extent of damage with a "local structural diagnosis system." Figure 5.6 illustrates the general architecture of the global-local structural diagnosis approach, and its operation flow.

The global diagnosis system, shown in Fig 5.7, is a modified version of the basic system described in Section 5.2. The operation of the global system still follows that of the basic system except that NNET2 is not included, and the output of NNET1 now defines the damaged substructures instead of damaged elements. After a specified global excitation is introduced, the time history response of the structure is transformed to a "global vibrational signature," which could be either the mode shapes or spectrums of response at various locations of the structure, before being further processed into the



Global Structural Diagnosis Approach



Global & Local Structural Diagnosis Approach

Figure 5.5: Global and Global & Local structural diagnosis approach.

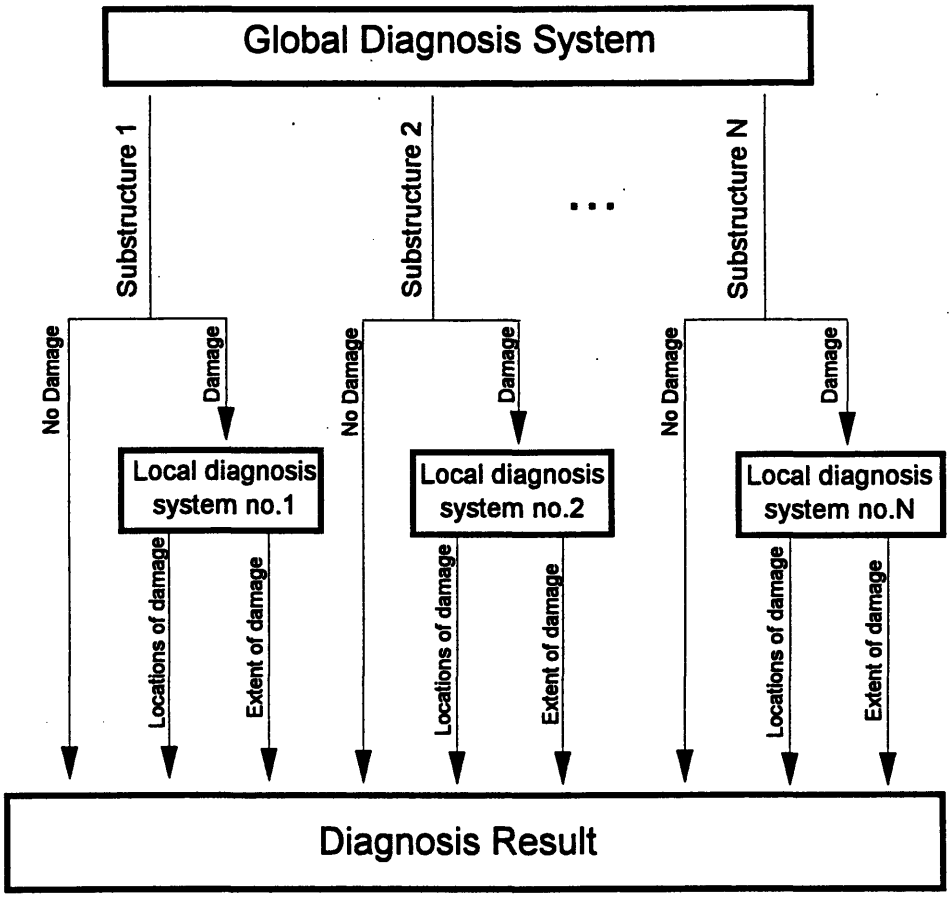


Figure 5.6: General architecture of neural network-based diagnosis system.

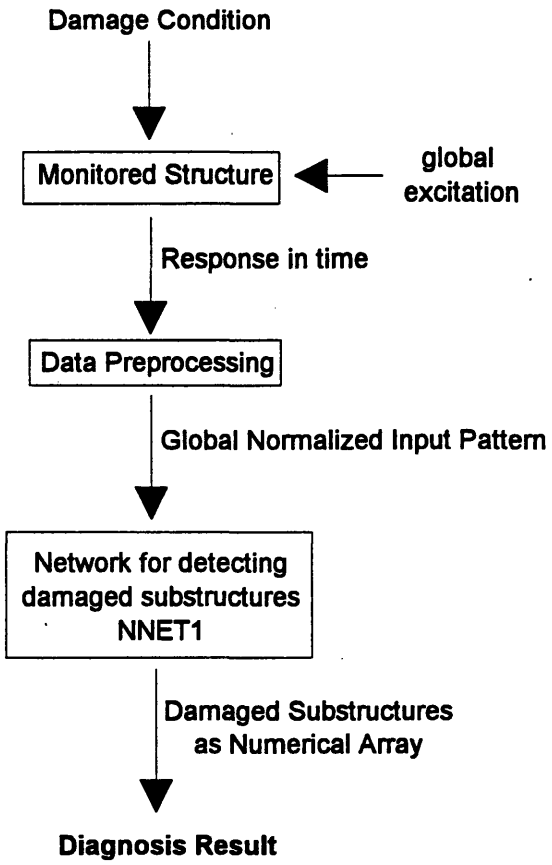


Figure 5.7: Global diagnosis system.

"global normalized input pattern" corresponding to the damage condition. Given the global normalized input pattern, NNET1 identifies the substructures that contain damage by defining each substructure as a possible location of damage. The training procedure for the basic system, which is illustrated in Fig 5.2, is also applicable here. Global structural diagnosis requires at least a global model of the monitored structure and a neural network (NNET1).

The local diagnosis system of a particular substructure is actually a basic neural network-based diagnosis system that is trained by a local training data set specifically created for the substructure. The number of local systems depends on the number of substructures. As mentioned earlier, the local system of a substructure will operate whenever the substructure is identified as being damaged by the global system. The architecture and operation of a local system are similar to those of the basic system described in Section 5.2 (see Fig 5.1). To operate a local system, the response of the monitored substructure due to a prespecified local excitation is transformed to a "local vibrational signature," and further processed into a "local normalized input pattern," which is used as the input for the NNET1 of the local system of the substructure. Given the local normalized input pattern, NNET1 will predict the locations of damage on the substructure, and send its output vector to NNET2 (which also uses the local normalized input pattern as a part of its input) to predict the extent of damage. The training procedure of the NNET1 and NNET2 for each local system is identical to the procedure described in Section 5.2 (see Figs 5.2 and 5.3).

Figure 5.8 illustrates the comparison between the global and global-local diagnosis scheme for a 2-span beam having 22 elements. With global diagnosis, one diagnosis system has to be trained with a set of damage cases involving the order of $1.124*10^{21}$ (factorial of 22) data pairs. With global-local diagnosis, one global and two local diagnosis systems need training. However, the global system now involves only the order of 2

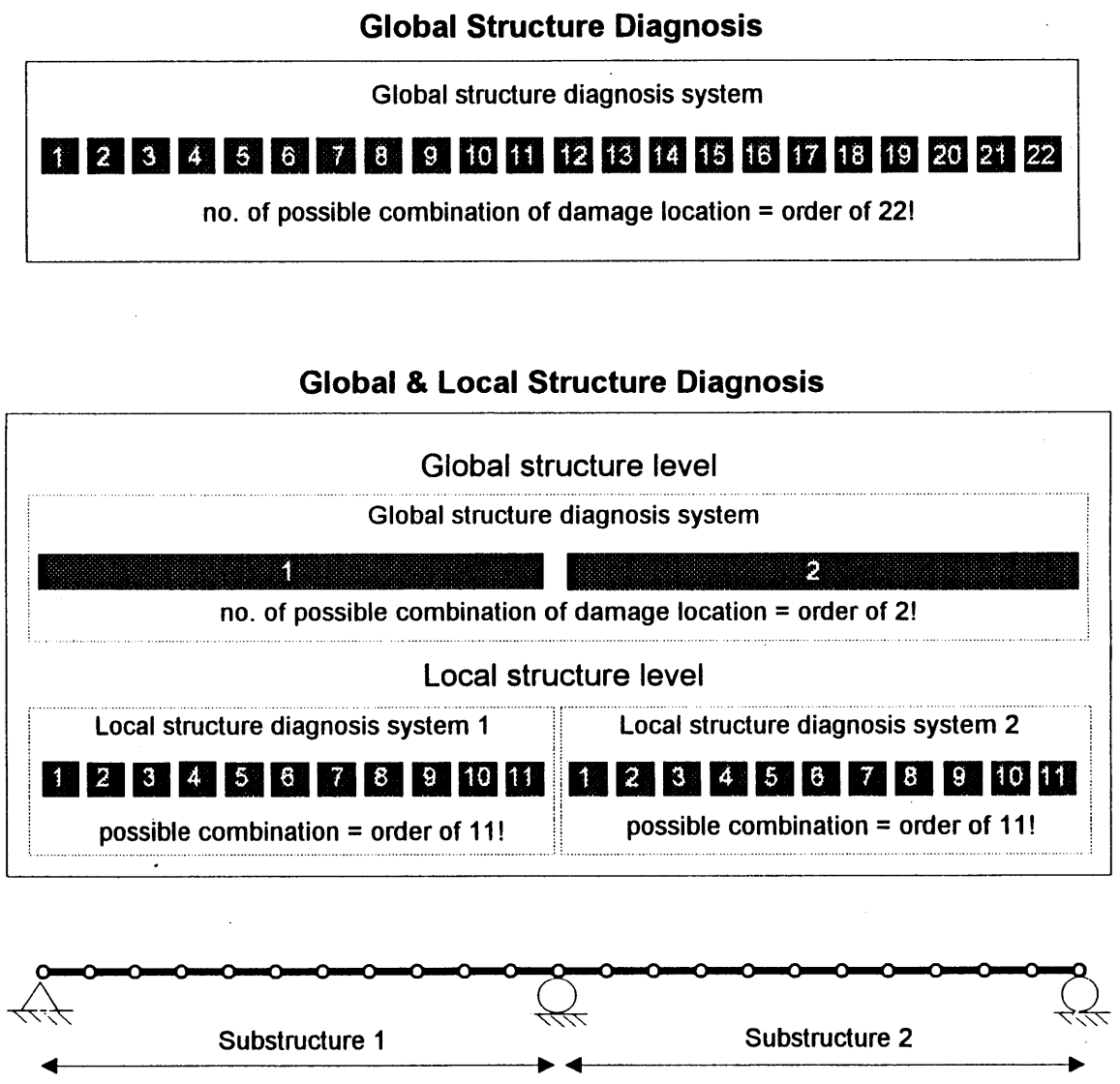


Figure 5.8 Global structure diagnosis, and Global & Local structure diagnosis, of a 2-span beam.

(factorial of 2) data pairs, while each of the two local diagnosis systems requires the order of $4*10^7$ (factorial of 11) data pairs. This reduction in training set is significant, and therefore combining global and local schemes makes the training of the individual networks much more feasible.

More details concerning the design methodology for the diagnosis systems based on the general architecture of neural network-based diagnosis system are presented in Chapter 7. Simulation studies of an idealized 4-span beam with multiple-point damage are performed. Two choices of global excitation and vibrational signature, ambient excitation & mode shape approach and prespecified impulse excitation & response spectrum approach, are evaluated. The prespecified random excitation & frequency transfer function approach is employed for local diagnosis systems.

Chapter 6

Single-Point Damage Diagnosis: A Case Study

6.1 Objective and Scope

This chapter describes the development and application of a neural network-based system for diagnosing single-point damage in a 2-span bending beam. The architecture and training procedure are based on the concepts described in Section 5.2 (see Figs 5.1 to 5.3). Two choices of excitation and vibrational signature are considered: i) ambient excitation & mode shape and ii) prespecified excitation & response spectrum. An assessment of the significant system design variables is performed for both choices of excitation and vibrational signature. The applicability of the diagnosis system to other structures with single-point damage is also discussed at the end of the chapter. The approach is extended to deal with damage prediction of a 4-span beam with multiple-point damage condition in the next chapter.

The scope of the problem is restricted to only linear planar bending behavior, while the contribution of transverse shear deformation is neglected. The multilayer feedforward network with back propagation training algorithm is the only neural network considered because the lack of a-priori knowledge of damage patterns in this problem does not suit the radial basis function network. Only one hidden layer is used since studies (see Chapter 3) indicate no clear performance advantage of including more than one hidden layer. The diagnosis system is supposed to identify both the location and extent of damage. Ambient

excitation is modeled by a single-wheel moving load, while prespecified excitation is modeled by impulse loading.

6.2 Description of Simulation Model

The 2-dimensional bending beam model shown in Fig 6.1 is taken as the model of a real 2-span beam. The mathematical model used is the Timoshenko's bending beam model. The length of the right and left span is 40 and 25 meters respectively. The model consists of 22 linear beam elements. Eleven elements are used to represent each span. Fig 6.2 defines the notation for the element nodal displacements. There are 4 degree of freedoms (DOFs) for each element; 2 translations and 2 rotations. The stiffness matrix of a beam element is given by

$$k = \frac{EI}{l}\begin{bmatrix} 4 & 6/l & 2 & -6/l \\ 6/l & 12/l^2 & 6/l & -12/l^2 \\ 2 & 6/l & 4 & -6/l \\ -6/l & -12/l^2 & -6/l & 12/l^2 \end{bmatrix} \qquad (6.1)$$

where E is the modulus of elasticity of the beam element, I is the moment of inertia of the bending axis, and l is the length of the beam element. The mass matrix for the element is

$$m = \frac{\rho\, l}{420}\begin{bmatrix} 4l^2 & 22l & -3l^2 & 13l \\ 22l & 156 & -13l & 54 \\ -3l^2 & -13l & 4l^2 & -22l \\ 13l & 54 & -22l & 156 \end{bmatrix} \qquad (6.2)$$

where ρ is the average mass per unit length of the element. The stiffness and mass matrices of the elements that are connected to the support of the beam are slightly different due to the effect of the boundary condition. Fig 6.3 shows the beam element no.1

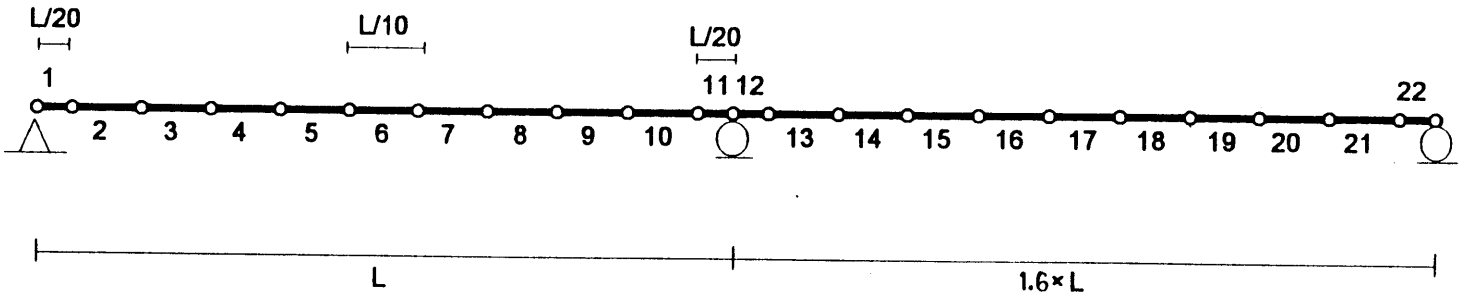


Figure 6.1: The 2-span beam model.

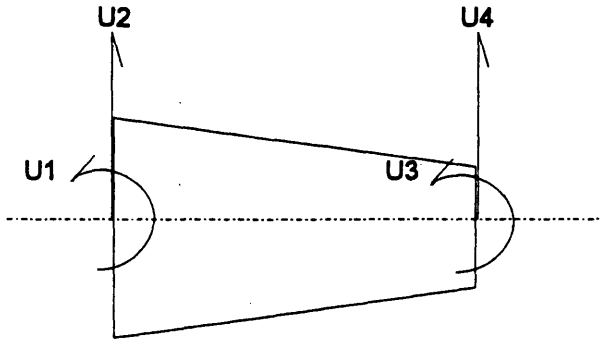


Figure 6.2: General beam element.

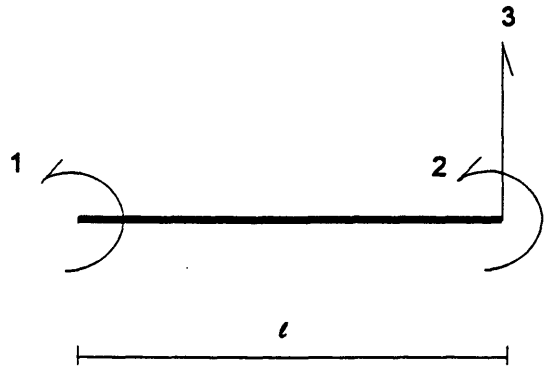


Figure 6.3: DOFs of beam element no.1

of the beam model. There is no translational movement at the left end of this element due to the presence of the hinge support, so the corresponding row and column of Eqs. 6.1 and 6.2 can be deleted. The reduced matrices are

$$k_1 = \frac{EI}{l}\begin{bmatrix} 4 & 2 & -6/l \\ 2 & 4 & -6/l \\ -6/l & -6/l & 12/l^2 \end{bmatrix} \tag{6.3}$$

and

$$m_1 = \frac{\rho\, l}{420}\begin{bmatrix} 4l^2 & -3l^2 & 13l \\ -3l^2 & 4l^2 & -22l \\ 13l & -22l & 156 \end{bmatrix} \tag{6.4}$$

The same approach is applied to elements 11, 12, and 22 in order to provide all the necessary boundary conditions for the equilibrium equations of the 2-span beam.

The stiffness and mass matrices of the complete beam model are generated by superimposing the contribution of the elements as indicated below.

$$K = \begin{bmatrix} k1 & & & \\ & k2 & & \\ & & \ddots & \\ & & & k22 \end{bmatrix}_{43 \text{ by } 43} \tag{6.5}$$

and

$$M = \begin{bmatrix} m1 & & & \\ & m2 & & \\ & & \ddots & \\ & & & m22 \end{bmatrix}_{43 \text{ by } 43} \qquad (6.6)$$

Note that there is overlap between each pair of neighboring k, or m, since they share some global degrees of freedom.

Assuming that the beam has proportional damping (i.e. Rayleigh damping), the damping matrix is

$$C = \alpha_0 M + \alpha_1 K \qquad (6.7)$$

where α_0 and α_1 are parameters that correspond to the pre-specified damping ratio of the first 2 modes of vibration of the model. These parameters can be determined by solving the equation

$$\frac{1}{2}\begin{bmatrix} 1/w_1 & w_1 \\ 1/w_2 & w_2 \end{bmatrix}\begin{bmatrix} \alpha_0 \\ \alpha_1 \end{bmatrix} = \begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix} \qquad (6.8)$$

where ϕ_1 and ϕ_2 are the prespecified damping ratios, and w_1 and w_2 are the undamped modal frequencies of the 1st and 2nd mode of vibration respectively. In this application the damping ratios of the first 2 modes are set to be 1%.

By defining

$$U = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{43} \end{bmatrix} \tag{6.9}$$

as the global displacement vector, which contains all the degree of freedom of the beam model (see Fig 6.4), the equation of motion of the model can be represented as

$$M\ddot{U} + C\dot{U} + KU = P \quad , \tag{6.10}$$

where

$$P = \begin{bmatrix} p_1(t) \\ p_2(t) \\ \vdots \\ p_{43}(t) \end{bmatrix} \tag{6.11}$$

is the force on the beam model as a function of time t. Given the force function and all the parameters of the model, the time history response can be determined by solving the equation of motion.

For this application, the flexural rigidity, EI, is taken as 5.34×10^9 N·m^2 throughout the length. The average mass per unit length is 9880 kg/m. Fig 6.5 shows the first three mode shapes and their corresponding frequencies. The unsymmetrical feature is reflected in the mode shapes. The response of the beam model due to an excitation is determined by a direct time integration method (Newmark method) performed in MATLAB. The time interval of the integration is taken as 0.01 second. Damage is introduced in the model by lowering the value of EI in the stiffness matrix of a beam element. This type of damage condition can be interpreted as degradation of the flange area of the beam.

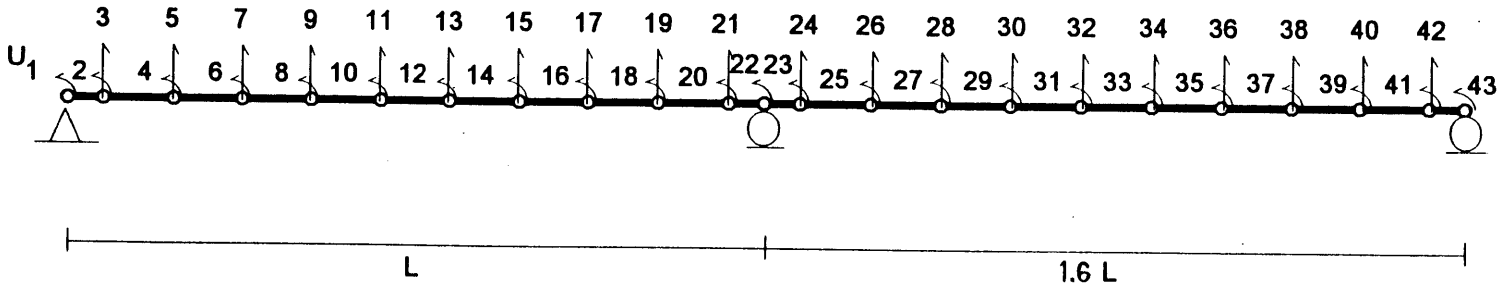


Figure 6.4: Global degree of freedoms of the 2-span beam model.

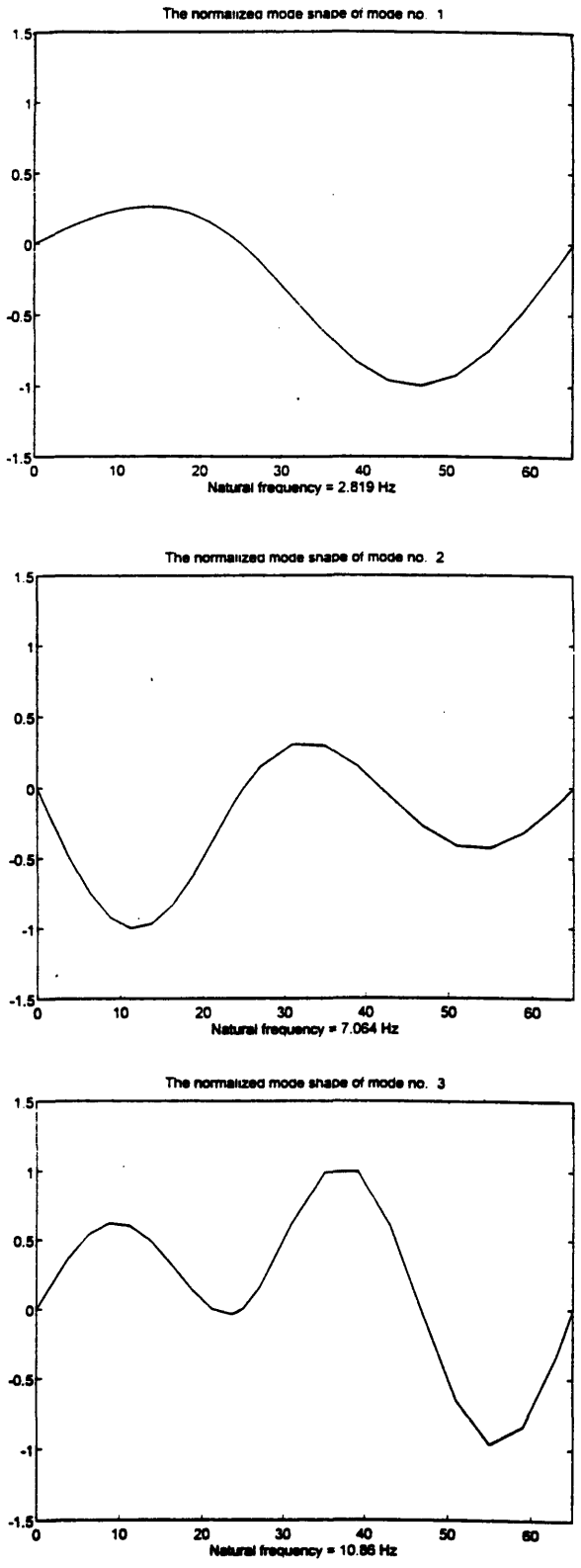


Figure 6.5: First 3 mode shapes of the unsymetrical 2-span beam.

Two types of vibrational signatures are used to monitor the state of the beam model; mode shape and response spectrum. As demonstrated in Appendix C, the changes of the vibrational signatures of a bending beam can be employed to identify the change of the beam's stiffness. The signatures are determined from the simulated free-vibration response of the model given a specified excitation. Ambient excitation is employed in creating mode shapes since the properties of the mode shapes do not change with excitations (Mazurek, 1992). Therefore, the neural network-based diagnosis system can employ the mode shapes of the model (with various damage conditions) corresponding to different excitations as its training data. Besides, using ambient excitation provides the diagnosis system the real-time operation ability, and does not require extra loading equipment. On the other hand, the properties of response spectrums depend directly on the characteristics of the excitation, so it is necessary for a neural network-based diagnosis system to employ the response spectrums of the model corresponding to a specified excitation as its training data.

6.3 Mode Shape Approach

Since the roadway roughness and vehicle velocity do not have any influence on the modal frequencies and shapes, and variable mass has only a minimal effect (Mazurek, 1992), a mass-consistent moving load with variable velocity is utilized for creating the training and testing data patterns corresponding to different damage conditions. A single-wheel load of 2000 kg (4.4 kips), with its velocity ranging from 40 to 60 mph, is moved through both spans of the model as the excitation. In this case, the duration of loading, or the amount of time that the moving load travel on the beam, is more than ten times of the first three fundamental periods of the beam. Therefore, the moving load effectively excites the first three modes of vibration of the beam (Humar, 1990 and Humar et al, 1993). In practice, the ambient excitation may be more heavily weighted to certain frequencies.

However, such variations in excitation are kept to a minimum in most cases (Green, 1995).

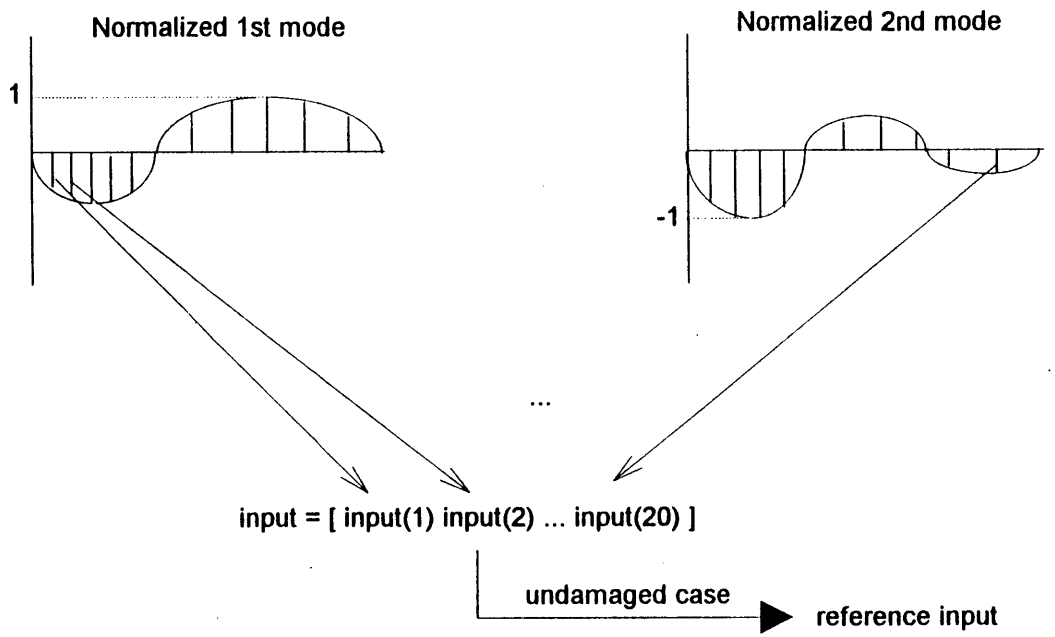
6.3.1 Data Preprocessing Strategy

The data preprocessing unit transforms the time history response of the structure having a particular damage condition into a corresponding input pattern for the neural networks NNET1 and NNET2. The time response at each monitoring point of the model is determined via simulation, and is taken as the sensory data of the real structure. In the mode shape approach, the free-vibration response data is processed by a modal analysis routine (see Appendix A) in order to determine the mode shapes of the model corresponding to different damage conditions. The data is collected right after the moving load passes the last support for a period of 20 seconds.

Fig 6.6 illustrates the case where the first two mode shapes, with 10 points representing each mode, are selected as input for the neural networks. All the mode shapes are similarly normalized so that their maximum amplitude is equal to unity. The value of the points representing the normalized mode shapes are then used to create a vector called "input" vector,

$$input = \begin{bmatrix} \text{data point no.1 of mode shape no.1} \\ \vdots \\ \text{data point no.10 of mode shape no.1} \\ \text{data point no.1 of mode shape no.2} \\ \vdots \\ \text{data point no.10 of mode shape no.2} \end{bmatrix} \in \Re[0,1], \tag{6.12}$$

The Reference Value Approach is employed next. This approach uses the difference between the damaged and undamaged data patterns as the input for the neural



input = input - reference input

input = norm(input)

output = [output(1) output(2) ... output(22)]

output(i) = 1 , when ith beam element is damaged.

= 0 , otherwise.

Figure 6.6: Data preprocessing approach.

networks (Elkordy et al, 1993 and Ballinger et al, 1995). The data generated through the simulation of the undamaged structural model is called the "reference data set."

$$reference\ \ data\ \ set = input\ \ ,\ \ given\ \ undamaged\ \ case. \tag{6.13}$$

The difference between the data corresponding to a simulation of any given damage condition and the *reference data set* is taken as the "*input pattern*" for the particular damage condition.

$$input\ \ pattern\ \ =\ \ input - reference\ \ data\ \ set. \tag{6.14}$$

Each *input pattern* is then normalized by dividing by the absolute value of the maximum data point in all available *input patterns* (*maxdata*).

$$normalized\ \ input\ \ pattern\ \ =\ \ \frac{input\ \ pattern}{\max data}\ \ ,\ \ \in \Re[0,1]\ \ . \tag{6.15}$$

Given a "*normalized input pattern,*" NNET1 gives the corresponding output,

$$output1 = \begin{bmatrix} out1_1 \\ out1_2 \\ \vdots \\ out1_{22} \end{bmatrix}\ , \tag{6.16}$$

where $out1_i \in \Re[0,1]$. Since the number of elements representing the model is 22, the size of the output vector is 22 by 1. The "expected" *output*1 corresponding to the same damage condition is also represented by Eq. 6.16, where $out1_i$ is 1 if there is reduction of *EI* of the *i*th element and 0 otherwise. By defining

$$out1_i = 0, \text{ when } out_i < threshold$$
$$= 1, \text{ when } out_i \geq threshold, \quad threshold \in \Re[0,1],$$

for every *output*1, a particular output of NNET1 is regarded as a correct classification whenever it is within a specified threshold of error. For example, if the network output is

$$\begin{bmatrix} 0.0154 \\ 0.7163 \\ 0.1209 \\ \vdots \\ 0.0286 \end{bmatrix}_{22\times 1},$$

and the expected output is

$$\begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}_{22\times 1},$$

which corresponds to damage in the second element, the classification is correct if *threshold* is set at 0.7. However, if the threshold were set at 0.8 or 0.1, the network prediction would be incorrect. When the amount of available training data is limited, the threshold is kept low in order to allow the diagnosis system to predict with lower level of confidence. As more training data becomes available, the threshold and the confidence of prediction increase. Since the optimum threshold value for a specific application depends on the training and testing data set, a trial and error approach is also required to find the optimum value.

The *normalized input pattern* and its corresponding expected *output*1 for a particular damage case can then be employed as a training or testing data pair of NNET1. Note that the error term used by the training algorithm of NNET1 (see Fig 5.2) is the difference between the expected *output*1 and the *output*1 that is not yet adjusted by the *threshold* value.

The input of NNET2, *normalized input pattern*2 , is the vector that combines the *normalized input pattern* with the threshold-adjusted output of NNET1 given the *normalized input pattern*.

$$normalized\ \ input\ \ pattern2\ = \begin{bmatrix} normalized\ \ input\ \ pattern \\ output1 \end{bmatrix} \tag{6.17}$$

The output of NNET2 corresponding to a *normalized input pattern*2 is

$$output2 = \begin{bmatrix} out2_1 \\ out2_2 \\ \vdots \\ out2_5 \end{bmatrix}, \quad out2_i \in \Re[0,1]. \tag{6.18}$$

After being adjusted by a *threshold* value, the *binary output*2 represents 5 categories of damage states: $out2_1$ = 1 indicates 0-20% reduction of *EI* of the damaged element identified by NNET1, and 0 otherwise; $out2_2$ does the same for 20-40% reduction of *EI* ; $out2_3$ for 40-60% ; $out2_4$ for 60-80%, and $out2_5$ for 80-100%. The *normalized input pattern*2 and the expected *output*2 corresponding to a particular damage are then taken as a training or testing data pair of NNET2. Note that the error term used by the training algorithm of NNET2 (see Fig 5.3) is the difference between the expected *output*2 corresponding to a particular damage condition and the *output*2 that is not yet adjusted by *threshold* value.

6.3.2 Configuration and Training of Neural Networks

Since this application is a static pattern mapping problem, a 1-hidden-layer feedforward network trained with the back propagation algorithm is appropriate. Figure 6.7 shows a NNET1 with 20 nodes in the input layer, corresponding to the size of the *normalized input pattern* used in Section 6.3.1, and 22 nodes in the output layer that represent the size of the binary output vector. A tangent sigmoid type transfer function is used for the hidden layer,

$$f(x) = \tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}, \quad f(x) \in \Re[-1,1], \tag{6.19}$$

while the log sigmoid function is employed for the output layer,

$$f(x) = \frac{1}{1+e^{-x}}, \quad f(x) \in \Re[0,1]. \tag{6.20}$$

Both transfer functions are illustrated in Fig 6.8. According to the discussion in Section 4.2.2, this transfer function assignment is appropriate for classification networks. The initial value of each of the connection weights is randomized value between -1 to 1. The training algorithm also utilizes the momentum term and adaptive learning rate.

The training data set contains the *normalized input patterns* and their corresponding expected *output*1s corresponding to 92 different damage cases, which are created by performing 92 numerical simulations. The first four times, the simulation is performed on the damage-free beam model. The remaining 88 times, it is performed with different damage cases that have 5%, 20%, 50% or 80% reduction of EI on a beam element. Table 6.1 demonstrates the damage cases for all simulations.

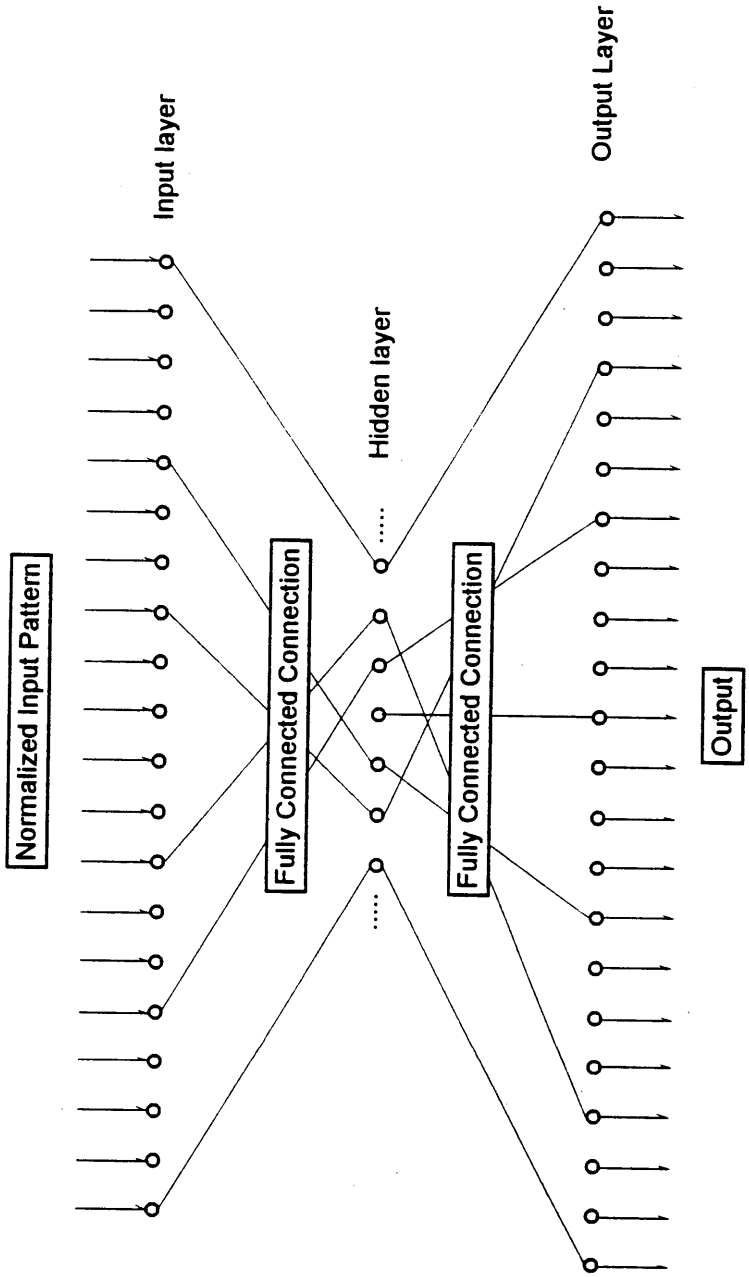
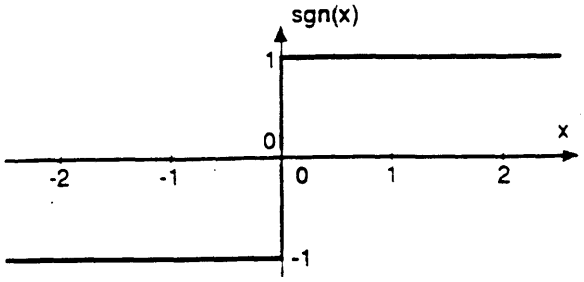
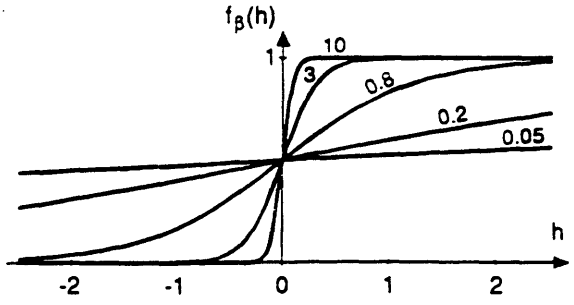


Figure 6.7: A 1-hidden-layer network used in damage diagnosis.





$$f_\beta(h) \equiv \frac{1}{1+\exp(-2\beta h)}$$

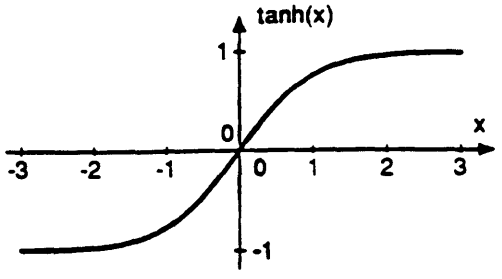


Figure 6.8: Example of transfer functions.

