Model Updating Using Neural Networks

by

Mauro J. Atalla

Dissertation submitted to the faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

Engineering Mechanics

APPROVED:

Daniel J. Inman, Chairman

Ali H. Nayfeh

Harry H. Robertshaw

O. H. Griffin, Jr.

Ronald D. Kriz

April, 1996

Blacksburg, Virginia

Keywords: Model Updating, Neural Networks, Adaptive Control

Model Updating Using Neural Networks

by

Mauro J. Atalla

Committee Chairman: Daniel J. Inman

Engineering Science and Mechanics

Abstract

Accurate models are necessary in critical applications. Key parameters in dynamic systems often change during their life cycle due to repair and replacement of parts or environmental changes. This dissertation presents a new approach to update system models, accounting for these changes. The approach uses frequency domain data and a neural network to produce estimates of the parameters being updated, yielding a model representative of the measured data.

Current iterative methods developed to solve the model updating problem rely on minimization techniques to find the set of model parameters that yield the best match between experimental and analytical responses. Since the minimization procedure requires a fair amount of computation time, it makes the existing techniques infeasible for use as part of an adaptive control scheme correcting the model parameters as the system changes. They also require either mode shape expansion or model reduction before they can be applied, introducing errors in the procedure. Furthermore, none of the existing techniques has been applied to nonlinear systems.

The neural network estimates the parameters being updated quickly and accurately without the need to measure all degrees of freedom of the system. This avoids the use of mode shape expansion or model reduction techniques, and allows for its implementation as part of an adaptive control scheme. The proposed technique is also capable of updating weakly nonlinear systems.

Numerical simulations and experimental results show that the proposed method has good accuracy and generalization properties, and it is therefore, a suitable alternative for the solution of the model updating problem of this class of systems.

Dedication

I dedicate this dissertation to my grandparents, Tuffi Atalla and Conceição Manzano Fernandes. Their hard work, dedication, and love made all this possible. Eighty years ago my Grandfather's family fled Lebanon and went to Brazil, where he worked extremely hard his whole life to provide his sons with the things he did not have. My Grandmother comes from a family of spanish immigrants. She also worked very hard to educate her daughters and to provide for them a better life. Without them I could not have written this dissertation.

Acknowledgements

My family is essential to me. Their unconditional love and support kept me going during the hard times. My parents' dedication, hard work, and perseverance made it possible to educate myself and my siblings. Thanks Mom, Dad, Márcio, and Miriam.

One of my best decisions was to come to the U.S. and work with Dr. Inman. I learned many things from him during these years. I thank him for letting me try my own ideas, even when he did not agree with them. These were great years of my life, and they would not have been so without him.

I also would like to thank my committee members, in particular Dr. Nayfeh and Dr. Robertshaw.

This dissertation would not be completed in time if not for my friends' assistance. Eric Austin, Greg Agnes, Deborah Pilkey, and Dino Sciulli spent many hours reading, correcting, and giving suggestions for the final version of this work. Thanks a lot!

Living in a foreign country can be a very difficult experience. I was extremely fortunate to work with people that were not only my officemates but also good friends. Brett Pokines, Joseph Slater, Ralph Rietz, and Don Leo helped me adjust to a new environment and provided many happy and unforgettable moments.

I thank Eric for the discussions, experience, knowledge, and of course, for having Ollie, Greg for the many suggestions and help with the experimental setup, Debbie and Dino for their friendship during these last years, and Clay Carter, Jens Cattarius, Prasad Gade, Margaretha Lam-Anderson.

I also wish to thank the following people for helping, supporting and encouraging me during these years: Liu Hsiang Hui, Hans I. Weber, Karim Tahboub, Michelle K. Idle, Márcio H. Atalla, Carlos A. P. Guido, Miriam P. Atalla, Ilmar F. Santos, and Ana P. N. Nogueira.

Professor Inman's advising of my work was funded in part by the Air Force Office of Scientific Research and by the Army Research Office.

This research was fully supported by the Brazilian government through the National Council for Scientific and Technological Development, CNPq, under scholarship 201371/92. In particular, I would like to thank Mr. Carlos Pittaluga for his prompt help when it became necessary.

Mauro J. Atalla

Contents

1 Introduction **1**

2 Review of Model Updating Techniques **5**

2.1 Introduction 5

2.2 Comparing Numerical and Experimental Data 6

2.2.1 The Modal Assurance Criterion 7

2.2.2 Complex Modes 8

2.2.3 Mode Shape Expansion 9

2.2.4 Model Reduction 10

2.2.5 Choosing Transducer Locations 12

2.3 Error Localization 14

2.3.1 Checking the Choice of Transducer Locations 14

2.4 Model Updating 15

2.4.1 Direct Techniques 15

2.4.2 Iterative Techniques Based on Modal Data 16

2.4.3 Iterative Techniques Based on Frequency Domain Data 20

2.5 Summary 24

3 A Brief Introduction to Neural Networks **26**

3.1 Introduction 26

3.2 The Neuron 28

3.3 Network Architectures 30
3.3.1 Multilayer Perceptron Networks 30
3.3.2 Radial Basis Function Networks 35
3.4 Summary 39

4 Model Updating Using Neural Networks 40
4.1 Introduction 40
4.2 Radial Basis Function Neural Networks - Review 42
4.3 Normalization and choice of the spread constants 48
4.4 Sensitivity of FRFs with respect to parameter changes 52
4.5 Choice of input data 55
4.5.1 Effect of noise in the input data 56
4.6 Weighting the input data 60
4.7 Influence of the number of parameters being updated 60
4.8 Influence of the number of FRFs used in the updating 63
4.9 Nonlinear systems 72
4.10 Computational load 74
4.11 Summary 76

5 Examples 78
5.1 Introduction 78
5.2 Kabe's example 79
5.3 Yang's example 85
5.4 Truck Suspension with a LQR Controller 93
5.5 Duffing oscillator 98
5.6 Cantilevered Beam Experiment 103
5.7 Flexible Frame Experiment 106
5.8 Summary 117

6 Conclusions and Future Work **119**

6.1 Conclusions 119

6.2 Future Work 123

A Experimental Hardware **130**

A.1 The beam experiment 130

A.2 The flexible frame experiment 132

List of Figures

3.1 Biological neuron 29
3.2 Examples of mathematical neurons 29
3.3 Radial basis functions. 37
3.4 Network architecture 38

4.1 Sample vector to illustrate the different variances of two rows of the normalized input matrix. The first row corresponds to the solid line and the second to the dashed line. 49
4.2 Example of possible ambiguity when measuring the distance between FRFs. 56
4.3 Frequency intervals (shaded areas) used to integrate the real and imaginary parts of the frequency response function of a cantilevered beam. 57
4.4 Worst case estimation - no weight factors. 61
4.5 Worst case estimation - $\rho_1 = \rho_2 = 2$. 61
4.6 Worst case estimation - $\rho_1 = \rho_2 = 4$. 62
4.7 Diagram of the truck suspension 63
4.8 Generalization characteristics - One parameter 64
4.9 Generalization characteristics - Two parameters 65
4.10 Generalization characteristics - Four parameters 66
4.11 Error statistics 68
4.12 Generalization characteristics when trained with one FRF 69
4.13 Generalization characteristics when trained with two FRFs 70

4.14 Generalization characteristics when trained with three FRFs 71
4.15 Frequency response of a Duffing oscillator around the primary resonance ($F = 2$, $\mu = 0.1$) 73

5.1 Kabe's example. 80
5.2 Frequency intervals used to update Kabe's problem. 81
5.3 Generalization characteristics of the network trained to solve Kabe's problem. 83
5.4 Generalization characteristics of the network trained to solve the damped version of Kabe's problem. 84
5.5 Yangs's example 86
5.6 Frequency intervals used to update Yang's problem. 88
5.7 Generalization characteristics for the network trained with 528 sample pairs to solve the low damping variation of Yang's problem. 89
5.8 Generalization characteristics for the network trained with 828 sample pairs to solve the low damping variation of Yang's problem. 90
5.9 Generalization characteristics for the network trained with 528 sample pairs to solve the high damping variation of Yang's problem. 91
5.10 Generalization characteristics for the network trained with 828 sample pairs to solve the high damping variation of Yang's problem. 92
5.11 Diagram of the truck suspension 94
5.12 Performance comparison for the expensive control effort case ($R/Q = 1 \times 10^{-2}$). 96
5.13 Performance comparison for the cheap control effort case ($R/Q = 1 \times 10^{-6}$). 97
5.14 The Duffing oscillator 99
5.15 Real and imaginary components of the frequency response function of the Duffing oscillator. 101
5.16 Generalization of the network trained to solve the Duffing oscillator problem. 102
5.17 Cantilevered beam: experimental setup 104
5.18 Frequency intervals used in the cantilevered beam experiment. 105

5.19 Measured, initial and updated frequency responses of the cantilevered beam. 107
5.20 Generalization characteristics of the network trained to solve the cantilevered beam experiment. 108
5.21 Flexible frame diagram. 109
5.22 Diagram of the joints used in the flexible frame. 109
5.23 Finite element mesh for the flexible frame experiment. 110
5.24 Experimental data obtained for the flexible frame experiment. 112
5.25 Frequency intervals used in updating the flexible frame model. 113
5.26 Generalization of the network trained to update the flexible frame model. . 115
5.27 Measured, initial and updated responses of the flexible frame experiment. . 116

A.1 Cantilevered beam: experimental setup. 131
A.2 Flexible frame: experimental setup. 132

List of Symbols

FRF : frequency response function

MLP : multilayer perceptron neural network

RBFNN : radial basis function neural network

NNUM : neural network updating method

FE : finite element

DOF : degree of freedom

flop : floating point operation

flops : floating point operations per second

$\mathcal{M}$: modal assurance criterion matrix

Φ_R : real matrix of eigenvectors

Φ_C : complex matrix of eigenvectors

$\Re(\cdot)$: the real part of a variable

$\Im(\cdot)$: the imaginary part of a variable

ω : frequency, in radians per second

W : weight matrix

M : mass matrix

C : stiffness matrix, or matrix of measurements

K : stiffness matrix

G : gyroscopic matrix

A : state-space matrix

B : matrix of inputs

D : feed-through matrix, or matrix of distances

$B(\omega)$: dynamic stiffness matrix

$H(\omega)$: frequency response function matrix

NI : number of rows in the input vector

N : number of sample pairs used to train the network

N_t : number of sample pairs used to test the network

NH : number of neurons in the hidden layer

NO : number of neurons in the output layer

x_i : i^{th} column input vector

$x_{m,i}$: m^{th} row of the i^{th} input vector

X : input matrix used for training, composed of N columns, x_i, and NI rows

h_j : j^{th} output vector from the hidden layer

$h_{n,j}$: n^{th} row of the j^{th} output vector from the hidden layer

H : output matrix from the hidden layer during training, composed of N columns, h_j, and NH rows

$w_{p,n}$: element of the weight matrix connecting the n^{th} hidden neuron to the p^{th} output neuron

y_k : k^{th} output vector

$y_{p,k}$: p^{th} element of the k^{th} output vector

Y : matrix of outputs, composed of N columns and NO rows

Chapter 1

Introduction

The mathematical model of a structure is verified by building a prototype of the structure, testing it, and comparing experimental and analytical responses. Since these responses often do not agree, the mathematical model must be modified until a good agreement is achieved. The increasing design competitiveness and the development of better and faster computers have allowed engineers to develop larger and more complex models in an attempt to better model the structure and reduce the time spent in the design stage of the production cycle. For the engineer, who has to design the structure based on the mathematical model, it is very important that the structure represented by the model is physically realizable. Having a reliable mathematical model allows the engineer to investigate variations of the original design and choose the best option. It also allows the design of high performance control laws based on an accurate model of the structure. The process of modifying the mathematical model in order to achieve a good agreement with the measured data is called model updating. Model updating differs from system identification in the sense that updating requires a good initial model of the structure and can yield mathematical models that are physically realizable, while system identification techniques do not require, nor do they yield a physically plausible model of the structure.

Model updating of mechanical systems has been an active field of research during the

past 15 years [1]. Many techniques have been proposed where most of the models are derived from finite element analysis. These techniques, developed for linear time-invariant systems, compute correction matrices or corrected parameters that render the mathematical model consistent with experimental measurements. Among the many techniques developed to solve the model updating problem, the ones that result in physically realizable models rely on minimization techniques to find the set of model parameters that yield the best match between experimental and analytical responses. The computational time required by the minimization procedure makes the existing techniques infeasible for use in a real-time fashion to correct the model parameters as the system changes. This is a necessary requirement if a control law acting on the structure is to be adjusted due to the parameter changes. The method proposed in this dissertation uses frequency domain data and a neural network for model updating of linear and weakly nonlinear systems in a way compatible with an adaptive control scheme.

Neural networks have been used in mechanical engineering problems since the early 1990's. The main areas of concentration have been control, identification, and damage detection. The majority of research in this area uses the neural network as a "black box", and the results are mostly limited to computer simulations. A few successful experimental results in the area of vibration control have been published by Chen et al.[2] and Rao et al.[3]. To date, only one work, published by Szewczyk & Hajela [4], used a neural network to solve a problem similar to the model updating problem. In their paper, the inverse eigenvalue problem was solved by taking eigenvalues as input to the network and estimating the corresponding model parameters. No experiment, only simulations were shown to illustrate the method. In the present work, a neural network is used to represent the mapping between frequency domain data and model parameters. Once trained, the neural network quickly yields accurate estimates of the model parameters based on the frequency domain response of the structure. Since the process of estimating the model parameters is fast (of the order of tenths of a second), this technique can be used to adjust the control law acting on the structure in real-time as long as parameter variations are slow

enough to allow for the updating of the system.

Traditional techniques for designing control systems assume a good understanding of the system under study. There are many instances, however, when such an assumption is not true and parameter identification/estimation techniques need to be used in the design of the control scheme. The combination of parameter estimation/identification and control techniques is known as adaptive control, a special type of nonlinear feedback control where the states change on a fast time scale while some parameters change on a slower time scale. [5, 6] Research in adaptive control was motivated in the 1950's by the problem of designing autopilots for airplanes operating in a wide variety of conditions. High-gain controllers were used to render the autopilot insensitive to the parameter variations, but it was found that a simple fixed-gain controller did not produce satisfactory results. An improvement was obtained with gain scheduling, which adjusts the gains in an open-loop fashion based on the measurement of some auxiliary variables. A further improvement is the use of self-adjusting controllers. Belonging to this class are the model-reference adaptive systems (MRAS) and self-tuning regulators (STR) techniques. These techniques estimate directly or indirectly the parameters of the control law as the system changes. Using the technique presented here to estimate the parameter changes results in a control law adjustment based on a change of the physical parameters of the system instead of identified parameters that may not have any physical meaning. This may be a desired feature in systems where tracking changes in physical parameters is of interest to the engineer.

This work concentrates in developing and demonstrating a new model updating technique based on a neural network, the neural network updating method (NNUM). The second chapter reviews the existing model updating techniques, showing their assumptions and derivations. Chapter three gives a brief introduction to the field of neural networks, concentrating on the architectures relevant to this dissertation. The proposed technique is introduced in chapter four. There, the issues involved in the development of this technique are discussed, and two systems are used to illustrate these issues: a cantilevered beam and a simplified truck suspension used by Szewczyk & Hajela [4].

Chapter five shows the performance of the proposed technique when applied to a variety of different problems. It is divided into two major sections: numerical simulations and experiments. Four problems are solved by numerical simulations:

- Kabe's example [7]. This is an eight-degree-of-freedom undamped lumped parameter system with very high modal density and stiffness parameters that differ by orders of magnitude, making the solution of the model updating very difficult.
- A 15-degree-of-freedom damped discrete system studied by Yang & Brown [8]. This system has repeated modes and high modal density. Two variations are analyzed: low and high damping.
- An adjustable LQR controller designed to control the driver's seat of the truck suspension problem illustrates how the technique can be used to adapt a control law acting on a structure.
- The Duffing oscillator. This weakly nonlinear equation is used to verify the behavior of the proposed method when applied to weakly nonlinear systems.

Two experiments are conducted to verify the performance of the proposed approach when used with experimental data.

- A simple cantilevered beam, and
- A flexible frame with high modal density.

The last chapter summarizes the research, emphasizing its contributions to the field of model updating, and outlines the next steps to be taken in the future.

Chapter 2

Review of Model Updating Techniques

2.1 Introduction

Physical structures are continuous systems with infinite degrees of freedom (DOF) and are usually governed by a system of partial differential equations (pde). The first in the analysis is to discretize the structure so one can work with a system of ordinary differential equations (ode). This is the first issue the engineer has to address: how many DOF are necessary to capture the relevant behavior of the system? This is an important decision. The results will not match if not enough DOF are kept in the discretized model. Alternatively, too many DOF result in an excessively long simulation time. Since the mathematical model has its accuracy measured against experimental data obtained from a prototype, it is very important to assure that the experiment, the data collection and the processing are done accurately. Assuming that the measurements are accurate, and that the model is able to reproduce the dynamics of the prototype, the differences between predicted and measured responses will be a result of incorrect values of the parameters in the mathematical model. Model updating is the set of techniques developed to find the correct values of these

parameters.

Most of the literature in model updating deals with models represented by a system of ordinary differential equations (ode), usually produced by a finite element analysis (FEA) of the structure. However, this need not to be the case: in theory, models represented by non-linear or partial differential equations (pde) can also be subjected to model updating.

When modeling a structure three basic sources of error that can occur are [1]:

i. Model structure errors - These errors occur when the model does not represent the physical behavior of the prototype. Examples of this type of error are the assumptions on the linearity of the system and the boundary conditions.

ii. Model order errors - As mentioned before, a sufficient number of DOF must be used in the analysis so that the actual behavior can be modeled. The model must have fully converged with respect to the natural frequencies.

iii. Model parameter errors - These errors occur when the model is good but the numerical values of the physical parameters are incorrect. This is the type of error targeted by model updating techniques.

Measurements must be conducted carefully. Attention should be paid to the process of attaching transducers to the structure, since they can affect its behavior. Special care must be taken in acquiring and discretizing the data to avoid effects like aliasing, leakage and noise contamination.

Summarizing, it is very important to conduct accurate tests and to derive mathematical models able to represent the structure being tested. Model updating is not intended to be a parameter identification scheme, but rather a "fine tuning" procedure.

2.2 Comparing Numerical and Experimental Data

Because experimental data are the basis for the updating of the mathematical model, it is important that equivalent quantities be compared. This might represent a problem

because the discrete mathematical model (DMM) contains many DOF, some of which are not accessible or measurable by current techniques. Examples consist of internal variables and rotational DOF. This means that the model identified by the experiment has fewer DOF than the DMM. Most of the techniques developed for model updating are based on the identified modal characteristics like natural frequencies and mode shapes. There are two ways of circumventing the problem of incompleteness. The first way is to expand the measured modes by filling the unknown entries of the measured mode shapes with data from the analytical model. Because the model is not correct it is necessary to iterate until convergence occurs. The second way is to condense the DMM to a model with fewer DOF by eliminating rotational and non-important DOF. The problem with this approach is that the condensed model loses physical meaning. This is very important if changes in the structure are necessary to achieve a specified goal.

An alternative approach is to compare frequency response functions (FRF) directly. This may be difficult since the analytical model seldom includes the effects of damping. One alternative is to assume proportional damping and identify the coefficients multiplying the mass and stiffness matrices.

In the next few sections some aspects involving the comparison of experimental and analytical data are addressed.

2.2.1 The Modal Assurance Criterion

When comparing experimental and analytical data it is very important that the eigenvectors are paired correctly. The modal assurance criterion (MAC) [9] is a popular technique to estimate the degree of correlation between analytical and measured mode shapes. The MAC is based on a matrix with entries calculated by

$$\mathcal{M}_{i,j} = \frac{\mid u_i^H v_j \mid^2}{(u_i^H u_i)(v_j^H v_j)}, \tag{2.1}$$

where u_i is an experimental mode shape, v_j is an analytical mode shape, and the ^{H} superscript means the transpose of the complex-conjugate. The value of $\mathcal{M}_{i,j}$ is a number between 0 and 1. The closer the value is to 1 the more similar the two vectors are. Ideally, the matrix $\mathcal{M}$ should be the identity matrix. In practice one should obtain values close to one in the diagonal and small values in the off-diagonal entries.

The MAC can be modified to be mass normalized. This results in smaller off-diagonal terms. The modified MAC is calculated by

$$\overline{\mathcal{M}}_{i,j} = \frac{\mid u_i^H M v_j \mid^2}{(u_i^H M u_i)(v_j^H M v_j)}. \tag{2.2}$$

2.2.2 Complex Modes

There is no physical explanation for the assumption of proportional damping. The reason for its use is simply convenience and because it works well for many lightly damped systems. When conducting experiments, because the damping is in general not proportional, the eigenvectors will be complex vectors. Because the DMM usually does not include damping effects, and if it does, it is of the proportional type, the complex eigenvectors are transformed, incorrectly, into real vectors.

A common way of transforming complex eigenvectors into real ones is to use a complex transformation like $\Phi_R = \Phi_C T$, where Φ_R are the real valued eigenvectors, Φ_C the complex eigenvectors and T the complex transformation matrix. Writing the equation above using its real and imaginary components yields [10]

$$\Phi_R = (\Re\Phi_C + \jmath\Im\Phi_C)(\Re T + \jmath\Im T), \tag{2.3}$$

and collecting the real and imaginary components results in

$$\Phi_R = (\Re\Phi_C\Re T - \Im\Phi_C\Im T) + \jmath(\Re\Phi_C\Im T + \Im\Phi_C\Re T).$$

Assuming that the real part of T is the identity matrix ($\Re T = I$), eliminating T and solving for Φ_R results in

$$\Phi_R = \Re\Phi_C + \Im\Phi_C \left(\Re\Phi_C{}^T \Re\Phi_C\right)^{-1} \left(\Re\Phi_C{}^T \Im\Phi_C\right) \ , \tag{2.4}$$

for rectangular Φ_C and Φ_R matrices.

It should be noted that errors are introduced by this approximation and these errors will be carried forward in the updating process.

2.2.3 Mode Shape Expansion

Many times in modal testing the number of mode shape elements that can be measured is smaller than the number of assumed degrees of freedom in the model. Thus, a method is required to "fill out" the remaining entries of the mode shapes.

Mode shape expansion involves completing the measured eigenvectors with data from the analytical model. Two approaches are mentioned here: expansion using mass and stiffness matrices and expansion using modal data.