Damage case of the simulation no.	Damaged Element	*EI* Reduction (% of original *EI*)
1-4	-	0
5-26	1, 2, 3, ..., 22	5
27-48	1, 2, 3, ..., 22	20
49-70	1, 2, 3, ..., 22	50
71-92	1, 2, 3, ..., 22	80

Table 6.1: The damage cases of the training data set.

Fig 6.9 shows the convergence of the Sum-Square Error (*SSE*) of the NNET1, with 7 processing elements in the hidden layer, trained by this data set.

Other 70 testing data pairs are then similarly created. Each of them are created from the simulation of the model that has the reduction of *EI* randomly range between 5% to 90% on one beam element. Table 6.2 demonstrates all the damage cases included in the testing data set.

Damage case of the simulation no.	Damaged Element	*EI* Reduction (% of original *EI*)
1-4	-	0
5-70	1, 2, 3, ..., 22, 1, 2, ..., 22,1, 2, 3, ..., 22	randomly between 5-90%

Table 6.2: The damage cases of the testing data set.

The accuracy of NNET1 is determined by operating the trained network on the training data set, and comparing the network outputs to the expected outputs, which are the binary vectors corresponding to the damage cases used to create the testing data set. In this application, the threshold value is set at 0.8 for both NNET1 and NNET2.

The configuration of NNET2, and the procedure followed to construct the training and testing data, are similar to those of NNET1. The only difference is that NNET2

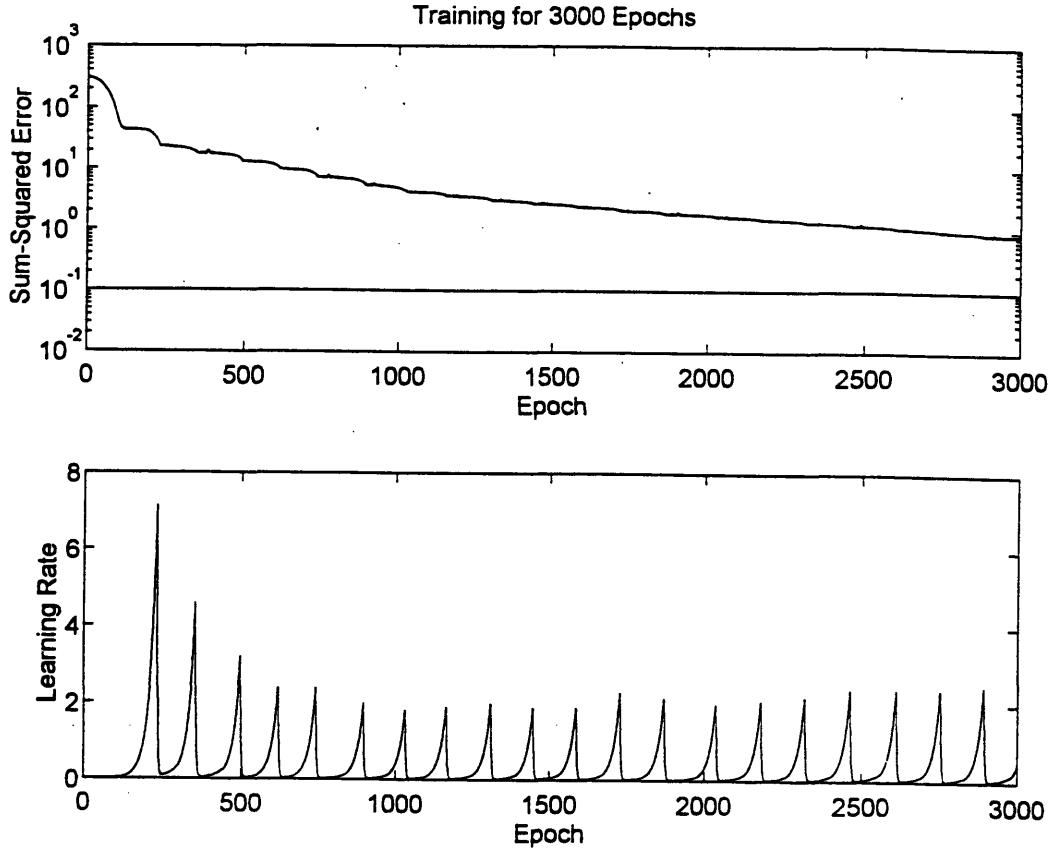


Figure 6.9: Convergence of the training of an example network.

requires more nodes in the input layer (corresponding to the size of *normalized input pattern* 2), and 5 output nodes in the output layer (corresponding to the size of *output*2).

6.3.3 Performance Studies

The performance-based process for selecting the optimum global diagnosis system for the 2-span beam problem based on the results of simulation studies is performed in this section. Two measures of the performance of a diagnosis system are considered: i) absolute accuracy, i.e., the best percentage of correct diagnosis that can be achieved by a diagnosis system, and ii) the number of training cycles required for neural networks to reach a specific level of accuracy.

Sensitivity studies are carried out for the following 3 variables; i) number of mode shapes used to create input, ii) number of points representing each mode, and iii) number of processing elements in the hidden layer of NNET1 and NNET2. Fig 6.10 shows the significance of the number of mode shapes on the accuracy of NNET1. The accuracy measure plotted the percentage of correct predictions by NNET1 based on the testing data. In this case, the NNET1 configuration has 7 processing elements in the hidden layer, and each mode shape is represented by 10 points. The result demonstrates that incorporating more modes in the training data improves the absolute accuracy, and reduces the number of training cycles needed to reach the absolute accuracy. Using all 3 mode shapes as input, this network trained for 4000 cycles can predict the location of damage with the absolute accuracy as high as 96%.

Fig 6.11 shows the influence of the number of points used to represent each mode shape, or the number of sensors, on the accuracy of NNET1. The first 3 mode shapes are employed to create the input. In this investigation, the number of points representing each mode shape is varied uniformly from 5 to 20 points, with an increment of one point. The result demonstrates that the absolute accuracy increases with the number of points until an optimum number is reached. In this application, the absolute accuracy stops increasing

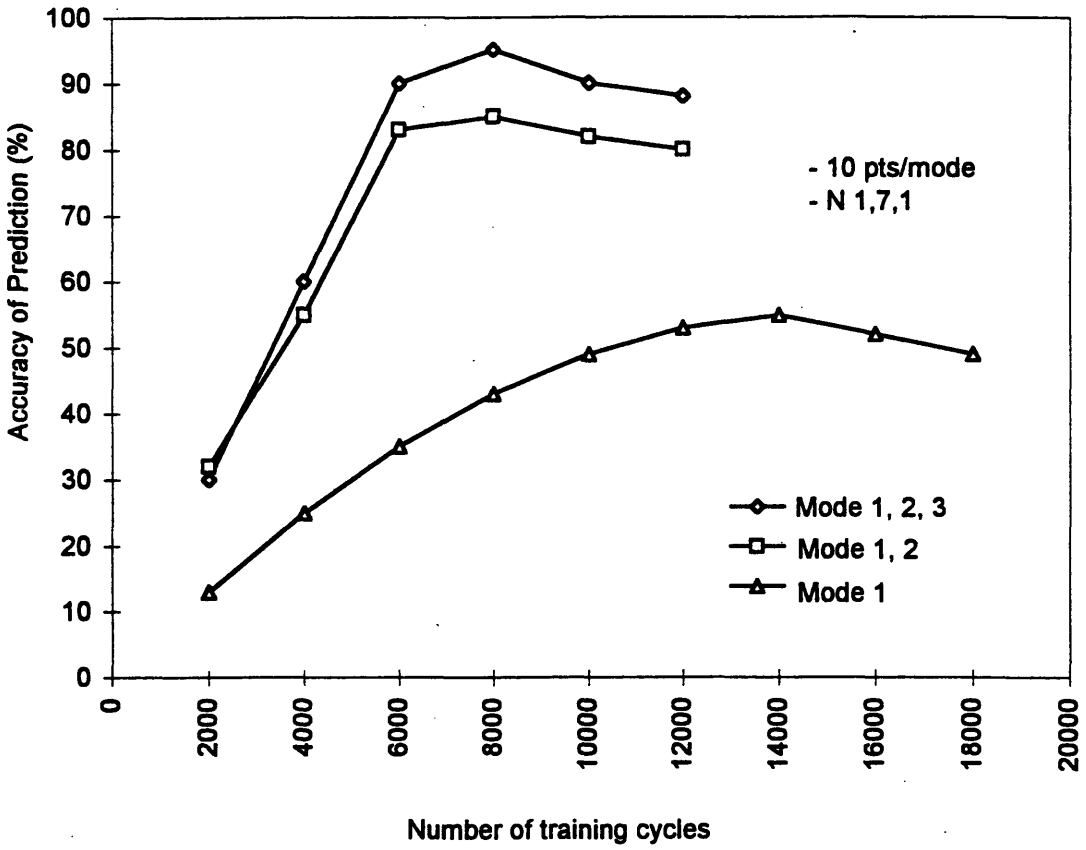


Figure 6.10: Significance of the no. of mode shapes to the accuracy of NNET1.

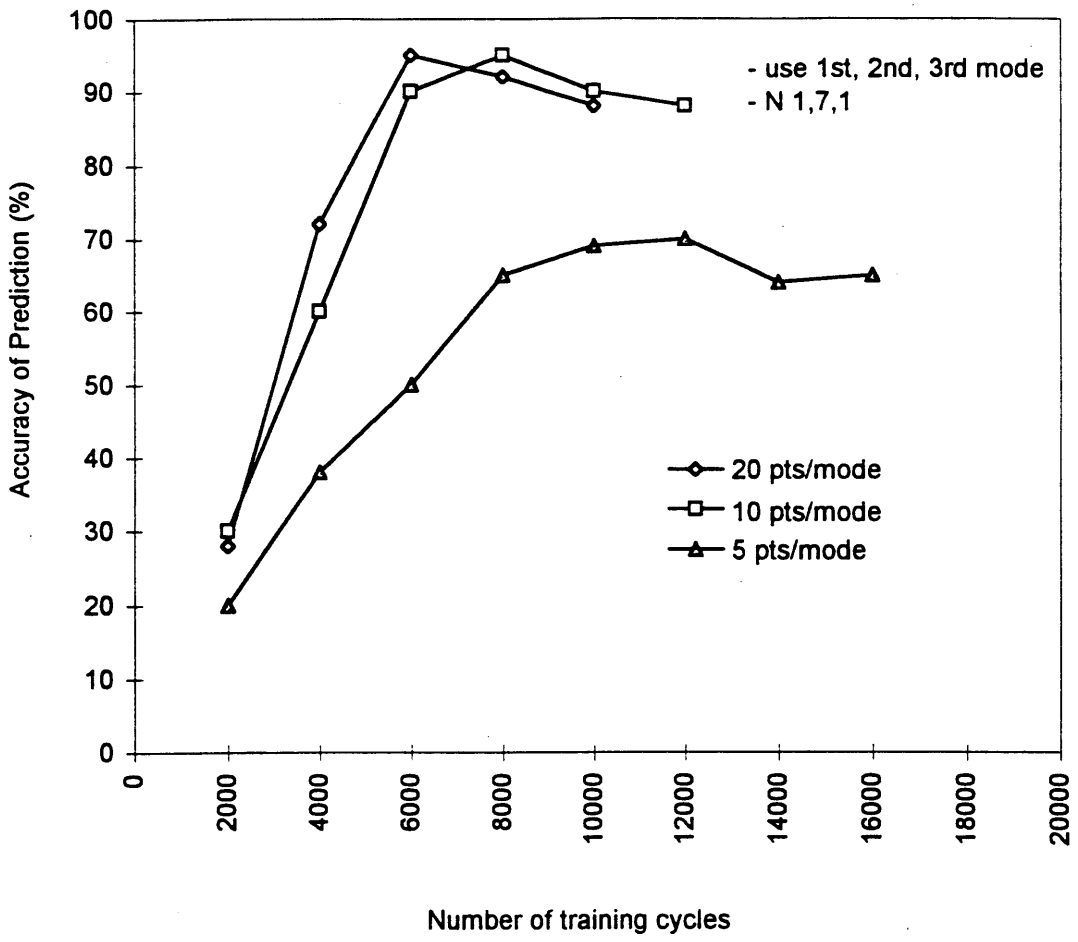


Figure 6.11: Significance of no. of points representing each mode shape to the accuracy of NNET1.

when the number of points representing each mode shape exceeds about 10, which is an indication that 10 points per mode shape is a reasonable choice.

The effect of the number of processing elements in the hidden layer on the accuracy is investigated by varying the number of elements uniformly from 2 to 20 elements with an increment of one element. The network is trained and tested with the data created from the first three mode shapes, each of which is represented by 10 points. The results indicate that the absolute accuracy increases with the number of elements up to about 7 elements, and then tends to decrease slightly beyond this number of elements. Fig 6.12 compares the accuracy corresponding to three different networks (having 5, 7, and 15 processing elements respectively). The figure shows that the 7 processing elements architecture has the best absolute accuracy. The 15 elements version performs better on the training data, but worse on the testing data. The reason is that larger networks tend to have less generalization ability, and are more susceptible to overfit the training data (Liu, 1995; Ling, 1995).

Different testing data sets, each of which has damage cases that have a specific extent of damage, are then employed to test the optimum NNET1, which is the NNET1 with all design variables set at optimum. The configuration includes using the first three mode shapes, each represented by 12 data points. The network has 12 hidden units, and 100 training samples are used. The result is demonstrated in Table 6.3.

Extent of Damage (% of *EI* reduction)	**Accuracy of Optimum NNET1 (% of correct diagnosis)**
10	96
5	96
4	81
3	54

Table 6.3: The damage sensitivity of the optimum NNET1.

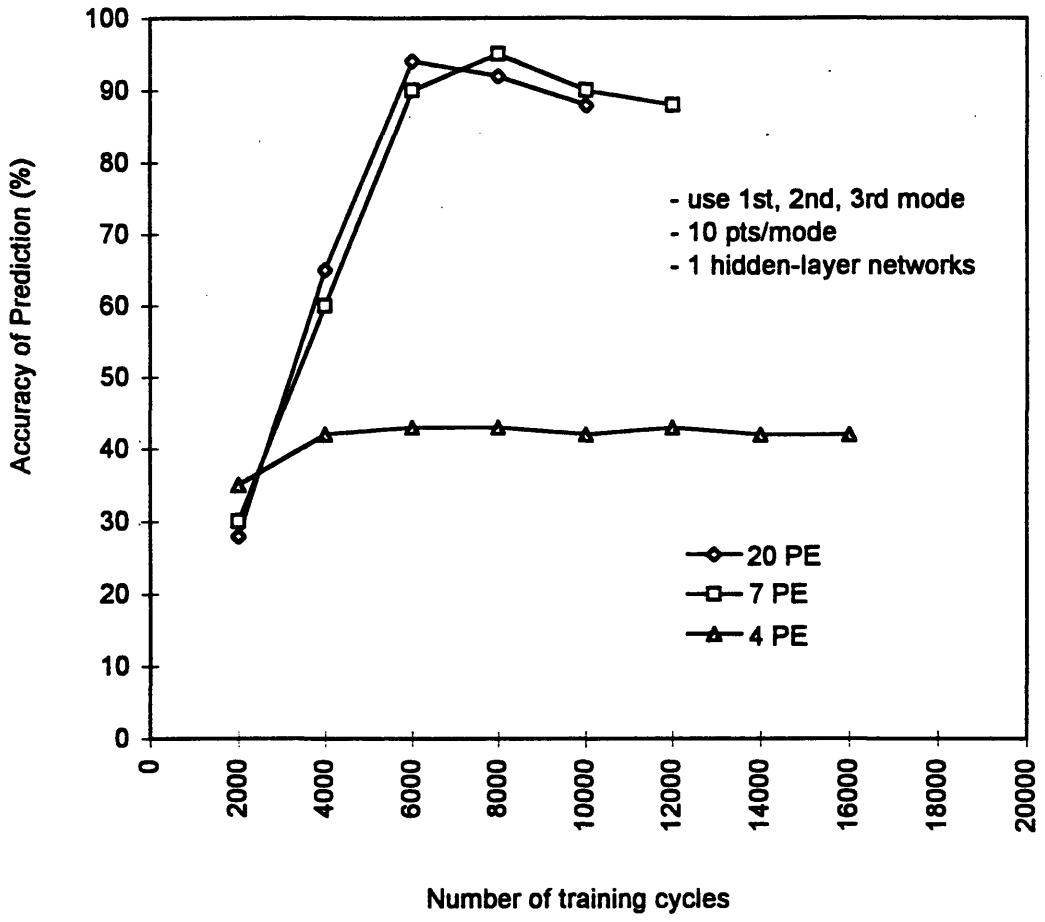


Figure 6.12: Significance of the no. of PE in the hidden layer to the accuracy of NNET1.

The optimum NNET1 is able to detect damage corresponding to a 5% reduction in *EI* at any location of the 2-span beam model with an accuracy of 96%. This performance is very impressive.

The same performance-based design process is also employed to find the optimum NNET2 configuration. Based on the results of simulation studies, the optimum NNET2 architecture has 21 processing elements. It's input data is generated from the first 3 mode shapes, each of which is represented by 10 points. The optimum NNET2 trained by the cross-validation approach is able to predict the extent of damage with absolute accuracy up to 84%.

6.3.4 Observation

Based on the results of performance studies, the optimum NNET1 architecture has 7 processing elements. It's input data is generated from the first 3 mode shapes, each represented by 10 data points. Properly trained NNET1 is able to predict the location of damage with an absolute accuracy up to 96%. Simulation results demonstrate the overtraining effect (see Chapter 3) when the network is trained to fit the training data so that it performs poorly with the testing data. Therefore, a trial and error approach, or the cross-validation method (see Chapter 3), is recommended for optimizing the NNET1.

These results also reveal the ability of the diagnosis system based on mode shape approach to effectively monitor the condition of an idealized structure with single-point damage. An additional benefit of this approach is that it requires no extra loading equipment for operation. Moreover, the diagnosis systems based on this approach can be operated in real time when ambient excitation is abundant.

6.4 Response Spectrum Approach

The excitation used in this application is a set of specified hammer impulses applied at specific locations of the structure. The simulation model, and the damage cases

for both the training and testing data sets, are similar to those employed in the previous section.

6.4.1 Data Preprocessing Strategy

The time-acceleration response of all DOFs is determined from numerical simulation, and is taken as the sensory data at the corresponding locations of the monitored structure. Each time-acceleration data, $x(t)$, is recorded during the time period T and then passed through an analog-to-digital converter to generate the discrete time series $\{x_r\}$, $r = 0, 1, 2, ..., (N\text{-}1)$. The Fast Fourier Transform (FFT) is then used to calculate the Discrete Fourier Transform (DFT) of this time series, $\{X_k\}$, $k = 0, 1, 2, ..., (N\text{-}1)$, and hence find the spectral estimate

$$\tilde{S}_{xx}(w_k) \cong \frac{T}{2\pi} X_k^* X_k \quad , \tag{6.21}$$

where X_k is the DFT of $\{x_r\}$, and X_k^* is the complex conjugate of X_k. The spectral estimates corresponding to $\{x_r\}$ of various sensors are then smoothed in order to improve their statistical reliability by a spectrum smoothing routine in MATLAB (based on the method suggested by Newland, 1993) before being used to create the input for neural networks.

In this application, the free-vibration response of the model is employed to create the response spectrums of the model corresponding to different damage conditions. Since the sampling interval is 0.01 second, the Nyquist frequency is 50 Hz, which effectively contains the response of the first 3 modes of vibration of the beam model (Newland, 1993, suggests the Nyquist frequency should be at least twice the frequency of interest). The time series signal is filtered to remove frequency components over 50 Hz in order to avoid the effect of aliasing. The frequency resolution is set at 1 Hz, and the maximum effective bandwidth of the calculation is set at 0.5 Hz. The ratio of the standard deviation to the

mean of measurement is set at 0.3, and the required measurement length is determined to be 20 seconds. The smoothing process involves averaging every 11 adjacent spectral estimates, and each value of the spectral estimates is the result of smoothing over 0.5 Hz frequency span. The smoothed spectral estimates are then taken as the response spectrums of the time series.

Fig 6.13 shows the 20-second acceleration response of DOFs 7 and 28 of the undamaged model given a hammer impulse, and their corresponding smoothed spectrums. Assuming that these 2 spectrums are used as a damage pattern for NNET1, each spectrum will be divided into a number of zones, *NUMZONE*. As shown in Fig 6.14, each zone covers the same size of frequency interval,

$$f\,\text{int} = \frac{MAXFREQ}{NUMZONE} \quad Hz, \tag{6.22}$$

where *MAXFREQ* is the maximum frequency (Hz) of the response spectrums. In this application, *MAXFREQ* is set at 15 Hz, which well covers the 3rd mode of vibration of the 2-span beam (see Fig 6.5). The area under the spectrum of each interval, which relatively represents the amount of energy released by the DOF (or by a specific area of the beam represented by the DOF) in that frequency interval, is then calculated. This provides a number of energy-related data points (*NUMZONE* points from each spectrum) for a particular damage case as a vector,

$$input = \begin{bmatrix} \text{data point no.1 of spectrum no.1} \\ \vdots \\ \text{data point no. } NUMZONE \text{ of spectrum no.1} \\ \vdots \\ \text{data point no.1 of spectrum no. } NUMSPEC \\ \vdots \\ \text{data point no. } NUMZONE \text{ of spectrum no. } NUMSPEC \end{bmatrix}, \tag{6.23}$$

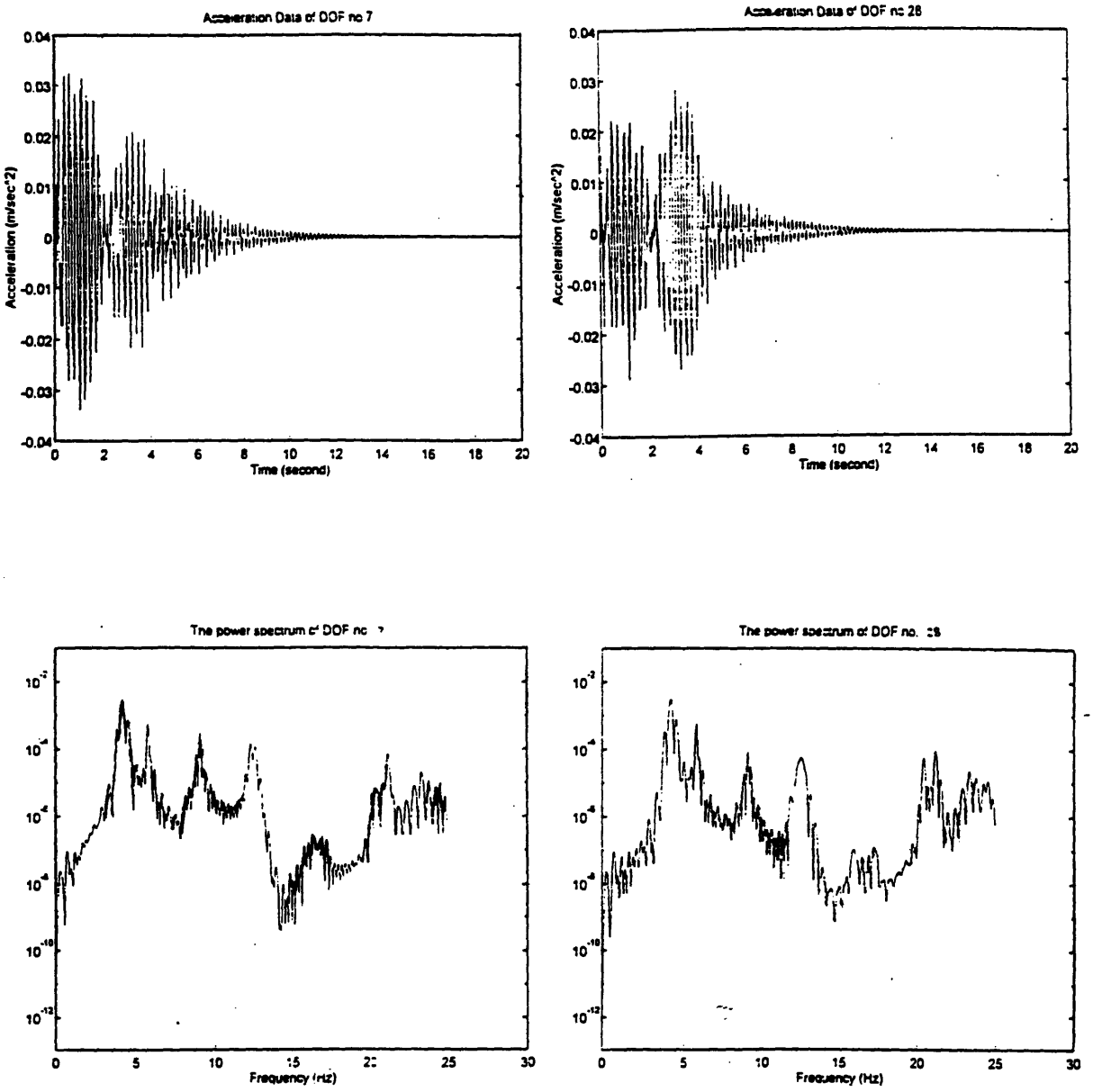
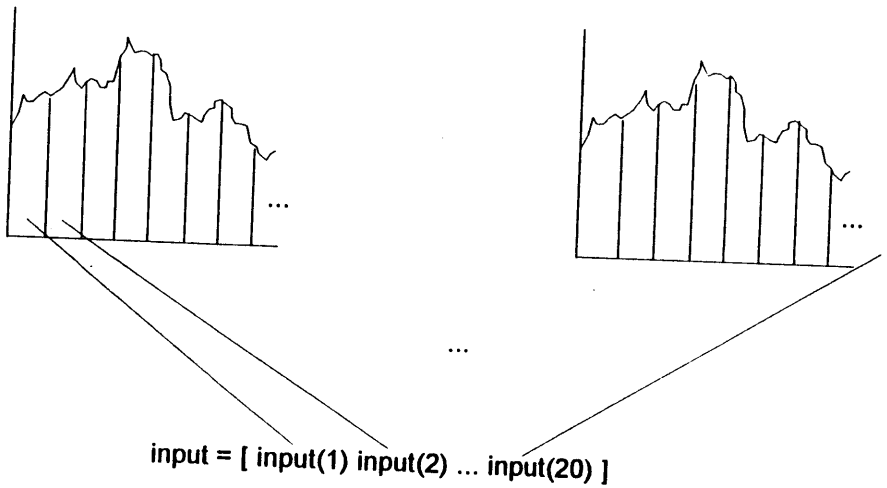


Figure 6.13: Example of simulated acceleration response and its spectrums.



input = [input(1) input(2) ... input(20)]

input = log10(input) —undamaged case→ reference input

input = input - reference input

input = norm(input)

output = [output(1) output(2) ... output(22)]

output(i) = 1 , when ith beam element is damaged.

= 0 , otherwise.

Figure 6.14: Data preprocessing approach.

where *NUMSPEC* is the total number of spectrums used to create the input for NNET1 and NNET2 (or the total number of sensors used to collect the data).

In order to keep the difference of the magnitude among all data points (data spread) low, the logarithm of base ten of the *input* vector is used.

$$input = \log_{10}(input) \quad . \tag{6.24}$$

The vector generated through the simulation of the undamaged model is referred to as the "*reference data set*,"

$$reference\ \ data\ \ set = input \quad , \text{ given undamaged case.} \tag{6.13}$$

The Reference Value Approach is then applied. The difference between the *input* vector corresponding to a particular damage case and the *reference data set* is taken as the "*input pattern*" for the damage case.

$$input\ \ pattern \ = \ input - reference\ \ data\ \ set. \tag{6.14}$$

Each *input pattern* is then normalized by dividing by the absolute value of the maximum data point in all the *input patterns* (*maxdata*).

$$normalized\ \ input\ \ pattern \ = \ \frac{input\ \ pattern}{\max data} \quad . \tag{6.15}$$

Each "*normalized input pattern*" and its corresponding expected outputs of NNET1 and NNET2, which are the same types of binary vectors used in the mode shape approach, can then be taken as a training or testing data pair of NNET1 and NNET2. More details on how to create the data are demonstrated in Section 6.3.1.

6.4.2 Configuration and Training of Neural Networks

The configuration of NNET1 and NNET2, and the damage cases for their training and testing data sets, of this diagnosis system are exactly the same as those used in section 6.3.2. However, the number of input nodes for both NNET1 and NNET2 now corresponds to the size of *normalized input pattern* used by the spectrum approach.

6.4.3 Performance Studies

Six design variables are investigated for their influence on the performance of the diagnosis system; i) the number of sensors, ii) the location of sensors, iii) the number of intervals representing each spectrum, iv) the number of hammers, v) the locations of hammers, and vi) the number of processing elements in the hidden layer of NNET1 and NNET2. Fig 6.15 demonstrates the significance of the location of sensors to the accuracy of NNET1. Each point represents the accuracy of a NNET1 trained and tested with data created from two response spectrums, each of which is created from the data collected from a sensor located at the same relative position of each span. Two hammer impulse loads, each of which is located at the middle-left point of each span, synchronously apply the impulse of 10000 Newton for 0.1 second as the prespecified excitation. Ten intervals are used to represent each response spectrum, and 15 processing elements are included in the hidden layer of NNET1. Fig 6.15 shows that the location of the sensors has almost no effect on the performance of NNET1 except when the sensors are close to supports. In this case, there is a slight drop in performance due to the fact that the sensor response is weak for these locations.

Fig 6.16 shows the significance of the number of sensors used to create the input data for NNET1. The loading and network parameters are the same as for the previous figure. The plot demonstrates that the absolute accuracy of NNET1 increases with the number of sensors per span. However, the accuracy does not significantly improve when

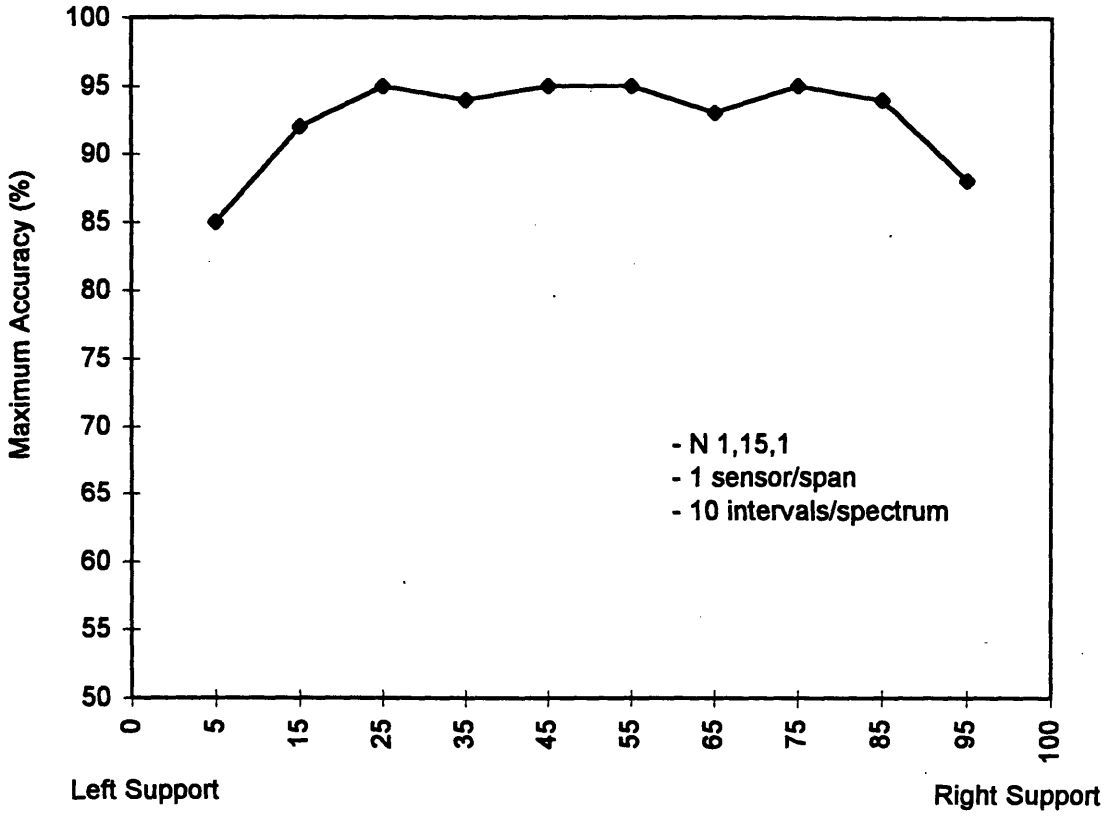


Figure 6.15: Effects of the location of sensors to the accuracy of NNET1.

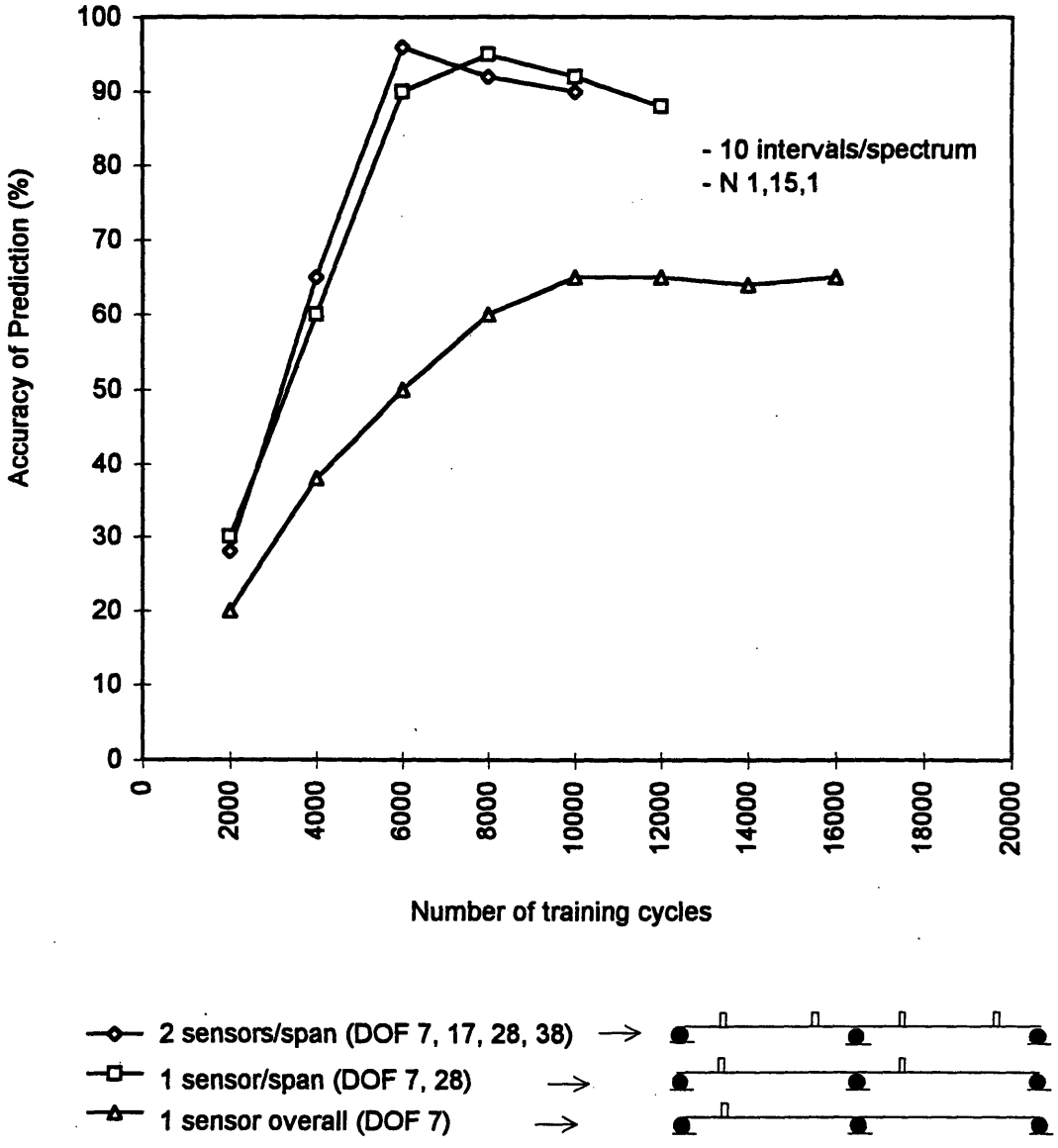


Figure 6.16: The significance of the no. of sensors to the accuracy of NNET1.

more than one sensor per span is employed. Therefore, one sensor per span is the recommended minimum number of sensors for this application.

The significance of the number of intervals representing each spectrum to the absolute accuracy of NNET1 is then investigated. Similar loading and network parameters are still employed. One sensor is located at the middle-left point of each span. The number of intervals representing each spectrum is varied from 5 to 20 intervals, with an increment of five intervals. The results show that the absolute accuracy of NNET1 increases with the number of intervals used to represent a response spectrum up to 10 intervals, and is eventually constant beyond the level. This is the indication that 10 intervals per response spectrum is essentially the optimum number for this diagnosis system configuration. Results for 5, 10, and 20 intervals are plotted in Figure 6.17.

The significance of the configuration of hammer load is also investigated. Firstly, the significance of the number of hammer impulse loads is examined. NNET1 still has 15 processing elements, and one sensor is located at the middle of the left half of each span. The number of intervals representing each spectrum is 10. NNET1 is then trained and tested with data set generated from three different configurations of hammer impulse load; i) a single hammer applied on the left span, ii) 1 hammer per span, and iii) 2 hammer per span. Fig 6.18 compares the performance of the networks that are trained with data from different excitations. It is evident that a single hammer is not sufficient since the network trained with the data created from this excitation has only around 40% absolute accuracy, compared to 92% for those networks trained with the data generated by the other excitations. This difference can be explained by the fact that 1 hammer is able to effectively excite only the first mode of vibration of the beam, whereas 1 or 2 hammers per span can excite higher modes of vibration. It was established in Section 6.3.3 that a minimum of 3 modes is needed to create the input of NNET1. Since using 2 hammers per span does not significantly improve the performance, only 1 hammer per span is considered for this application.

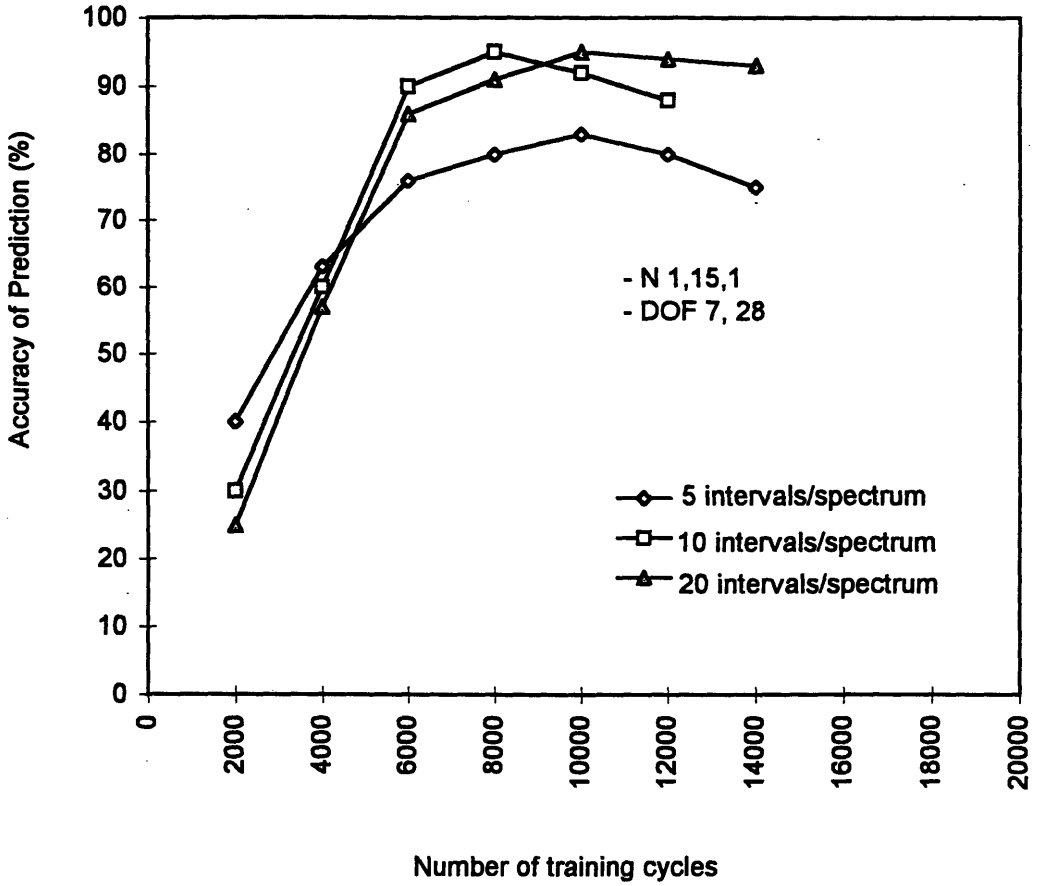


Figure 6.17: Significance of the no. of intervals to the accuracy of NNET1.

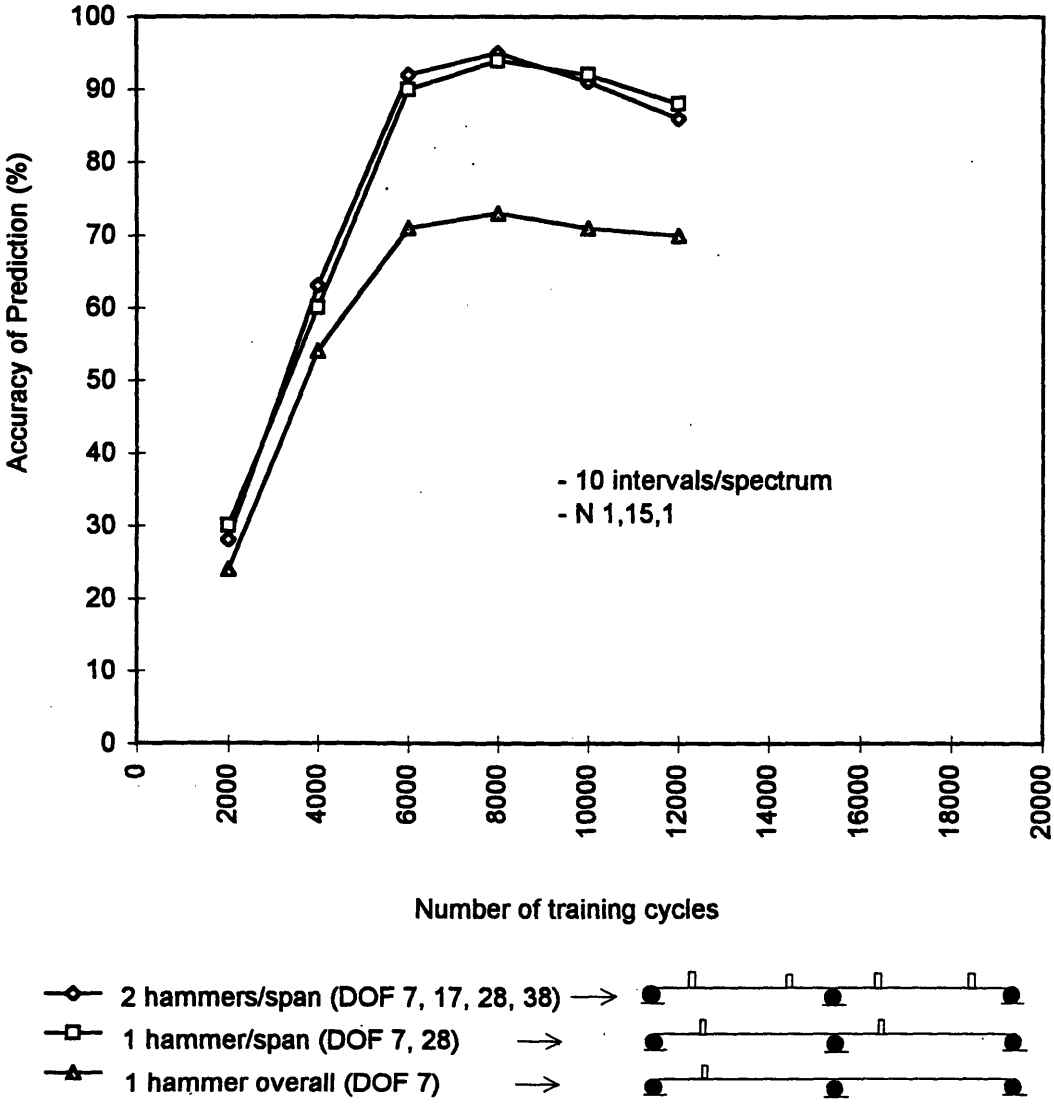


Figure 6.18: The significance of the no. of hammers to the accuracy of NNET1.

The influence of the locations of the hammer loads is then examined. The architecture of NNET1, and the data preprocessing strategy, are the same as used in the previous investigation. Fig 6.19 shows that the location of the single-hammer load has very little effect on the performance of NNET1. Fig 6.20 shows 4 different configurations of excitations employing 1 hammer load per span. Separate networks are trained and tested with the training and testing data generated by these 4 excitations respectively. The results indicate that the performance of the individual networks is approximately similar, which is up to 93% of absolute accuracy. Therefore, it is reasonable to concede the location of hammer load to be unimportant.

The significance of the number of processing elements in the hidden layer is investigated by varying the number of elements from 4 to 30, with an increment of one element. The network is trained and tested with the data created from the response spectrums of two sensors, each of which is located at the middle-left point of each span. Each spectrum is represented by 10 intervals. Fig 6.21 compares the performance of four networks (with 4, 7, 15 and 30 processing elements respectively). The result indicates that the absolute accuracy increases with the number of elements until the number exceeds 15, and tends to decrease slightly afterward.

Different testing data sets, each consisting of damage cases with a specified extent of damage, are then employed to test the damage sensitivity of the optimum NNET1. The result is shown in Table 6.4, which demonstrates accuracy improvement with the extent of damage. The accuracy is very good when the extent of damage is higher than 5% reduction of EI, but becomes poor for the damage with lower extent.

The performance-based design process used to design NNET1 can also be employed to optimize NNET2. Simulation studies indicate that NNET2 should have 27 processing elements, the time response data should be collected using at least 1 sensor per span, and each response spectrum should be represented by 12 intervals. For hammer impulse loads, at least one hammer load per span is needed. Based on the result of the

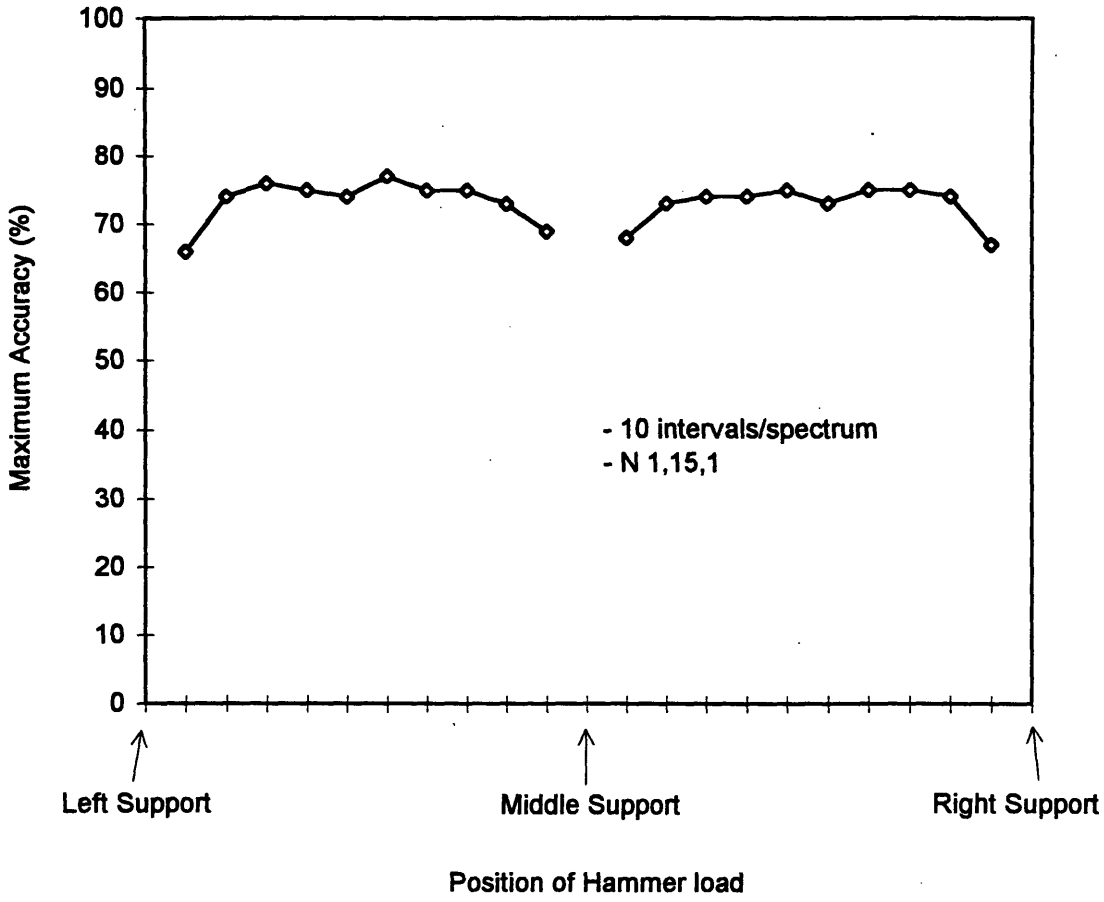


Figure 6.19: Significance of the location of single hammer load.

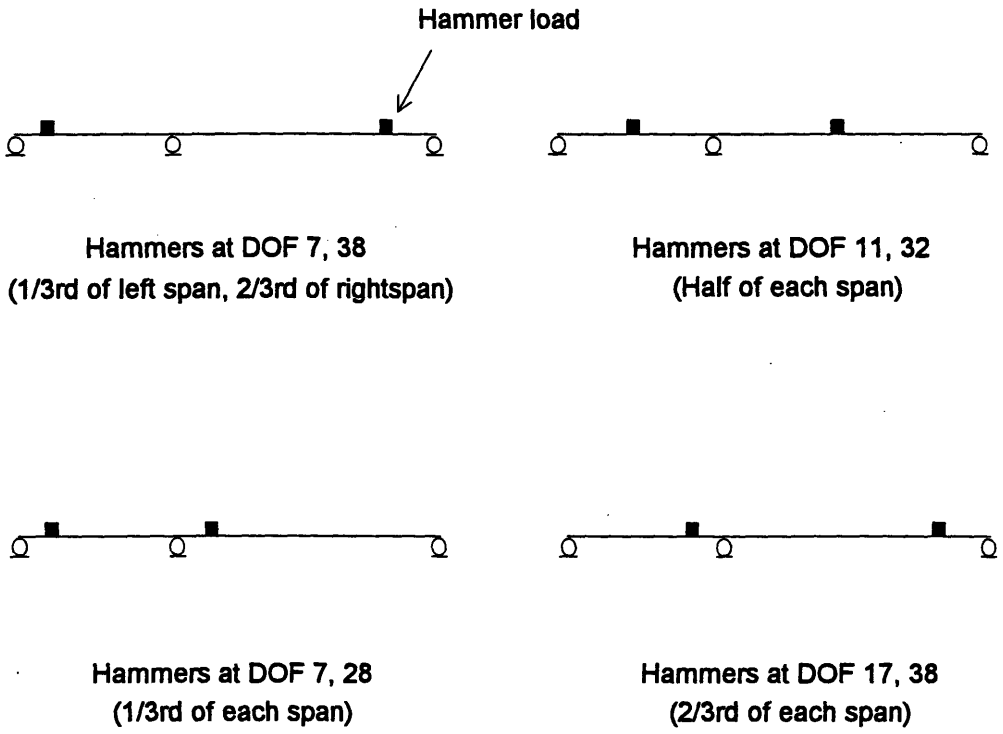


Figure 6.20: Combination of locations of the 2-hammer case (1 hammer/span).

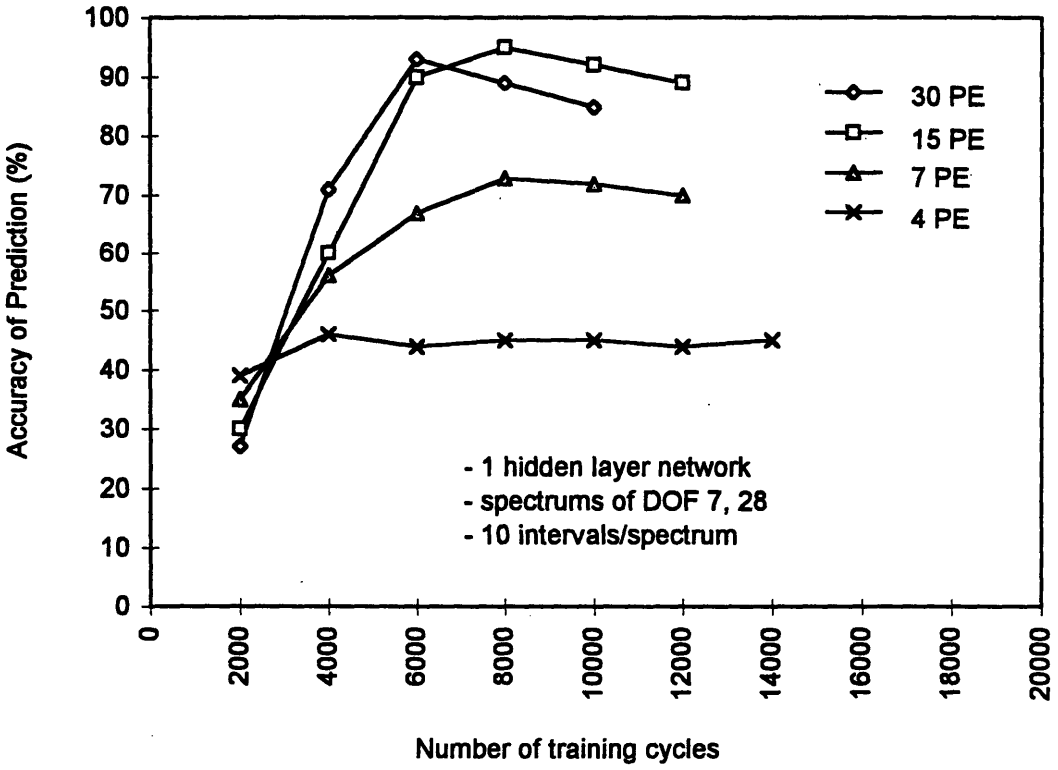


Figure 6.21: Significance of the number of processing elements in the hidden layer to the accuracy of NNET1.

performance study using the training and testing data mentioned earlier, a properly trained optimum NNET2 is able to predict the extent of damage with the absolute accuracy up to 83%.

Extent of Damage (% of *EI* reduction)	Accuracy of Optimum NNET1 (% of correct diagnosis)
10	96
5	92
4	78
3	48

Table 6.4: The damage sensitivity of the optimum NNET1.

6.4.4 Observation

To avoid overtraining, a trial and error-based approach, or the cross-validation method described in Chapter 3, needs to be employed to find the optimum configuration of NNET1 and NNET2. The results also indicate that a 1-hidden-layer network with 15 processing elements is optimal for this application. The time response data should be collected from at least 1 sensor per span, and each response spectrum should be represented by 10 intervals. At least one hammer load per span should be used as the excitation. A properly trained optimum NNET1 is able to predict the location of damage as small as 5% reduction of *EI* with the absolute accuracy more than 90%.

Simulation data indicates that the performance of the diagnosis system based on the response spectrum approach is sufficient for this single-point damage application. The approach is feasible when the monitored structures are subject to inconsistent ambient excitation (low frequency, or very weak, excitation), which makes it difficult to employ the ambient vibration & mode shape approach.

6.5 Applicability to Other Structures

There are many potential applications for single-point damage diagnosis. For example, a single-point damage diagnosis system for the frame structures shown in Fig 6.22 can be established by following the same procedures employed in building the diagnosis systems for the 2-span beam demonstrated in the previous section.

The mathematical model, or a scaled model, of the frame can be constructed and employed to create the training and testing data. Either the mode shape or response spectrum approach can be employed. If there is regular earthquake or wind gust in the area, it can be used as the ambient excitation for generating training and testing data. Hammer impulse generators or mechanical vibrators can also be used to generate prespecified excitation, if ambient excitation is not available.

The training and testing data can be created by using the procedure previously shown in this chapter. The time history response of the frame due to an excitation should be recorded from the sensors on each floor, and then used in creating mode shapes or response spectrums that correspond to different damage cases. The vibrational signatures are then further processed by the data preprocessing procedure similar to those employed in the previous sections, and used as input of NNET1 and NNET2. Only minor adjustments on the size of the input and output layers of NNET1 and NNET2 are needed to correspond to the size of their new inputs and outputs. The performance test, which is used for configuring the optimum architecture of the diagnosis system, can be similarly performed after the all the design variables are identified.

6.6 Discussion and Summary

The simulation studies of the preliminary application on an idealized 2-span beam demonstrate good potential of the neural network-based structural diagnosis system on single-point damage diagnosis. The performance of the optimized diagnosis systems, based on either the mode shape or response spectrum approach, is also similar. Therefore, the

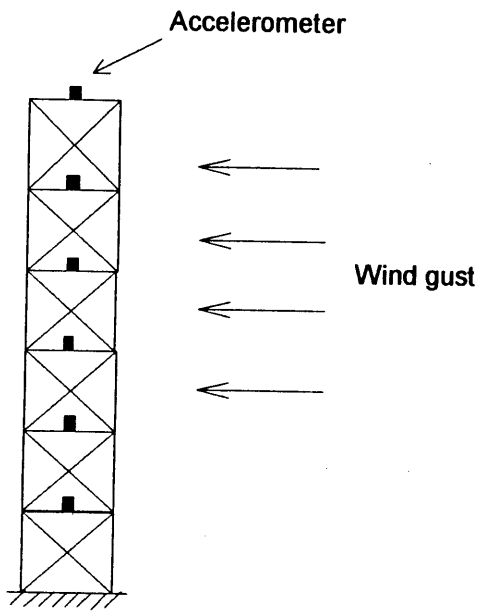
choice of excitation and vibrational signature for a specific application could be made by considering the practical applicability of the choices (Green, 1995). Increasing the information contained in the input of NNET1 and NNET2, such as increasing the number of mode shapes or response spectrums used to create each *normalized input pattern*, usually leads to better prediction performance of the diagnosis system until an optimum point is reached.

Finally, the optimum diagnosis systems, mode shape or response spectrum approach, are tested with the testing data set that consists of 50 multiple-point damage cases, each of which has more than one damaged beam element. The damaged beam elements of each case are randomly selected, and so is the extent of damage. The result demonstrates that the diagnosis systems that are optimized for single-point damage can predict the location of multiple-point damage with less than 10% accuracy. Therefore, it can be assumed that the diagnosis systems that are optimized for single-point damage cannot be applied to multiple-point damage problem.

However, the neural network-based diagnosis system for single-point damage can also be applied to other structural diagnosis problems that single-point damage condition applies. Only some detail adjustment in the design procedure is required to apply the basic architecture to a specific application. Therefore, the approach is proved very flexible, and should be very beneficial for structural engineers.

Application for high-rise structure

Ambient excitation & mode shape approach
- wind gust as excitation.
- perform performance analysis to determine location of sensors, and optimum diagnosis system.



Application for low-rise structure

Fixed excitation & response spectrum approach
- cyclic loading generator provides excitation.
- perform performance analysis to determine location of sensors, and optimum diagnosis system.

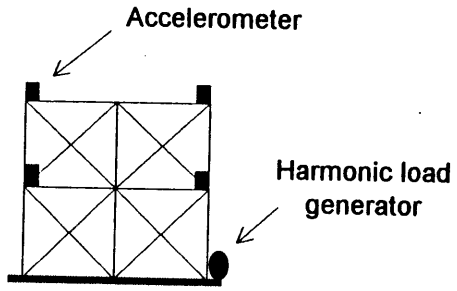


Figure 6.22: Application on frame structures.

Chapter 7

Multiple-Point Damage Diagnosis: A Case study

7.1 Introduction

In this chapter, a neural network-based diagnosis system for a 4-span bending beam with multiple-point damage is developed and evaluated. Since multiple-point damage leads to an excessive number of different damage cases, a combination of global and local structural diagnosis systems is employed. The global diagnosis system is used to identify which spans are damaged. Each damaged span is then diagnosed with a local diagnosis system that has been customized for the particular span to detect the location and extent of damage. The architecture and operation of the combination of diagnosis systems is similar to the model described in Section 5.4.2. Two choices of excitation and vibrational signature are employed for global diagnosis: i) ambient excitation & mode shape and, ii) prespecified excitation & response spectrum. However, only the prespecified excitation & frequency transfer function approach is employed for local diagnosis. Simulation studies are performed with the global diagnosis system, considering both the mode shape and response spectrum signatures. The applicability of this global/local strategy to other structures with multiple-point damage is also discussed.

The research domain is similar to that of the 2-span beam application discussed in Chapter 6. However, the objective here is to identify only the locations of damage, not their extent, so the NNET2 networks are not considered. The actual physical system is

modeled as a linear beam with negligible transverse shear deformation. The neural networks are taken as one-hidden-layer feedforward networks with back propagation training. Ambient excitation is modeled by a single-wheel moving load, while prespecified excitation is modeled by a set of hammer impulses.

7.2 Description of Simulation Model

The 2-dimensional unsymmetrical 4-span bending beam model shown in Fig 7.1 is taken as the model of the real 4-span beam. The model consists of 16 beam elements, with 4 elements representing each span. A typical beam element is shown in Fig 7.2. The mathematical model used is the Timoshenko's bending beam formulation. The length of the two right spans is 40 meters, while the two left spans are 25 meters long. Each beam element has 4 degree of freedoms (DOFs); 2 translations and 2 rotations. The element stiffness and mass matrices are defined by Eqs. 6.1 and 6.2 respectively, which are listed here for convenience.

$$k = \frac{EI}{l}\begin{bmatrix} 4 & 6/l & 2 & -6/l \\ 6/l & 12/l^2 & 6/l & -12/l^2 \\ 2 & 6/l & 4 & -6/l \\ -6/l & -12/l^2 & -6/l & 12/l^2 \end{bmatrix}, \qquad (6.1)$$

$$m = \frac{\rho\, l}{420}\begin{bmatrix} 4l^2 & 22l & -3l^2 & 13l \\ 22l & 156 & -13l & 54 \\ -3l^2 & -13l & 4l^2 & -22l \\ 13l & 54 & -22l & 156 \end{bmatrix}, \qquad (6.2)$$

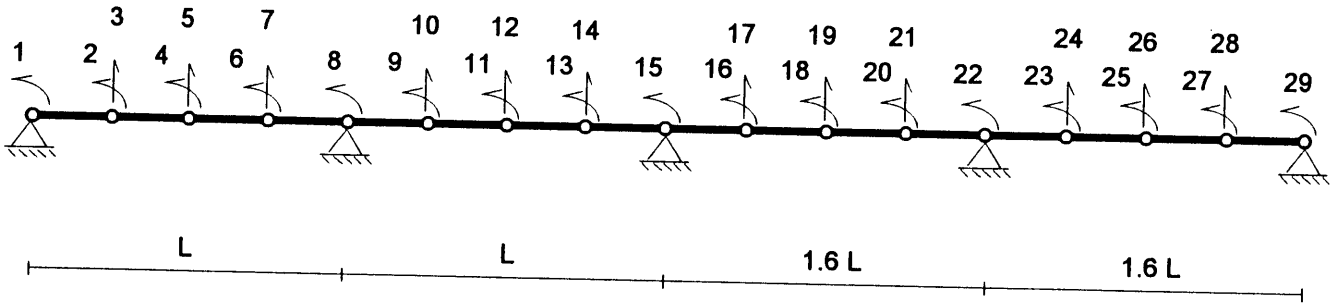


Figure 7.1: 4-span beam model.

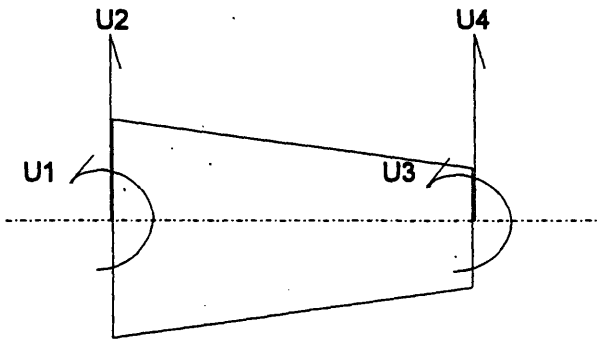


Figure 7.2: General beam element.

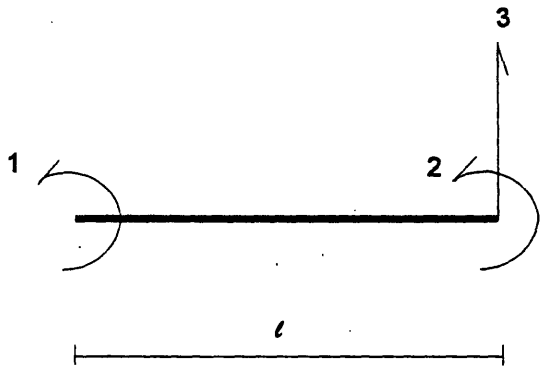


Figure 7.3: DOFs of beam element no.1

where E is the modulus of elasticity, I is the moment of inertia of the bending axis, l is the length, and ρ is the average mass per unit length of the element. The matrices of the elements that are connected to the supports, such as beam element no.1 shown in Fig 7.3, are slightly different due to the boundary condition. This element has no translational movement at the left side due to the hinge support, so the corresponding row and column can be deleted. The reduced matrices are

$$k_1 = \frac{EI}{l}\begin{bmatrix} 4 & 2 & -6/l \\ 2 & 4 & -6/l \\ -6/l & -6/l & 12/l^2 \end{bmatrix} \tag{6.3}$$

and

$$m_1 = \frac{\rho\, l}{420}\begin{bmatrix} 4l^2 & -3l^2 & 13l \\ -3l^2 & 4l^2 & -22l \\ 13l & -22l & 156 \end{bmatrix} \tag{6.4}$$

The same approach is applied to elements 4, 5, 8, 9, 12, 13 and 16 to provide all the necessary boundary conditions for the equilibrium equations of the beam.

The stiffness and mass matrices of the complete beam model are generated by superimposing the contributions of the elements

$$K = \begin{bmatrix} k1 & & & \\ & k2 & & \\ & & \ddots & \\ & & & k16 \end{bmatrix}_{29 \text{ by } 29} \tag{7.1}$$

and

$$M = \begin{bmatrix} m1 & & & \\ & m2 & & \\ & & \ddots & \\ & & & m16 \end{bmatrix}_{29 \text{ by } 29} \qquad (7.2)$$

Note that there is overlap between each pair of neighboring k, or m, since they share some global DOFs.

Assuming that the beam has proportional damping (i.e. Rayleigh damping), the damping matrix C can be constructed following the method described in Section 6.2. The damping ratios for the first 2 modes are still set at 1%.

By defining

$$U = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{29} \end{bmatrix} \qquad (7.3)$$

as the global displacement vector, which contains all the degree of freedoms (see Fig 7.1), the equation of motion of the beam model can be represented as

$$M\ddot{U} + C\dot{U} + KU = P \quad , \qquad (6.10)$$

where

$$P = \begin{bmatrix} p_1(t) \\ p_2(t) \\ \vdots \\ p_{29}(t) \end{bmatrix} \tag{7.4}$$

is the force on the beam model as a function of time, t. Given the force function and all other parameters of the beam model, the time history response of the beam model can be determined by solving the equation of motion.

For this application, the beam is assumed to have a constant EI of 5.34×10^9 N·m^2 over its length. The average mass per unit length is 9880 kg/m. Fig 7.4 shows the first three mode shapes, and their corresponding frequencies. The response of the beam model due to the excitation is determined by a direct time integration method (Runge Kutta method) performed in MATLAB. The time interval of the integration is still taken as 0.01 second.

Similarly to the 2-span beam case study, damage is introduced in the model by lowering the EI in the stiffness matrices corresponding to the elements that are selected as damaged elements. However, in this application, a damage case may involve more than one damaged element.

7.3 Global Structural Diagnosis: Mode Shape Approach

The global diagnosis system is a modified version of the basic neural network-based diagnosis system (shown in Fig 5.7). The diagnosis system has only one neural network, NNET1, which predicts the damaged spans given the global excitation. As earlier mention in Chapter 6, the modal approach does not require a consistent excitation, and both ambient and prespecified excitations can be employed. In this research, ambient excitation is used for its convenience and its real-time applicability. The excitation is a mass-consistent single-wheel load of 2000 kg (4.4 kips) with velocity ranging from 40 to

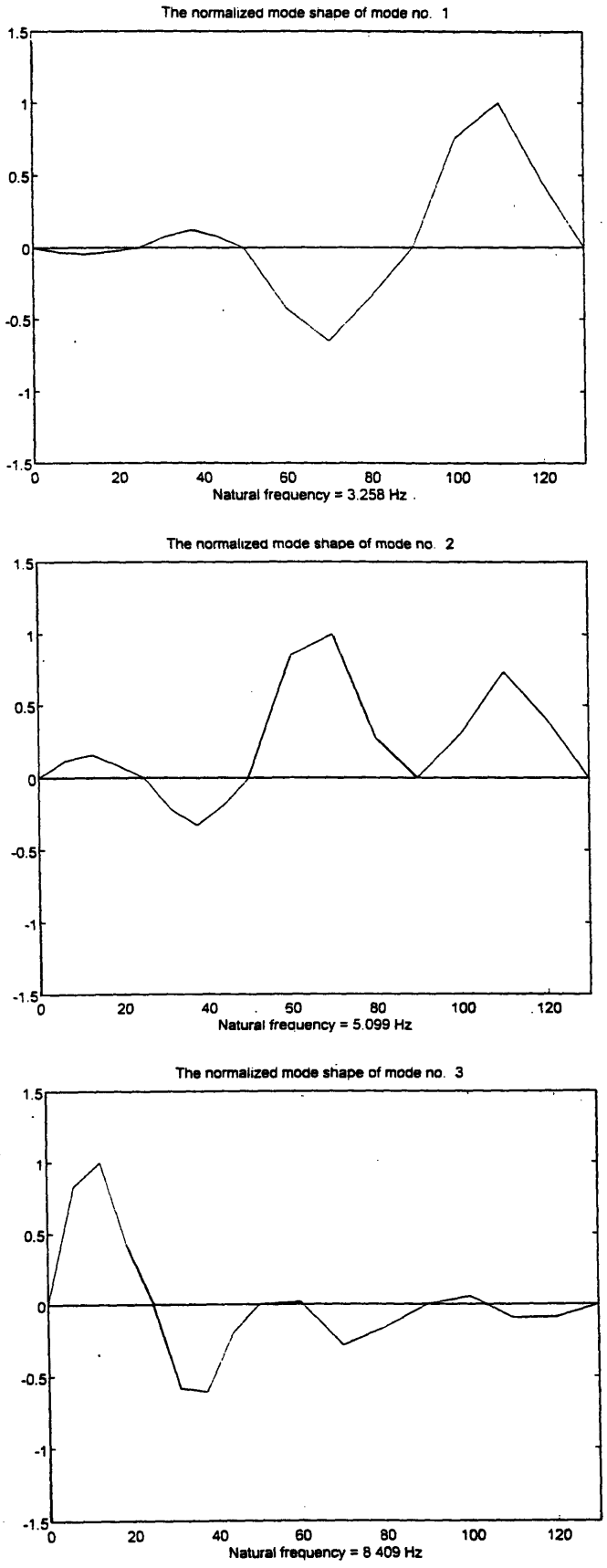


Figure 7.4: First 3 modes of vibration of the 4-span beam.

60 mph. The travel time is still more than ten times of the first three fundamental periods of the beam. Therefore, the moving load effectively excites the first three modes of vibration (Humar, 1990 and Humar et al, 1993).

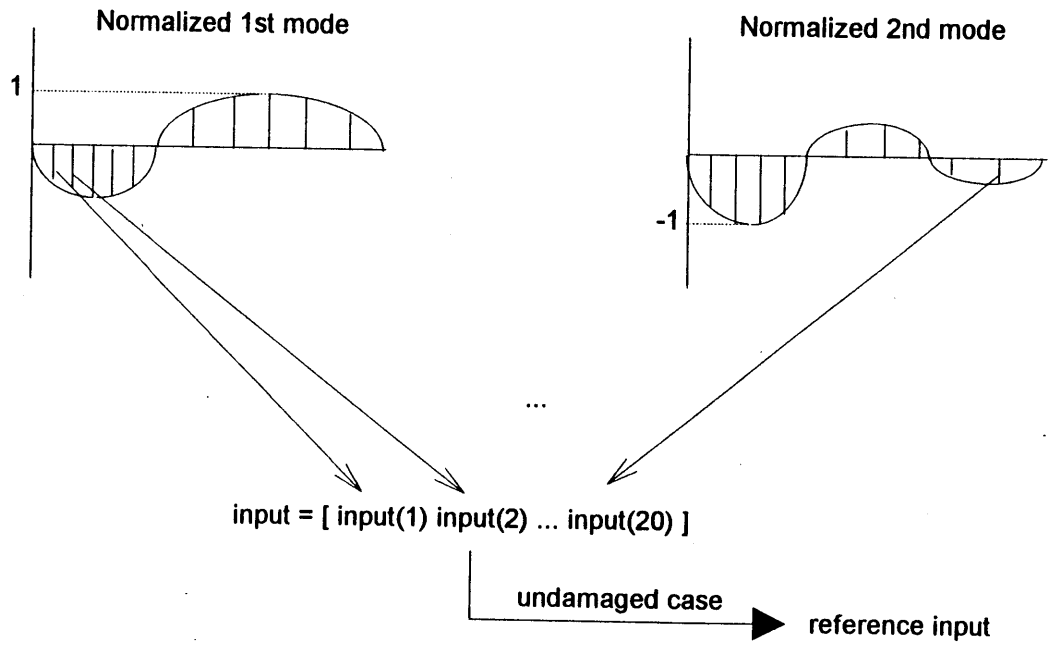
7.3.1 Data Preprocessing Strategy

The free-vibration response of the simulation model with various damage conditions is collected and processed by a modal analysis routine (Appendix A) in order to determine the corresponding mode shapes of the 4-span beam. The mode shapes, and the number of points representing each mode shape, which are used to create the input for NNET1 then have to be specified. Fig 7.5 illustrates the case when two mode shapes, with 10 points representing each mode, are used as input for NNET1. The mode shapes corresponding to a particular damage condition are normalized so that their maximum amplitude is equal to unity. The value of the points representing the normalized mode shapes are then used to create a vector called "input" vector,

$$input = \begin{bmatrix} \text{data point no.1 of mode shape no.1} \\ \vdots \\ \text{data point no.}m\text{ of mode shape no.1} \\ \text{data point no.1 of mode shape no.2} \\ \vdots \\ \text{data point no.}m\text{ of mode shape no.}n \end{bmatrix}_{mn\times 1}, \qquad (7.5)$$

where m is the number of points representing each mode shape, and n is the number of mode shapes considered.

The Reference Value Approach is then employed, and the remaining data preprocessing procedures follow the procedures described in Section 6.3.1. As mentioned earlier, the global structural diagnosis system identifies only damaged spans, not damaged



input = input - reference input

input = norm(input)

output = [output(1) output(2) ... output(22)]

output(i) = 1 , when ith beam element is damaged.

= 0 , otherwise.

Figure 7.5: The data preprocessing approach.

beam elements. Therefore, the *output* vector corresponding to each *normalized input pattern* is

$$output = \begin{bmatrix} out_1 \\ out_2 \\ out_3 \\ out_4 \end{bmatrix}, \tag{7.6}$$

where $out_i \in \Re[0,1]$. The "expected" *output* vector corresponding to a particular damage case is also represented by Eq. 7.6, where out_i is 1 if there is reduction of EI of any beam element in the ith span from the left, and 0 otherwise. Since the beam has 4 spans, the size of the *output* vector is 4 by 1. Each *normalized input pattern* and its corresponding expected *output* can then be employed as a training, or testing, sample of NNET1.

7.3.2 Configuration and Training of Neural Networks

A 1-hidden-layer feedforward network with back propagation training algorithm is employed as NNET1. According to Eq. 7.5, NNET1 needs *mn* nodes in the input layer, which corresponds to the size of the *normalized input pattern*. It also needs 4 output nodes, which represent the size of the *output* vector. The transfer function of the processing elements in the hidden layer is the tangent sigmoid function (Eq. 6.19), while that of the output layer is the log sigmoid function (Eq. 6.20). Both functions are illustrated in Fig 6.8. The initial value of all the connection weights is still randomized value between -1 to 1. The threshold value is set at 0.8. The training algorithm also utilizes the momentum term and adaptive learning rate.

Four training data sets, containing 50, 100, 500, and 1000 different damage cases respectively, are created. The first four damage cases of each set are taken as the damage-free case. The remaining cases are variations of the basic cases shown in Table 7.1.

Case No.	Damaged Span(s)
1	1
2	2
3	3
4	4
5	1,2
6	1,3
7	1,4
8	2,3
9	2,4
10	3,4
11	1,2,3
12	1,2,4
13	1,3,4
14	2,3,4
15	1,2,3,4

Table 7.1: The basic damage cases for the global diagnosis system.

Combination No.	Damaged Beam Element(s)
1	1
2	2
3	3
4	4
5	1,2
6	1,3
7	1,4
8	2,3
9	2,4
10	3,4
11	1,2,3
12	1,2,4
13	1,3,4
14	2,3,4
15	1,2,3,4

Table 7.2: The possible combinations of damaged elements in a particular span.