Expansion Using The Coefficient Matrices

The analytical model is partitioned into master and slave DOF. Denoting the measured natural frequencies by ω_i and the measured master mode shapes by Φ^e_{mi} one can write the eigenvalue problem as

$$\left(\omega_i^2 \begin{bmatrix} M_{mm} & M_{ms} \\ M_{sm} & M_{ss} \end{bmatrix} + \begin{bmatrix} K_{mm} & K_{ms} \\ K_{sm} & K_{ss} \end{bmatrix} \right) \begin{Bmatrix} \Phi_{mi} \\ \Phi_{si} \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix} . \tag{2.5}$$

Usually the number of measured DOF is smaller than the number of unmeasured DOF, therefore the lower part of the equation above is chosen to solve for the unknown part of the eigenvector yielding

$$\Phi_{si} = -\left(-\omega_i^2 M_{ss} + K_{ss}\right)^{-1}\left(-\omega_i^2 M_{sm} + K_{sm}\right)\Phi_{mi}. \tag{2.6}$$

Expansion Using Modal Data

One can use the modal data from the analytical model to determine the missing entries of the measured eigenvector. It is assumed that the measured master modes are a linear combination of the analytical modes:

$$\Phi_m^e = \Phi_m^a T. \tag{2.7}$$

Usually only modes that correlate well between predicted and measured data are considered. The equation above can be solved for the transformation matrix T, yielding

$$T = \left(\Phi_m^{a^T}\Phi_m^a\right)^{-1}\Phi_m^{a^T}\Phi_m^e. \tag{2.8}$$

This transformation is then used to determine Φ_s^e as

$$\Phi_s^e = \Phi_s^a T. \tag{2.9}$$

It can also be used to smooth the measured modes

$$\overline{\Phi}_m^e = \Phi_m^a T, \tag{2.10}$$

where $\overline{\Phi}_m^e$ are the "smoothed" experimental mode shapes.

2.2.4 Model Reduction

Measurements are taken at a limited number of locations on the structure and over a limited frequency range. As a result, only a small number of natural frequencies and

mode shape vectors with a reduced number of entries are determined. An alternative to expanding the measured data is to reduce the number of DOF in the analytical model. Master DOF are chosen and they are the ones retained in the reduced model while the slave DOF are discarded. There are a few techniques used to reduce the order of the model. Two are mentioned here: Guyan or Static reduction [11] and the System Equivalent Reduction Expansion Process - SEREP [12].

Guyan Reduction

The simplest way of reducing the number of DOF is to partition the mass and stiffness matrices and the forcing vector into master and slave DOF. An usual assumption is that no force is applied to the slave DOF. One can write the equations of motion as

$$\begin{bmatrix} M_{mm} & M_{ms} \\ M_{sm} & M_{ss} \end{bmatrix} \begin{Bmatrix} \ddot{x}_m \\ \ddot{x}_s \end{Bmatrix} + \begin{bmatrix} K_{mm} & K_{ms} \\ K_{sm} & K_{ss} \end{bmatrix} \begin{Bmatrix} x_m \\ x_s \end{Bmatrix} = \begin{Bmatrix} f_m \\ 0 \end{Bmatrix}. \tag{2.11}$$

In the Guyan reduction the inertia matrices M_{sm} and M_{ss} are neglected, hence the name static reduction. The second set of equations then becomes

$$K_{sm}x_m + K_{ss}x_s = 0 \Rightarrow x_s = -K_{ss}^{-1}K_{sm}x_m. \tag{2.12}$$

This expression can be used to eliminate the slave DOF so that

$$\begin{Bmatrix} x_m \\ x_s \end{Bmatrix} = \begin{bmatrix} I \\ -K_{ss}^{-1}K_{sm} \end{bmatrix} \{x_m\} = T_s x_m, \tag{2.13}$$

and therefore

$$M_r = T_s^T M T_s \qquad K_r = T_s^T K T_s, \tag{2.14}$$

where M_R and K_R are the reduced mass and stiffness matrices.

It is important to note that because some of the inertia submatrices have been neglected, the frequency response function is only exact at zero frequency.

System Equivalent Reduction Expansion Process - SEREP

The System Equivalent Reduction Expansion Process - SEREP uses the eigenvectors from the model to generate the transformation matrix between master and slave DOFs. The analytical DOF are partitioned into master and slave DOF and after some manipulation the transformation matrix is found to be

$$T_U = \left\{ \begin{array}{c} \Phi_m \\ \Phi_s \end{array} \right\} \left(\Phi_m^T \Phi_m\right)^{-1} \Phi_m^T, \tag{2.15}$$

and the reduced mass and stiffness matrix are calculated using an expression similar to equation (2.14).

The reduced mass and stiffness matrices will generate natural frequencies and mode shapes that exactly reproduce the ones of the complete model.

2.2.5 Choosing Transducer Locations

The choice of transducer locations greatly impacts the accuracy of the updating procedure. For example, a transducer will not give useful data if it is placed at a node of a mode shape of interest, or its weight may change the dynamic behavior of the structure. Two methods to help selecting the locations are considered. The first is based on classical Guyan reduction [13] and the second on the effective independence distribution vector method [14, 15].

Guyan Reduction

The Guyan reduction method described earlier was derived to reduce the number of DOF of the DMM. It makes sense to assume that the master DOF are the ones where transducers

should be placed. The procedure starts by removing rotational and non accessible DOF from the original model. Then an iterative procedure is performed. Because inertia matrices are neglected during the reduction process, the master DOF should be the ones with small k_{ii}/m_{ii}. Terms with large k_{ii}/m_{ii} can be removed safely since the inertia associated with these DOF is small. One DOF is removed at each iteration until the order of the model matches the number of desired transducers.

Effective Independence Distribution Vector Method

The effective independence distribution vector method selects measurement locations that render the mode shapes of interest as linearly independent as possible. This is accomplished by using the Fisher information matrix.

First, a Guyan reduction is performed eliminating rotational and non accessible DOF. Then the Fisher information matrix is computed by

$$A = \Phi^T \Phi, \tag{2.16}$$

where Φ is the reduced modal matrix. Then the matrix E is calculated by

$$E = \Phi A^{-1} \Phi^T. \tag{2.17}$$

The matrix E is called an idempotent matrix, meaning $E = E^2$. The useful property of this matrix is that its trace equals its rank, therefore the diagonal elements of E represent the contribution of each measurement location to the rank of E. The DOF corresponding to the smallest element in the diagonal of E is removed since it contributes the least to the independence of the vectors in Φ. The procedure of calculating Φ, A and E is then repeated until the number of remaining coordinates matches the number of transducers.

2.3 Error Localization

The purpose of error localization is to determine those substructures which seem to be the cause of the observed error. By reducing the number of parameters being updated the numerical conditioning of the problem is improved. An approach based on the sensitivity matrix is detailed here [16].

It is assumed that a matrix sensitivity equation for θ can be written in the form

$$S\theta = b \ , \tag{2.18}$$

where b is the vector of observations. To avoid the problems with least-squares solution of the above equation, a subspace approach is used.

First, the column of S that best resembles the vector b is found. Then the combination of 2, 3 columns and so on, in order to determine the best sub-basis for the representation of b. If b^P is the best representation of b in the subspace of dimension P, then the distance between b and b^P is given by

$$e^P = \frac{\|b - b^P\|}{\|b\|} \ . \tag{2.19}$$

The selection of the substructures which are likely to contain dominant modeling errors is made by analyzing the errors e^P, obtained with subspaces of increasing dimension.

2.3.1 Checking the Choice of Transducer Locations

The quality of the set of locations can be checked by one of the following criteria: modal assurance criterion, singular value decomposition, modal energy, and the Fisher information matrix.

2.4 Model Updating

In contrast with system identification, which requires no initial model and is usually done *on-line*, model updating is usually done *off-line* and requires a very good initial model. The objective is to obtain an improved model based on measured data.

Many different approaches to this problem have been developed. Most of them deal with the updating of a model obtained from a finite element analysis of the system being studied. Here we discuss the most popular approaches, and some of the methods in particular.

Finite element models are based upon the geometry of the structure and on the properties of its constituent materials. In the finite element method shape functions determine the distribution of mass and stiffness properties. Therefore, it is unwise to update individual terms of the mass and stiffness matrices. A better approach is to update the physical parameters associated with groups of finite elements. Using this approach, the updated mass and stiffness matrices can be written as [17]

$$M = M_0 + \theta_1 M_1 + \theta_2 M_2 + \ldots + \theta_N M_N, \text{ and} \tag{2.20}$$

$$K = K_0 + \theta_1 K_1 + \theta_2 K_2 + \ldots + \theta_N K_N,$$

where M_j and K_j are the mass and stiffness matrices of the j^{th} substructure, and θ_j is the j^{th} parameter being updated. The substructure matrices must be separately constructed finite element models because of the interconnections between elements.

2.4.1 Direct Techniques

Direct methods are based on modal data. A great advantage of these methods is that they do not require iteration, therefore they are fast compared to iterative techniques. Another advantage is that they reproduce the modal data exactly. Iterative methods do not possess this property.

Because the model will reproduce the measurements exactly, it is crucial that the measurements be as exact as possible. Thus direct methods are appropriate when the modal data is known with high confidence.

Direct methods require mode shape information. Because the number of analytical and experimentally measured DOF are different, model expansion or reduction is necessary. The expansion, or reduction, of the mode shapes will introduce error in the procedure.

An important characteristic of this class of methods is that the updated mass and stiffness matrices have little or no physical meaning. Because of that, physical changes to the finite elements in the original model are not possible. The connectivity of nodes is not enforced and generally the updated matrices are fully populated instead of banded. Another problem is that starting with a symmetric positive-definite system may result in asymmetric updated matrices. Recently, Minas & Inman [18] and Smith & Beattie [19] developed iterative methods to avoid the problem of connectivity. Lam & Inman [20] and Starek and Inman [21] have addressed the problems of symmetry and positive-definiteness.

2.4.2 Iterative Techniques Based on Modal Data

Iterative methods based on modal data minimize an objective function involving modal data such as eigenvalues and eigenvectors. An example of such function is the sum-squared-difference between measured and estimated modal data. The objective functions are often nonlinear functions of the parameters being updated, therefore convergence problems may occur. Because these methods rely on iterative procedures, it is necessary to evaluate the analytical model at every iteration.

Iterative methods allow any parameter of the model to be updated as well as a wide choice of objective functions. The objective functions can contain both experimental and analytical data, and they can also contain weighting factors.

There are three issues to be addressed when using this class of methods. First, the natural frequencies and mode shapes of the experimental and analytical data must be paired correctly. This can be difficult to do in systems with high modal density. The

modal assurance criterion can be used to solve this problem. If $\mathcal{M}_{i,j}$ (eq. 2.2) between experimental and analytical mode shapes is close to one, then this pair of modes is used in the updating procedure, otherwise it is discarded.

The second problem is mode shape normalization. Analytical mode shapes are often mass normalized, and measured mode shapes can also be mass normalized. However, if the mass matrices of the analytical and experimental models are different, then the scaling of the mode shapes will be inconsistent. To avoid this problem the measured mode shape may be scaled to the analytical mode shape by multiplying it by the modal scale factor (MSF) [9] given by

$$MSF_i = \frac{\phi_i^{a^T} \phi_i^e}{\phi_i^{e^T} \phi_i^e}, \tag{2.21}$$

where ϕ_i^a and ϕ_i^e are analytical and experimental mode shapes respectively. Multiplying the experimental mode shape by the corresponding MSF also solves the problem that the measured and analytical mode shapes could be 180^o out of phase.

The third issue involves damping. FEM based models typically do not account for damping. In these cases real mode shapes must be estimated from the complex mode shapes obtained experimentally. If damping is included in the analysis, then complex eigenvalues and mode shapes may be used in the updating procedure. An alternative to using complex eigenvalues is the use of natural frequencies and damping ratios. The updating algorithms require the sensitivity of the damping ratio, natural frequencies and complex mode shapes to the unknown parameters to be calculated.

A particular class of iterative methods is detailed next: the penalty function methods.

Penalty Function Methods

Penalty function methods express the modal data as a function of the unknown parameters using a truncated Taylor series expansion. The series is truncated to yield the linear approximation:

$$\delta z = S_j \delta\theta, \tag{2.22}$$

where $\delta\theta = \theta - \theta_j$, θ_j is the current value of the parameter vector, θ is the estimated vector, $\delta z = z_e - z_j$, z_e is the measured output, z_j is the current estimate of the output, S_j is the sensitivity matrix containing the first derivative of the eigenvalues and mode shapes with respect to the parameters, evaluated at the current parameter estimate θ_j. Calculating this matrix is very computationally intensive.

- Overdetermined Systems

  In the case when there are more measurements than parameters, equation (2.22) is overdetermined. The penalty function is defined as

$$J(\delta\theta) = \epsilon^T \epsilon \ , \tag{2.23}$$

  where $\epsilon = \delta z - S_j \delta\theta$ is the error in the predicted measurements based on the current parameter estimate.

  Substituting the expression for ϵ into equation (2.23) gives

$$J(\delta\theta) = \delta z^T \ \delta z - 2\delta\theta^T S_j^T \delta z + \delta\theta^T S^T S \delta\theta. \tag{2.24}$$

  Minimizing J involves differentiating J with respect to each element of θ and setting the expression equal to zero. The result is

$$\delta\theta = \left[S^T S\right]^{-1} S^T \delta z \ , \ or \quad \theta_{j+1} = \theta_j + \left[S^T S\right]^{-1} S^T \delta z \ . \tag{2.25}$$

  The problem with the expression above is that it equally weighs eigenvalues and eigenvectors. This is a problem because eigenvalues are measured with higher accuracy

than mode shapes. Also, higher frequencies are not measured as accurately as lower frequencies. Weighting matrices can be used to compensate for these experimental differences.

The weighted penalty function is written as

$$J(\delta\theta) = \epsilon^T W_{\epsilon\epsilon} \epsilon, \tag{2.26}$$

where $W_{\epsilon\epsilon}$ is a positive-definite weighting matrix, usually a diagonal matrix whose elements are given by the reciprocal of the variance of the corresponding measurement. Solving the minimization problem yields

$$\theta_{j+1} = \theta_j + \left[S_j^T W_{\epsilon\epsilon} S_j\right]^{-1} S_j W_{\epsilon\epsilon} \delta z \ . \tag{2.27}$$

Because it was assumed that the number of measurements is larger than the number of parameters, the matrix $S_j^T W_{\epsilon\epsilon} S_j$ is square and probably full rank. It will not be full rank in two cases: when either one of the parameters does not affect the measurements or when at least two combinations of parameters have the same effect on the measured data. If the matrix is ill-conditioned, a singular value decomposition is used to solve the system of equations.

- Underdetermined Systems

  In most practical cases the number of unknown parameters is larger than the number of measurements. In this case SS^T is rank deficient. The set of equations is underdetermined and equation (2.22) has an infinite number of solutions. Among all the possible solutions, the one of interest is the one which represents the smallest change in the parameters. The optimization problem is now constrained and stated as

$$J(\delta\theta) = \delta\theta^T \ \delta\theta \ , \quad s.t. \quad \delta z = S\delta\theta. \tag{2.28}$$

For the same reasons stated in the previous topic, a weighting matrix that accounts for the different variances of the parameters can be used. In this case the matrix accounts for variances of the θ_j parameters. The optimization problem is now written as

$$J(\delta\theta) = \delta\theta^T W_{\theta\theta}\delta\theta \ , \quad s.t. \quad \delta z = S\delta\theta. \tag{2.29}$$

The solution to this problem is obtained as

$$\theta_{j+1} = \theta_j + W_{\theta\theta}^{-1} S_j^T \left[S_j W_{\theta\theta}^{-1} S_j^T\right]^{-1} \delta z. \tag{2.30}$$

Another alternative is to penalize the errors between measured and predicted outputs and the parameter change relative to the initial parameter set. The optimization problem is stated as

$$J(\delta\theta) = \epsilon^T W_{\epsilon\epsilon}\epsilon^T + \delta\theta^T W_{\theta\theta}\delta\theta \ , \tag{2.31}$$

and solving for θ_{j+1} yields

$$\theta_{j+1} = \theta_j + \left[S_j^T W_{\epsilon\epsilon} S_j + W_{\theta\theta}\right]^{-1} \left\{ S_j^T W_{\epsilon\epsilon}\delta z - W_{\theta\theta}(\theta_j - \theta_0) \right\}. \tag{2.32}$$

2.4.3 Iterative Techniques Based on Frequency Domain Data

It can be difficult to extract modal data from a structure with high modal density. Using frequency response functions (FRF) data as the objective function of the identification procedure avoids the need of modal parameters. When using FRFs it is necessary to include damping in the analytical model. It is a common practice to assume viscous proportional

damping, and this is the approach taken here. However, any damping model can be used without difficulty.

Two different measures of error that can be used in the objective function: equation and output error. Taking the Laplace transform of the equations of motion one obtains the equations of motion in the frequency domain, given by

$$[-\omega^2 M + \jmath\omega C + K]x(\omega) = f(\omega) \ , \tag{2.33}$$

or

$$B(\omega)x(\omega) = f(\omega)$$

where $B(\omega) = -\omega^2 M + \jmath\omega C + K$ is the dynamic stiffness matrix. The first measure of error is called equation error and is calculated by

$$\epsilon_E = f(\omega) - B(\omega)x(\omega) \ . \tag{2.34}$$

It is usually assumed that the force has a flat spectrum over a wide frequency range, and therefore the vector $f(\omega) = 1$. Since the spectrum of the force is considered to be flat, the displacement can be replaced by the receptance $\alpha(\omega)$. One great advantage of this approach is that the error is a linear function of the entries in $B(\omega)$. Its disadvantage is the necessity to measure all modeled DOF.

The second measure of error is called output error and is given by

$$\epsilon_O = B^{-1}(\omega)f(\omega) - x(\omega) \ . \tag{2.35}$$

Once again, the spectrum of the force is assumed to be flat over a wide frequency range and the displacements are replaced by the receptances. The advantages of this approach are that it minimizes the error between the measured and the analytical data, and that it

is not necessary to measure all DOF. The disadvantage is that the error is now a highly nonlinear function of the entries in $B(\omega)$.

Independent of the approach used, it is very important that the FRF data contain enough information about the parameters being updated. When choosing the parameters to be updated, the designer should be aware that the matrices might be linear or nonlinear functions of the parameters. In the case when the matrices are nonlinear functions of the parameters a Taylor series expansion of the matrices is often used,

$$A(\theta) = A_0 + A_1\delta\theta_1 + A_2\delta\theta_2 + ... + A_p\delta\theta_p, \tag{2.36}$$

where p is the number of unknown parameters, $\delta\theta = \theta - \overline{\theta}$ and $\overline{\theta}$ is the current value of the parameter. It is also recommended, for numerical conditioning purposes, that the original or the current parameter estimates be normalized to unit.

Equation Error Methods

The simplest form of objective function minimizes the Euclidean norm of the equation error

$$J(\theta) = \|\epsilon_E\|^2 = \sum_{i=1}^{n}\sum_{k=1}^{m}\left|f_i(\omega_k) - \left(B(\theta,\omega_k)x(\omega_k)\right)_i\right|^2, \tag{2.37}$$

where n is the number of DOF and m is the number of measured frequencies. The objective function J is minimized by solving

$$\begin{bmatrix} B_1(\omega_1)x(\omega_1) & B_2(\omega_1)x(\omega_1) & \dots & B_p(\omega_1)x(\omega_1) \\ B_1(\omega_2)x(\omega_2) & B_2(\omega_2)x(\omega_2) & \dots & B_p(\omega_2)x(\omega_2) \\ \vdots & \ddots & \ddots & \vdots \\ B_1(\omega_m)x(\omega_m) & B_2(\omega_m)x(\omega_m) & \dots & B_p(\omega_m)x(\omega_m) \end{bmatrix} \begin{Bmatrix} \delta\theta_1 \\ \delta\theta_2 \\ \vdots \\ \delta\theta_p \end{Bmatrix} = \begin{Bmatrix} f(\omega_1) - B_0(\omega_1)x(\omega_1) \\ f(\omega_2) - B_0(\omega_2)x(\omega_2) \\ \vdots \\ f(\omega_m) - B_0(\omega_m)x(\omega_m) \end{Bmatrix}, \tag{2.38}$$

or

$$A\delta\theta = b, \tag{2.39}$$

where A and b are the real components of the matrices above.

Using the pseudo-inverse of A one can solve for $\delta\theta$ yielding

$$\delta\theta = \left(A^T A\right)^{-1} A^T b. \tag{2.40}$$

Often the parameters are known within some margin of confidence. This fact can be used to build a weighting matrix. A good choice for the weighting matrix is the inverse of the covariance matrix [1]. Calling this weighting matrix $W_{\theta\theta}$, J can be written as

$$J(\theta) = \|\epsilon_E\|^2 + (\theta - \theta_a)^T W_{\theta\theta} (\theta - \theta_a), \tag{2.41}$$

where θ_a is the analytical estimate of the parameter. Substituting ϵ_E by eq. (2.39) yields

$$J(\theta) = \left(A\delta\theta - b\right)^T \left(A\delta\theta - b\right) + (\theta - \theta_a)^T W_{\theta\theta} (\theta - \theta_a) \tag{2.42}$$

The solution for $\delta\theta$ then becomes

$$\delta\theta = \left[A^T A + W_{\theta\theta}\right]^{-1} \left(A^T b - W_{\theta\theta}(\theta_e - \theta_a)\right). \tag{2.43}$$

To overcome the need to measure all DOF, model reduction techniques are often used.

This method can be improved by using weighted objective functions [22] or by using instrumental variables [23, 24].

Output Error Methods

The objective function minimizes the Euclidean norm of the output error

$$J(\theta) = \|\epsilon_O\|^2 = \sum_{i=1}^{r} \sum_{j=1}^{q} \sum_{k=1}^{m} \left| \left[\alpha_m(\omega_k) - DB^{-1}(\theta, \omega_k) f\right]_{ij} \right|^2, \tag{2.44}$$

where D is the matrix of transducer locations and α_m is the measured receptance. This is a highly nonlinear function of the parameter θ. It is necessary to know B^{-1} if gradient based techniques are used in the minimization. This can be easily obtained by differentiating $B^{-1}B = I$ with respect to θ_j,

$$\frac{\partial(B^{-1}B)}{\partial\theta_j} = 0 \Rightarrow \frac{\partial B^{-1}}{\partial\theta_j} = -B^{-1}\frac{\partial B}{\partial\theta_j}B^{-1}. \tag{2.45}$$

Some numerical methods require the use of the second derivative of B^{-1}. This matrix is given by differentiating equation (2.45) with respect to θ_k to yield

$$\frac{\partial^2 B^{-1}}{\partial\theta_j\partial\theta_k} = -\Big(\frac{\partial B^{-1}}{\partial\theta_k}\frac{\partial B}{\partial\theta_j}B^{-1} + B^{-1}\frac{\partial^2 B}{\partial\theta_j\partial\theta_k}B^{-1} + B^{-1}\frac{\partial B}{\partial\theta_j}\frac{\partial B^{-1}}{\partial\theta_k}\Big). \tag{2.46}$$

The pseudo-output method [25] is an improvement over this method.

Frequency Domain Filters

Frequency domain filters can be used to avoid direct minimization. This technique recursively estimates the parameters by introducing the measured data at each frequency one at a time [26, 27, 28, 29].

2.5 Summary

Model updating techniques are intended to "fine tune" physical parameters of the model, looking for a better match between analytical and experimental results. They differ from parameter/system identification techniques in the sense that they need a good initial model of the system being studied. Because the model is corrected based on experimental data, every effort should be made to obtain clean and accurate data.