Note that, for a particular damage case, the combination of the damaged elements in any damaged span is randomly selected from the basic combinations shown in Table 7.2. The extent of damage of any damaged element randomly ranges between 5 to 80% reduction of *EI*.

The training procedure of NNET1 follows the approach described in Section 5.2 (see Fig 5.2). Fig 7.6 shows the convergence of the Sum-Square Error (*SSE*) of a NNET1 with 16 processing elements in the hidden layer that is trained by a data set with 100 training samples.

The testing data, which contains a total of 50 different damage cases, is similarly created. The damage cases cover the basic damage cases shown in Table 7.1. The combination of the damaged elements in any damaged span is also randomly selected from the basic combinations shown in Table 7.2. The extent of damage of any damaged element randomly ranges between 5 to 90% reduction of *EI*. In order to assess the ability of the "trained" diagnosis systems to detect an arbitrary damage pattern, all damage cases in the testing data set are different from the damage cases of the training data set .

The testing data are then used to test the performance of the trained NNET1 in predicting the damaged spans. The testing is performed by passing the *normalized input patterns* of the testing data set through the trained network, and comparing the network *outputs* to the "expected" *outputs* corresponding to the damage cases.

7.3.3 Performance Studies

Four variables are investigated for their influence on the accuracy of the global diagnosis system; i) number of mode shapes used to create input, ii) number of points representing each mode, iii) number of processing elements in the hidden layer of NNET1, and iv) number of training samples of the training data set. Fig 7.7 demonstrates the influence of the number of mode shapes used to create the input for NNET1 on the accuracy of NNET1 that is trained by the training data set with 100 training samples.

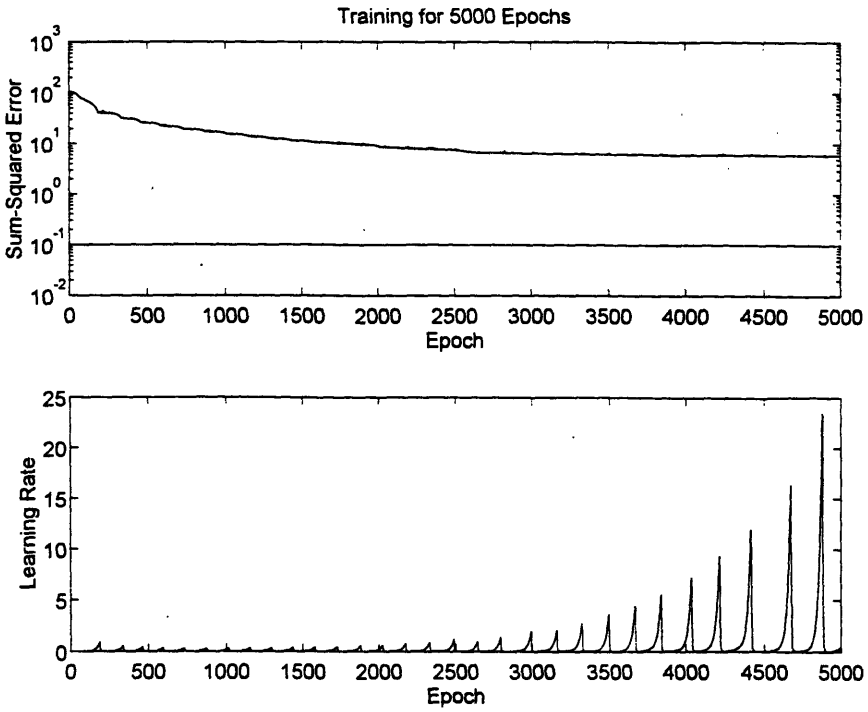


Figure 7.6: The convergence of the Sum-Squared Error of the NNET1 of the global structural diagnosis.

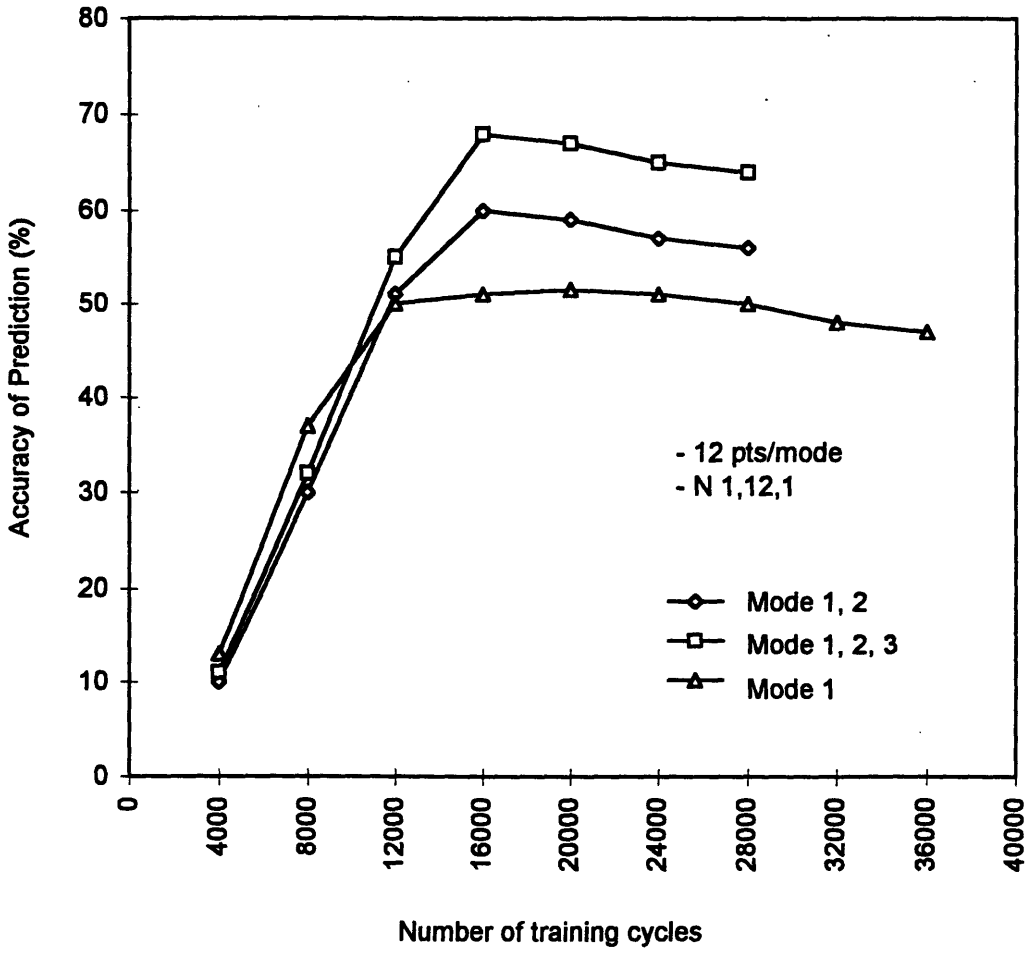


Figure 7.7: The significance of the no. of mode shapes to the accuracy of NNET1.

The accuracy shown in the plot is the percentage of correct predictions by NNET1 on the testing data. This particular NNET1 has 12 processing elements in the hidden layer, and employs 12 points to represent each mode shape. The result shows that the accuracy of NNET1 increases with the number of mode shapes, while the training cycles needed to reach the absolute accuracy decreases. By using the first 3 mode shapes, the NNET1 trained for 16000 cycles can predict the damaged spans with as high as 68% accuracy.

Fig 7.8 demonstrates the effect of the number of points representing each mode shape on the accuracy of NNET1. In this case, NNET1 has 12 processing elements, and the input data is created from the first three mode shapes. The training data set with 100 damage cases is used. In this investigation, the number of points representing each mode shape is varied from 5 to 25 points, with a one point increment. The result indicates that the absolute accuracy of NNET1 increases with the number of points up to the level of 12 points. Beyond this level, there is no additional accuracy. This suggests that 12 points per mode shape is the optimum number for this application.

The significance of the number of processing elements in the hidden layer of NNET1 to its accuracy is investigated by varying the number of elements from 2 to 25 elements, with a one element increment. Fig 7.9 shows the performance of three NNET1's (each with 6, 12, and 18 processing elements in their hidden layer respectively) that are trained and tested with the input data created from the first three mode shapes, each represented by 12 points. The training data set with 100 damage cases is still used. The simulation result indicates that NNET1 with 12 processing elements in the hidden layer is optimal.

The significance of the number of damage cases in the training data set is demonstrated in Fig 7.10. The NNET1 has 12 processing elements in the hidden layer, and uses all three mode shapes to create the input data. The number of points representing each mode shape is 12. Five NNET1's are respectively trained with 5 different training data sets, which contain 50, 100, 500, and 1000 damage cases respectively (see Section

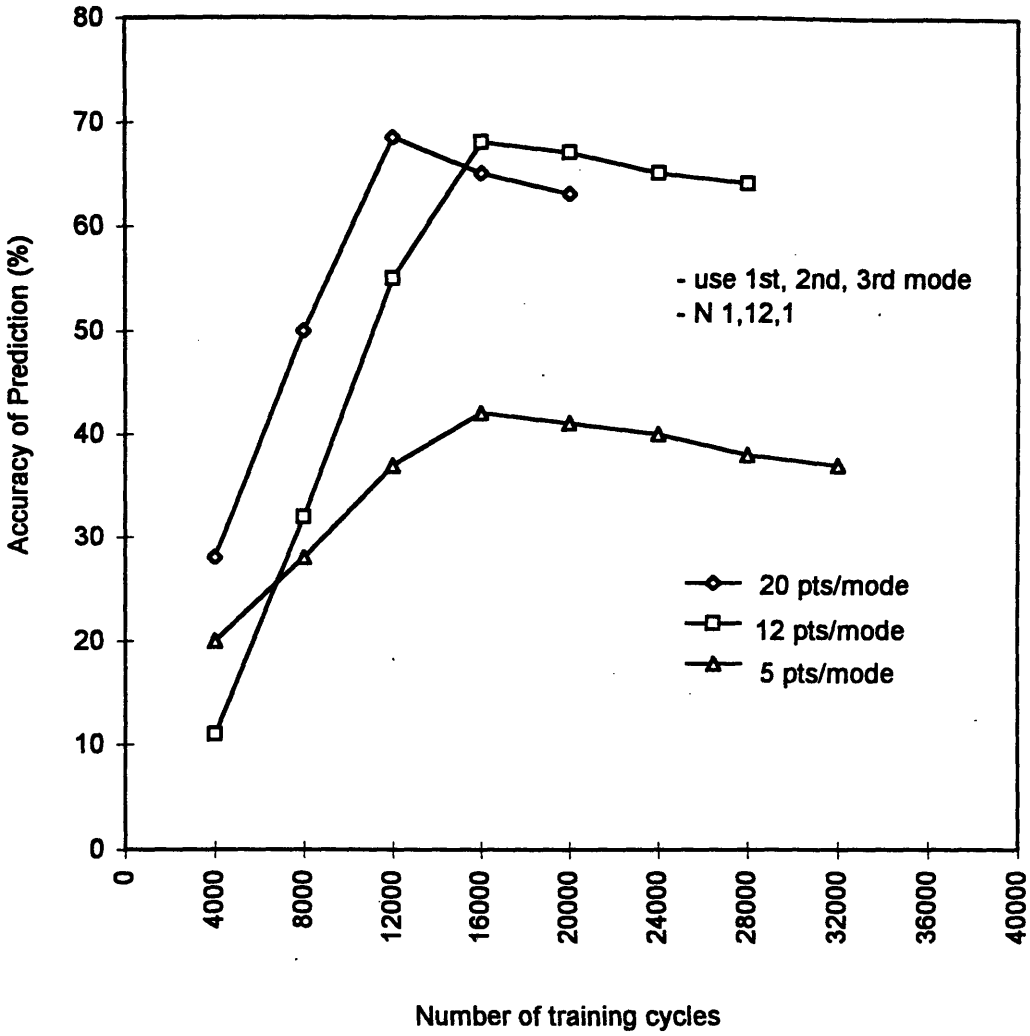


Figure 7.8: The significance of no. of points representing each mode shape to the accuracy of NNET1.

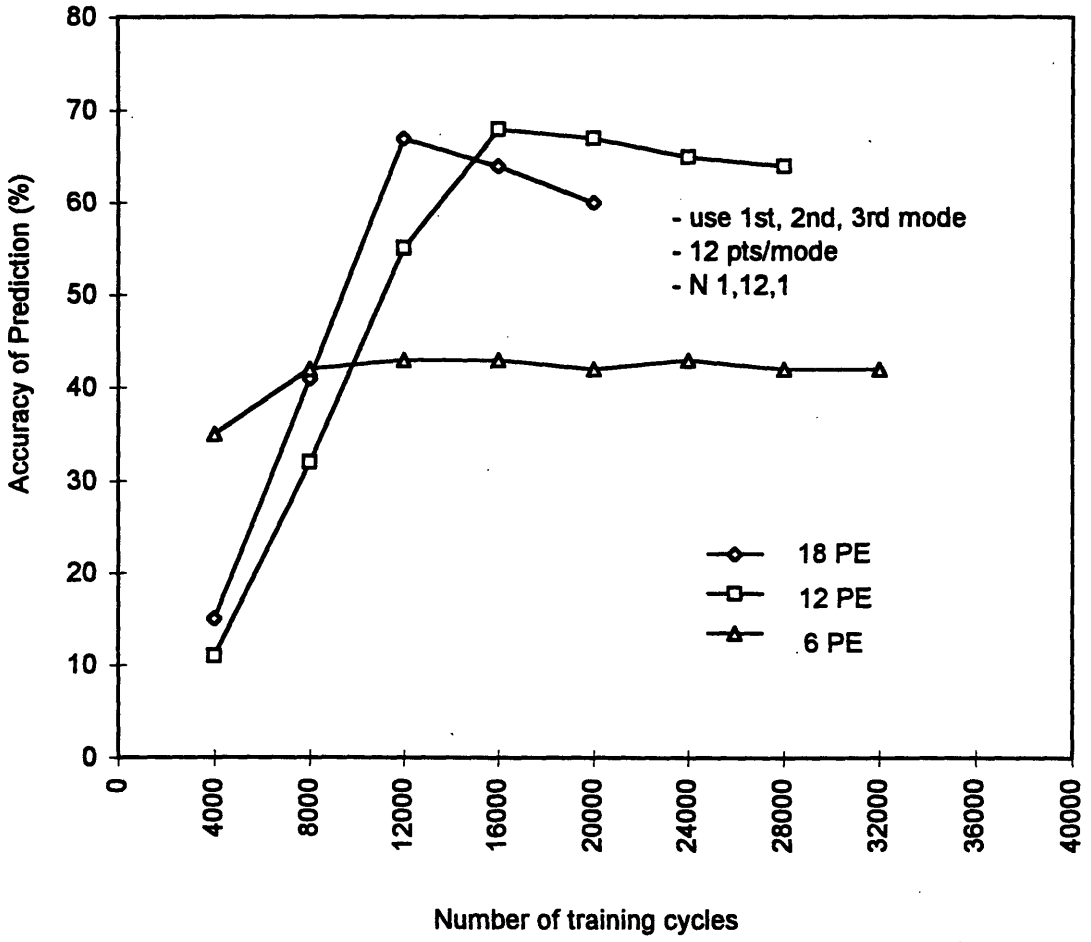


Figure 7.9: The significance of the no. of processing elements in the hidden layer to the accuracy of NNET1.

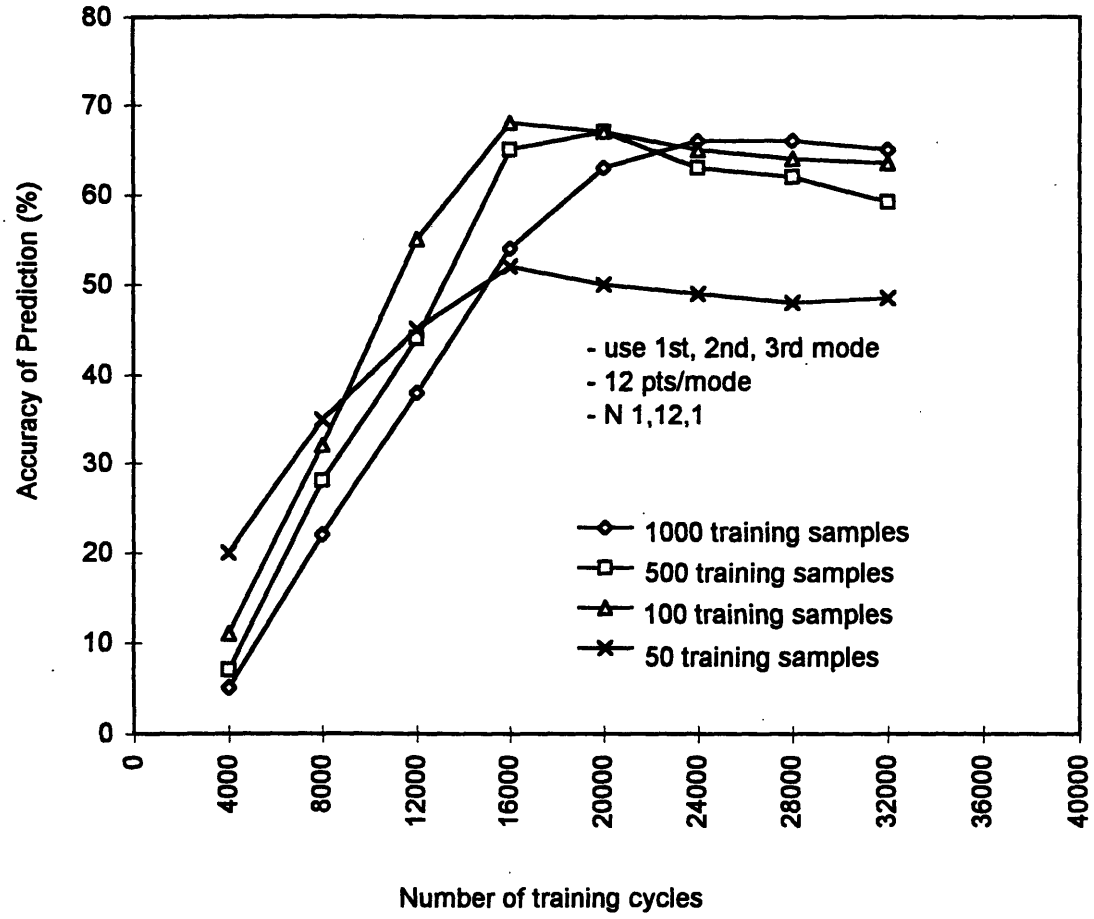


Figure 7.10: The significance of the no. of training samples to the accuracy of NNET1.

7.3.2 for more detail). Fig 7.10 demonstrates that the absolute accuracy of NNET1 increases with the size of the training data set. However, the accuracy does not significantly improve when the number of damage cases exceeds 100.

All simulation results exhibit the effect of overtraining. Therefore, a trial and error approach, or a cross-validation approach, has to be employed in order to find the optimum NNET1. The results also indicate that a 1-hidden-layer network with 12 processing elements is a satisfactory choice for NNET1. The input data should be created from the first three mode shapes, and each should be represented by 12 points. The training data set should contain at least 100 different damage cases.

The result from simulation studies shows that the information from mode shapes alone is not sufficient for NNET1 to predict the damaged spans at an acceptable level of accuracy. The changes of modal frequencies, which are also available from the mode shape compilation process (Appendix A), may provide the additional information needed. In what follows, each existing input of NNET1, which is created from the mode shapes corresponding to a particular damage case, is incorporated by their corresponding modal frequencies. New sets of training and testing data are created using the damage cases mentioned in Section 7.3.2. The prediction performance of the optimum NNET1 based on the new data set is also investigated.

The procedure for creating an input-output pair for the training or testing data set of NNET1 is similar to the one previously employed in this section. However, the new approach also incorporates information about the modal frequencies into the *input* vector as follows.

$$input = \begin{bmatrix} \text{data point no.1 of mode shape no.1} \\ \vdots \\ \text{data point no.}m\text{ of mode shape no.1} \\ \text{data point no.1 of mode shape no.2} \\ \vdots \\ \text{data point no.}m\text{ of mode shape no.}n \\ \text{modal frequency of mode no.1 (Hz)} \\ \vdots \\ \text{modal frequency of mode no.}n\text{ (Hz)} \end{bmatrix}_{(mn+n)\times 1}, \tag{7.7}$$

where m is the number of points representing each mode shape, and n is the number of mode shapes considered.

The *input* vector that is obtained from the non-damage model is still called the "*reference data set.*" The difference between the *input* vector corresponding to any damage condition and the *reference data set* is then called the "*input pattern*" of the corresponding damage condition. The data points of each *input pattern* that represent the changes of mode shapes are normalized by the approach previously employed. The remaining data points, which represent the changes of modal frequencies, are normalized by the maximum data points of the same row among all *input patterns*. The "*normalized input patterns*" and their corresponding expected *outputs* can then be used as the training or testing data of NNET1.

The performance study of the global diagnosis system is then carried out using new training and testing data sets, which are created following the procedure described in Section 7.3.2. All the design variables investigated are the same as for the previous investigation. The test results indicate similar behavior with the previous study. The accuracy of NNET1 still increases with the number of mode shapes, the number of points representing each mode shape, the number of processing elements of NNET1, and the number of damage cases of the training data, until an optimum point is reached. For this application, the optimum NNET1 has 19 processing elements. The input data should be

created from the first 3 mode shapes, and each mode should be represented by 12 points. Most importantly, the new optimum NNET1 is able to predict the damaged spans with as high as 92% absolute accuracy, which is a significant improvement over the NNET1 that is trained by the training data that does not contain the information of modal frequencies.

Different testing data sets, each of which contains the same damage cases that have a specific extent of damage, are then employed to test the new optimum NNET1. The result is demonstrated in Table 7.3.

Extent of Damage (% of *EI* reduction)	**Accuracy of Optimum NNET1 (% of correct diagnosis)**
10	94
5	90
4	71
3	38

Table 7.3: The damage sensitivity of the optimum NNET1.

The result demonstrates that the optimum NNET1 is able to detect the damaged spans with an accuracy as high as 90%, when the reduction of *EI* of any damaged beam element is as small as 5%.

7.3.4 Observation

Simulation data indicates that, when the multiple-point damage condition is applied, the changes of mode shapes due to damages alone do not provide sufficient information for NNET1 to predict the damaged spans at an acceptable level of accuracy. However, adding the changes in modal frequencies does significantly improve the performance of the global diagnosis system based on the mode shape approach, and makes the applicability of the approach much more feasible.

7.4 Global Structural Diagnosis: Response Spectrum Approach

The design of the global diagnosis system is based on the approach described in Section 5.4.2. A set of hammer impulses located at specific locations along the beam is used to excite the structure. Each impulse generates 10000 Newton of force for a time period 0.1 second. The simulation model and all damage cases of the training and testing data sets employed in the previous study are also used here.

7.4.1 Data Preprocessing Strategy

The spectrums of the numerical simulated free-vibration acceleration responses at various locations along the beam (given a prespecified hammer impulse) are used as the signatures for the damaged structure. The Fast Fourier Transform is employed to transform these time histories into response spectra which are later smoothed with the spectrum smoothing routine in MATLAB. The procedure for creating and smoothing the response spectrums is described earlier in Section 6.4.1. Fig 7.11 shows the 20-second segment of the acceleration response for DOFs 3 and 7, and the corresponding spectrums.

In this application, the spectrums of the frequency between 0 to 10 Hz, which well cover the first 3 modes of vibration of the 4-span beam (see Fig 7.4), is employed. These spectrums are then further processed into a "*normalized input pattern,*" corresponding to the damage condition by the same data preprocessing procedure used in Section 6.4.1. The expected output of NNET1, *output*, corresponding to the damage case is created following the procedure described in the previous section. By performing simulations of the beam model with various damage conditions, the *normalized input patterns* and *outputs* corresponding to the damage cases can be used as the training or testing samples of the NNET1 of the global structure diagnosis system.

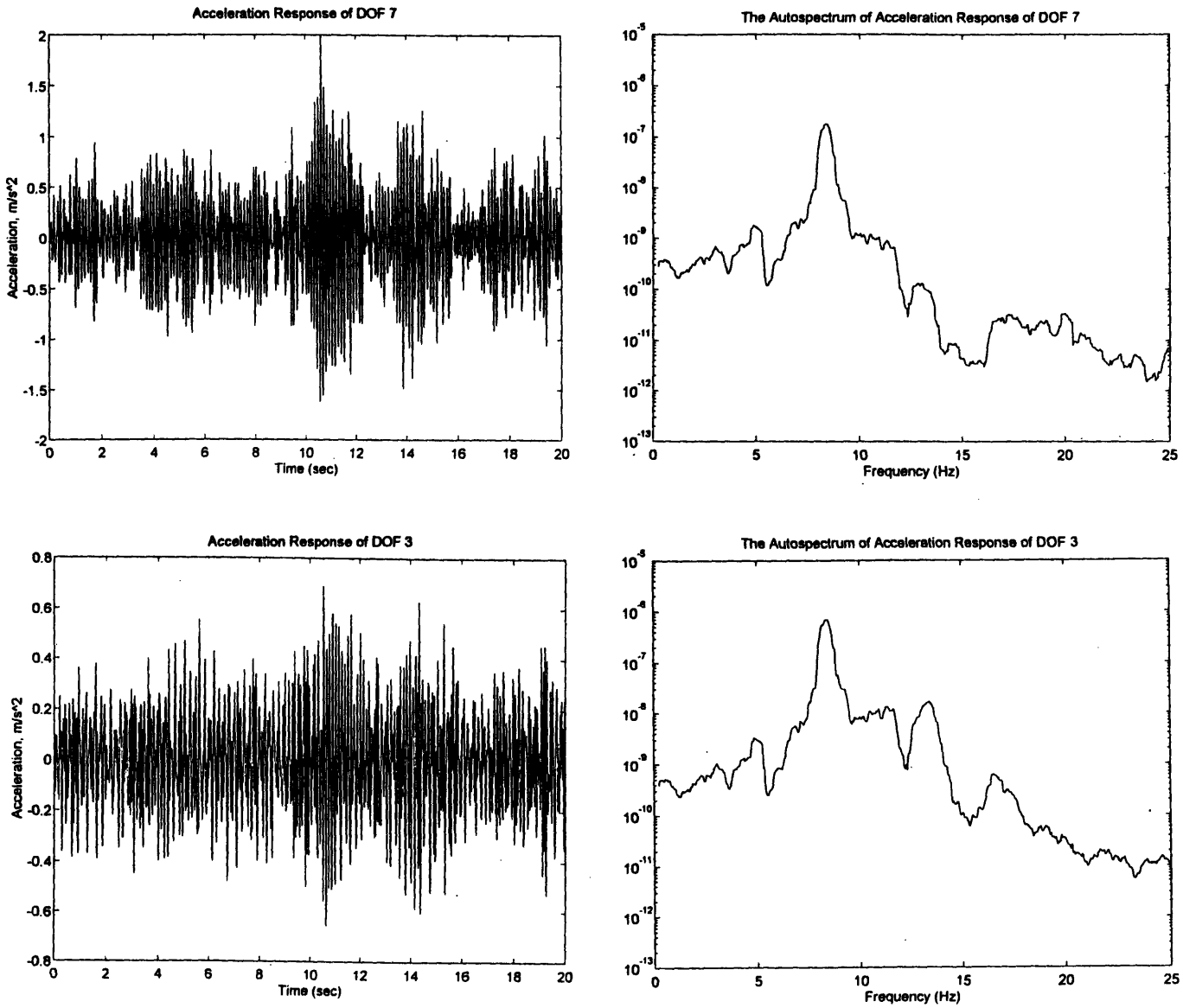


Figure 7.11: The acceleration response of DOFs 3 and 7, and their autospectrums, due to a hammer impulse.

7.4.2 Configuration and Training of Neural Networks

The configuration of NNET1, and the damage cases for the training and testing data sets used in the mode shape approach is also used here. However, the number of nodes in the input layer now corresponds to the size of the *normalized input pattern* used for the spectrum approach (see Eq. 6.23).

7.4.3 Performance Studies

Seven design variables are investigated for their influence on the performance of the diagnosis system; i) number of sensors, ii) location of the sensors, iii) number of intervals representing each response spectrum, iv) number of hammer impulse generators, v) location of hammer impulses, vi) number of processing elements in the hidden layer of NNET1, and vii) number of damage cases in the training data set.

Three NNET1's are trained, each with a different training data set. The first data set corresponds to a sensor located at the middle-left point of each of the four spans; the second and third data sets are created by locating sensors at the middle and middle-right points respectively. Each training data set contains 500 different damage cases. Four hammer impulse generators, each located at the middle-right point of each span, synchronously apply a force of 10000 Newtons for an interval of 0.1 seconds as the excitation. Ten intervals are used to represent each spectrum, and NNET1 has 18 processing elements in the hidden layer. The results, as shown in Fig 7.12, show that the location of the sensors has essentially no effect on the performance of NNET1.

Figure 7.13a demonstrates the significance of the number of sensors. The locations of the sensors are shown in Fig 7.13b. The loading and network parameters are the same as for the previous investigation. There is a substantial improvement when the number is increased to one per span. Beyond this point, there is no observable improvement. Therefore, one sensor per span is recommended as the minimum number for this application.

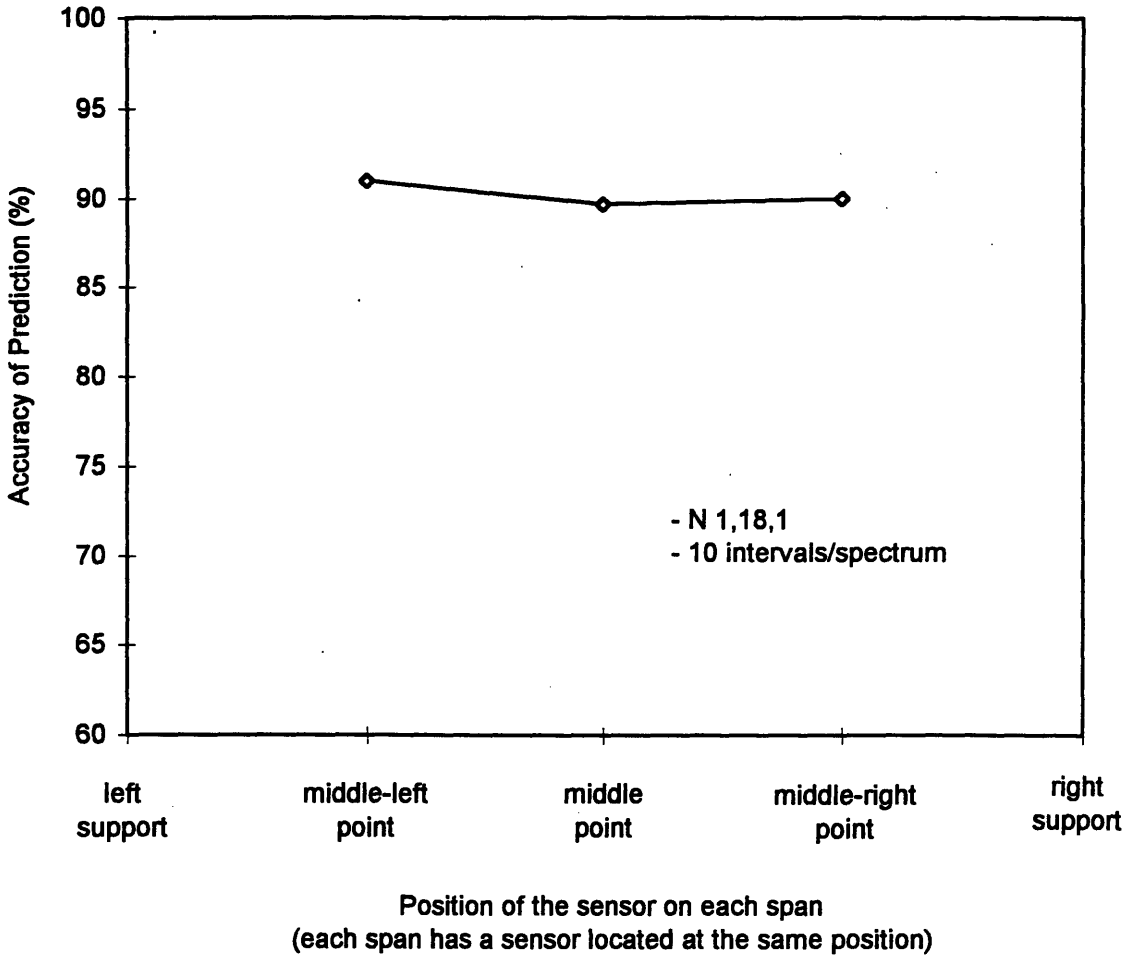
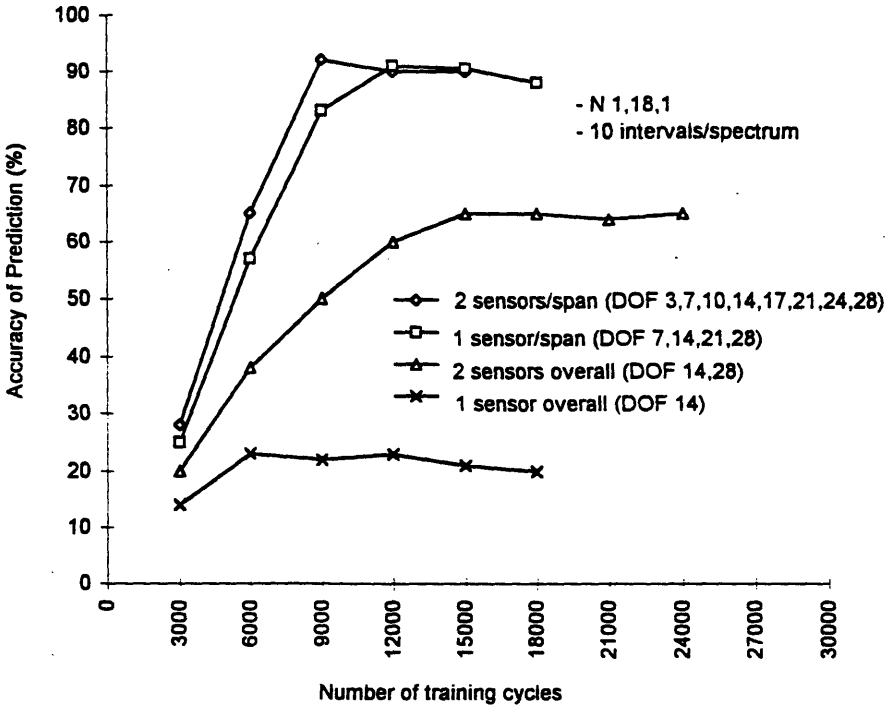
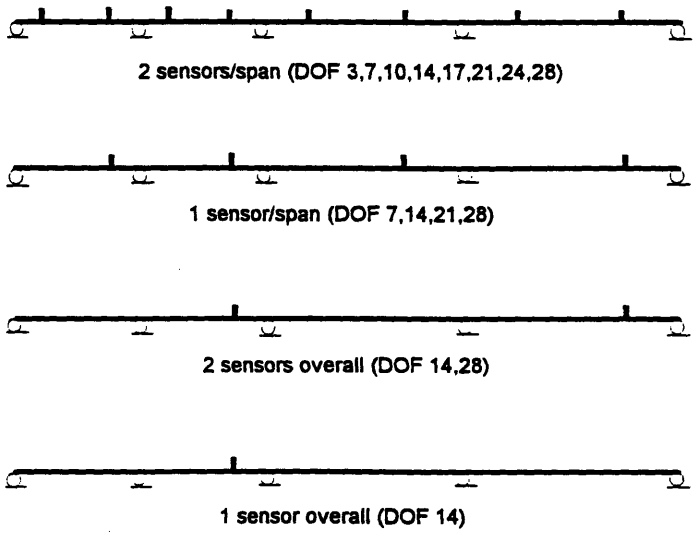


Figure 7.12: Effects of the location of sensors to the accuracy of NNET1.



a: The significance of the no. of sensors to the accuracy of NNET1.



b: The illustration of the locations of sensors.

Figure 7.13

The sensitivity of the absolute accuracy to the discretization of the response spectrum is then investigated. One sensor is located at the middle-right point of each span, and all other parameters are the same. The number of intervals representing each spectrum is varied from 5 to 20, with increments of five intervals. Figure 7.14 demonstrates the accuracy for 5, 10, and 20 intervals are employed. It appears that the absolute accuracy increases with the number of intervals up to 10 intervals, and then decreases slightly. Therefore, it is reasonable to take ten intervals per response spectrum as the optimum number for this diagnosis system configuration.

The significance of the configuration of the hammer impulse excitation is also investigated. Firstly, the significance of the number of hammer impulses is examined. NNET1 still has 18 processing elements in the hidden layer, and one sensor is located at the middle-right point of each span. Ten intervals are used to represent each spectrum. NNET1 is trained with three data sets created from three difference configurations of hammer load; i) 1 hammer on the leftmost span, ii) 1 hammer per span, and iii) 2 hammer per span (see Fig 7.15a). Each data set contains 500 different damage cases. Fig 7.15b demonstrates that one hammer is not enough to create quality training data, since the absolute accuracy is around 40%, compared to 91% accuracy of those networks trained with data created with 1 or 2 hammer-per-span excitations. This can be explained by the fact that 1 or 2 hammers-per-span excitation is more capable of exciting the 2nd and 3rd mode of vibration, which are essential as the information for the diagnosis system (based on the previous performance test on the significance of the number of mode shapes to the performance of NNET1 shown in Section 7.2.1). Therefore, it is appropriate to use at least 1 hammer-per-span excitation for this application.

Fig 7.16 demonstrates the effect of the location of hammers of the 1-hammer-per-span excitation to the performance of NNET1. The first excitation configuration has each hammer located at the middle-left point of each span. The second and third configuration have each hammer located at the middle and middle-right point of each span respectively.

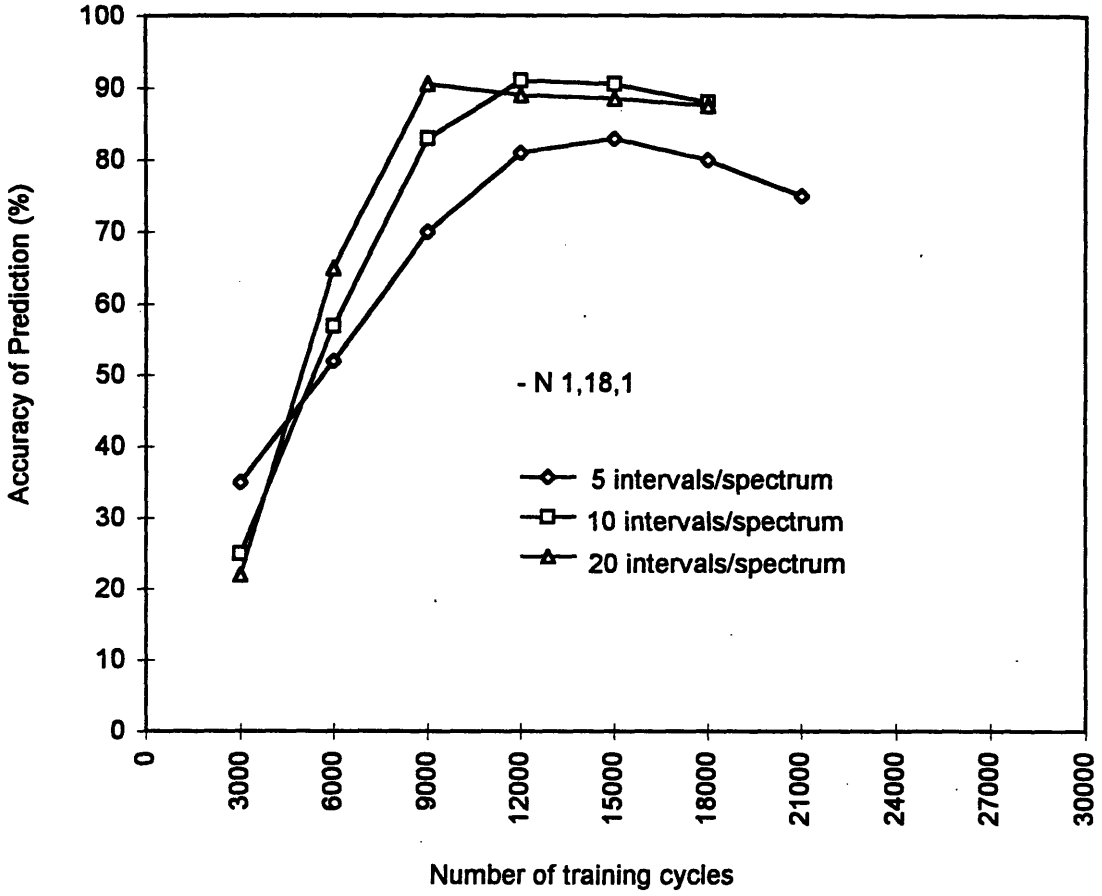
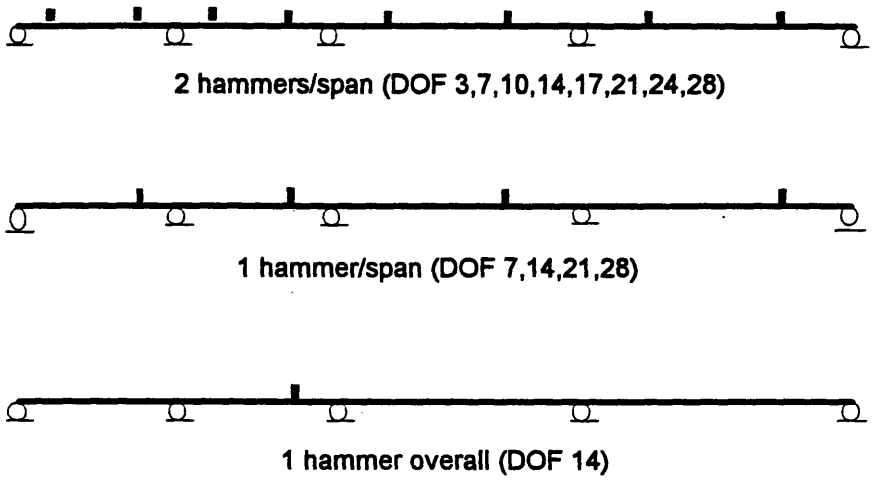
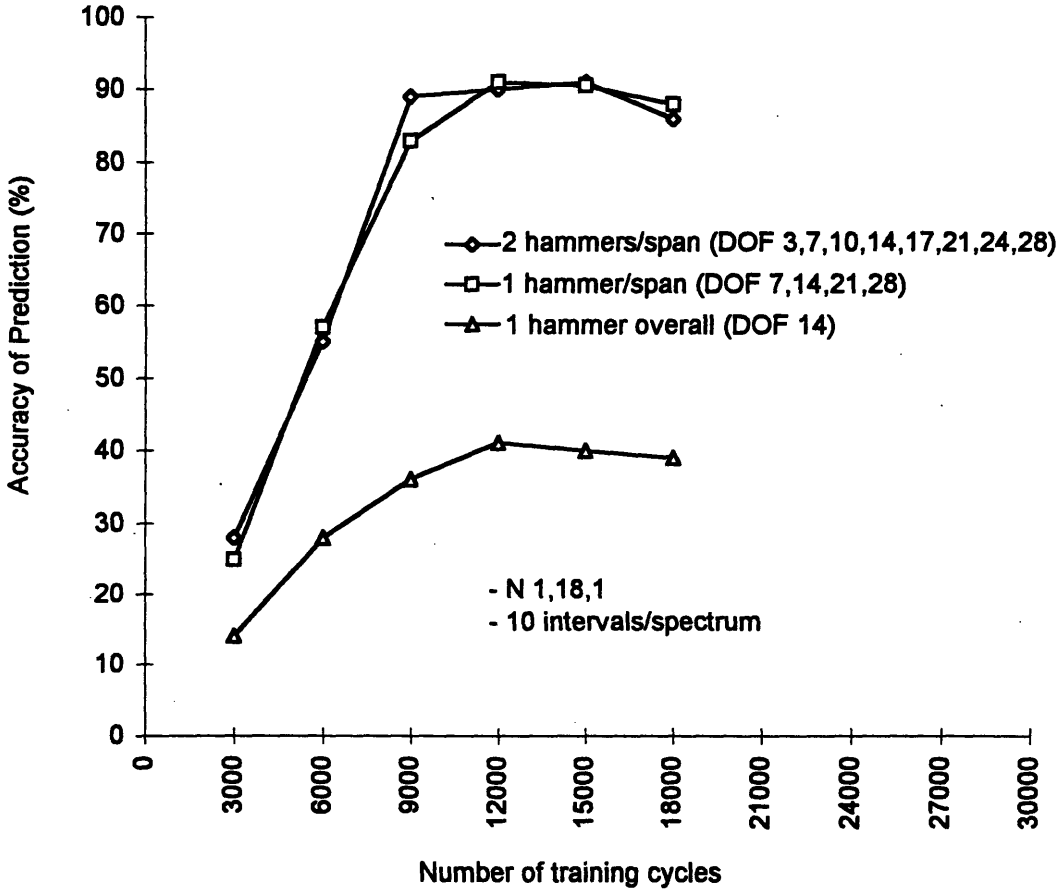


Figure 7.14: The significance of the no. of intervals to the accuracy of NNET1.



a) The locations of hammers.



b) The significance of the number of hammers to the accuracy of NNET1.

Figure 7.15

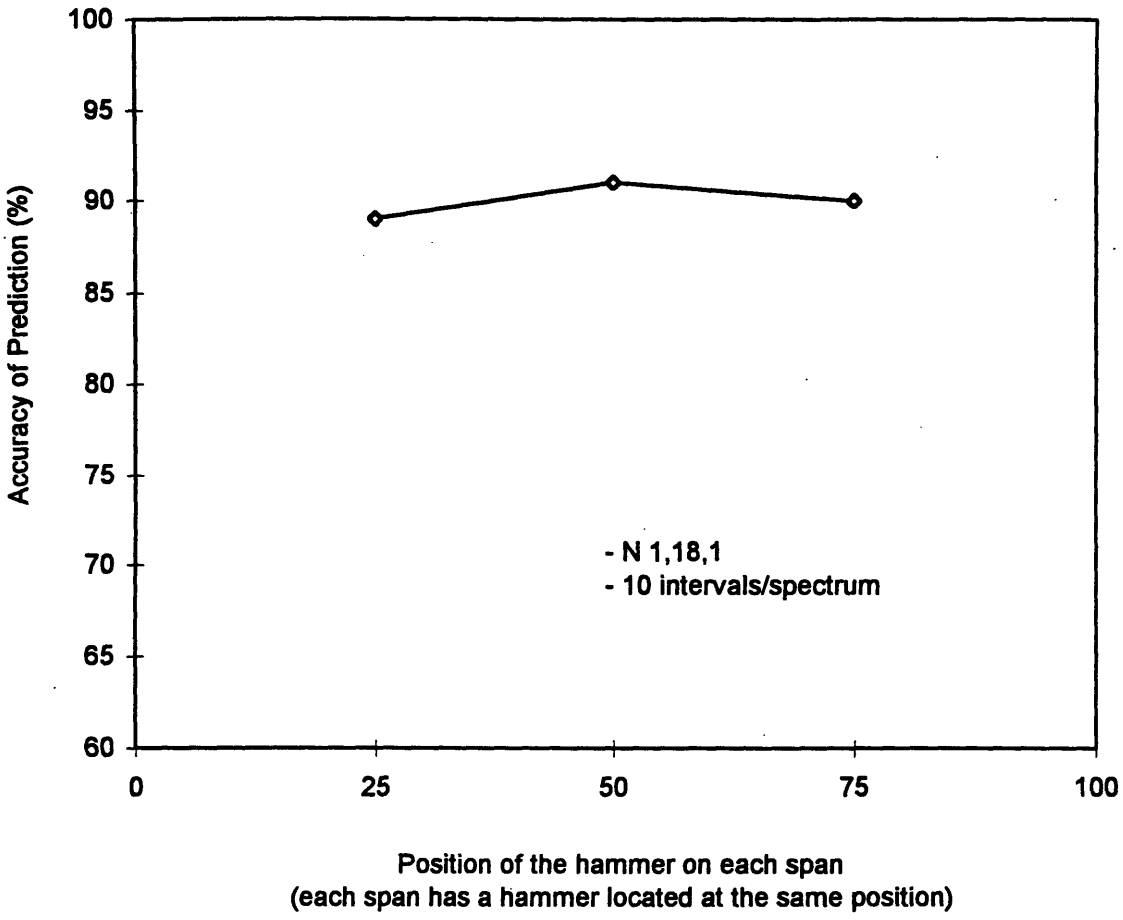


Figure 7.16: The effects of the location of hammers to the accuracy of NNET1.

Note that the architecture for NNET1 and data preprocessing strategy used in the previous investigation is also used here. The result shows that the prediction performance is essentially independent of the location of hammers.

The significance of the number of processing elements in the hidden layer of NNET1 to its accuracy is investigated by varying the number of elements from 4 to 40, with a constant increment of two elements. One sensor is located at the middle-right point of each span. Each spectrum is represented by 10 intervals, and the training data set with 500 damage cases is used. The result indicates that the absolute accuracy increases with the number of elements until the number reaches 18, and then tends to decrease beyond the level. Fig 7.17 compares the performance of three NNET1's (with 6, 18 and 36 processing elements respectively). The figure shows that NNET1 with 18 processing elements has the best absolute accuracy, and this configuration is considered as the optimum for this application.

The significance of the number of damage cases in the training data set is demonstrated in Fig 7.18. For this investigation, NNET1 has 18 processing elements, and one sensor is located at the middle-right point of each span. Ten intervals are used to represent each spectrum. Excitation is generated by four hammer impulses applied at the location of the sensors. Five NNET1's are trained with 5 different training data sets, which consists of 50, 100, 500, and 1000 different damage cases. Fig 7.18 shows that the absolute accuracy of NNET1 increases with the number of damage cases of the training data set. However, the accuracy does not significantly improve when the number is greater than 500, which is considered to be the optimum size of the training data set for this application.

The results of the performance studies indicate that a 1-hidden-layer network with 18 processing elements is a satisfactory choice of NNET1 for this application. At least 1 sensor per span should be employed, and each spectrum should be represented by 10

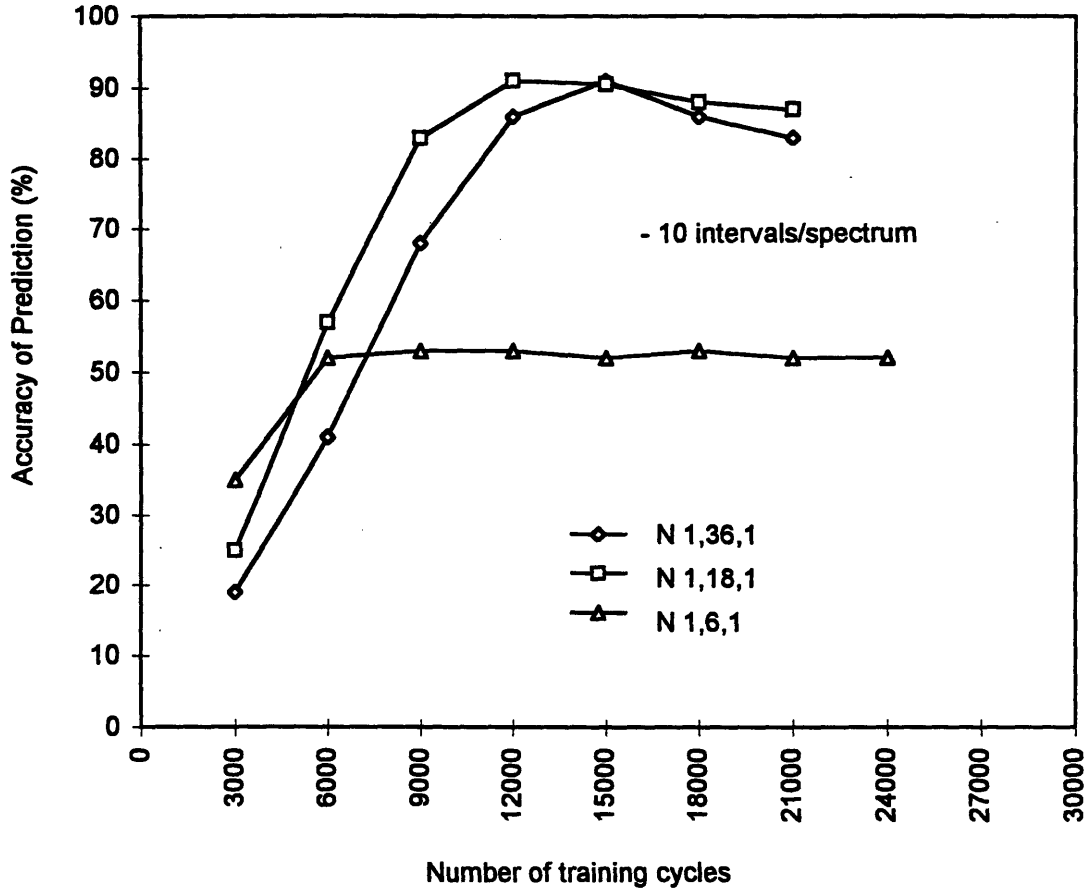


Figure 7.17: The significance of the no. of processing elements in the hidden layer to the accuracy of NNET1.

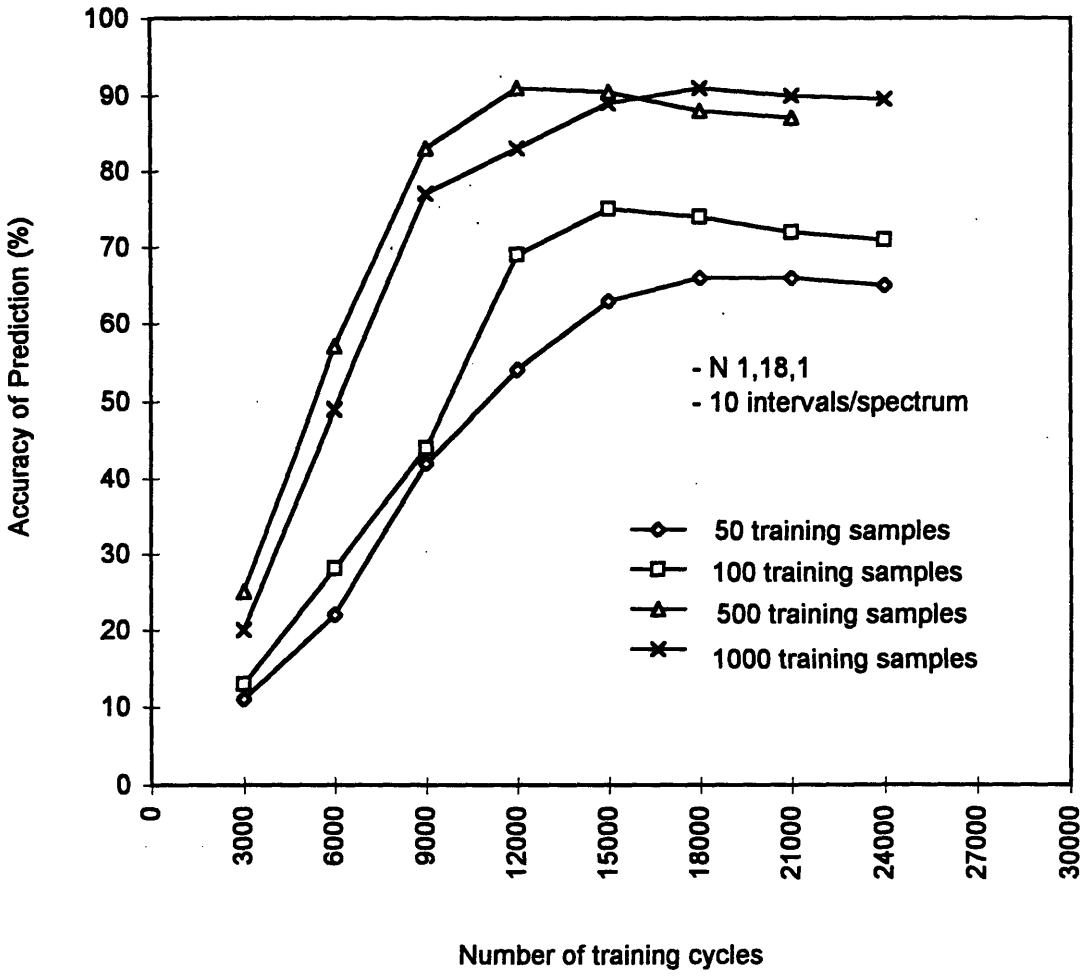


Figure 7.18: The significance of the no. of training samples to the accuracy of NNET1.

intervals. One hammer load per span is recommended as excitation, and the hammer location has very little significant. The size of the training data set has to be sufficiently large to include all the damage scenarios, which total 500 for this application. A properly trained NNET1 is able to predict the damaged spans with up to 90% absolute accuracy.

Four different testing data sets, each of which contains the damage cases that have a specified extent of damage, are employed to test the optimum NNET1. The sensitivity to damage of the optimum NNET1 is demonstrated in Table 7.4. The result demonstrates that the optimum NNET1 is able to detect the damaged spans with an accuracy as high as 89%, when the reduction of *EI* of any damaged beam element is as small as 5%.

Extent of Damage (% of *EI* reduction)	**Accuracy of Optimum NNET1 (% of correct diagnosis)**
10	91
5	89
4	64
3	47

Table 7.4: The damage sensitivity of the optimum NNET1.

7.4.4 Observation

Simulation data indicates that the change in response spectrum due to damage condition provide sufficient information for NNET1 to predict the damaged spans with an acceptable level of accuracy. A properly trained global diagnosis system can identify the damaged spans with absolute accuracy up to 90%.

7.5 Local Structural Diagnosis: Frequency Transfer Function Approach

Four separate neural network-based systems for local structure diagnosis, each of which is a basic neural network-based diagnosis system that is trained by a local training data set specifically created for each span, are employed to predict the damaged beam

elements in the individual spans. The frequency transfer functions (FTFs) of the response at various locations of a particular span, due to a prespecified local excitation, are employed as the local vibrational signature for the corresponding local diagnosis system.

7.5.1 Data Preprocessing Strategy

Figure 7.19 shows a middle span of a multispan beam. When the excitation is generated outside the span, it can be assumed that moments x_1 and x_2, which are the moments at the left and right support (assumed measurable), are the span's only inputs. If the vertical displacement at a specific location of the span (y) is the output of interest, a linear system can be created to represent the relation between the inputs and output, as shown in Fig 7.20 (Newland, 1994). Given the measured input and output data over a time period, the frequency transfer functions between the inputs and the output, $H_1(w)$ and $H_2(w)$, can be constructed and used to represent the condition of the linear system (Bregant et al, 1995 and Zimmerman et al, 1995). More details are demonstrated in Appendix B.

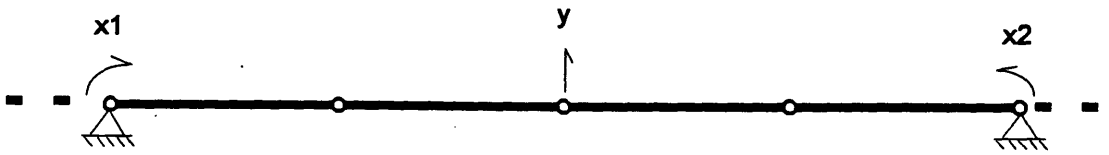


Figure 7.19: A middle span of a multispan beam.

For a linear system with two correlated inputs (x_1, x_2) and one output (y) shown in Figs 7.19 and 7.20, the frequency transfer functions are

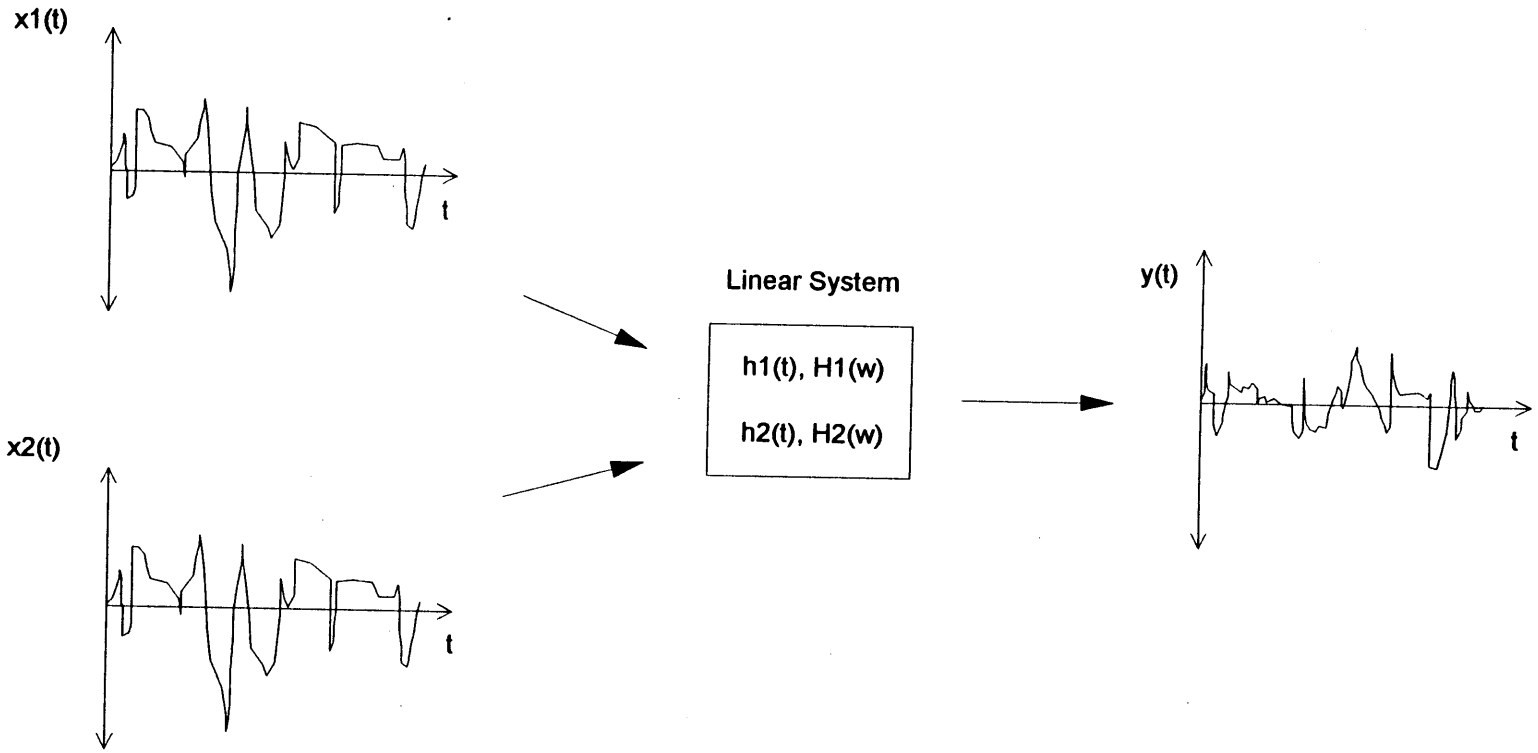


Figure 7.20: A linear system.

$$H_1(\omega) = \frac{S_{x_2} S_{x_1 y} - S_{x_2 y} S_{x_1 x_2}}{S_{x_1} S_{x_2} - S_{x_1 x_2} S_{x_2 x_1}} ,$$

$$H_2(\omega) = \frac{S_{x_1} S_{x_2 y} - S_{x_1 y} S_{x_2 x_1}}{S_{x_1} S_{x_2} - S_{x_1 x_2} S_{x_2 x_1}} . \qquad (7.8)$$

Equation 7.8 demonstrates that the frequency transfer functions of the response at a specific location of a middle span can be constructed from the spectrums and cross-spectrums of the inputs and output. These spectrums can be approximated by performing discrete fourier transform on the discrete time history data to create corresponding fourier spectrums, and then using these fourier spectrums to create the spectrums and cross-spectrums needed. The employed fourier spectrums are formerly smoothed in order to increase the statistic reliability of the FTFs created by this procedure. More detail on the smoothing process is described in Appendix B.

Since the properties of FTFs over a frequency range are desirable, prespecified white-noise impulse loads on nearby spans, as shown in Fig 7.21, are employed as the local excitation for a particular span. The white-noise impulses have the maximum amplitude of 10000 Newton, and are applied to the model throughout the period that the sensory data is recorded. By measuring the time history bending moments at both ends of the span as inputs, and recording the acceleration response at various locations of the span as outputs, the FTFs of the span (corresponding to the span's damage condition) can be constructed. The FTFs are then used to create the training and testing data set of the local diagnosis system of the specific span.

In this application, the time interval of all simulations is set at 0.01 second, so the FFT spectrums cover the frequency range of 50 Hz. The frequency transfer functions that are generated from the FFT spectrums cover the same frequency range. As shown in Figs 7.22 and 7.23, this frequency range well covers the first 3 modes of vibration of any

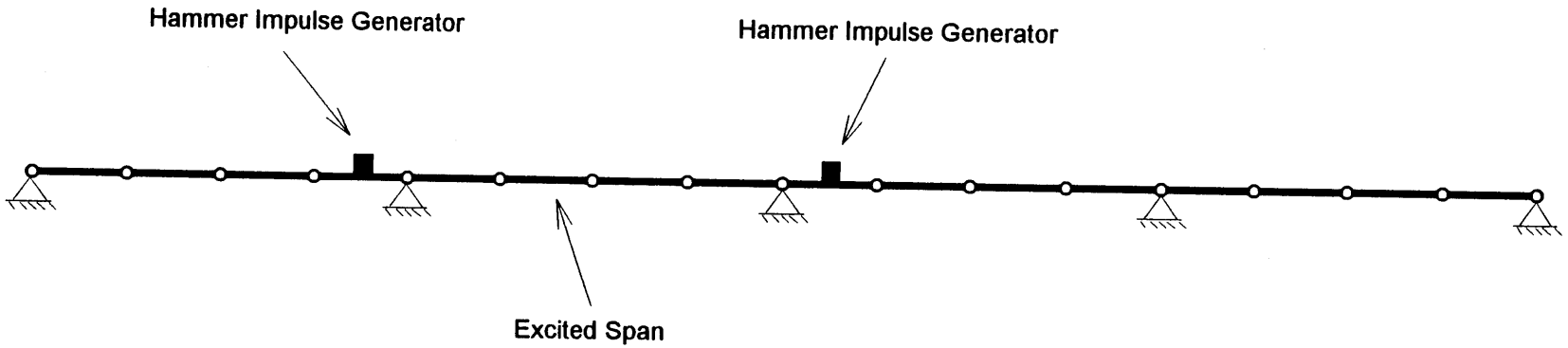


Figure 7.21: Location of impulse generators for middle span (2nd span).

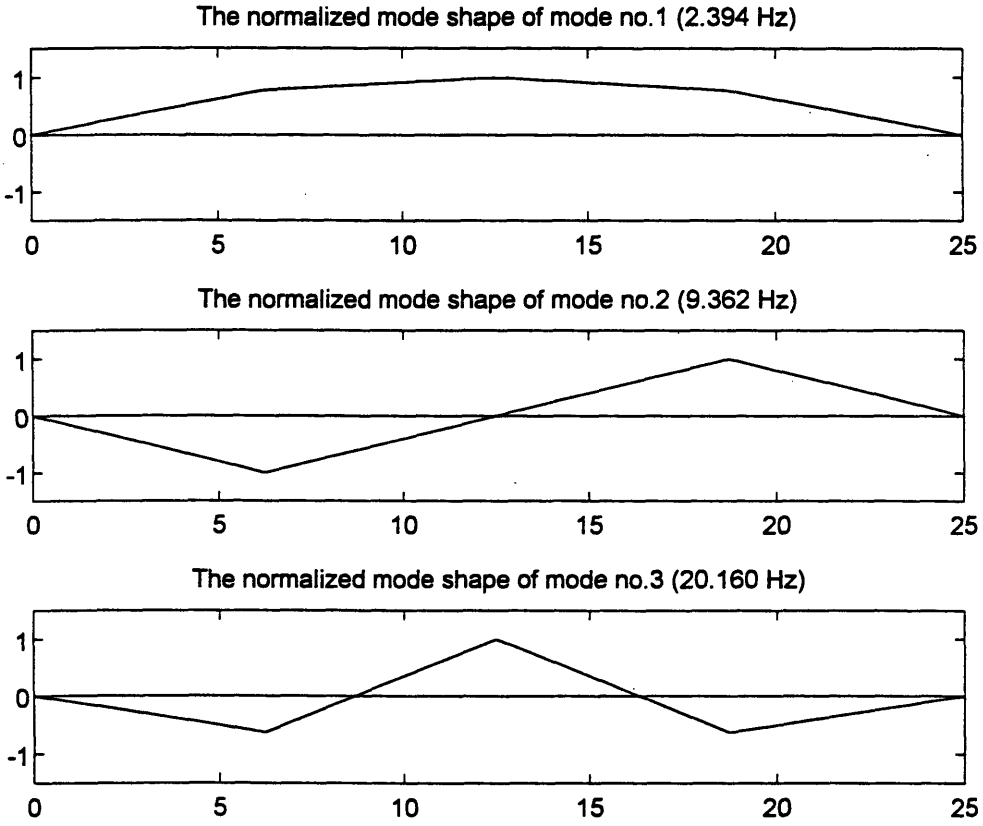


Figure 7.22: The first 3 modes of vibration of the 1st and 2nd span of the 4-span beam.

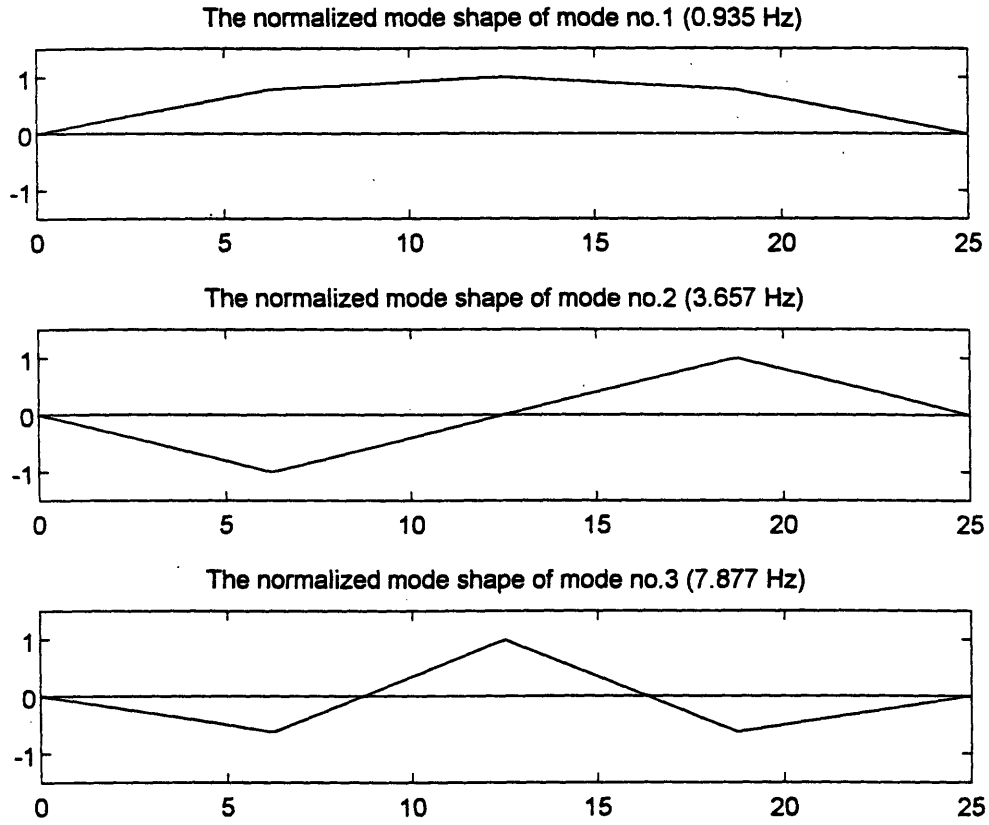


Figure 7.23: The first 3 modes of vibration of the 3rd and 4th span of the 4-span beam.

particular span, which is now considered as a simply supported beam with moments at both ends as its excitation (Fig 7.5). Fig 7.24 shows the 20-second acceleration response at the middle-right point of the 2nd span and their corresponding frequency transfer functions due to a 20-second excitation on the 1st and 3rd span. Note that there are two FTFs that correspond to the time history response of a particular location of the inner-spans, such as the 2nd and 3rd span, since they have two bending moments as inputs. However, there is only one FTF for that of the outer-spans since there is only one moment as their input.

Each frequency transfer function is then divided into a number of zones, *NUMZONE*. As shown in Fig 7.25, each zone covers the same frequency interval,

$$f\,\text{int} = \frac{MAXFREQ}{NUMZONE} \quad Hz, \tag{7.9}$$

where *MAXFREQ* is the maximum frequency concerned. The area under the frequency transfer function of each interval is then calculated. This provides a number of data points (*NUMZONE* points from each frequency transfer function) for each damage case as a vector

$$input = \begin{bmatrix} \text{data point no.1 of FTF no.1} \\ \vdots \\ \text{data point no. } NUMZONE \text{ of FTF no.1} \\ \vdots \\ \text{data point no.1 of FTF no. } NUMFTF \\ \vdots \\ \text{data point no. } NUMZONE \text{ of FTF no. } NUMFTF \end{bmatrix}, \tag{7.10}$$

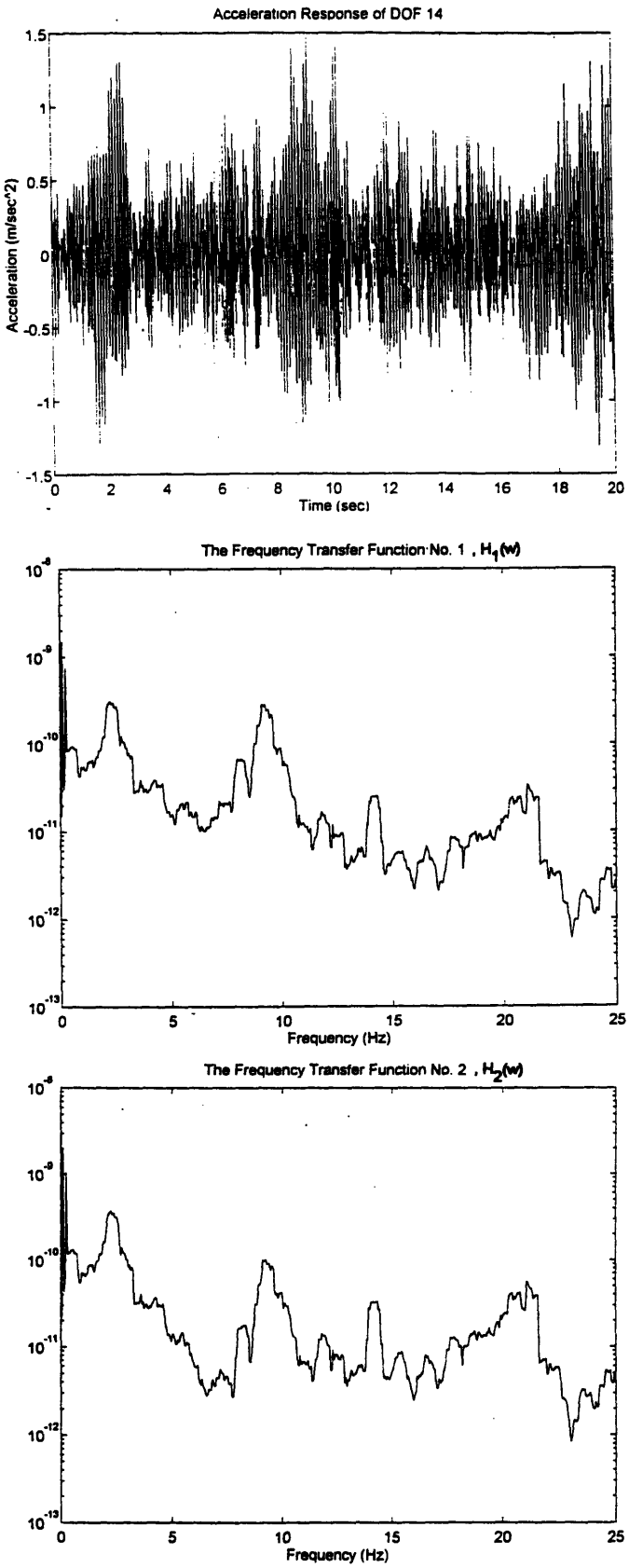


Figure 7.24: The acceleration response and frequency transfer functions of DOF 14.

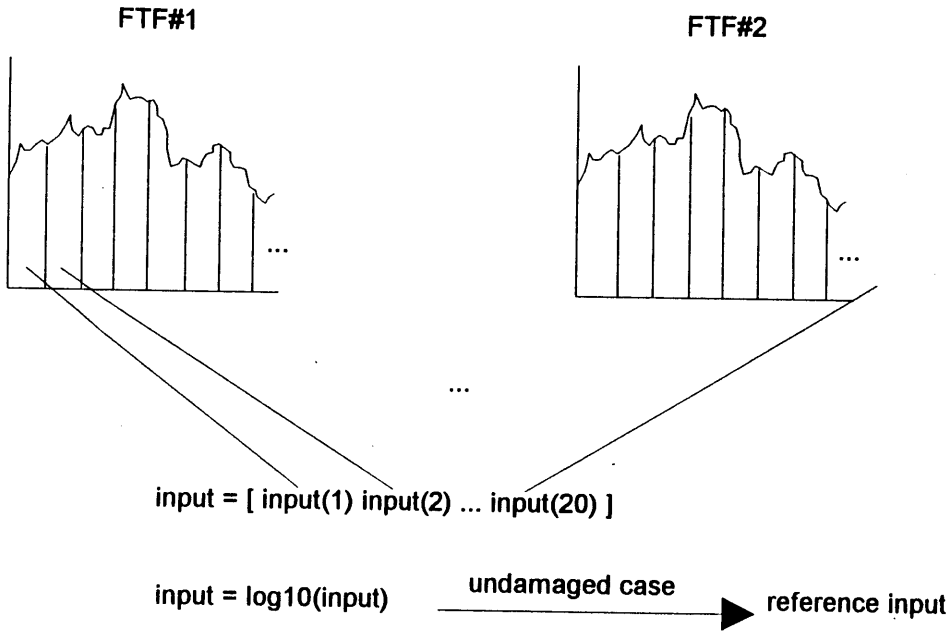


Figure 7.25: Data preprocessing approach.

where *NUMFTF* is the total number of frequency transfer functions used to create the input for the local diagnosis system.