When modal data can be obtained with high accuracy one should update the model using either direct techniques or iterative techniques based on modal data. The choice

between these two approaches depends mainly on whether or not the updated model will be used to modify the physical system. Direct methods produce updated models that do not obey the FEM formulation, while iterative techniques do. Other factors are computational cost and convergence issues associated with iterative techniques.

When modal data cannot be accurately obtained from experimental data, the choice is to use iterative methods based on frequency domain data. In this case one should be aware of the high computational cost and convergence issues associated with these techniques.

The choice of parameters to be updated should be based on the results of error localization techniques. These techniques determine which substructures contribute the most from the discrepancy between analytical and experimental data.

Chapter 3

A Brief Introduction to Neural Networks

3.1 Introduction

A common engineering problem is estimating a function based on the knowledge of some example input-output pairs. This process is called supervised learning by the neural network community. Other names are *function approximation* (numerical analysis), *regression analysis* (statistics) or *system identification* (control theory). The training set (examples) is composed of pairs of values for the independent (input) and dependent (output) variables. In general the neural network will be playing the role of $\phi(\cdot)$ in

$$y = \phi(z), \tag{3.1}$$

where x is the vector of inputs and y is the vector of outputs.

The supervised learning problem can be subdivided into parametric and nonparametric models. In parametric estimation, the form of the functional relationship is known but it may contain free parameters that are determined during the learning process. Usually the free parameters of a parametric model have meaningful interpretations in terms of

the physical parameters of the system. An example of a parametric model is polynomial regression.

Nonparametric models are different in the sense that there is no *a priori* knowledge of the form of the function being estimated. The function is still modeled using an equation with many free parameters but in a way which allows the class of functions which the model can represent to be very broad. Neural networks, as well as Fourier series, spline functions, and wavelets are nonparametric models.

A neural network is a massively parallel distributed processor that has the ability of storing knowledge and making it available for use. It resembles the brain in three aspects:

- The knowledge is acquired by the network through a learning process.
- Interneuron connection strengths, known as synaptic weights, are used to store the knowledge.
- The network is capable of generalization.

The learning process is performed by a learning algorithm. The objective of this algorithm is to change the synaptic weights of the network to attain a desired design objective. Once the network has been trained it is capable of generalization. Generalization refers to the neural network producing reasonable outputs for inputs not encountered during the training process.

The characteristics of neural networks relevant to this work are

- Input-output mapping. One popular class of training algorithms is called supervised learning. The network is presented input samples and the network weights are modified so as to minimize the difference between the network output and the desired output. The training proceeds until the network reaches a state where there are no further significant changes in weights.
- Nonlinearity. A neuron is basically a nonlinear function. Therefore a neural network made up of a collection of neurons is itself nonlinear.

- Adaptivity. A neural network trained to perform a specific task in a certain environment (input-output pairs) can be easily retrained to deal with minor changes in this environment.

The most popular type of neural network for supervised learning is the multi-layer perceptron (MLP) which became prevalent in 1986 with the development of the backpropagation algorithm [30]. MLPs have feedforward connections with adaptable weights, *i.e.*, the free parameters. Training a MLP means estimating the best set of weights so that the error is minimized. This requires the solution of a nonlinear optimization problem which is usually done by means of gradient descent in the weight-space [30]. The weights are incrementally adjusted to decrease the error, and this process is iterated until the error can no longer be minimized. Nonlinear optimization is typically slow to converge and can get stuck in local minima.

There are nonparametric models that are linear in the free parameters and still provide enough flexibility to model a wide class of functions. The linear model may be written as:

$$y_i(z) = \sum_{j=1}^{p} w_{ij}\phi_j(z). \tag{3.2}$$

The advantage of using such models is that linear optimization techniques can be used. This type of model has a single global minimum whose weight-space coordinates can be calculated from a formula. An increasingly popular architecture is the radial basis function (RBF) neural network.

3.2 The Neuron

Neurons (fig. 3.1) behave as functions. They transform an input signal into an output signal $\phi(x)$. The function $\phi(\cdot)$ can assume many forms. It can be a differential equation or simple function such as the sigmoid, the threshold, the radial basis, or the linear functions (fig. 3.2).

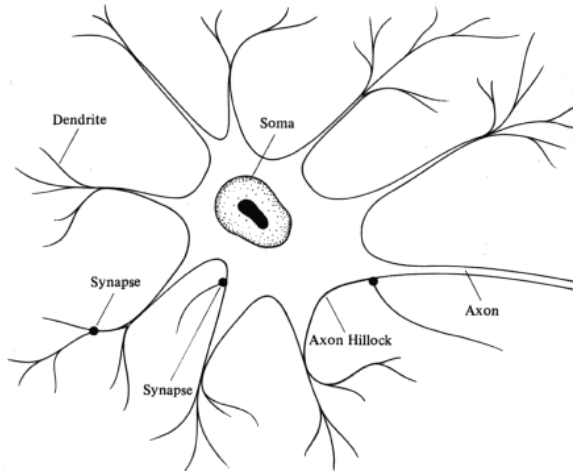

Figure 3.1: Biological neuron

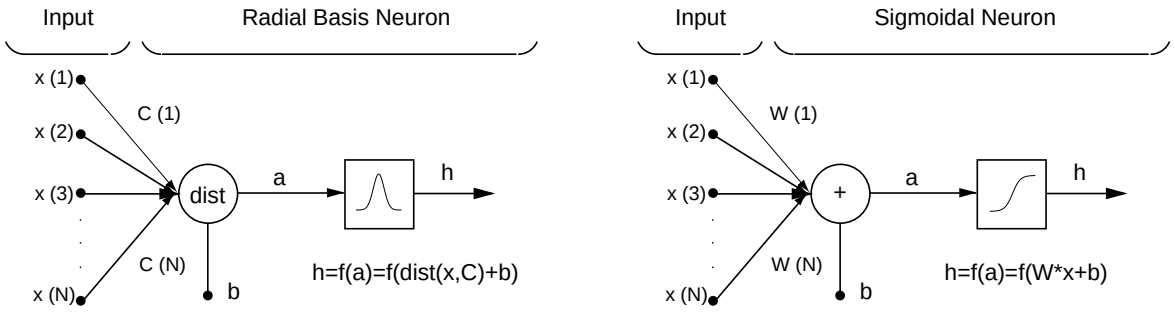

Figure 3.2: Examples of mathematical neurons

The physiological interpretation of the activation (or input) signal x and the output signal y involves electrical impulses of potential difference and their temporal summation. Activations involve small membrane pulses while output signals involve large axonal pulses.

Mathematically, the activation x represents the membrane potential or voltage difference across the neuron's surface membrane. The activation can be positive or negative. The output signal h represents the output induced by the activation x.

In general, signal functions are monotone nondecreasing. Increasing activation values can only increase the output signal or leave it unchanged. They can never decrease the signal. In practice this means signal functions have an upper bound saturation value.

An important exception to signal monotonicity are the Gaussian functions. Generalized Gaussian functions define potential basis functions as defined later in the chapter (eq. 3.24).

3.3 Network Architectures

The way in which the neurons of a neural network are arranged and connected strongly influences the learning algorithm used to train the network. The many architectures can be roughly divided into four classes:

- Single-layer feedforward networks
- Multilayer feedforward networks
- Recurrent networks
- Lattice networks

Multilayer feedforward networks are used in the present work because they are the most widely used architecture in function approximation problems. From the many existing variations in architecture and training algorithms, two are detailed here:

- Multilayer perceptron networks trained by the error backpropagation algorithm, and
- Radial basis function networks.

3.3.1 Multilayer Perceptron Networks

Multilayer perceptron networks have been applied successfully to many different problems since the advent of the error backpropagation learning algorithm [30]. This network architecture and the corresponding learning algorithm can be viewed as a generalization of the popular least-mean-square (LMS) algorithm.

A multilayer perceptron network consists of an input layer, one or more hidden layers of computation nodes, and an output layer of computation nodes. The input signal propagates through the network in a forward direction, layer by layer.

The error backpropagation learning algorithm consists of two phases. The first is usually referred to as the presentation phase or forward pass and the second as the backpropagation phase or backward pass. In the presentation phase an input vector x is presented to the network resulting in an output y at the output layer. During this phase the synaptic weights are all fixed. In the backpropagation phase the weights are adjusted based on the error between the actual and desired outputs.

The Presentation Phase

For simplicity we analyze a network with just one hidden layer. The following is a list of symbols used in the remainder of this chapter.

- NI : number of neurons in the input layer.
- NH : number of neurons in the hidden layer.
- NO : number of neurons in the output layer.
- x : input vector.
- h^H : input for the hidden layer.
- h^O : input for the output layer.
- y^H : output of the hidden layer.
- y : output of the network.
- w_{ji} : matrix $NH \times NI$ of synaptic weights connecting the input and hidden layers.
- w_{kj} : matrix $NO \times NH$ of synaptic weights connecting the hidden and output layers.
- b : bias, or threshold vector.
- $\phi(\cdot)$: the nonlinear function performed by the neuron.

- $i = [1 : NI]$: a neuron in the input layer.
- $j = [1 : NH]$: a neuron in the hidden layer.
- $k = [1 : NO]$: a neuron in the output layer.

Once the input vector is presented to the input layer we can calculate the input to the hidden layer as

$$h_j^H = b_j + \sum_{i=1}^{NI} w_{ji} x_i. \tag{3.3}$$

Each neuron of the hidden layer takes its input h_j^H and uses it as the argument for a function and produces an output given by

$$y_j^H = \phi(h_j^H). \tag{3.4}$$

Now the input to the neurons of the output layer are calculated as

$$h_k^O = b_k + \sum_{j=1}^{NH} w_{kj} y_j^H, \tag{3.5}$$

and the network output is then given by:

$$y_k = \phi(h_k^O) \ . \tag{3.6}$$

The Error Back-propagation Learning Algorithm

An error vector can be defined as being the difference between the network output and the desired output value.

$$e_k = d_k - y_k \ . \tag{3.7}$$

Based on the error vector we can calculate the sum of squared errors vector as

$$\mathcal{E} = \frac{1}{2}\sum_{k=1}^{NO} e_k^2. \tag{3.8}$$

This is the cost function to be minimized during the learning process. The sum-squared-error $\mathcal{E}$ is a function of all the variables of the network.

Using the chain rule we can calculate the gradient of the error with respect to the weight matrix connecting the hidden layer to the output layer as follows

$$\frac{\partial \mathcal{E}}{\partial w_{kj}} = \frac{\partial \varepsilon}{\partial e_k}\frac{\partial e_k}{\partial y_k}\frac{\partial y_k}{\partial h_k^O}\frac{\partial h_k^O}{\partial w_{kj}} \ . \tag{3.9}$$

Computing each term of this expression yields

$$\frac{\partial \mathcal{E}}{\partial e_k} = e_k$$

$$\frac{\partial e_k}{\partial y_k} = -1$$

$$\frac{\partial y_k}{\partial h_k^O} = \phi_k'(h_k^O)$$

$$\frac{\partial h_k^O}{\partial w_{kj}} = y_j^H$$

Combining the expressions above results in

$$\frac{\partial \varepsilon}{\partial w_{kj}} = -e_k\phi_k'(h_k^O)y_j^H \ . \tag{3.10}$$

The correction Δw_{kj} applied to the weight matrix connecting the hidden layer to the output layer is

$$\Delta w_{kj} = -\eta\frac{\partial \mathcal{E}}{w_{kj}} = \eta e_k\phi_k'(h_k^O)y_j^H, \tag{3.11}$$

where η is a constant known as the step-size or the learning rate. The equation above can be rewritten as

$$\Delta w_{kj} = \eta \delta_k y_j^H, \tag{3.12}$$

where $\delta_k = e_k \phi_k'(h_k^O)$ is the local gradient term. To update the weights connecting the input layer to the hidden layer we need to repeat the procedure above,

$$\frac{\partial \mathcal{E}}{\partial w_{ji}} = \frac{\partial \mathcal{E}}{\partial e_k} \frac{\partial e_k}{\partial y_k} \frac{\partial y_k}{\partial h_k^O} \frac{\partial h_k^O}{\partial y_j^H} \frac{\partial y_j^H}{\partial h_j^H} \frac{\partial h_j^H}{\partial w_{ji}} \ . \tag{3.13}$$

After calculating each of the terms above, the correction to the weight matrix is written as

$$\Delta w_{ij} = -\eta \delta_j x_i, \tag{3.14}$$

where $\delta_j = \phi_j'(h_j^H) \sum_{k=1}^{NO} \delta_k w_{kj}$. In general, the correction term is calculated by

$$\Delta w_{lm} = \eta \delta_m x_l = \text{learning rate} \times \text{local gradient} \times \text{input to the layer.} \tag{3.15}$$

Neuronal Functions

In general the functions $\phi(\cdot)$ of the hidden layer are different from the ones in the output layer. The linear function is normally used in the output layer while in the hidden layer the sigmoidal class of functions is the preferred choice. If the functions in the hidden and output layers are linear we have the LMS network.