This *input* vector is then further processed by the same procedure employed in Section 6.4.1. The logarithm of base ten of each vector is used instead of the vector itself in order to keep the data spread low. The *input* vector (after the $\log_{10}$ step) acquired from the undamaged model is then referred to as the "*reference data set*." The Reference Value Approach is then employed. The difference between the *input* vector for a particular damage condition and the reference data set is taken as the "*input pattern*" for the damage case. Each *input pattern* is then normalized by dividing by the absolute value of the maximum data point in all the *input patterns*. Each "*normalized input pattern*" and its expected "*localoutput*" vector, which is the corresponding damage condition represented by the binary vector

$$localoutput = \begin{bmatrix} localout_1 \\ localout_2 \\ localout_3 \\ localout_4 \end{bmatrix}_{4\times 1}, \qquad (7.11)$$

where $localout_i$ is 1 if there is reduction of *EI* of the *i*th beam element of the span of interest, and 0 otherwise. Note that the size of the *localoutput* vector is 4 by 1 since each span is represented by 4 beam elements. By performing simulations with various damage conditions on a specific span, the *normalized input patterns*, and their corresponding *localoutput* vectors, can be used as the training or testing samples for the NNET1 of the local diagnosis system of the span.

7.5.2 Configuration and Training of Neural Networks

The configuration of NNET1 of each local diagnosis system is the same as those described in Sections 7.3.2 and 7.4.2. NNET1 is a 1-hidden-layer network with back propagation training algorithm. The transfer function of the processing elements in the hidden layer is hyperbolic tangent function, while the sigmoidal function is used for the processing elements in the output layer.

The local system of each span requires its own training data set. Since four beam elements are used to represent a span, each span has 15 different basic combinations of damaged beam elements as shown in Table 7.2. In this study, four training data sets are created for the local diagnosis system of the 2nd span, and each contains 30, 100, 300, and 600 different damage cases respectively. The first four damage cases of each set are the damage-free cases. The remaining cases are the variations of the basic cases shown in Table 7.2. The extent of damage of any damaged element varying randomly between 5 to 90% reduction of EI. The testing data set with 50 different damage cases is similarly created. Note that all damage cases of the testing data set are different from those for the training data set.

The training procedure of the NNET1 of a local diagnosis system still follows the procedure described in Section 5.2 (see Fig 5.2). The testing data set is used to test the performance in predicting the damaged locations of the 2nd span. The threshold value is set at 0.8 for this application.

7.5.3 Performance Studies

Five design variables are investigated for their influence on the performance of local diagnosis systems; i) number of sensors, ii) locations of sensors, iii) number of intervals representing each FTF, iv) number of processing elements in the hidden layer of the NNET1, and v) number of damage cases of the training data set. In this investigation, only the performance of the local structural diagnosis system of the 2nd span is studied.

Fig 7.26 demonstrates the significance of the location of sensor to the accuracy of the NNET1 that predicts the location of damage in the 2nd span. NNET1 is trained by 3 different data sets, each of which is created from the data from a sensor located at a particular point of the span. Each training data set contains the same 300 different damage cases. Two hammer loads, located at the location of DOFs 7 and 17 respectively, apply the white-noise impulse loading throughout the data recording period. *MAXFREQ* is set at 25 Hz, which covers all three modes of vibration of the 2nd span. Fifteen intervals are used to represent each FTF. NNET1 has 35 processing elements in the hidden layer. Fig 7.26 shows that the location of sensor has almost no effect on the performance when one sensor is used. The result also shows that the absolute accuracy of NNET1, which is trained by the training data from one sensor, is not yet acceptable since it has around 60% maximum accuracy.

Fig 7.27 shows the performance of NNET1 in predicting the location of damages in the 2nd span when the data from different number of sensors is used to create its input. The loading and network parameters are the same as for the previous figure, and the training data set with 300 damage cases are employed. The result demonstrates that the absolute accuracy of NNET1 increases with the number of sensors. In this application, at least 3 sensors per span is recommended.

The significance of the number of intervals used to represent each FTF is shown in Fig 7.28. The same local excitation and training damage cases are still employed, and NNET1 still has 35 processing elements. Three sensors are located at the middle-left, middle, and middle-right point of the span respectively. The number of intervals representing each FTF is varied from 5 to 20 intervals, with an increment of five intervals. The result demonstrates that the absolute accuracy increases with the number of intervals used to represent each FTF up to 15 intervals, and drops slightly beyond this level. This is the indication that 15 intervals per FTF is the most appropriate for this application.

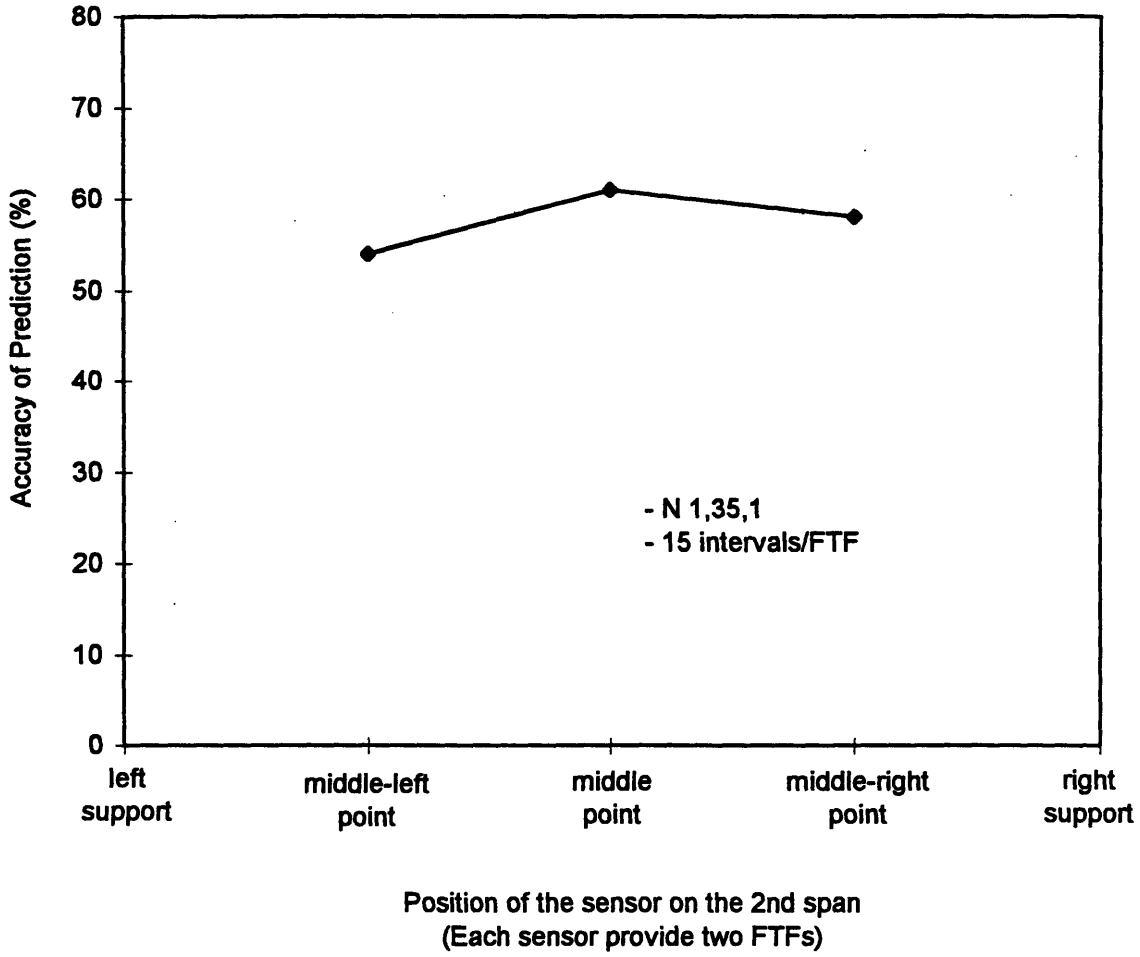


Figure 7.26: The effects of the location of sensors to the accuracy of NNET1.

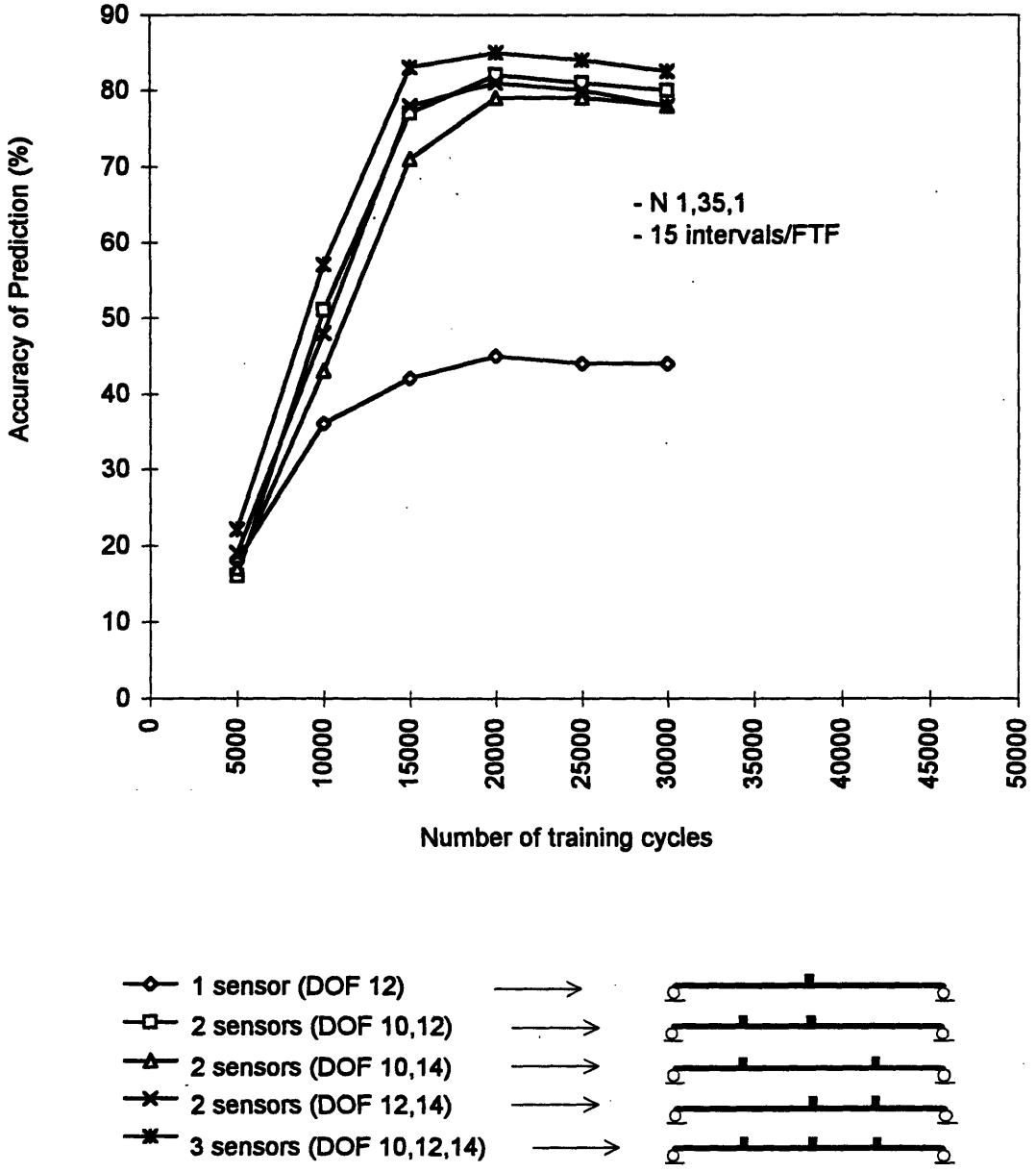


Figure 7.27: The effects of the no. of sensors to the accuracy of NNET1.

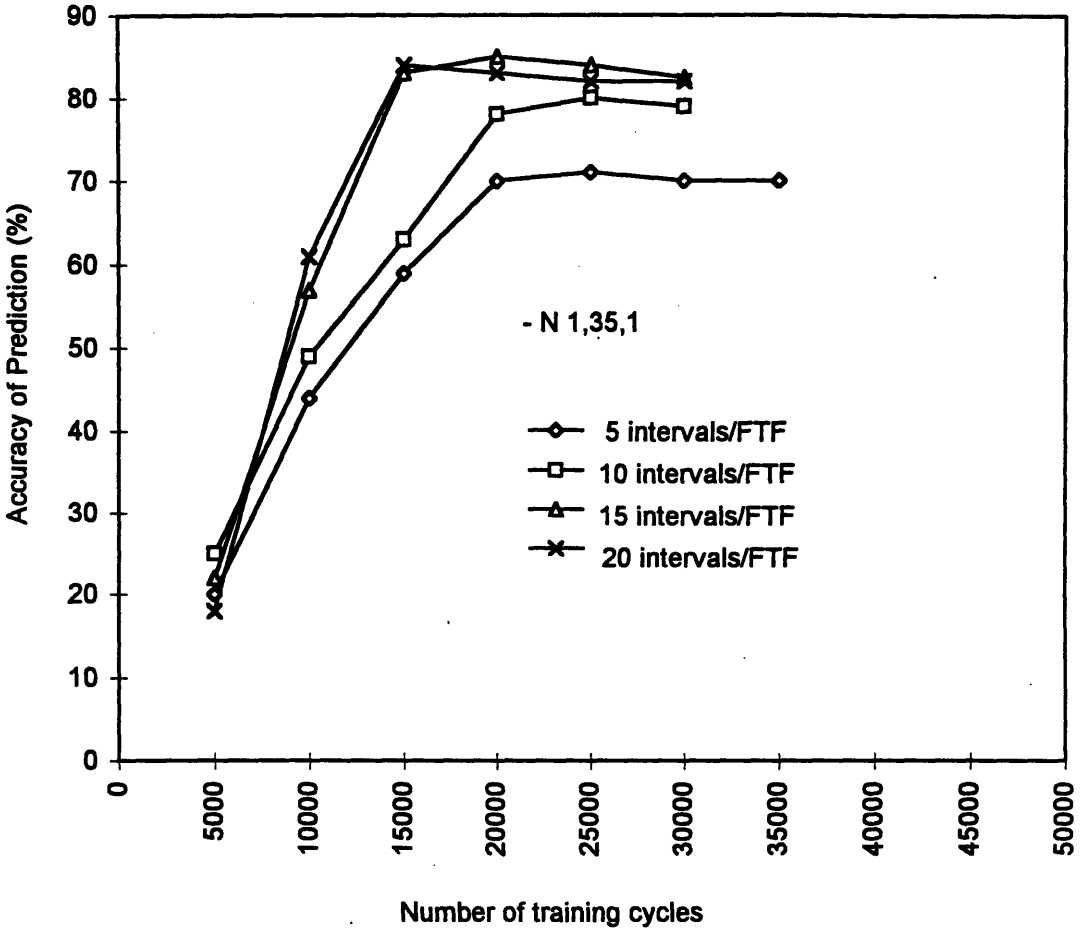


Figure 7.28: The significance of the no. of intervals to the accuracy of NNET1.

The significance of the number of processing elements in the hidden layer is investigated by varying the number of elements from 5 to 60 elements, with an increment of five elements. Fig 7.29 compares the performance of three NNET1's (with 20, 35, and 60 processing elements respectively) that are trained by 300 damage cases. Each FTF is represented by 15 intervals. The result indicates that the absolute accuracy increases with the number of elements until the number exceeds 35, and is constant beyond the level. Therefore, NNET1 with 35 processing elements should be an appropriate choice for this application.

The significance of the number of damage cases of the training data set is demonstrated in Fig 7.30. NNET1 has 35 processing elements, and is trained with the data that is created from the FTFs of three sensors respectively located at the middle-left, middle, and middle-right point of the span. Each FTF is represented by 15 intervals. The local excitation similar to the one in previous investigations is employed. Five NNET1's are respectively trained with 5 training data sets, which contain 30, 100, 300, and 600 damage cases respectively. The simulation results indicate that the absolute accuracy of NNET1 increases with the number of training samples. However, the accuracy does not significantly improve when the number exceeds 300, which is assumed to be the optimum size for this application.

From the results of performance studies, the optimum NNET1 is a 1-hidden-layer network with 35 processing elements. The input data should be created from the FTFs of 3 sensors, which are located at the middle-left, middle, and middle right point of the span respectively. Each FTF should be represented by 15 intervals, and the training data set should contain at least 300 different damaged cases. Properly trained NNET1 is able to predict the locations of damage on the 2nd span with the absolute accuracy up to 85%.

Different testing data sets, each of which contains the damage cases that have a specified extent of damage, are then employed to test the optimum local system. As demonstrated in Table 7.5, the result shows that the optimum NNET1 is able to detect any

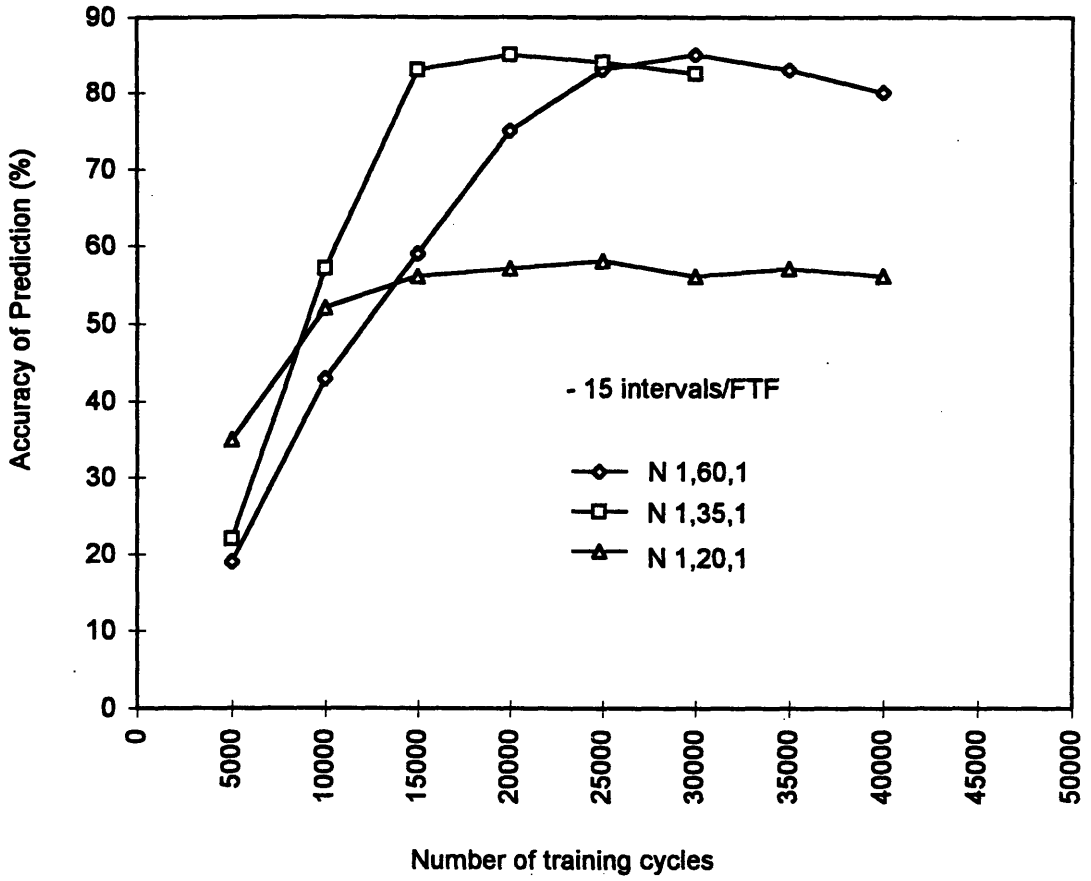


Figure 7.29: The significance of the no. of processing elements in the hidden layer to the accuracy of NNET1.

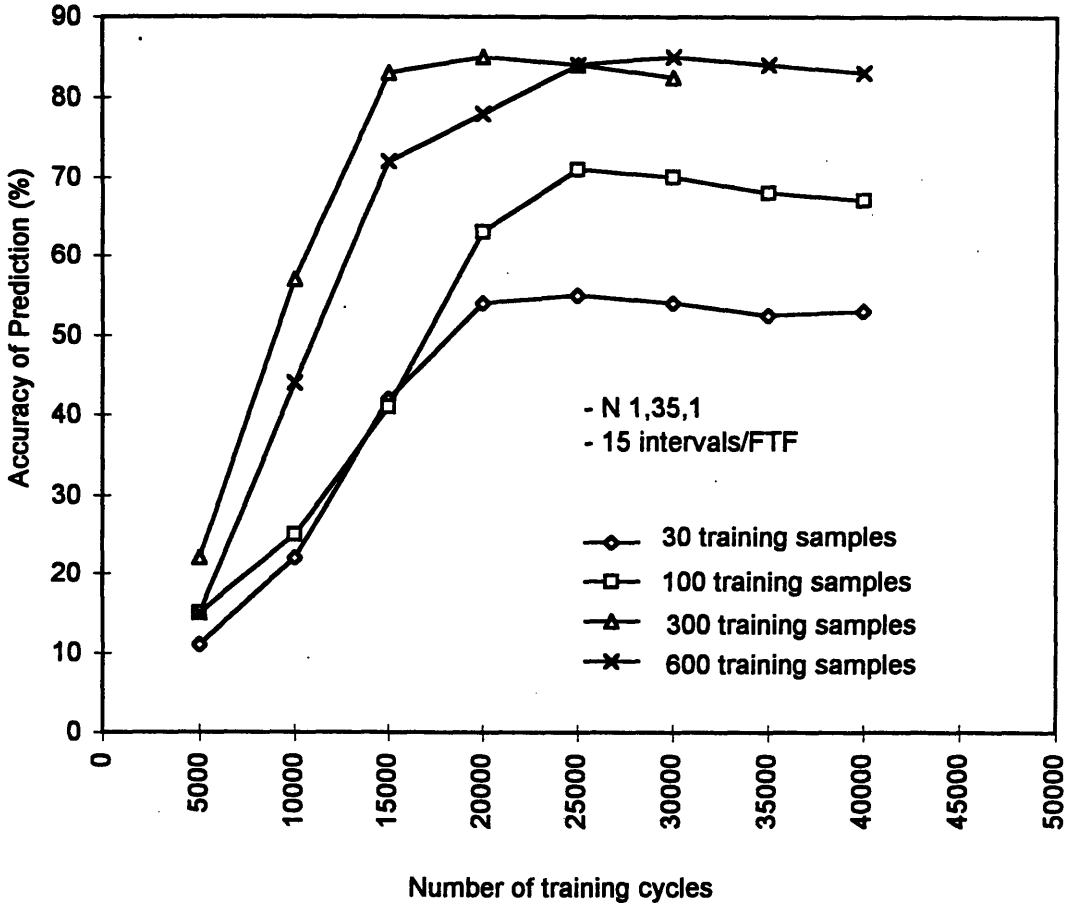


Figure 7.30: Significance of the no. of training samples to the accuracy of NNET1.

damaged beam element in the 2nd span, which has damage as small as 10% reduction of *EI*, with the accuracy up to 87%.

Extent of Damage (% of *EI* reduction)	Accuracy of Optimum NNET1 (% of correct diagnosis)
10	87
5	78
4	59
3	33

Table 7.5: The damage sensitivity of the optimum NNET1.

7.5.4 Observation

In order to find the appropriate configuration of the local structural diagnosis systems of the remaining spans, the same performance-based process earlier demonstrated for the 2nd span can be employed. The design approach for the NNET2 for each span is also similar.

The simulation data indicates that the changes of FTF's due to local damages provide sufficient information for the local system to predict the locations of local damage at an acceptable level of accuracy. For this case study, a properly configured NNET1 is able to predict the locations of local damage of the simulation model with the absolute accuracy as high as 85%, which is impressive.

7.6 Applicability to Other Structures

There are also other potential applications for the neural network system for multiple-point damage diagnosis. A diagnosis system for a frame structure, which is shown in Fig 6.23, with multiple-point damage can be set up by the approach earlier described in this chapter.

A mathematical model, or a scaled model, of the frame can then be created and employed to simulate the training and testing data for the neural network systems. A

global structural diagnosis system is required to predict the damaged substructures (see Fig 7.31). If there is earthquake, or wind gust, regularly in the area, they can be employed as the ambient excitation for generating the training and testing data of the global system. Otherwise, hammer impulse generators or mechanical floor vibrators can be used to generate prespecified excitation. If both ambient excitation & mode shape approach and prespecified excitation & response spectrum approach are practically feasible, the approach that performs better in the simulation will be used.

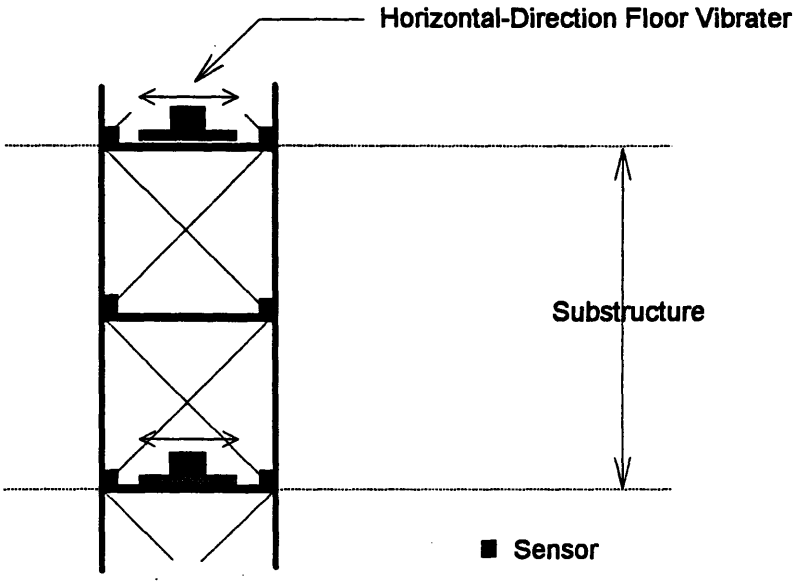


Figure 7.31: An example of a substructure of a frame and its local excitation.

The strategy for creating training and testing data also follows the procedure described in the previous sections. For the global structural diagnosis, the time history response at various locations of the frame due to a global excitation can be used to create the mode shapes or response spectrums that correspond to the damage conditions. The vibrational signature is then processed by the data preprocessing procedure, and then used

as the input for the NNET1 of the global diagnosis system. The performance-based design procedure can be performed to configure the optimum design variables.

As for local structural diagnosis, the whole frame can be divided into several substructures as shown in Fig 7.31. Each substructure will require a designated neural network-based diagnosis system that can predict the locations of damage, and optionally the extent of the damage, in the substructure. Designated local excitation for each substructure, such as a set of white-noise horizontal-direction floor vibrators on the floors next to the substructure (see Fig 7.31), is required for creating the training and testing data for the local structure diagnosis system. During the excitation period, the response of the substructure relative to its neighbors and all interaction forces between the substructures are measured and used to create the FTFs that represent the condition of the substructure. These FTFs are then processed into the input for training the local diagnosis system of the substructure (based on the process shown in Section 7.2.3). The performance-based design procedure can then be performed to configure the optimum design variables of each local diagnosis system.

7.7 Discussion and Summary

The simulation studies on an idealized 4-span beam indicate that a neural network-based system for multiple-point damage diagnosis has potential. The global diagnosis system based on either mode shape or response spectrum signature performs well. According to the simulation data, the performance of the global system based on the mode shape approach improves to the same level of accuracy as that of the spectrum approach when the changes in modal frequencies are also used. Therefore, the practical feasibility of the approaches on a specific application will decide which approach is more appropriate (Green, 1995). Local structural diagnosis systems based on the frequency transfer function approach also perform well in this restricted research domain.

However, the proposed multiple-pointed damage diagnosis system, which is based on the general architecture of neural network-based diagnosis system, still has the limitation on the number of different damage cases that the system can handle. This limitation depends on several factors including the types and amount of neural networks used, and the dimension of the input patterns for the neural networks. This limitation should be considered as a major design issue, especially when the monitored structure is very large or complex, or when the diagnosis result has to be highly specific. The simulation data also reveals that the performance of the neural network-based system for multiple-point damage diagnosis highly depends on the amount (or the diversity) of its training data. This relationship should be further investigated.

Including more information in the input, such as increasing the number of mode shapes used to create each *normalized input pattern,* usually improves the prediction performance up to a certain point. After this point, the performance does not significantly improve, or may even deteriorate. Therefore, redundant information should also be avoided in order to maintain the efficiency.

The global and local structure approach can be applied to a broad range of structural damage diagnosis problems. As demonstrated in the example of a frame structure, only some details need to be adjusted to customize the system for a specific application.

Chapter 8

Summary and Discussion

8.1 Introduction

In this chapter, the general architecture of a neural network-based diagnosis system, guidelines for configuring the system, and observations based on the simulation studies are summarized. The practical feasibility of neural network systems for structural damage diagnosis is discussed, and solutions to potential difficulties are recommended. Lastly, the contributions of this research are identified and topics for further research are recommended.

8.2 Summary

8.2.1 Neural Network-Based Diagnosis System

Two types of damage are considered in this research; single-point, and multiple-point. Single-point damage assumes that only one component of the system of interest is damaged at any specific time, while multiple-point damage allows many components to be damaged simultaneously.

In this research, a basic neural network-based diagnosis system is first proposed and its applicability for structural damage diagnosis is then evaluated. Since the basic system employs only two neural networks to classify all possible damages in the structure,

it can handle only a limited number of different damage cases. Therefore, its applicability is restricted to problems that involve a small number of damage states, ideally only single point damage. In order to overcome this problem, a general architecture of neural network-based diagnosis system is proposed. The approach is based on considering the whole structure to consist of a set of interacting substructures, and then using a neural network-based diagnosis system called "global structural diagnosis" to identify which substructures are damaged. Each individual damaged substructure is then independently examined to establish the locations and extent of damage by a "local structural diagnosis system." By transforming a structural damage diagnosis problem involving a large number of potential damage cases to a set of less complex problems that can be handled separately, the applicability of neural network-based diagnosis is enhanced.

Two implementation approaches for global structural diagnosis, and one approach for local structural diagnosis, are developed. The first global scheme employs ambient excitation as the excitation, and uses the mode shapes derived from this excitation as input for neural networks. The mode shape approach is suitable for neural network system that globally monitors structures that experience consistent significant ambient excitation. For example, high-rise structures with wind load, or bridges with normal traffic, can be considered in this category. The diagnosis systems designed by this approach do not require extra loading equipment, and could be operated in real-time basis if ambient excitation is available regularly. However, the mode shape approach (without incorporating corresponding modal frequencies in the input of the diagnosis system) does not perform well when there is multiple point damage, and it needs to be modified by including the changes in the modal frequencies, in addition to the changes in mode shapes, as input for the global diagnosis system.

The second global structural diagnosis approach employs prespecified excitation (i.e. a set of hammer load) as the excitation, and uses the response spectrums at various locations on the monitored structure as the input for neural networks. The response

spectrum approach is more appropriate when the ambient excitation is weak. However, the approach does require a prescribed loading, such as hammer load generators, where the mode shape approach requires arbitrary loading.

The only implementation approach for local structural diagnosis considered in this study uses localized white-noise hammer impulses to excite each substructure, and then employs the frequency transfer functions at various locations of the corresponding substructure as the input for the neural networks of the corresponding local diagnosis system.

Extensive data on the performance of a neural network-based diagnosis system applied to a multispan beam model with a range of damage states is obtained. This data, and the design expertise acquired, is then used to develop guidelines for the design and optimization of neural network-based diagnosis systems for a certain class of structures.

8.2.2 Performance-Based Design Methodology

A methodology is proposed as a framework for designing neural network-based damage diagnosis systems for engineering structures. The methodology is based on a consideration of the results of the simulation studies performed to assess the influence of the various design variables. The system configuration that provides optimum performance, or the "optimum diagnosis system," is then employed. This approach is actually a trial and error method, or even similar to the cross-validation approach (Wahba, 1980), used to configure neural networks. The entire procedure for designing a neural network-based damage diagnosis system for engineering structures is described in the following sections.

Simulation model

All types of damage that could occur to the structure of interest have to be defined. The models that can closely simulate both the behaviors of the physical structure

and the effects of concerned damage conditions are then chosen. It is possible to use several types of model to simulate several types of concerned damage in order to create the best possible training data.

Excitation and vibrational signature

The choice of excitation and vibrational signature is made separately for global and local diagnosis. For global diagnosis, the selection as to whether to use ambient excitation & mode shape approach, or prespecified excitation & response spectrum approach, is strongly influenced by the practical feasibility of the specific application. The mode shape approach is preferable since it requires no special loading equipment, and has real-time capability when there is adequate periodicity of ambient excitation. However, it is essential that the ambient excitation adequately excites all the modes of vibration of interest. The response spectrum approach is more appropriate when the ambient excitation is weak. For local damage, only the frequency transfer function approach is employed.

Neural network type

Any supervised-trained Artificial Neural Network (ANN), or system of supervised-trained neural networks, that is capable of performing function approximation can be employed in the diagnosis systems. If there is no a-priori knowledge of the nature of damage patterns, the multilayer feedforward network with back propagation training algorithm is recommended.

Data preprocessing strategy

The main objective of the data preprocessing unit is to transform the selected vibrational signatures into numerical patterns that can be used as input for the neural networks. Therefore the data preprocessing strategy depends largely on the type of neural network employed in the diagnosis system. Three major steps are involved in this task.

- Reduce, as much as possible, the maximum difference in magnitude of the elements of the input vector of neural networks. Using the logarithm of discretized spectrums as data is an example of this method.

- Employ the Reference Value Approach. This approach uses the difference between the data pattern of damage and non-damage case as the input for neural networks.

- Normalize all input vectors of neural networks, until the maximum value in the vectors is less than unity, in order to avoid the difficulty in training some types of networks (for example the problem of saturated parameters in multilayer networks with back propagation learning algorithm).

Training and testing data sets

The training and testing data sets should cover all concerned damage situations. However, the damage cases in the testing data set should be different from those of the training data set so that the generalization ability of the diagnosis systems can be evaluated.

Optimize the diagnosis systems

Before performing the performance analysis of either a global or local structural diagnosis system, all design variables that affect the performance of the diagnosis systems need to be identified. For example, the size, the architecture, and the type of neural networks employed are also considered as variables. Some researchers such as Green (1995) and French et al (1995) suggest the optimum excitation configuration for bridges and other beam-type structures. The diagnosis systems are set at their optimum

configuration, which is the configuration that provides the best performance based on the results of the performance studies. The cross-validation method should be used for training each neural network in order to maximize its generalization ability (Liu, 1995).

Calibration and testing

After the optimum configuration of the diagnosis systems is established, the systems are tested with data sets containing damage cases not included in the training data. This difference is achieved by varying the extent or location of damage. The performance measure of the diagnosis system is taken as the percentage of correct diagnose for the testing data.

If the test results indicate that the diagnosis systems have an acceptable prediction accuracy, the systems that are trained by simulation data should be further tested with the data from real physical structures, if available, in order to observe the performance of the diagnosis system under real operating condition. Note that the systems have to be calibrated for real structure by substituting the *reference data set* of each optimum diagnosis system (see Section 6.3.1), which is the base-line vibrational signature from the simulation of the undamaged simulation model (see Sections 7.3.1, 7.4.1 and 7.5.1), by another *reference data set* that corresponds to the undamaged structure.

It is also important to note that this design methodology only provides the optimum configuration, given specific training and testing data sets. The whole procedure of configuring the optimal configuration should be repeated whenever additional training and testing data is available. The influence of the training and testing data on the performance of the neural network-based diagnosis system is an area of ongoing research.

8.3 Observations Based on Simulation Studies

Simulation data indicates that the diagnosis system employing the mode shape approach predicts structural damage with essentially the same level of performance as that

of the diagnosis system based on the response spectrum approach. This observation suggests the two approaches are related, which is rational considering how the mode shapes and response spectrums are constructed in this research.

According to the mode shape compilation procedure described in Appendix A, the amplitudes of a particular peak of various response spectrums, which are created from the time-response data from the sensors located at various locations of the structure, are compiled to create the particular mode of vibration of the structure. The frequency corresponding to the particular peak is considered as the modal frequency.

Since the mode shapes are constructed from response spectrums, theoretically they contain the same information about the condition of the monitored structure. The only difference is the normalization method used by each approach. The mode shapes created from several simulations are normalized by their maximum amplitude, while the response spectrums are normalized by the use of a specified excitation for every simulation.

The data also reveals that the performance of the neural network-based diagnosis system largely depends on the quality of training and testing data. Increasing the information in the input data of neural networks, or the number of damage cases in the training data set, usually improves the prediction performance of the diagnosis system until an optimum point is reached.

8.4 The Feasibility of Neural Network-Based Diagnosis System with Simulation Training Approach

Since the data from the real physical structure is usually unavailable, the use of simulation data is necessary. In this section, the practical feasibility of the diagnosis systems is evaluated by examining the performance of the systems, which are trained by data generated with a simulation model, on the data generated with an alternate model.

The optimum diagnosis systems for a 4-span bending beam, presented in Chapter 7, are tested with data generated by an alternate simulation model. The *EI* of each beam element of the alternate model varies randomly between 0 to 25% from that of the original model. The alternate test data is created by performing simulations on the alternate model with the same damage cases used for the original test data (see Sections 7.3.2 and 7.5.2). The optimal diagnosis systems for the original model are then tested using the test data from the alternate model. The Reference Value Approach (see Section 6.3.1) is applied by substituting the *reference data set* of each optimum diagnosis system, which is the base-line vibrational signature from the simulation of the undamaged original model (see Sections 7.3.1, 7.4.1 and 7.5.1), by another *reference data set* that corresponds to the undamaged alternate model. The results of this investigation are listed in Table 8.1. The performance of the diagnosis systems with the original *reference data sets* is also demonstrated for comparison.

Optimized Diagnosis System	Accuracy (%)		
	Original testing data	Alternate testing data (new reference)	Alternate testing data (old reference)
Global diagnosis system: Mode shape approach	92	87	69
Global diagnosis system: Response spectrum approach	90	84	63
Local diagnosis system of the 2nd span	85	77	54

Table 8.1: The performance of optimized diagnosis systems.

The optimum global diagnosis systems with new *reference data sets* are able to predict the damage of the alternate model with less than 10% reduction in accuracy. The reduction increases to about 25% when the original *reference data set* is employed. The

performance of the optimum local diagnosis system of the 2nd span on the alternate testing data is also reasonable when a new *reference data set* is employed. In this case, the accuracy reduction is only 8%, compared to 31% when the original *reference data set* is used.

Next, the optimum diagnosis systems are tested by two testing data sets that are generated from another two alternate simulation models. Each model is an assemblage of 32 and 64 beam elements respectively (8 and 16 elements for each span). The formulation of the numerical models follows the method described in Section 6.2. The dimension, bending stiffness (EI) and mass per unit length of the models are the same as those of the original model. The damage cases of the alternate testing data sets are assigned to be the same as those of the original testing data by selecting the damaged beam elements that are located within the locations of damage of the corresponding original damage cases. The extent of damage at each location is also similar to that of the original cases. The diagnosis systems optimized for the original model are then tested by data generated with the alternate models. The Reference Value Approach (see Section 6.3.1) is applied similarly to the previous investigation. The results are demonstrated in Table 8.2.

Optimized Diagnosis System	Accuracy		
	Original testing data (%)	Alternate 1 (32 elements) (%)	Alternate 2 (64 elements) (%)
Global diagnosis system: Mode shape approach	92	89	83
Global diagnosis system: Response spectrum approach	90	87	79
Local diagnosis system of the 2nd span	85	83	78

Table 8.2: The performance of optimized diagnosis systems.

The results show that there is less than 5% reduction of the prediction accuracy when the alternate model consists of 32 beam elements. However, the accuracy reduction increase to about 7 to 11% when being tested with the data from the alternate model that has 64 beam elements. The accuracy reduction caused by the difference between the training and testing models also increases with the difference between the models.

The optimum diagnosis systems are then tested with another two data sets. One is generated from the alternate model that has the right span 5% longer than that of the original model, while the other has the right span 5% shorter. The number of beam elements representing the 4-span beam is still 16. The formulation of the numerical models follows the method demonstrated in Section 6.2 except that the length of each beam element of the right span is 5% longer or shorter. The diagnosis systems that are optimized for the original model are then tested with the test data from the alternate models. The Reference Value Approach is applied. The results are contained in Table 8.3. The optimum global diagnosis systems have only 2-3% of accuracy reduction due to the use of the alternate testing data, while the performance of the optimum local diagnosis system of the 2nd span reduces only 1%.

Optimized Diagnosis System	Accuracy		
	Original testing data (%)	Alternate 1 (5% longer) (%)	Alternate 2 (5% shorter) (%)
Global diagnosis system: Mode shape approach	92	90	89
Global diagnosis system: Response spectrum approach	90	87	87
Local diagnosis system of the 2nd span	85	84	84

Table 8.3: The performance of optimized diagnosis systems.

The optimum diagnosis systems are then tested with the data generated from the alternate model which is similar to the original model except that the damping ratio of the first two modes is changed from 1% to 5%. The diagnosis systems that are optimized for the original model are then tested by the testing data from the alternate model with Reference Value Approach applied. The results, which are demonstrated in Table 8.4, show minimal change in prediction accuracy due to the change of damping.

Optimized Diagnosis System	Accuracy		
	Original testing data (%)	Alternate testing data (%)	Change (%)
Global diagnosis system: Mode shape approach	92	92	0
Global diagnosis system: Response spectrum approach	90	89	-1
Local diagnosis system of the 2nd span	85	85	0

Table 8.4: The performance of optimized diagnosis systems.

Finally, The optimum diagnosis systems are tested with data generated from the alternate model whose flexural rigidity varies randomly between 0 to 25% of the original model. This model is an assemblage of 64 beam elements (16 elements for each span), and has the right span 5% longer than that of the original model. The damping ratio of the first two modes of the alternate model is 5%. The diagnosis systems that are optimized for the original model are then tested by the testing data from the alternate model with Reference Value Approach applied. The results, which are demonstrated in Table 8.5, show substantial degradation of prediction accuracy due to the change of model properties.

Optimized Diagnosis System	Accuracy		
	Original testing data (%)	Alternate testing data (%)	Change (%)
Global diagnosis system: Mode shape approach	92	73	-19
Global diagnosis system: Response spectrum approach	90	64	-26
Local diagnosis system of the 2nd span	85	67	-18

Table 8.5: The performance of optimized diagnosis systems.

These results are encouraging due to the fact that the diagnosis systems are trained with the data from one simulation model, and tested with the data from different models. The tests also provide a preliminary indicator of how a neural network-based diagnosis system with simulation training would perform in real world applications, where the training data needs to be generated from simulation models, and the testing data is obtained via sensors located on the real structure. The difference between the model and the real structure always degrades the damage detection ability of the diagnosis systems that are trained through simulation. However, the degradation can be offset by employing the Reference Value Approach, and is still acceptable if the difference is minimal.

8.5 Other Potential Problems and Suggested Solutions

Although this research indicates an impressive performance of the neural network-based diagnosis system on simplified problems, the performance on practical application is not yet examined. In this section, some of the issues involving the possible difficulties in real world application are identified and discussed. Recommended solutions are also presented.

Since the neural network-based diagnosis system are trained by simulation data, it is essential that the simulation models can simulate all possible damage conditions,

especially the multiple-type damage cases (see Fig 8.1). The difference between the behaviors of the models and real physical structures is also a crucial issue. Various models, or even combinations of models, should be employed to simulate the effects of a particular type of damage condition on the vibrational signatures of the structure. As demonstrated in Section 8.4, the reference value approach used in the proposed diagnosis systems can suppress the effects of the problem, but only to a certain level.

Another potential difficulty is the problem of noisy sensory data. Optimizing the performance of the diagnosis system by employing the proposed performance-based design methodology should handle the problem since properly trained neural networks have the generalization ability to deal with noisy data (see Chapter 3). However, the effect of noisy data on the performance of the diagnosis system in real world applications is also another topic that needs further investigation.

Occasionally a neural network-based diagnosis system has to classify types of damage that it has never been trained for, and these damage cases certainly would lead to incorrect diagnosis by the diagnosis system. A subsystem that can identified unseen damage cases, or unseen data patterns, should be developed and integrated into each diagnosis system (see Fig 8.2) in order to identified these types of damage. This subsystem may be an unsupervised learning neural network (see Section 2.3), a vector quantization unit, or any clustering unit that has the ability recognize unseen data pattern. Future research in this area is also recommended.

Problems related to variable environment, such as the change of stiffness or damping of structural materials due to seasonal temperature, could cause inconsistent performance by the diagnosis system. Employing several *reference data sets* for several testing conditions can lessen this problem. The problem of deteriorating or aging structures can also be avoided by regularly updating the *reference data set* to cope with the change of structures. However, it is essential that the updated *reference input pattern* is created from the responses of structures with no damage. In practice, the pattern can be

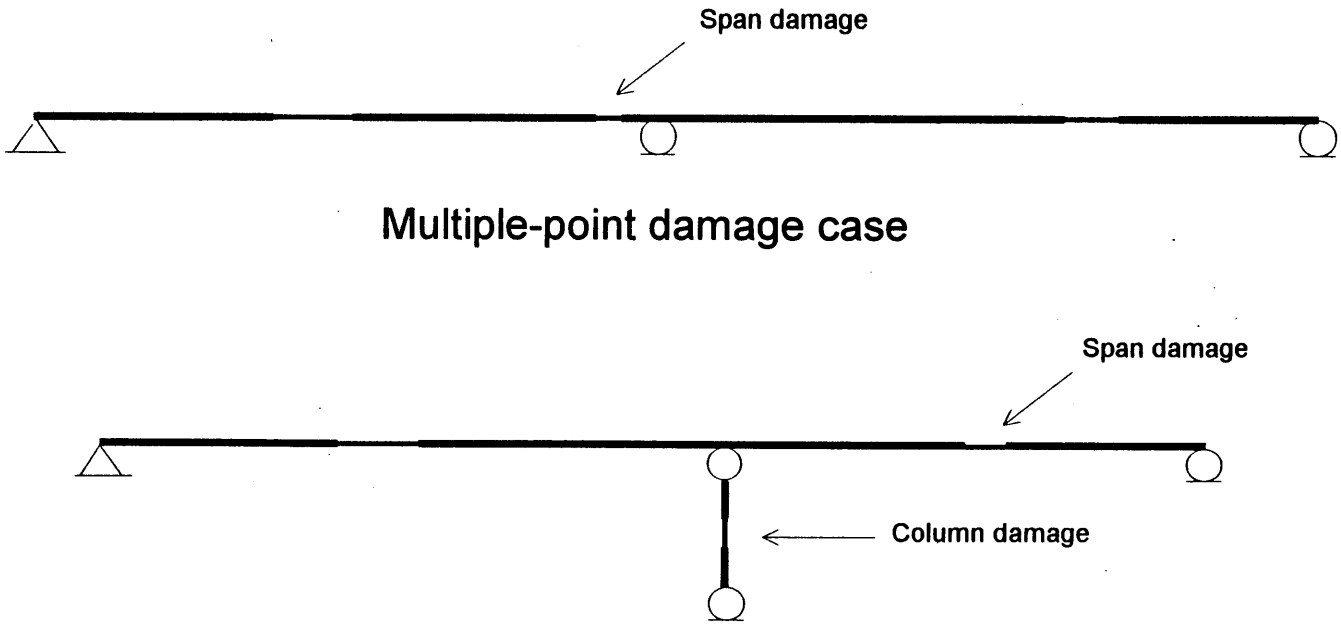


Figure 8.1: Examples of multiple-point damage case & multiple-type damage case.

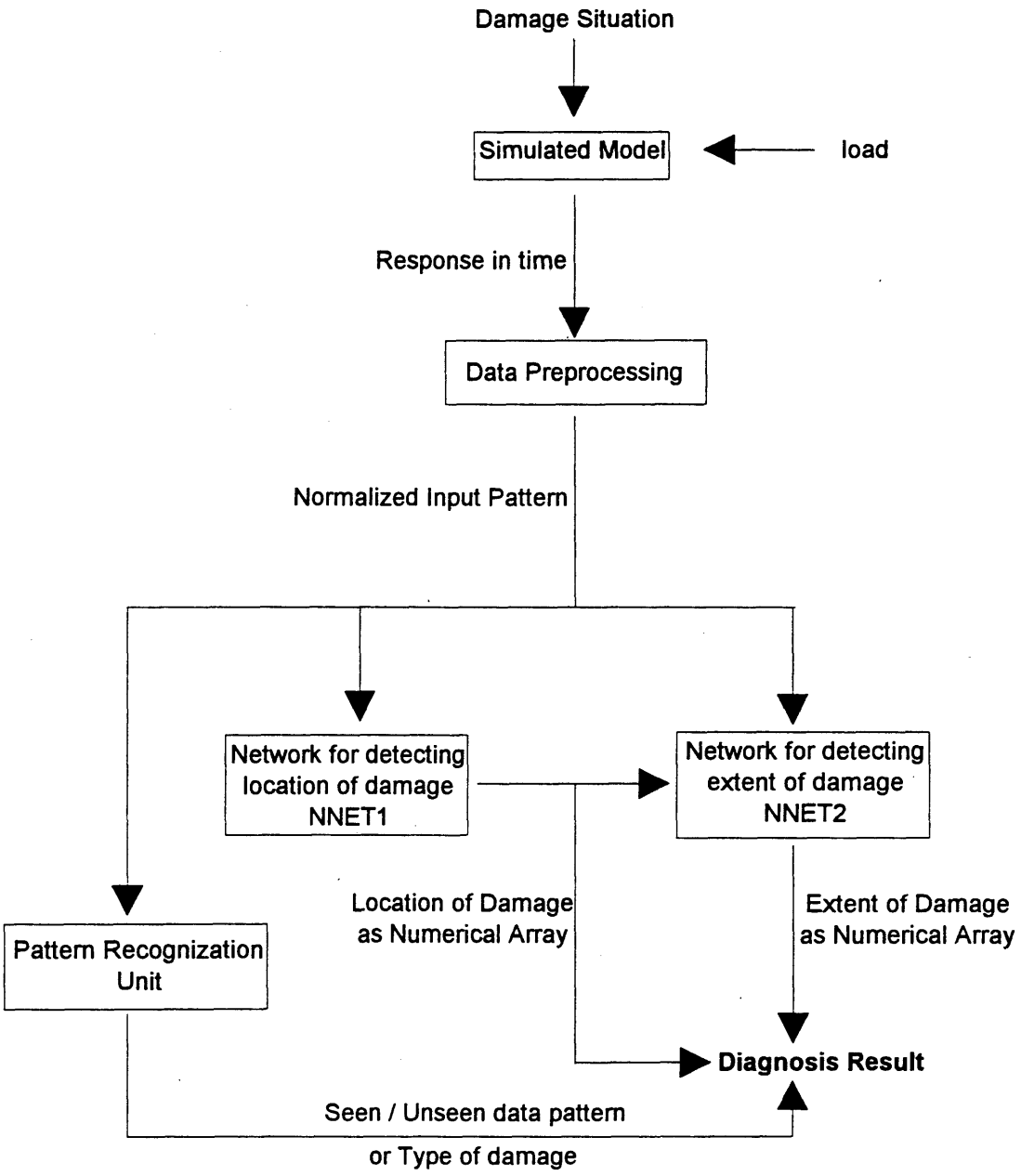


Figure 8.2: Recommended approach for detecting unseen damage case.

updated whenever a human inspection, or other types of major inspection, indicates that the structure has no damage.

8.6 Conclusion

Combining global and local diagnosis, as demonstrated in Chapter 5, considerably improves the practical feasibility of multiple-point damage diagnosis in comparison to other existing neural network-based approaches. The combined approach also has more potential for large-scale problems since it can be easily scaled up. Preliminary results obtained with neural network-based diagnosis systems for multispan beams (Chapter 6, 7) demonstrate potential. These systems can also be employed as a basis for designing diagnosis systems for other types of engineering structures.

Although the preliminary feasibility study shows promising results, the performance of the neural network-based diagnosis system on real world applications has not yet been examined. Difficulties in developing good simulation models of large-scaled civil engineering structures, and the extensive amount of possible damage states that the structures involve, are the major practical problems of this damage diagnosis approach. This problem may not be overcome in the near future, and it may result in limited applicability of this approach in the field of civil structures. However, this neural network-based damage diagnosis approach can also be applied to smaller-scale problems in other engineering fields such as mechanical engineering where the training data from real structures is available, or when good simulation models of the structures are easier to be developed. The approach is also a strong candidate when remote sensing is required. For example, the damage diagnosis of small-scaled space structures, under-water structures, or structures in hazardous environment would benefit from this approach.

It is also very essential to note that, although there are recent successes in developing automated damage diagnosis approaches, the importance of human inspection should not be diminished. There are always unexpected damages that is able to interfere

operation of the automated system. There also may be deficiencies in the diagnosis systems itself. The performance of these "intelligent" systems is only as good as the knowledge employed in developing the systems. Therefore, human inspection is always the first option, and automated damage diagnosis systems should be considered as a supplement to human inspection.

8.7 Recommended Research Topics

Even though the results from this research are promising, additional research and development is needed before the approach can be proposed for application to real structures. The research issues of significant importance are:

- the application of other data preprocessing strategies and supervised-trained neural networks.

- the problem of modeling real physical structures.

- the application to structures with multiple-type damages.

- the problem of noisy data.

- the methodology for detecting unanticipated damage cases.

- the feasibility of neural network-based approach on real large-scaled applications.

- the significance of the training and testing data to the performance and training efficiency.

References

Ackley, D. H., Hinton G. E., and Sejnowski T. J. 'A learning algorithm for Boltzmann machines.' *Cognitive Science*, vol.9, pp. 147-160,1985.

Albus, J. 'A theory of cerebellar function.' *Mathematical Bioscience*, vol.10, pp.25-61, 1971.

Arbib, M. A. *Brains, Machines and Mathematics.* Springer-Verlag, 1987.

Baldi, P. 'Gradient Descent Learning Algorithm Overview: A general dynamical systems perspective.' *IEEE Transactions on Neural Networks*, vol.6, no.1, pp.182-195, 1995.

Ballinger, R. S. and Herrin, D. W. 'Design health monitoring of mechanical dynamic structures.' *Proceedings of the 13th International Modal Analysis Conference* (Nashville, TN), vol.2, pp.1861-1868, 1995.

Barai, S. V., and Pandey, P. C. 'Performance of the generalized delta rule in structural damage detection.' *Engineering Applications of Artificial Intelligence*, vol.8, no.2, pp. 211-221, 1995.

Barai, S. V., and Pandey, P. C. 'Vibration signature analysis using artificial neural networks.' *Journal of Computing in Civil Engineering*, vol.9, no.4, pp.259-265, 1995.

Batur, C. and Srinivasan, A. 'Hopfield/ART-1 neural networks based fault detection and isolation.' *Proceeding of the 1992 American Control Conference*, pp.540-544, 1992.

Bertero, M., Poggio, T. and Torre, V. 'Ill-posed problems in early vision,' *Proceedings of the IEEE*, vol.76, pp.869-889, 1988.

Bianchini, M. and Frasconi, P. 'Learning without local minima in radial basis function networks.' *IEEE Transactions on Neural Networks*, vol.6, no.3, pp.749-755, 1995.

Billings, S. and Chen, S. 'Neural networks and system identification.' *Neural networks for control and systems*, 1992, Short Run Press Ltd., England.

Bregant, L., Otte, D. and Sas, P. 'FRF Substructure Synthesis: Evaluation and validation of data reduction methods.' *Proceedings of the 13th International Modal Analysis Conference* (Nashville, TN), vol.2, pp.1592-1597, 1995.

Broomhead, D. S. and Lowe, D. ' Multivariable functional interpolation and adaptive networks,' *Complex Systems*, vol.2, pp. 321-355, 1988.

Bullock, D., Garrett, J. H. and Hendrickson, C. T. 'A Prototype Neural Network for Vehicle Detection.' In *Proceedings of Artificial Neural Networks in Engineering*, INNS, November 1991.

Carpenter, G. A. and Grossberg, S. 'ART 2: self-organization of stable category recognition codes for analog input patterns.' *Applied Optics*, vol.26, pp. 4919-4930, 1987.

Carpenter, G. A. and Grossberg, S. 'The ART of Adaptive Pattern Recognition by a Self-Organizing Neural Network.' *Computer*, March 1988, pp. 77-88.

Carpenter, G. A. and Grossberg, S. 'A massively parallel architecture for a self-organizing neural pattern recognition machine.' *Computer Vision, Graphics, and Image Processing*, vol.37, pp. 54-115, 1987.

Carpenter, W. C. and Barthelemy, J. F. 'Common misconceptions about neural networks as approximators.' *Journal of Computing in Civil Engineering*, vol.8, no.3, pp.345-358, 1994.

Carrol, S. M. and Dickinson, B. W. 'Construction of neural nets using the Radon transform.' In *Proceedings of IJCNN*, 1989.

Cater, J. P. 'Successfully Using Peak Learning Rates of 10 (and Greater) in Back-Propagation Networks with the Heuristic Learning Algorithm.' In *IEEE First International Conference on Neural Networks* (San Diego, 1987), eds. M. Caudill and C. Butler, vol.2, pp. 645-651.

Chen, S., Billings, S. A. and Grant, P.M. 'Practical identification of NARMAX models using radial basis functions,' *Internatinal Journal of Control*, vol. 52, pp. 1327-1350, 1990.

Chen, S., Billings, S. A. and Grant, P. M. 'Nonlinear System Identification Using Neural Networks.' *Internatinal Journal of Control*, vol.51, No. 6, pp.1191-1214, 1990.

Chu, S. R. and Shoureshi, R. 'A Hopfield-based neuro-diagnostic system.' *Proceeding of the 1992 American Control Conference*, pp.2629-2633, 1992.

Conte, J. P., Shelton, R. O. and Durrani, A. J. 'System identification of seismically excited structures using artificial neural networks.' *Proceedings of ANNIE*, 1992.

Cybenko, G. 'Continuous valued neural networks with two hidden layers are sufficient.' Technical Report, Department of Computer Science, Tufts University, 1988.

Cybenko, G. 'Approximation by superpositions of a sigmoidal function.' *Mathematical Control Signal Systems*, vol. 2, pp. 303-314, 1989.

DARPA Neural Network Study, Lexington, MA: MIT Lincoln Laboratory, 1988.

Dyn, N. 'Interpolation and approximation by radial and related functions.' In C. K. Chui, L.L. Shumaker, and D.J. Ward, editors, *Approximation Theory*, VI, pp.211-234, Academic Press, New York,1991.

Elkordy, M. F., Chang, K. C. and Lee, G. C. 'Neural network classifier in vibrational signature analysis.' *Proceedings of the 8th Annual Conference of Computing in Civil Engineering.*, ASCE, June, pp.1155-1162, 1992.

Elkordy, M. F., Chang, K. C. and Lee, G. C. 'Neural networks trained by analytically simulated damage state.' *Journal of Computing in Civil Engineering*, vol. 7, no. 2, April, 1993.

Flood, I. and Kartam, N. 'Neural networks in civil engineering II: systems and application.' *Journal of Computing in Civil Engineering*, vol. 8, no. 2, pp.149-162, 1994.

Franzini, M. A. 'Speech Recognition with Back Propagation.' In *Proceedings of the 9th Annual Conference of the IEEE Engineering in Medicine and Biology Society* (Boston, MA), pp. 1702-1703, 1987.

Freeman, J. A. S. and Saad, D. 'Learning and generalization in radial basis function networks.' *Neural Computation*, vol.7, pp.1000-1020, 1995.

French, M. and Gordon, B. 'Short time period excitation of structures.' *Proceedings of the 13th International Modal Analysis Conference* (Nashville, TN), vol.1, pp.1-7, 1995.

Funahashi, K. I. 'On the approximate realization of continuous mappings by neural networks.' *Neural Networks*, vol. 2, pp. 183-192, 1989.

Garrett, J. H. ' Neural networks and their applicability within civil engineering.' In *Proceedings of the 8th Conference on Computing in Civil Engineering*, pp. 1155-1162, June 1992.

Ghaboussi, J., Garrett, J. H. and Wu, X. 'Knowlege Based Material Behaviour with Neural Network.' *Journal of Engineering Mechanics*, vol.1, pp. 132-153, 1991.

Girosi, F. 'Regularization theory, radial basis functions and networks.' *From Statistics to Neural Networks*, (V. Cherkassky, J. H. Friedman, H. Wechsler, eds.), Springer-Verlag, Subseries F, Computer and Systems Sciences, 1993.

Girosi, F., Jones, M. and Poggio, T. Priors. 'Stabilizers and basis functions: From regularization to radial, tensor and additive splines.' A.I. Memo No. 1430, Artificial Intelligence Laboratory, MIT, 1993.

Girosi, F., Jones, M. and Poggio, T. 'Regularization theory and neural networks architectures.' *Neural Computation*, vol.7, no.2, pp.219-269, 1995.

Green, M. F. 'Modal test methods for bridges: A review.' *Proceedings of the 13th International Modal Analysis Conference* (Nashville, TN), vol.1, pp.552-558, 1995.

Grossberg, S. 'Competitive Learning: From Interactive Activation to Adaptive Resonance.' *Cognitive Science*, vol.11, pp. 23-63, 1987.

Gu, S. N. , Jiang, J. S. and He, C. A. 'The transfer function of linear vibrating system with proportional and non-proportional damping.' Technical notes, No. SHJ 8449 of Northwestern Polytechnical University, 1984.

Hampson, S. E. and Volper, D. J. 'Disjunctive models of Boolean category learning.' *Biological Cybernetics*, vol.56, pp. 121-137, 1987.

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

Hertz, J. A., Krogh, A. and Palmer, R. *Introduction to the theory of neural computation.* Addison-Wesley, Redwood City, CA, 1991.

Hinton, G. E. and Sejnowski, T. J. 'Learning and relearning in Boltzmann machines.' In *Parallel Distributed Processing*, (D. E. Rumelhart and J. L. McClelland, eds.), ch.7, Cambridge, MA:MIT Press, 1986.

Hopfield, J. J. ' Neural networks and physical system with emergent collective computational abilities.' In *Proceedings of the National Academy of Sciences*, USA, vol.79, April 1982.

Hornik, K., Stinchcombe, M. and White, H. 'Multilayer feedforward networks are universal approximators.' *Neural Networks*, vol. 2, PP. 359-366, 1989.

Huang, W. Y. and Lippmann, R. P. 'Neural net and traditional classifiers.' In *Conference on Neural Information Processing Systems - Natural and Synthetic*, IEEE, Nov. 1987.

Huckelbridge, A. A. and Lawrence, C. 'Identification of structural interface characteristics using component mode synthesis.' *The 1987 ASME Design Technology Conferences - 11th Biennial Conference on Mechanical Vibration and Noise*, Boston, MA, vol. 3, 1987.

Humar, J. L. *Dynamics of Structures.* Englewood Cliffs, New Jersey: Prentice Hall, 1990.

Humar, J. L. and Kashif, A. M. 'Dynamic response of bridges under traffic loads.' *Proceedings of the 5th International Conference in Computing in Civil Engineering*, vol.1, pp.477-484, 1993.

Hurty, W. C. 'Dynamic analysis of structural systems by component mode synthesis.' JPL-TR-32-350, NASA CR-53057, 1964.

Jacobs, R. A. ' Increased Rates of Convergence Through Learning Rate Adaptation.' *Neural Networks*, vol.1, pp. 295-307, 1988.

Karunanithi, N., Grenney, W. J., Whitley, D. and Bovee, K. 'Neural networks for river flow prediction.' *Journal of Computing in Civil Engineering*, Vol. 8, No. 2, pp.201-220, 1994.

Kawato, M., Furukawa, K. and Suzuki, R. 'A hierarchical model for control and learning of voluntary movement.' *Biological Cybernatics*, vol.57, pp. 169-185, 1987.

Kirkpatrick, S., Gelatt Jr., C. D. and Vecchi, M. P. 'Optimization by Simulated Annealing.' *Science*, vol.220, pp. 671-680, 1983.

Kohonen, T. ' Self-Organized Formation of Topologically Correct Feature Maps.' *Biological Cybernetics*, vol.43, pp. 59-69, 1982.

Krishnaiah, P. R. and Kanal, L. N. 'Classification, Pattern Recognition, and Reduction of Dimensionality.' *Handbook of Statistics*, vol.2. Amsterdam: North Holland.

Lippmann, R. P. and Martin, E. A. 'Multi-style training for robusted isolated-word speech recognition.' In *ICASSP 87*, pp. 705-708,April 1987.

Ling, C. X. 'Overfitting and generalization in learning discrete patterns.' *Neuro Computing*, vol.8, no.3, pp.341-347, 1995.

Liu, P. L. 'Identification and damage detection of trusses using modal data.' *Journal of Structural Engineering*, vol.121, no.4, pp.599-608, 1995.

Liu, Y. 'Unbiased estimate of generalization error and model selection in neural network.' *Neural Networks*, vol.8, no.2, pp.215-219, 1995.

MacIntyre, J., Tait, J., Kendal, S., Smith, P., Harris, T. and Brason, A. 'Neural networks application in condition monitoring.' in *Applications of Artificial Intelligence in Engineering, Proceedings of the 9th International Conference*, PA, USA, pp.37-48, July 1994.

Madych W. R. and Nelson, S. A. 'Multivariate interpolation and conditionally positive definite functions.' *Mathematics of Computation*, 54(189):211-230, January 1990.

von der Malsburg, C. 'Self-Organization of Orientation Sensitive Cells in the Striate Cortex.' *Kybernetik*, vol.14, pp. 85-100, 1973.

von der Malsberg, C. 'Frank Rosenblatt: Principle of Neurodynamics: Perceptron and the theory of brain mechanisms.' In *Brain Theory*, (G. Palm and A. Aertzen, eds.), Berlin: Springer Verlag, 1986.

Marroquin, J. L., Mitter, S. and Poggio, T. 'Probabilistic solution of ill-posed problems in computational vision.' *J. Amer. Stat. Assoc.*, vol.82, pp.76-89, 1987.

Mazurek, D. F. and DeWolf, J. T. ' Experimental study of bridge monitoring technique.' *Journal of Structural Engineering.*, ASCE, 116(9), pp.2532-2549, 1992.

McCulloch, W. S. and Pitts, W. 'A logical calculus of the ideas immanent in nervous activity.' *Bulletin of Mathematical Biophysics*, vol.5, pp. 115-133. 1943.

Miller, W. T., Glanz, F. H. and Kraft, L. G. 'Application of a general learning algorithm to the control of robotic manipulators.' *International Journal of Robotics*, vol.6, pp. 84-98, Febuary 1987.

Minsky, M. A. and Papert, S. A. *Perceptrons - Expanded Edition.* Cambridge, MA: MIT Press, 1988.

Montague, G. A., Morris, A. J. and Willis, M. J. 'Artificial neural networks: Methodologies and applications in process control.' *Neural networks for control and systems*, 1992, Short Run Press Ltd., England.

Musavi, M. T., Chan K. H., Hummels, D. M. and Kalantri, K. 'On the generalization ability of neural network classifier.' IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.16, no.6, pp.659-663, 1994.

Narendra, K. S. and Parathasathy, K. 'Identification and control of dynamical systems using neural networks.' *IEEE Transactions on Neural Networks*, vol.1, no.1, pp.4-27, 1990.

Narendra, K. S. and Wakatsuki, K. 'A comparative study of two neural network architectures for the identification and control of nonlinear dynamical systems.' Technical report, Center for Systems Science, Yale University, New Haven, CT, 1991.

Newland, D. E. *An Introduction to Random Vibrations, Spectral & Wavelet Analysis*, 1993, Longman Scientific & Technical, England.

Newmark, N. M. 'A Method of Computation for Structural Dynamics.' *Journal of Engineering Mechanics, ASCE 3*, pp. 67-94, 1959.

Ney, H. 'On the probabilistic interpretation of neural network classifiers and discriminative training criteria.' IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.17, no. 2, pp.107-119, 1995.

Nguyen, D. and Widrow, B. 'Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights.' In *International Joint Conference of Neural Networks*, vol.3, pp. 21-26, July 1990.

Nikolaou, M., You, Y. and Sarimveis, H. 'Process modelling with recurrent neural networks.' *Proceedings of ANNIE* , 1991.

Park, J. and Sandberg, I. W. 'Universal approximation using radial-basis-function networks.' *Neural Computation*, 3, pp. 246-257, 1991.

Parker, D. B. 'Learning logic.' Tech. Rep. TR-47, Center for Computational Research in Economics and Management Science, MIT, April 1985.

Pham, D. T. 'Neural networks in engineering.' in *Applications of Artificial Intelligence in Engineering, Proceedings of the 9th International Conference*, PA, USA, pp.3-35, July 1994.

Plaut, D., Nowlan, S. and Hinton, G. 'Experiments on Learning by Back Propagation.' Technical Report CMU-CS-86-126, Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA.

Poggio, T. and Girosi, F. 'A theory of networks for approximation and learning.' A.I. Memo No. 1140, Artificial Intelligence Laboratory, MIT, 1989.

Poggio, T. and Girosi, F. 'Networks for approximation and learning.' In *Proceedings of IEEE*, 78(9), September 1990.

Poggio, T. and Girosi, F. 'Extension of a theory of networks for approximation and learning: dimensionality reduction and clustering.' In *Proceedings Image Understanding Workshop*, pp. 597-603, Pittsburgh, Pennsylvania, September 1990.

Ranjithan, S., Eheart, J. W. and Garrett, J. H. 'Neural Network-based Screening for Groundwater Reclamation Under Uncertainty.' *Journal of Water Resources Research*, 1992.

Rehak, D. R. and Garrett, J. H. 'Strucural System Control using Neural Networks.' In *Proceedings of the 8th Conference on Computing in Civil Engineering*, June 1992.

Rosenblatt, F. *Principle of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms.* Sparton Books, 1962.

Rumelhart, D. E., Hinton, G. E. and Williams, R. J. 'Learning internal representations by error propagation.' In *Parallel Distributed Processing*, (D. E. Rumelhart and J. L. McClelland, eds.), ch. 8,Cambridge, MA: MIT Press, 1986.

Rumelhart, D. E., Hinton, G. E. and Williams, R. J. 'Interactive processes in speech perception: the trace model.' In *Parallel Distributed Processing*, (D. E. Rumelhart and J. L. McClelland, eds.), ch. 15,Cambridge, MA:MIT Press, 1986.

Salane, H. J., Baldwin, J. W. and Duffield, R. C. 'Dynamic approach for monitoring bridge deterioration.' *Transp. Res. Rec.*, pp.832, 1981.

Steinbuch, K. and Piske, U. 'Learning matrices and their applications.' In *IEEE Transactions on Electronic Computers*, pp. 846-862, 1963.

Szewczyk , Z. P. and Hajela, P. 'Neural networks based damage detection in structures.' *Tech. Rep.*, RPI, Troy, N.Y, 1992.

Szewczyk , Z. P. and Hajela, P. 'Damage detection in structures based on feature-sensitive neural networks.' *Journal of Computing in Civil Engineering.*, ASCE, vol.8, no.2, pp. 163-178, 1994.

Valiant, L. G. 'Learning disjunctions of conjunctions.' In *Proceedings 9th International Joint Conference on Artificial Intelligence*, pp. 560-566, August 1985.

Ventura, C. E., Schuster, N. D. and Felber, A. 'Ambient vibration testing of a 32 storey reinforced concrete building during construction.' *Proceedings of the 13th International Modal Analysis Conference* (Nashville, TN), vol.3, pp.1164-1170, 1995.

Vogl, T. P., Mangis, J. K., Rigler, A. K., Zink, W. T. and Alkon, D. L. 'Accelerating the Convergence of the Back Propagation Method.' *Biological Cybernetics*, vol. 59, pp. 257-263, 1988.

Wahba, G., 'Spline bases, regularization, and generalized cross-validation for solving approximation problems with large quantities of noisy data.' In J. Ward and E. Cheney, editors, *Proceedings of the International Conference on Approximation theory in honour of George Lorenz*, Austin, TX, January 8-10 1980. Academic Press.

Wahba, G. *Splines Model for Observational Data. Series in Applied Mathematics*, Vol. 59, SIAM, Philadelphia, 1990.

Waibel, A., Hanazawa, T., Hinton, G., Shikano, K. and Lang, K. 'Phoneme recognition using time-delay neural networks.' Technical Report TR-1-006, ATR Interpreting Telephony Research Laboratories, Japan, 1987.

Watanabe, S. and Fukumizu, K. 'Probabilistic design of layer neural networks based on their unified framework.' *IEEE Transaction of Neural Networks*, vol.6, no.3, pp.691-702, 1995.

Werbos, P. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences.* Ph.D thesis, Harvard, August 1974.

Werbos, P. 'Neural networks for control and system identification.' In *IEEE Conference on Decision and Control (Florida)*, IEEE, New York, 1989.

Widrow, B. and Hoff, M. E. 'Adaptive switching circuits.' In *1960 Wescon Convention Record Part IV*, pp. 96-104, August 1960.

Wu, X. 'A Neural Network Approach to material Modeling.' Ph.D. Dissertation, Department of Civil Engineering, University of Illinois at Urbana-Champaign, Urbana, IL, 1991.

Wu, X., Ghaboussi, J. and Garrett Jr, J. H. 'Use of neural networks in detection of structural damage.' *Computer and Structure*, 42(4), pp.649-659, 1992.

Yee, E. K. L. and Tsuei, Y.G. 'An investigation to the solution of component modal synthesis.' *Proceedings of the 6th International Modal Analysis Conference at Orlando*, FL, 1988.

Yuille, A., N. 'Grzywacz. The motion coherence theory.' In *Proceedings of the International Conference on Computer Vision*, pp.344-354, Washington D.C., December 1988. IEEE Computer Society Press.

Zimmerman, D. C., Simmermacher, T. and Kaouk, M. 'Structural damage detection using frequency response functions.' *Proceedings of the 13th International Modal Analysis Conference* (Nashville, TN), vol.1, pp.179-184, 1995.

Appendix A

Mode Shape Compilation Process

When the ambient excitation and mode shape approach is employed, a mode shape compilation routine is needed for creating the mode shapes that correspond to various damage conditions of the monitored structure. The mode shapes are then used as input patterns for the diagnosis system. Details of how this mode shape compilation routine works are demonstrated as follows.

Firstly, given the ambient excitation, the time-displacement free vibration data from the sensors located at the interested parts of the structure is collected. Assuming that the time-displacement free vibration data, *x(t)*, from a specific sensor is recorded during the time period T and then passed through an analog-to-digital converter to generate the discrete time series $\{x_r\}$, $r = 0, 1, 2, ..., (N\text{-}1)$. The Fast Fourier Transform (FFT) is then used to calculate the Discrete Fourier Transform (DFT) of this time series, $\{X_k\}$, $k = 0, 1, 2, ..., (N\text{-}1)$, and hence find the spectral estimate

$$\tilde{S}_{xx}(w_k) \cong \frac{T}{2\pi} X_k^* X_k \ ,$$

where X_k is the DFT of $\{X_k\}$, and X_k^* is the complex conjugate of X_k. These spectral estimates are then smoothed in order to improve their statistical reliability by a spectrum

smoothing routine in MATLAB (based on the method suggested by Newland, 1993) before being used for mode shape compilation. Since the sampling interval is set at 0.01 second, the Nyquist frequency is set to be 50 Hz, which well covers the frequency range of interest (Newland, 1993, suggests Nyquist frequency to be at least twice the frequency of interest). The time series signal from all sensors is filtered to remove frequency components over 50 Hz in order to avoid the effect of aliasing when the signal is used for creating spectral estimates. The frequency resolution is set at 1 Hz, and the maximum effective bandwidth of the calculation is set at 0.5 Hz. The ratio of the standard deviation of measurement to the mean of measurement is set at 0.3, and the required measurement length is determined to be 20 seconds. The smoothing process involves averaging every 11 adjacent spectral estimates, and each value of spectral estimates is the result of smoothing over 0.5 Hz. The effective frequency range of the spectral estimates is from 0 to 50 Hz, which also means the spectrums well cover the response of the first 3 modes of vibration of the beam model. Finally, the smoothed spectral estimates are then employed in the mode shape compilation process.

The amplitudes of a particular peak of various spectral estimates, which are created from the time-response data from sensors located at various locations of the structure, are then compiled to create the particular mode of vibration of the structure. The frequency corresponding to the particular peak is considered as the corresponding modal frequency. For example, the amplitudes of the first peaks of all spectrums can be compiled to create the first mode of vibration of the structure, while the amplitudes of the second and the third peaks can be compiled to create the second and the third mode respectively. Figure A.1 illustrates the process of the mode shape compilation routine.