The sigmoidal function is defined as

$$y = \frac{1}{1 + e^{-x}} \ ; \qquad -\infty < x < +\infty \tag{3.16}$$

and

$$\frac{\partial y}{\partial x} = \frac{e^{-x}}{(1+e^{-x})^2} = y \times (1-y) \ . \tag{3.17}$$

This special form of the derivative makes this function very attractive for computer implementation. A variation of the sigmoidal function is

$$y = a \tanh(bx) = a \times \frac{1-e^{-bx}}{1+e^{-bx}} = \frac{2a}{1+e^{-bx}} - a \ . \tag{3.18}$$

Common choices for a and b are $a = 1.716$, and $b = 2/3$.

Comments

The multilayer perceptron network has been proven to be a universal approximator, meaning it can approximate any continuous multivariate function to any degree of accuracy provided that enough hidden neurons exist. [31, 32]

As any gradient-descent based algorithm, the error backpropagation algorithm suffers from slow convergence and high probability of converging to local minima. There are ways to improve the algorithm by using adaptive learning rates and momentum terms.

Other training algorithms have been developed for the multilayer perceptron networks. The most popular classes of algorithms are the ones based on conjugate-gradient techniques and on Newton's method.

3.3.2 Radial Basis Function Networks

Radial basis functions were first used in the context of neural networks by Broomhead & Lowe [33] in 1988. They are special types of linear models (eq. 3.2) where the argument z has the form

$$z = (x - c_j)^T R_j^{-1} (x - c_j), \tag{3.19}$$

and considering the Euclidean metric, the argument z is written as

$$z = \frac{\|x - c_j\|^2}{\sigma_j^2}, \tag{3.20}$$

where σ_j is a constant. The output of the network is given by equation (3.2). Here, y is the vector denoting the output of the network, x is the input vector to the network, R is a matrix defining the metric, and ϕ is the radial basis function. The vectors c_j, which belong to the input space, are called *centers*. For the network to be a linear function of the free parameters w_{ij}, the number of hidden neurons (p), their position (c_j), and the metric (R or σ_j) have to be constant. Often a constant term, called bias or threshold, is added to the weighted sum (eq. 3.2) so that the function now becomes

$$y_k(z) = w_{0k} + \sum_{j=1}^{NH} w_{kj}\phi_j(z). \tag{3.21}$$

Park & Sandberg [34] showed that RBF networks are suitable for nonparametric regression and that under certain conditions they are universal approximators. The conditions are

- The hidden layer must contain enough neurons, and
- The function $\phi(\cdot)$ must be continuous, bounded and monotonic in $[0, \infty)$.

Common choices for $\phi(\cdot)$ are the Gaussian function of the form

$$\phi(z) = e^{-z}, \tag{3.22}$$

and the Cauchy function of the form

$$\phi(z) = \frac{1}{1+z}. \tag{3.23}$$

The Gaussian and Cauchy functions then become

$$\phi_j(x) = e^{-\frac{\|x-c_j\|^2}{\sigma_j^2}}, \text{ and} \tag{3.24}$$

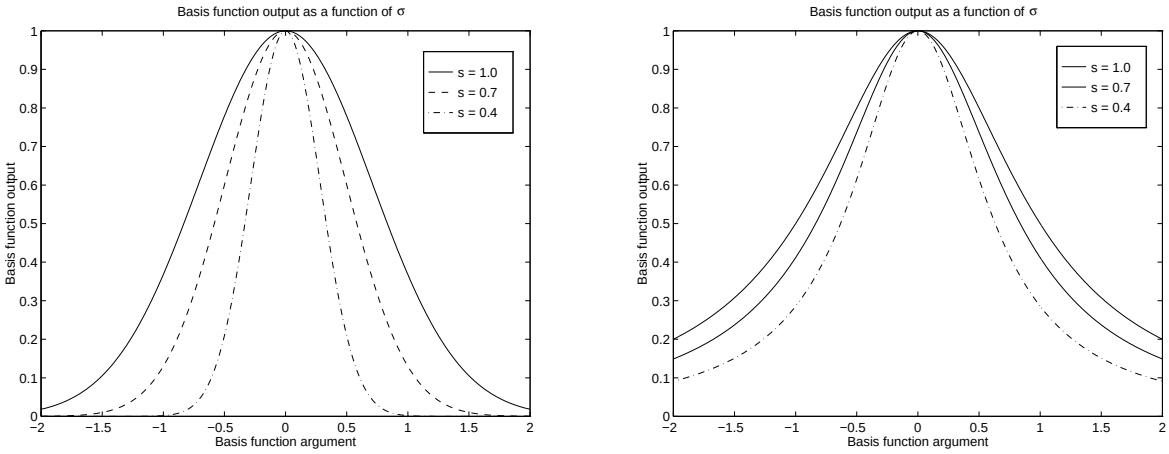

Figure 3.3: Radial basis functions.

$$\phi_j(x) = \frac{1}{1 + -\frac{\|x - c_j\|^2}{\sigma_j^2}}. \tag{3.25}$$

Examples of a Gaussian and of a Cauchy functions for $c = 0$ are shown in figure (3.3). As seen from the figure, the parameter σ (the spread constant) determines how the basis function responds to an input. For large values of σ the range of activation of the basis function is broader. The center c determines where the maximum output will occur.

A Radial Basis Function (RBF) network has the architecture shown in figure (3.4). When the input vector x is presented to the network its distance to each of the centers c_j is measured and each neuron in the hidden layer will output a number between 0 and 1 according to the proximity of the input vector to the neuron's center. This output is weighted by the connections between the hidden and output layers to yield the network output y. Neurons with centers far from the input vector will have output close to zero. This small output will have only a small effect on the output neurons. In contrast, neurons with centers close to the input vector x will output values close to one, and will influence the final output, y.

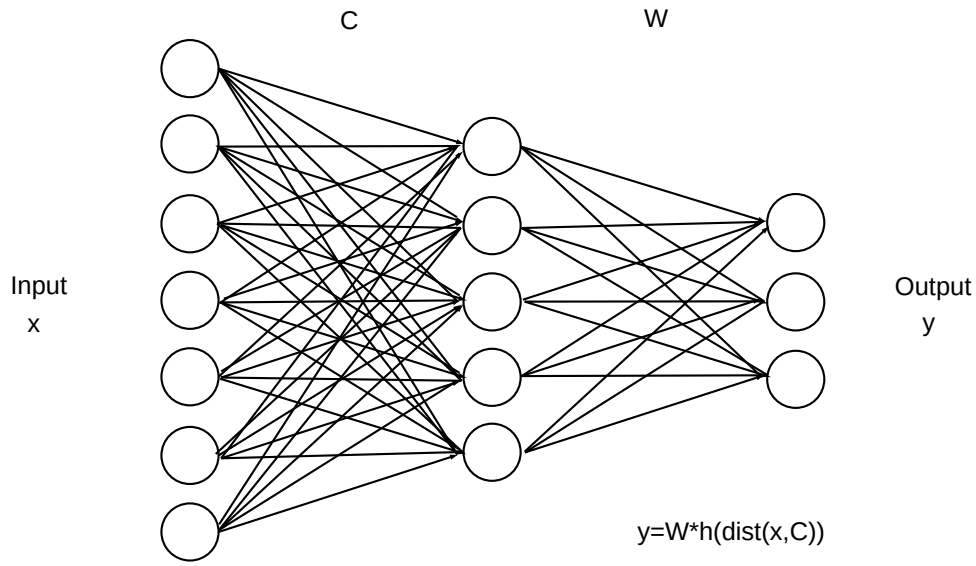

Figure 3.4: Network architecture

The Learning Process

A RBF network is defined by equations (3.19, 3.21, and 3.22 or 3.23). Figure (3.4) is a graphic representation of a RBF network. Each radial basis function defines a hyperspherical receptive field in $\Re^n$. The i^{th} neuron outputs a signal that is close to one for sample activation vectors x that fall in its receptive field, and a smaller value otherwise.

The training of a RBF network comprises the following steps:

- Selection of the centers : There are different ways of selecting the centers. The most used are: to use the input vectors as centers; to randomly choose vectors from the input set as centers; to use the K-means algorithm [35, 36] to cluster the input vectors and select the centers based on the number of clusters and their variance.

- Choice of σ : σ can also be chosen in a variety of ways. It is a measure of the variance of the input set. One way of calculating σ is to use the P-nearest neighbor algorithm:

$$\sigma_j = \frac{1}{P}\sum_{k=1}^{P}(\|c_j - c_k\|^2)^{1/2}, \tag{3.26}$$

where c_k is the P-nearest neighbor of σ_j. P is determined heuristically.

- Calculation of the weights : The output of the network is given by

$$y = W\phi(x, c, \sigma) \Rightarrow W = y\phi^T(\phi\phi^T)^{-1} \tag{3.27}$$

The error of the network is zero for the training input vectors, and the quality of the generalization depends on how well the training set represents the solution space.

3.4 Summary

Neural networks are nonparametric models able to represent any function with arbitrary accuracy, provided enough neurons exist. Supervised learning is the preferred algorithm to train a neural network for function approximation and from the many existing architectures, feedforward and radial basis function networks are the most widely used.

The backpropagation learning algorithm is the most popular, producing global generalization and depending nonlinearly on the connection weights. This results in long training periods and possible convergence to local minima. Radial basis function networks produce local approximations and depend linearly on the connection weights. When choosing between one of these two networks one should consider the characteristics of the function being approximated, the available amount of training data, and the amount of time required to train the network to produce acceptable results.

Chapter 4

Model Updating Using Neural Networks

4.1 Introduction

The technique being proposed here fills a gap in the existing literature by being the first to address the model updating problem of structures that have parameters that change with time. This chapter presents a technique, that unlike existing techniques, allows for its implementation as part of an adaptive control scheme, updating the model as the structure changes, and for updating weakly nonlinear systems.

Model updating belongs to a larger class of problems called inverse problems. In the field of linear vibrations, the direct problem is to determine the response of the structure given the coefficient matrices M, C, and K, while the inverse problem is to determine these coefficient matrices from the response of the structure. Since the solutions of the inverse problem are not unique [1] the task is to to determine which solution is the most appropriate to the problem at hand. It is usually desirable to seek the solution closest to the original model by minimizing not only the difference between predicted and measured responses, but also the distance between the nominal and updated parameters. Existing

techniques require the measurement of all degrees of freedom of the model, and since this is infeasible, either mode shape expansion (section 2.2.3) or model reduction (section 2.2.4) must be used. Most of these methods use minimization techniques to search for the solution, solving the direct problem many times during the procedure but not making any use of these intermediary solutions. Here is a key difference between the proposed technique and existing techniques: the solutions of the direct problem that are discarded by existing techniques are used by the neural network to build a map between frequency response functions and model parameters. If the model parameters change later in time, this map can be used to update these parameters without solving the direct problem again. The updated parameters generate an updated model consistent with the experimental data and can be used to adapt the control scheme acting on the system. Frequency response functions are chosen over modal data as the source of system information because they are not restricted to linear systems. Furthermore, it is difficult to identify modal parameters in structures with high modal density.