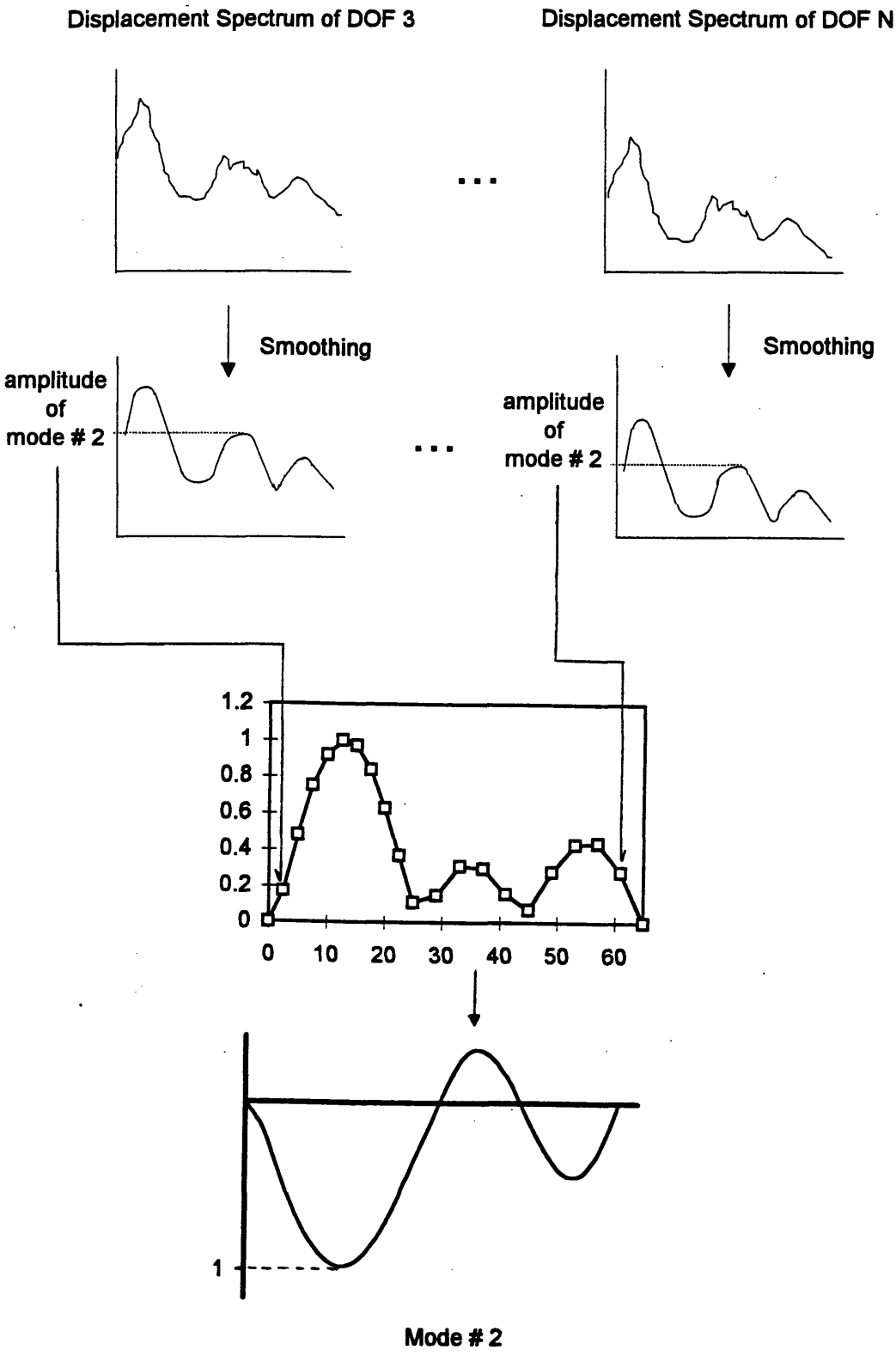


Figure A.1 Modal Analysis Approach

Appendix B

Frequency Transfer Functions of Linear Systems

Figure B.1 demonstrates the beam model of a middle span of a multispan beam model. When the excitation of the beam is generated outside the area of the span, it can be assumed that moment *x1* and *x2*, which are the moment at the left and right support respectively, are the only excitations on the span. Assuming that the vertical displacement at a specific location of the span, *y*, is the output of interest, a linear system with 2 inputs and 1 output can be created to represent the relation between input and output as shown in Fig B.2 (Newland, 1994). The frequency transfer functions, *H1(w)* and *H2(w)*, can be used to represent the linear system. Given the measured input and output data during a time period, the frequency transfer functions between the inputs and the output can be constructed.

Using Convolution Integral Method for a linear system (Newland, 1994), the output *y* can be determined from

$$y(t) = \int_{-\alpha}^{t} h_1(t-\tau)\ x_1(\tau)\ d\tau + \int_{-\alpha}^{t} h_2(t-\tau)\ x_2(\tau)\ d\tau ,$$

where *h1(t)* and *h2(t)* are the frequency transfer functions as function of time *t* (or the unit impulse response functions). If we define

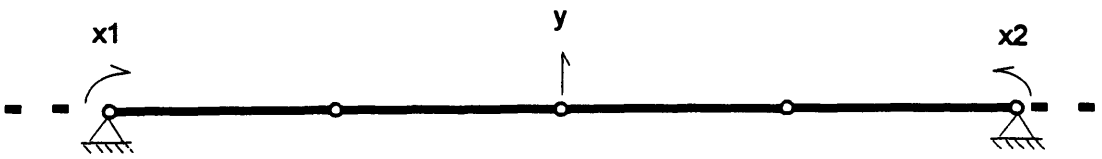


Figure B.1: A middle span of a multi-span beam.

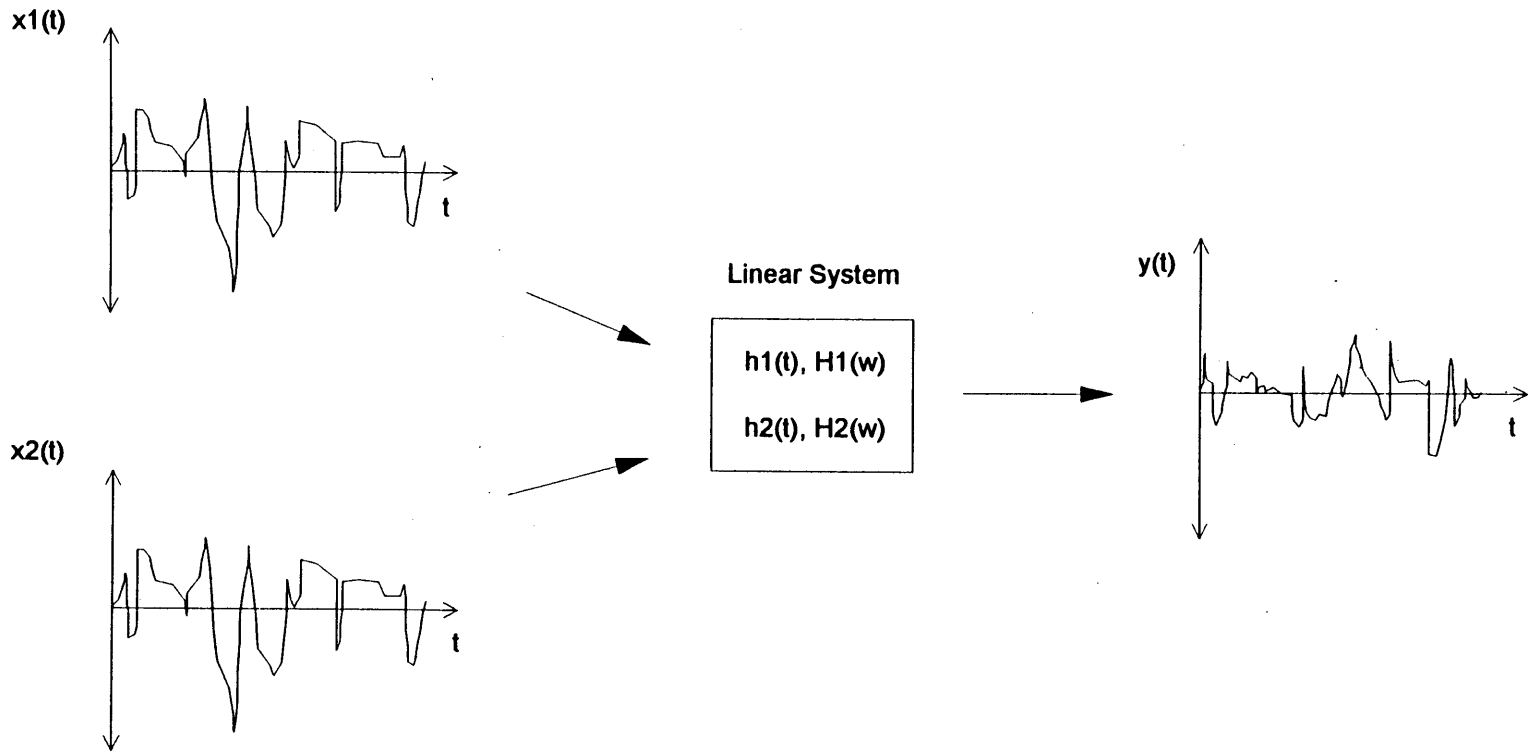


Figure B.2: A linear system.

$$\theta = t - \tau ,$$

output y can be represented as

$$y(t) = \int_{\alpha}^{0} h_1(\theta)\ x_1(t-\theta)\ (-d\theta) + \int_{\alpha}^{0} h_2(\theta)\ x_2(t-\theta)\ (-d\theta) ,$$

$$= \int_{0}^{\alpha} h_1(\theta)\ x_1(t-\theta)\ d\theta + \int_{0}^{\alpha} h_2(\theta)\ x_2(t-\theta)\ d\theta .$$

Since $h(\theta) = 0$ for $\theta < 0$ (there is no output before the impulse), the lower limit of the integral can be extended from 0 to $-\alpha$,

$$y(t) = \int_{-\alpha}^{\alpha} h_1(\theta)\ x_1(t-\theta)\ d\theta + \int_{-\alpha}^{\alpha} h_2(\theta)\ x_2(t-\theta)\ d\theta . \qquad \text{(B.1)}$$

The cross-correlation between the output y and an input x_1 can be demonstrated by

$$R_{x_1y}(\tau) = E\left[x_1(t)y(t+\tau)\right] . \qquad \text{(B.2)}$$

Substitute Eq. B.1 into B.2 gives

$$R_{x_1y}(\tau) = E\left[x_1(t)\int_{-\alpha}^{\alpha} h_1(\theta)\ x_1(t+\tau-\theta)\ d\theta + x_1(t)\int_{-\alpha}^{\alpha} h_2(\theta)\ x_2(t+\tau-\theta)\ d\theta\right] .$$

Since $x_1(t)$ is not a function of θ, it may be moved under the integral signs, and the averaging process is carried out to give

$$R_{x_1y}(\tau) = \int_{-\alpha}^{\alpha} h_1(\theta)\ E\left[x_1(t)x_1(t+\tau-\theta)\right] d\theta + \int_{-\alpha}^{\alpha} h_2(\theta)\ E\left[x_1(t)x_2(t+\tau-\theta)\right] d\theta .$$

Since the autocorrelation of input

$$R_x(\tau) = E[x(t)x(t+\tau)] ,$$

and the cross-correlation between input and output

$$R_{xy}(\tau) = E[x(t)y(t+\tau)] ,$$

the cross-correlation between input $x_1(t)$ and output $y(t)$ is

$$R_{x_1y}(\tau) = \int_{-\alpha}^{\alpha} h_1(\theta)\ R_{x_1}(\tau-\theta)\ d\theta + \int_{-\alpha}^{\alpha} h_2(\theta)\ R_{x_1x_2}(\tau-\theta)\ d\theta .$$

(B.3)

Equation B.3 expresses the cross-correlation between input $x_1(t)$ and output $y(t)$ in terms of the autocorrelation of $x_1(t)$, the cross-correlation between $x_1(t)$ and the other input $x_2(t)$, the frequency response function between x_1 and y ($h_1(t)$), and the frequency response function between x_2 and y ($h_2(t)$).

Performing the Fourier transform of both sides of Eq. B.3 gives

$$S_{x_1y}(\omega) = \frac{1}{2\pi}\int_{-\alpha}^{\alpha} d\tau\ e^{-i\omega\tau}\left[\int_{-\alpha}^{\alpha} h_1(\theta)\ R_{x_1}(\tau-\theta)\ d\theta + \int_{-\alpha}^{\alpha} h_2(\theta)\ R_{x_1x_2}(\tau-\theta)\ d\theta\right]$$

$$= \frac{1}{2\pi}\int_{-\alpha}^{\alpha} d\theta\ h_1(\theta)e^{-i\omega\theta} \int_{-\alpha}^{\alpha} d\tau\ R_{x_1}(\tau-\theta)e^{-i\omega(\tau-\theta)}$$

$$+ \frac{1}{2\pi}\int_{-\alpha}^{\alpha} d\theta\ h_2(\theta)e^{-i\omega\theta} \int_{-\alpha}^{\alpha} d\tau\ R_{x_1x_2}(\tau-\theta)e^{-i\omega(\tau-\theta)} .$$

(B.4)

The integrals with respect to τ have constant θ. If $(\tau-\theta)$ is replaced by ϕ, then $d\tau$ becomes $d\phi$. Since

$$S_x(\omega) = \frac{1}{2\pi} \int_{-\alpha}^{\alpha} R_x(\tau) e^{-i\omega\tau} d\tau \quad , \tag{B.5}$$

$$S_{xy}(\omega) = \frac{1}{2\pi} \int_{-\alpha}^{\alpha} R_{xy}(\tau) e^{-i\omega\tau} d\tau \quad , \tag{B.6}$$

and

$$H(\omega) = \int_{-\alpha}^{\alpha} h(t)\ e^{-i\omega t} dt \quad . \tag{B.7}$$

Eqs. B.5 and B.6 can be employed to evaluate integrals in Eq. B.4 with respect to ϕ. Equation B.7 is then used to evaluate integrals in Eq. B.4 with respect to θ that still remains. Then we obtain

$$S_{x_1 y}(\omega) = H_1(\omega) S_{x_1}(\omega) + H_2(\omega) S_{x_1 x_2}(\omega) \quad .$$

When the system has N separate inputs, of which $x_r(t)$ is a typical one, the system can be represented by

$$S_{x_r y}(\omega) = \sum_{s=1}^{N} H_s(\omega) S_{x_r x_s}(\omega) \quad , \tag{B.8}$$

where

$$S_{x_r x_r} = S_{x_r} \quad .$$

For uncorrelated inputs, it can be shown that

$$S_{xy}(\omega) = H(\omega)S_x(\omega) \ ,$$

$$\therefore H(\omega) = \frac{S_{xy}(\omega)}{S_x(\omega)} \ .$$

For the linear system with two correlated inputs (x_1, x_2) and one output (y) as shown in Figs B.1 and B.2,

$$S_{x_1y}(\omega) = H_1(\omega)S_{x_1}(\omega) + H_2(\omega)S_{x_1x_2}(\omega) \ ,$$

$$S_{x_2y}(\omega) = H_2(\omega)S_{x_2}(\omega) + H_1(\omega)S_{x_2x_1}(\omega) \ .$$

The frequency transfer functions can then be demonstrated by

$$H_1(\omega) = \frac{S_{x_2}S_{x_1y} - S_{x_2y}S_{x_1x_2}}{S_{x_1}S_{x_2} - S_{x_1x_2}S_{x_2x_1}} \ ,$$

$$H_2(\omega) = \frac{S_{x_1}S_{x_2y} - S_{x_1y}S_{x_2x_1}}{S_{x_1}S_{x_2} - S_{x_1x_2}S_{x_2x_1}} \ . \qquad \text{(B.9)}$$

These equations prove that the frequency transfer functions of a linear system can be constructed from spectrums and cross-spectrums of the inputs and the corresponding outputs of the system. These spectrums can be created by performing discrete fourier transform on the discrete time history data of the inputs and output to create fourier spectrums, and then using these fourier spectrums to create the spectrums and cross-spectrums.

Assuming that *x(t)* and *y(t)* are recorded during the same time period T, and then passed through an analog-to-digital converter to generate the discrete time series $\{x_r\}$ and

$\{y_r\}$, $r = 0, 1, 2, ..., (N\text{-}1)$. FFT can then be used to calculate the DFT's (Discrete Fourier Transforms) of these two time series, $\{X_k\}$ and $\{Y_k\}$, $k = 0, 1, 2, ..., (N\text{-}1)$. Spectral estimates can then be demonstrated as

$$\tilde{S}_{xx}(w_k) \cong \frac{T}{2\pi} X_k^* X_k \quad , \quad \tilde{S}_{xy}(w_k) \cong \frac{T}{2\pi} X_k^* Y_k \quad ,$$

$$\tilde{S}_{yx}(w_k) \cong \frac{T}{2\pi} Y_k^* X_k \quad , \quad \tilde{S}_{yy}(w_k) \cong \frac{T}{2\pi} Y_k^* Y_k \quad , \qquad \text{(B.10)}$$

where X_k^* is the complex conjugate of DFT X_k , and Y_k^* is the complex conjugate of DFT Y_k . The spectral estimates will satisfy equations above whether or not there is noise present in the output process *y(t)*, and whether or not the system is linear (Newland, 1993).

For accurate assessment of frequency transfer functions, spectral estimates have to be properly smoothed in order to improve their statistical reliability before being substituted into Eq. B.9. Smoothing spectrum routine in MATLAB is employed for this purpose. Since the sampling interval is set at 0.01 second, the Nyquist frequency is set to be 50 Hz, which well covers the frequency range of interest (Newland, 1993, suggests Nyquist frequency to be at least twice the frequency of interest). The time series signal from all sensors is filtered to remove frequency components over 50 Hz in order to avoid the effect of aliasing when the signal is used for creating spectral estimates. The frequency resolution is set at 1 Hz, and the maximum effective bandwidth of the calculation is set at 0.5 Hz. The ratio of the standard deviation of measurement to the mean of measurement is set at 0.3, and the required measurement length is determined to be 20 seconds. The smoothing process involves averaging every 11 adjacent spectral estimates, and each value of spectral estimates is the result of smoothing over 0.5 Hz. The effective frequency range of the spectral estimates is from 0 to 50 Hz, which also means the spectrums well cover

the response of the first 3 modes of vibration of the beam model. The smoothed spectral estimates are then employed in creating frequency transfer functions. Detail on smoothing spectral estimates and methods for evaluating the quality of frequency transfer functions, which are created from spectral estimates, is discussed in Newland (1993).

Appendix C

Relation of Vibrational Signatures and Damage of Bending Beam

C.1 Relation of Mode Shapes and Damage

Figure C.1a shows a beam with flexural rigidity $EI(x)$ and mass $m(x)$ per unit length. The beam is shown as simply supported, but other support conditions are equally admissible. Transverse vibration is allowed under the action of a distributed force $p(x,t)$. The transverse displacement at any point along the beam is presented by $u(x,t)$, which is a function of both the spatial coordinate x and time t.

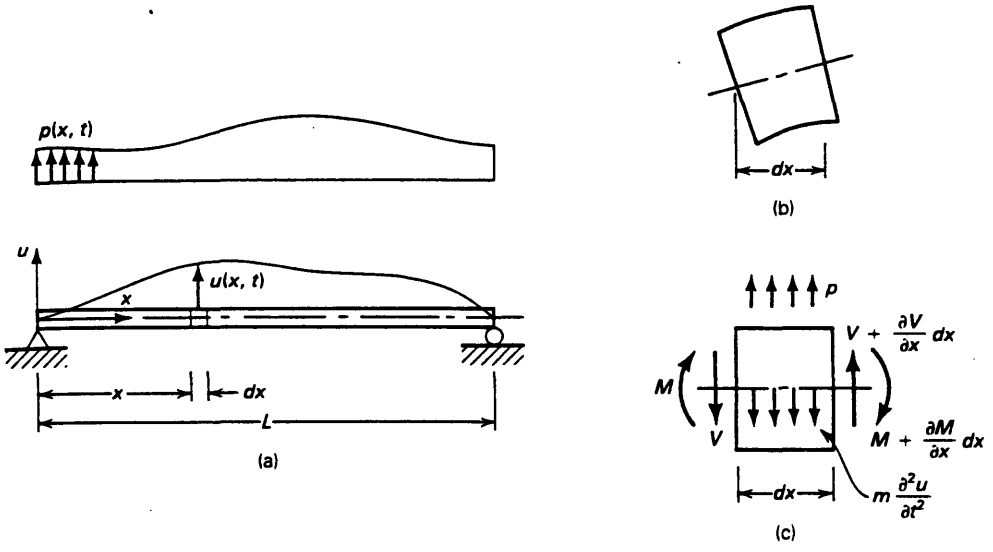


Figure C.1: Transverse vibration of a beam (Humar, 1990).

A small beam element of length dx is shown in Fig C.1b in its deformed position and the forces acting on the element are identified in Fig C.1c. These forces consist of an external force pdx ; the inertia force $m\partial^2 u/\partial\ t^2 dx$; the shear force V and the moment M on the left-hand face of the element; and the shear force $V+\partial\ V/\partial\ xdx$ and the moment $M+\partial\ M/\partial\ xdx$ on the right-hand face. The inertia moment caused by angular acceleration of the element, and the damping force, are neglected.

The infinitesimal element is in equilibrium under the forces and moments shown in Fig C.1c. The shear forces, which act in the direction perpendicular to the elastic axis, are slightly inclined to the vertical. For small rotation, their vertical components can be taken as equal to the shear force values. Equilibrium of the element in the vertical direction gives

$$\frac{\partial\ V}{\partial\ x}dx - m\frac{\partial^2 u}{\partial\ t^2}dx + pdx = 0 \tag{C.1a}$$

or

$$\frac{\partial\ V}{\partial\ x} - m\frac{\partial^2 u}{\partial\ t^2} + p = 0. \tag{C.1b}$$

Equating the sum of moments about the left-hand face to zero gives

$$\left(V+\frac{\partial\ V}{\partial\ x}dx\right)dx + pdx\frac{dx}{2} - m\frac{\partial^2 u}{\partial\ t^2}dx\frac{dx}{2} + M + \frac{\partial\ M}{\partial\ x}dx - M = 0. \tag{C.2}$$

On neglecting the higher order terms, Eq. C.2 becomes

$$V+\frac{\partial\ M}{\partial\ x} = 0 \tag{C.3}$$

From elementary beam theory,

$$M = EI\frac{\partial^2 u}{\partial\ t^2}. \tag{C.4}$$

Substitution of Eq. C.4 into C.3 gives

$$V = \frac{-\partial}{\partial x}\left(EI\frac{\partial^2 u}{\partial x^2}\right). \tag{C.5}$$

Differentiating Eq. C.5 and then substituting in Eq. C.1b gives

$$\frac{\partial^2}{\partial x^2}\left(EI\frac{\partial^2 u}{\partial x^2}\right)+m\frac{\partial^2 u}{\partial t^2} = p. \tag{C.6}$$

Equation C.6 is the equation governing the transverse vibration of the beam. To obtain a unique solution to this equation, four boundary conditions and two initial conditions must be specified. For a simply supported beam, the four boundary conditions are

$$u = 0 \quad \text{at} \quad x = 0$$
$$u = 0 \quad \text{at} \quad x = L \tag{C.7}$$

and

$$EI\frac{\partial^2 u}{\partial x^2} = 0 \quad \text{at} \quad x = 0$$
$$EI\frac{\partial^2 u}{\partial x^2} = 0 \quad \text{at} \quad x = L. \tag{C.8}$$

Other type of boundary conditions can as easily be identified for other types of supports.

The equation of undamped free transverse vibrations of a beam is obtained from Eq. C.6 by setting $p = 0$:

$$\frac{\partial^2}{\partial x^2}\left(EI\frac{\partial^2 u}{\partial x^2}\right)+m\frac{\partial^2 u}{\partial t^2} = 0. \tag{C.9}$$

This is a fourth-order linear homogeneous partial differential equation. Assuming

$$u = f(x)g(t), \tag{C.10}$$

where $f(x)$ is a function of x alone and $g(t)$ is a function of t alone. Substitution of Eq.. C.10 into Eq. C.9 gives

$$g(t)\frac{d^2}{dx^2}\left(EI\frac{d^2f(x)}{dx^2}\right)+mf(x)\frac{d^2g(t)}{dt^2}=0 \tag{C.11}$$

or

$$\frac{1}{mf(x)}\frac{d^2}{dx^2}\left(EI\frac{d^2f(x)}{dx^2}\right)=\frac{-1}{g(t)}\frac{d^2g(t)}{dt^2}. \tag{C.12}$$

The terms on the left-hand side of Eq. C.12, including m and EI, are all functions of x alone, while the terms on the right-hand side are functions of t alone. The equality can hold only provided that each of the two sides of the equation is equal to a constant, normally referred to as a "separation constant." Eq. C.12 leads to two separate equations, as follows:

$$\frac{d^2g(t)}{dt^2}+w^2g(t)=0 \tag{C.13}$$

$$\frac{d^2}{dx^2}\left(EI\frac{d^2f(x)}{dx^2}\right)=w^2mf(x) \tag{C.14}$$

where w^2 is used as the constant. The solution of Eq. C.13 is given by

$$g(t) = A\sin wt + B\cos wt \tag{C.15}$$

where A and B are constants that can be determined from the two initial conditions: the initial displacement and velocity profiles of the beam.

Equation C.14, along with the boundary conditions given by Eq. C.7 and C.8, represent an eigenvalue problem.

$$\frac{d^2}{dx^2}\left(EI\frac{d^2 f(x)}{dx^2}\right) = w^2 m f(x) \tag{C.14}$$

with the boundary conditions

$$f(x) = 0 \quad \text{at} \quad x = 0$$
$$f(x) = 0 \quad \text{at} \quad x = L \tag{C.16}$$

and

$$EI\frac{d^2 f(x)}{dx^2} = 0 \quad \text{at} \quad x = 0$$
$$EI\frac{d^2 f(x)}{dx^2} = 0 \quad \text{at} \quad x = L. \tag{C.17}$$

A nontrivial solution for Eq. C.14 is possible for special value of w^2. There is an infinite number of such values separated by discrete intervals. These values are referred to as "eigenvalues" of the system. The square root of an eigenvalue is known as the "frequency" of the system. Corresponding to each eigenvalue, there is a solution for *f(x)*, called an "eigenfunction" or a "mode shape," which also satisfies the boundary conditions of Eqs. C.16 and C.17.

Given a specific distribution of *EI* over the length of the beam, *EI(x)*, the solution of the eigenvalue problem of Eq. C.14 is a specific set of "mode shapes" and their corresponding "frequencies," $\{f_i(x), w_i\}_{i=1}^{\infty}$. In another perspective, if a specific *EI(x)* is defined as a certain damage condition, the corresponding $\{f_i(x), w_i\}_{i=1}^{\alpha}$ is the particular set of mode shapes and their corresponding frequencies that corresponds to the damage condition. Figure C.2 illustrates the effect of the change of *EI(x)* to a mode shape of the 2-span beam demonstrated in Chapter 6.

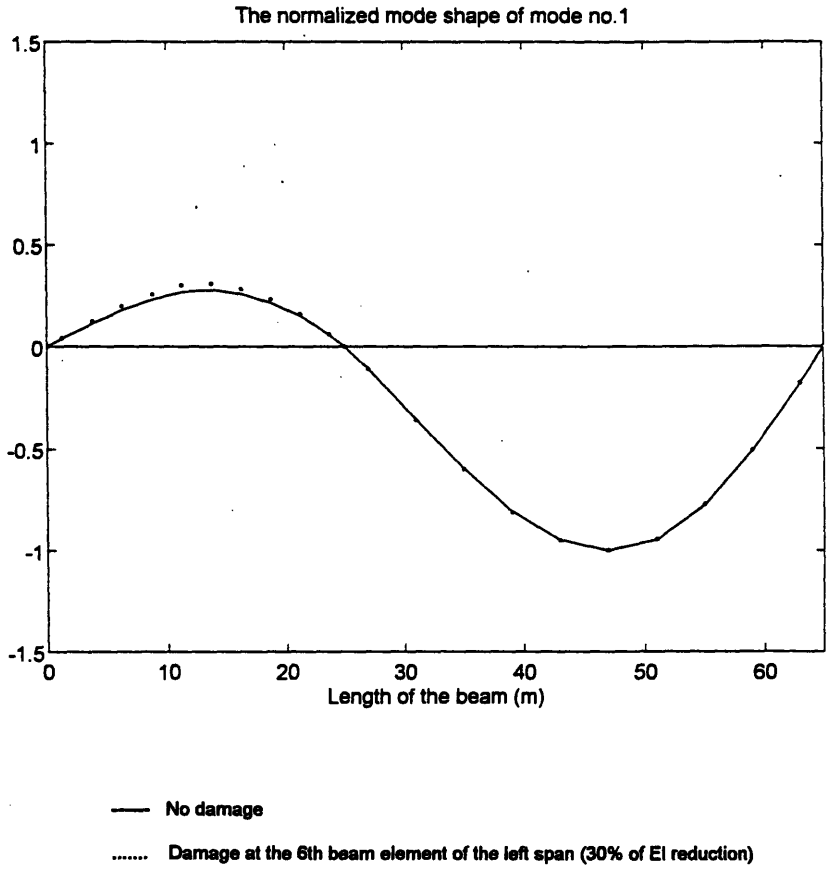


Figure C.2: Change of the 1st mode shape due to a damage condition

C.2 Relation of Response Spectrums and Damage

Response spectrums at proper locations of structures contain as much information about the condition of the structure as mode shapes do. The only difference is that the response spectrums are normalized by the excitation, whereas the mode shapes are normalized by their amplitude (see Appendix A). This makes mode shapes independent of excitation. Figure C.3 illustrates the effect of the change of *EI(x)* to a response spectrum of the 2-span beam shown in Chapter 6.

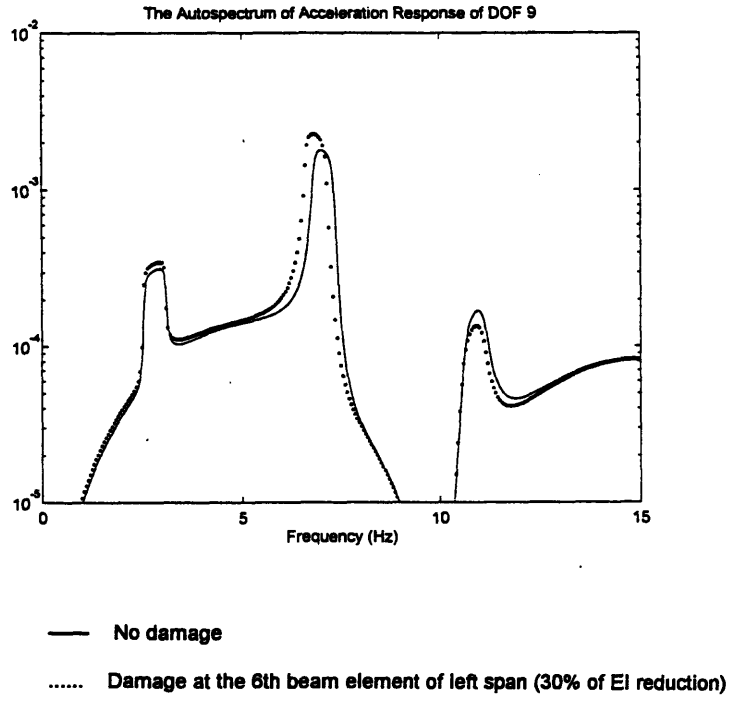


Figure C.3: Change of the response spectrum of DOF 9 due to a damage condition.

C.3 Relation of Frequency Transfer Functions and Damage

Figure C.4 shows a beam similar to the one shown in Fig C.1a, but it is now under the action of the moment at the left-end of the beam M_1. The beam has flexural rigidity *EI(x)*, damping *c(x)*, and mass *m(x)* per unit length. The transverse displacement at any point along the beam is presented by *u(x,t)*.

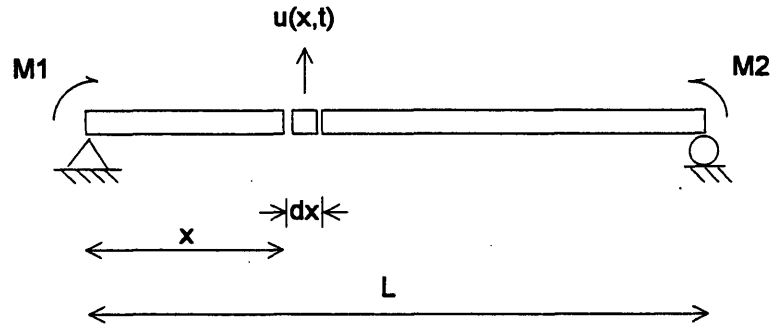


Figure C.4: Transverse vibration of a beam with moment at the left support.

The equation of motion of the beam is still based on the general equation of motion of the transverse vibration of bending beam described by Eq.C.6,

$$\frac{\partial^2}{\partial x^2}\left(EI\frac{\partial^2 u}{\partial x^2}\right)+m\frac{\partial^2 u}{\partial t^2}=0, \tag{C.18}$$

where $p = 0$ since there is no distributed force on the beam. The four boundary conditions are

$$u=0 \quad \text{at} \quad x=0$$
$$u=0 \quad \text{at} \quad x=L \tag{C.7}$$

and

$$EI\frac{\partial^2 u}{\partial x^2}=M_1 \quad \text{at} \quad x=0$$
$$EI\frac{\partial^2 u}{\partial x^2}=0 \quad \text{at} \quad x=L, \tag{C.19}$$

which provide the presence of the moment M_1 at the left support.

Given a specific distribution of EI over the length of the beam, $EI(x)$, the solution of the eigenvalue problem of Eq. C.14 is still the specific set of "mode shapes" and their corresponding "frequencies," $\{f_i(x), w_i\}_{i=1}^{\alpha}$ of the beam.

Now consider the force response of the beam,

$$u(x,t) = \sum_{i=1}^{\alpha} f_i(x) y_i(t). \tag{C.20}$$

The mode shapes are normalized such that

$$\int_0^L f_i(x) f_j(x) dx = L\delta_{ij} \tag{C.21}$$

where $\delta_{ij} = 0$ except when $i = j$, when it is unity. Substitute Eq. C.21 into Eq. C.18 gives

$$\sum_{i=1}^{\alpha} \left\{ \frac{\partial^2}{\partial x^2} \left(EI \frac{\partial^2 f_i(x)}{\partial x^2} \right) y_i + m f_i(x) \frac{\partial^2 y_i}{\partial t^2} \right\} = 0. \tag{C.22}$$

Multiplying Eq. C.22 by $f_j(x)$, and integrating over x, give

$$\frac{\partial^2 y_j}{\partial t^2} + \frac{1}{mL} \int_0^L \sum_{i=1}^{\alpha} \left\{ \frac{\partial^2}{\partial x^2} \left(EI \frac{\partial^2 f_i(x)}{\partial x^2} \right) f_j(x) y_i \right\} dx = 0$$

$$\frac{\partial^2 y_j}{\partial t^2} + w_j^2 \left(EI(x), m, L, j \right) y_j = 0 \tag{C.23}$$

where the undamped natural frequencies $\{w_j\}_{j=1}^{\alpha}$ of the beam are given by a function

$$w_j^2 \left(EI(x), m, L, j \right) y_j = \frac{1}{mL} \int_0^L \sum_{i=1}^{\alpha} \left\{ \frac{\partial^2}{\partial x^2} \left(EI \frac{\partial^2 f_i(x)}{\partial x^2} \right) f_j(x) y_i \right\} dx \tag{C.24}$$

Substitute Eq. C.20 into Eq. C.18 gives

$$\sum_{i=1}^{\alpha}\left\{\left(EI(x)\frac{\partial^2 f_i(x)}{\partial x^2}\right)y_i(t)\right\} = M_1(t) \quad , \quad x = 0$$

(C.25)

Multiplying Eq. C.25 by $f_j(x)$, and integrating over x, give

$$\int_O^L \sum_{i=1}^{\alpha}\left\{\left(EI(x)\frac{\partial^2 f_i(x)}{\partial x^2}\right)f_j(x)y_i(t)\right\}dx = \int_O^L f_j(x)M_1(t)dx \quad , \quad x = 0$$

$$y_j(t)\int_O^L\left\{\frac{\partial^2 f_j(x)}{\partial x^2}f_j(x)EI(x)\right\}dx = M_1(t)\int_O^L f_j(x)dx \quad , \quad x = 0$$

$$y_j(t) = M_1(t)\frac{\int_O^L f_j(x)dx}{\int_O^L\left\{\frac{\partial^2 f_j(x)}{\partial x^2}f_j(x)EI(x)\right\}dx} \quad , \quad x = 0.$$

(C.26)

Since

$$w_j \propto \int_O^L\left\{\frac{\partial^2 f_j(x)}{\partial x^2}f_j(x)EI(x)\right\}dx \quad ,$$

$$y_j(t) = M_1(t)Q_j\left(f_j(x), EI(x), w_j, L\right) \quad , \quad x = 0.$$

(C.27)

Assuming that the moment at the left support of the beam is

$$M_1(t) = e^{iwt},$$

(C.28)

Eq. C.20 becomes

$$u(x,t) = H(x,w)e^{iwt} = \sum_{i=1}^{\alpha} f_i(x)y_i(t), \tag{C.29}$$

where $H(x,w)$ is the frequency transfer function when the response $u(x,t)$ is measured at x for unit harmonic moment e^{iwt} applied at the left end of the beam. Therefore, from Eqs. C.25 and Eq. C.23, it can be concluded that

$$H(x,w) = \sum_{i=1}^{\alpha} f_i(x)Q_i\left(f_i(x), EI(x), w_j, L\right). \tag{C.30}$$

Given a specific distribution of EI over the length of the beam, $EI(x)$, the solution of the frequency transfer function of the beam, when the response $u(x,t)$ is measured at x, exists (see Eq. C.30). In another perspective, if a specific $EI(x)$ is defined as a certain damage condition, the corresponding $\left\{\{H(x,w)\}_{x=0}^{L}\right\}_{w=0}^{\alpha}$ is the particular set of frequency transfer functions that corresponds to the damage condition.

Figure C.5 illustrates the effect of the change of $EI(x)$ to a frequency transfer function, $H(w)$, corresponding to the response at a location of the left-most span of the 4-span beam demonstrated in Chapter 7.

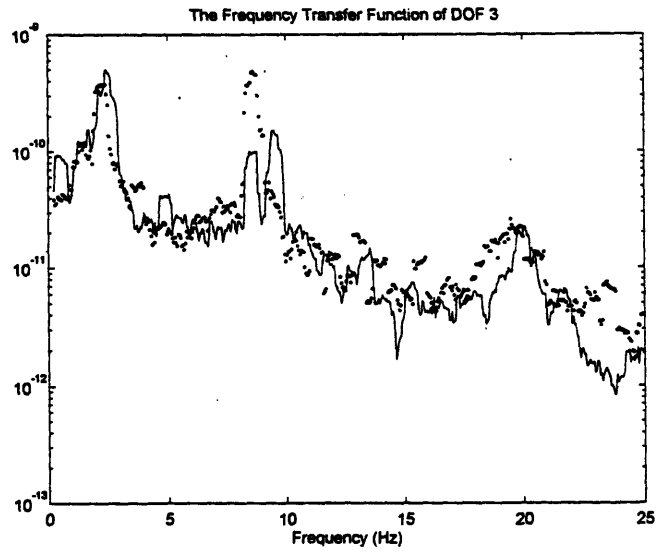


Figure C.5 Change of the frequency transfer function of DOF 3 due to a damage condition