In section two, radial basis function networks are reviewed focusing on the model updating problem. Error measures are defined to study the performance of network and to allow the comparison of different study cases. Important issues are the distance measure used by the neural network and the choice of input data used to train the network. The third section discusses the normalization of the input data, a problem closely related to the one of measuring the distances between vectors. A general rule for choosing the spread constants and a gradient-descent-based procedure to adjust these constants in order to minimize the generalization error of the network are presented.

The fourth and fifth sections address the choice of data used to train the network. The FRF sensitivity to model parameter changes is analyzed in order to determine the most sensitive frequency points, the ones used in the updating procedure. Then a new way of using frequency domain data to update a model is presented. It uses the integral of the real and imaginary parts of the FRF to train the network. The effect of noise in the FRF over the network performance is analyzed illustrating the utility and improvement offered

by the proposed choice.

Sometimes it is desirable to increase the accuracy of the model over a certain frequency range. This is possible by means of weighting the input data. Section six presents this technique and illustrates its effect on the response of a cantilevered beam. Sections seven and eight analyze the performance of the network as a function of the number of model parameters being updated and of the number of different frequency responses of the system used to train the network. As a test case, the truck suspension problem presented by Szewczyk & Hajela [4] is used to illustrate the effects of varying these two variables. Section nine discusses the model updating of weakly nonlinear systems, concentrating on the choice of input data used to train the network.

The computational load required to train the network and to produce parameter estimates is presented in section ten. This section shows how quick it is to calculate the parameter estimates, once the network has been trained. This chapter ends with a review of the key aspects of the proposed technique.

4.2 Radial Basis Function Neural Networks - Review

The neural network is a mapping between its inputs and outputs based on a number of known sample input-output pairs. In general, the more samples available to train the network, the more accurate the representation of the real mapping will be. These samples are obtained by solving the direct problem N_t times. The collection of frequency response data obtained from the solution of the direct problem is put in an input matrix X and organized such that each column corresponds to a solution of the direct problem, and each row contains a piece of information from the frequency response. The collection of model parameters used to generate the frequency responses stored in X are organized in a matrix Y such that each column corresponds to a set of model parameters used to generate a FRF and each row corresponds to a model parameter. The columns of X and Y should be paired, that is, the i^{th} column of Y should contain the model parameters used to generate

the response stored in the i^{th} column of X. For example, for a one degree-of-freedom system that needs updating of the damping and stiffness parameters, the matrices X and Y could be

$$
\begin{aligned}
X &= \begin{bmatrix} H_1(\omega_1) & H_2(\omega_1)\cdots & H_N(\omega_1) \\ H_1(\omega_2) & H_2(\omega_2)\cdots & H_N(\omega_2) \\ H_1(\omega_3) & H_2(\omega_3)\cdots & H_N(\omega_3) \end{bmatrix} \\
Y &= \begin{bmatrix} k_1 & k_2 & \cdots & k_N \\ c_1 & c_2 & \cdots & c_N \end{bmatrix}
\end{aligned} \tag{4.1}
$$

where $H(\omega_i)$ is a frequency response function evaluated at ω_i. The number of rows of X is the number of neurons in the input layer (NI) while the number of rows of Y is the number of neurons in the output layer (NO). The number of columns of X and Y is the number N_t of sample pairs used to train the network.

As mentioned in chapter three there are many ways to select the centers of a RBFNN. In this dissertation the centers are chosen to be the input vectors used to train the network, that is, the matrix of centers C is identical to the input matrix X.

Once the network has been trained its accuracy is verified by solving the direct problem N times, with a new randomly generated set of model parameters. The responses corresponding to these N set of model parameters are input to the network and the corresponding estimates are compared to the model parameters used to generate the input data, resulting in a performance measure of the network. This performance measure is used to decide if enough data (N_t) has been used to train the network. The network need to be retrained with a larger amount of data (larger N_t) if the performance is low.

Training a RBFNN comprises the following steps:

i. The distance between each column of the input matrix X and each neuron in the hidden layer (represented by its center) is measured, resulting in the matrix D of distances. The element $d_{j,i}$ of this matrix is the distance between the i^{th} column of

X and the j^{th} neuron in the hidden layer.

ii. The output of the hidden layer is calculated by $h_{j,i} = e^{-d_{j,i}}$ The elements of H, which measure how much the i^{th} input vector resembles the center of the j^{th} neuron, range between zero and one. If these vectors are identical, the distance $d_{j,i}$ is zero, resulting in $h_{j,i} = 1$, a perfect match. The distance $d_{j,i}$ decays to a value close to zero for vectors that have little resemblance to the centers of the hidden layer.

Based on the above, one can also interpret the network as a classifier, where the neurons in the hidden layer output a number close to one for a close match, and a smaller number otherwise.

iii. The network output Y is calculated by a linear combination of the components of the matrix H,

$$Y = WH \ , \tag{4.2}$$

where W is the weight matrix connecting the hidden layer to the output layer.

Solving the equation above for W yields

$$W = YH^{\dagger} \ , \tag{4.3}$$

where $H^{\dagger}$ is the pseudo-inverse of H, since, in the general case H is not square, *i.e.*, there can be more training samples than neurons in the hidden layer ($N \geq NH$). The matrix W combines the rows of H linearly to produce the output matrix Y. Once it has been trained, the network can be used to estimate model parameters based on new input vectors.

iv. The network generalization characteristics should be verified before implementing it. Generalization is the behavior of the network output when presented with an input

vector not used to train the network. The definition of generalization is similar to that of interpolation: it is desirable that the network will produce reasonable estimates when presented with input vectors not "seen" before. The data used to test the network should have the same characteristics as the data used to train the network, so a fair analysis can be conducted. There are many different ways to measure the network generalization characteristics. The measures used in this work are

a) Mean absolute error, defined as

$$e_p^{avg} = \sum_{k=1}^{N_t} \left| \frac{y_{p,k}^a - y_{p,k}^d}{y_{p,k}^d} \right| , \qquad k = 1, 2, \ldots N , \tag{4.4}$$

where e_p^{avg} is the mean absolute error in the estimation of the p^{th} parameter, y^a is the actual output of the network, and y^d is the desired output.

This measure indicates how much error, on average, can be expected from the network when estimating model parameters.

b) Maximum error, defined as

$$e_p^{max} = \max_k \left| \frac{y_{p,k}^a - y_{p,k}^d}{y_{p,k}^d} \right| , \qquad k = 1, 2, \ldots N . \tag{4.5}$$

This measure represents a worst-case scenario.

c) Sum-squared-error, defined as

$$\mathcal{E}_p = \frac{1}{2N} \sum_{k=1}^{N_t} \left(y_{p,k}^d - y_{p,k}^a \right)^2 , \qquad k = 1, 2, \ldots N . \tag{4.6}$$

This is an error measure with continuous derivatives that can be used to fine-adjust parameters of the network.

Various issues have to be addressed when training a network. The most relevant to this work are

i. The choice of training data is crucial. The mapping constructed by the network reflects the data presented to it, therefore, it is essential that the training data represents the mapping being approximated. The training data should represent the largest possible range of input data, however, in view of equation (4.3), care should be taken to avoid vectors that are very close to each other since this will result in a numerical ill-conditioning of $H^{\dagger}$. The technique being proposed assumes that the engineer knows the boundaries within which the model parameters vary. It was found during this research that a random generation of the model parameters used to train the network yields the best results. The training data should reflect the probability distribution of the model parameters being updated, if this information is known. This is often the case when material properties, dimensions of the structure or environmental conditions are know to vary about a mean value with a certain variance.

   The type of data used for training is also very important. It should be sensitive to changes in the model parameters, and, if possible, be such that ambiguities are avoided. For example, input data that lead to the case where two similar input vectors result in very different outputs, or where two very different input vectors result in similar outputs, should be avoided.

ii. Measuring the distance between two vectors proved to be one of the most important issues in developing this technique. The solution to this problem involves the normalization and weighting of the different positions of the input vector, as discussed later in this chapter. The difficulty arises when the different rows of the input matrix have very different sensitivities to changes in the model parameters, or when they differ in orders of magnitude. This problem can be illustrated using the Euclidean norm

$$d = \Big(\sum_{m=1}^{NI} \big(x_m - c_m\big)^2 \Big)^{1/2} \ . \tag{4.7}$$

If x and c are, for example,

$$x = \left\{ \begin{array}{c} x_1 \pm \sigma_1 \\ x_2 \pm \sigma_2 \end{array} \right\} \quad \text{and} \quad c = \left\{ \begin{array}{c} c_1 \\ c_2 \end{array} \right\} \ , \tag{4.8}$$

where x_1 and x_2 are the average values of the first and second positions of x and σ_1 and σ_2 are the expected variations of those positions, then distance is given by

$$d = \Big(\big(x_1 - c_1 \pm \sigma_1\big)^2 + \big(x_2 - c_2 \pm \sigma_2\big)^2 \Big)^{1/2} \ . \tag{4.9}$$

If $x_1 \gg c_1$ or if $\sigma_1 \gg \sigma_2$, the distance of the first position will dominate the distance measure. This should be avoided since it is assumed that the whole input vector is relevant to the estimation process.

iii. The order of magnitude of the elements of the weight matrix should be checked. High orders of magnitude are a symptom of either improper distance measurement or choice of input data. Either case can result in a matrix H such that its entries are all numerically very similar to each other, causing the elements of W to have high orders of magnitude in order to enhance the difference among the elements of H. High orders of magnitude in W yield poor generalization characteristics, since the output of the hidden layer is greatly enhanced by W, and any small deviation in H will greatly affect Y.

iv. The choice of output values also affects the performance of the network. The output vector should be such that its elements have approximately the same order of magnitude, to avoid numerical ill-conditioning of W. One way to avoid this problem is

to use the logarithm of the model parameters as the output of the network. Another way is to use multiplying factors of the updated parameter, for example, 0.8 k, 1.0 k, and 1.2 k.

Most of the remarks above can be interpreted as being common sense, but many times are not taken into account, leading to failure when using a neural network as a function approximator or classifier.

4.3 Normalization and choice of the spread constants

It is easy to see that rows of the input matrix X with large magnitude variation will dominate the value of the distance, making inputs with small magnitude differences irrelevant to the estimation process. To overcome this problem, the input data is normalized so that each entry of the input vector falls within prescribed bounds. A common choice is the interval [0, 1], and for this choice the transformation is linear of the form

$$x_m^{new} = \frac{x_m^{old} - \min(x_m^{old})}{\max(x_m^{old}) - \min(x_m^{old})} \ , \tag{4.10}$$

where x_m is a row of the input matrix.

The normalization given above does not solve all the problems. For example, consider the two normalized rows shown in figure (4.1). It can be clearly seen that the two lines above have very different variances. Further scaling is necessary to improve the distance measure between the two vectors. This is accomplished by using the vector σ of spread constants, one associated with each row of the input matrix. The choice of σ is arbitrary and has the effect previously shown in figure (3.3). It was found that for the model updating problem an appropriate way to choose the vector σ is to make it dependent on the standard deviation of the rows of the normalized set of training vectors. The vector σ is then calculated by

$$\sigma_m = \gamma \ std(x_m) \tag{4.11}$$

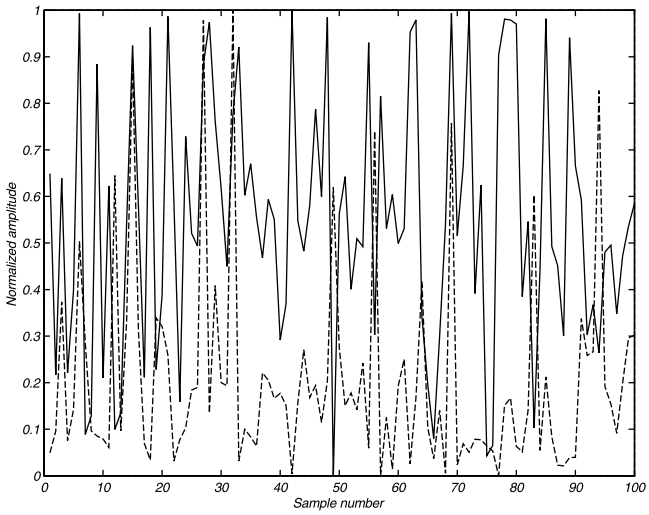

Figure 4.1: Sample vector to illustrate the different variances of two rows of the normalized input matrix. The first row corresponds to the solid line and the second to the dashed line.

where $std(\cdot)$ is the standard deviation and γ is a scaling constant multiplying it, being the value $\gamma = 1$ a very good initial guess.

When training a neural network one is always interested in minimizing the generalization error, *i.e.*, the resulting output error when an input vector not used to train the network is presented to the network. Let's assume that N input vectors are presented to the network. Each input has an output error vector associated with it, defined as

$$e_i = y_i^d - y_i^a \ , \tag{4.12}$$

where y_i^d is the vector of desired outputs and y_i^a is the actual output vector when the vector x_i is presented to the network. It is desirable to minimize the squared sum of all error vectors. Minimizing this scalar is synonymous to improving the generalization characteristics of the network. Therefore, it is desirable to find the constant γ such that the sum-squared-error

$$\mathcal{E} = \frac{1}{2N} \sum_{i=1}^{N} \sum_{q=1}^{NO} e_q^2(i) \tag{4.13}$$

is minimized. Any minimization technique can be used to find the optimal value of γ. Here, a gradient-descent formulation is presented. The gradient of $\mathcal{E}$ with respect to γ is given by

$$\frac{\partial \mathcal{E}}{\partial \gamma} = \sum_{i=1}^{N} \frac{\partial \mathcal{E}}{\partial e_j(i)} \frac{\partial e_j(i)}{\partial y_k^a(i)} \frac{\partial y_k^a(i)}{\partial \gamma} \ , \tag{4.14}$$

where $y_k^a = \sum_m w_{k,m} h_m^a$ and $h_m^a = e^{-\gamma d_m}$. Analyzing term by term, equation (4.14) yields

$$\frac{\partial \mathcal{E}}{\partial e_j(i)} = \frac{1}{N} e_j(i) \tag{4.15}$$

$$\frac{\partial e_j(i)}{\partial y_k^a(i)} = -\delta_{jk} \tag{4.16}$$

$$\frac{\partial y_k^a(i)}{\partial \gamma} = \sum_{p=1}^{N_t} \frac{\partial w_{k,p}}{\partial \gamma} h_p^a(i) + \sum_{p=1}^{N_t} w_{k,p} \frac{\partial h_p^a(i)}{\partial \gamma} \tag{4.17}$$

The first term of the above expression can be calculated by using eq. (4.3)

$$\frac{\partial W}{\partial \gamma} = \frac{\partial Y H^{-1}}{\partial \gamma} = Y \frac{\partial H^{-1}}{\partial \gamma} = -Y H^{-1} \frac{\partial H}{\partial \gamma} H^{-1} = -W \frac{\partial H}{\partial \gamma} H^{-1} \ , \tag{4.18}$$

where H is the matrix output of the hidden layer when presented with the training vectors, and use of eq. (2.45) has been made. Writing the expression above in summation form and calculating the derivative of H with respect to γ yields

$$\frac{\partial \sum_{p=1}^{N_t} w_{k,p}}{\partial \gamma} = \sum_{p=1}^{N_t} \sum_{n=1}^{N} w_{k,p} (d_{p,n} h_{p,n}) h_{n,l}^I \ , \qquad \begin{matrix} k = 1, 2, \ldots NO \\ l = 1, 2, \ldots N_t \end{matrix} \ , \tag{4.19}$$

where $h^I_{n,k}$ is the (n,k) element of the matrix H^{-1}.

The second term of eq. (4.17) is given by

$$\frac{\partial h^a_k(i)}{\partial \gamma} = -(d^a_k(i)h^a_k(i)) \tag{4.20}$$

Collecting the results from eqs. (4.19) and (4.20), eq. (4.17) can be written as:

$$\frac{\partial y^a_k(i)}{\partial \gamma} = \sum_{p=1}^{N_t} w_{k,p} \left(\sum_{n=1}^{N_t} \sum_{l=1}^{N_t} (d_{p,n}h_{p,n})h^I_{n,l}h^a_l(i) - (d^a_p(i)h^a_p(i)) \right) , \tag{4.21}$$

or, in matrix form:

$$\frac{\partial y^a(i)}{\partial \gamma} = W(\overline{H}H^{-1}h^a(i) - \overline{h}^a(i)) , \tag{4.22}$$

where $\overline{H}_{p,n} = d_{p,n}h_{p,n}$, and $\overline{h}^a_p(i) = d^a_p(i)h^a_p(i)$.

Now, collecting the results obtained from eqs. (4.15), (4.16), and (4.21), one can write $\frac{\partial \mathcal{E}}{\partial \gamma}$ as

$$\frac{\partial \mathcal{E}}{\partial \gamma} = -\frac{1}{N} \sum_{i=1}^{N} e_j(i) \sum_{p=1}^{N_t} w_{k,p} \left(\sum_{n=1}^{N_t} \sum_{l=1}^{N_t} (d_{p,n}h_{p,n})h^I_{n,l}h^a_l(i) - (d^a_p(i)h^a_p(i)) \right) , \tag{4.23}$$

or in matrix form as

$$\frac{\partial \mathcal{E}}{\partial \gamma} = -\frac{1}{N} \sum_{i=1}^{N} e^T_j(i)(W(\overline{H}H^{-1}h^a(i) - \overline{h}^a(i))) , \tag{4.24}$$

The change in the γ parameter should be in the direction opposite to the gradient $\frac{\partial \mathcal{E}}{\partial \gamma}$, therefore, the iteration scheme for minimizing $\mathcal{E}$ is

$$\gamma^{k+1} = \gamma^k + \eta \frac{\partial \mathcal{E}}{\partial \gamma} \tag{4.25}$$

where η is the step size, usually a small number between 10^{-1} and 10^{-2}.

The estimation problem has to be solved at each iteration. This can be efficiently done by recognizing that

$$e^{-\gamma d} = \left(e^{-d}\right)^{\gamma} \ . \tag{4.26}$$

Therefore it is not necessary to measure the distance matrix at each iteration. One need only perform the following operation:

$$d(\gamma^{k+1}) = \left(d(\gamma = 1)\right)^{\gamma^{k+1}} \ . \tag{4.27}$$

Equations (4.24), (4.25), and (4.27) represent the necessary steps to efficiently adjust the constant γ in a gradient-descent fashion in order to minimize the sum-squared-error of the network over the validation set.

4.4 Sensitivity of FRFs with respect to parameter changes

Frequency response functions (FRFs) contain a large amount of redundant information since there are many more points in a FRF than parameters in the model. It is therefore necessary to choose which points of the FRF should be used in updating the model. Intuitively one expects that these points should be the ones that are most sensitive to changes in the model parameters. To determine which points are most sensitive, one should first write the expression for the frequency response matrix. In state-space notation, the system is described by a set of first-order differential equations

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t) \ , \end{aligned} \tag{4.28}$$

where A is the system's state-matrix, B is the matrix of inputs, C is the matrix of measurements, D is the feed-through matrix, $x(t)$ is the state-vector, $u(t)$ is the vector of control forces, and $y(t)$ is the vector of measurements. Taking the Laplace transform of the equation above yields a set of algebraic equations in the Laplace variable s. Assuming zero initial conditions, equation (4.28) becomes

$$\begin{aligned} sX(s) &= AX(s) + BU(s) \\ Y(s) &= CX(s) + DU(s) \ . \end{aligned} \tag{4.29}$$

The first equation can be solved for $X(s)$ yielding

$$X(s) = (sI - A)^{-1} BU(s) \ . \tag{4.30}$$

This expression for $X(s)$ can be substituted into eq. (4.29) to yield

$$Y(s) = \Big[C(sI - A)^{-1} B + D\Big] U(s) \ , \tag{4.31}$$

or, in condensed form,

$$Y(s) = H(s)U(s) \ . \tag{4.32}$$

Assuming the force vector is periodic, with frequency ω, the equation above can be rewritten as

$$Y(\omega) = H(\omega)U(\omega) \ , \text{ or } Y_i(\omega) = \sum_{j=1}^{N_t} H_{i,j}(\omega) U_j(\omega) \ , \tag{4.33}$$

where $H(\omega)$ is the matrix of frequency response functions.

The sensitivity of $H(\omega)$ with respect to a variation of the generic parameter α is determined next. The derivative of $H(\omega)$ with respect to α is

$$\frac{\partial H(\omega)}{\partial \alpha} = \frac{\partial \Big[C(\jmath\omega I - A)^{-1} B + D \Big]}{\partial \alpha} = C \frac{\partial (\jmath\omega I - A)^{-1}}{\partial \alpha} B + C(\jmath\omega I - A)^{-1} \frac{\partial B}{\partial \alpha} \tag{4.34}$$

making the reasonable assumption that C, and D are not functions of the model parameter α. The derivative of the inverse of a matrix can be easily calculated using equation (2.45), yielding

$$\begin{aligned} \frac{\partial H(\omega)}{\partial \alpha} &= -C(\jmath\omega I - A)^{-1} \frac{\partial (\jmath\omega I - A)}{\partial \alpha} (\jmath\omega I - A)^{-1} B + C(\jmath\omega I - A)^{-1} \frac{\partial B}{\partial \alpha} \\ &= C(\jmath\omega I - A)^{-1} \left(\frac{\partial A}{\partial \alpha} (\jmath\omega I - A)^{-1} B + \frac{\partial B}{\partial \alpha} \right) . \end{aligned} \tag{4.35}$$

Equation (4.35) is the sensitivity of the FRF due to model parameter changes and now the issue of which points to select can be analyzed. For values of ω close to a natural frequency of the system, the determinant of the matrix $(\jmath\omega I - A)$ becomes a small number, since this expression is very similar to the eigenvalue problem. In fact, it becomes zero for undamped systems at the natural frequency. In the general damped case, this small number makes the sensitivity $\frac{\partial H(\omega)}{\partial \alpha}$ a large number. Therefore, the general rule for selecting points of the frequency response function to be used in the updating procedure is that the chosen points should be as close as possible to the natural frequencies. In general, however, experimental frequency response functions are not very accurate at the natural frequencies. This occurs for many reasons, among them, for lightly damped systems, the response becomes too large and a linear model is no longer valid. Because of this, the points selected for updating should be nearby the natural frequencies so a high sensitivity is obtained, but not exactly at the natural frequencies, so inexact values are not used in the updating.

An extra guideline is that one should use the smallest number of frequency points necessary to solve the problem. Extra points do not contain any new information, increase the computational load, and can cause numerical instabilities.

4.5 Choice of input data

Existing techniques based on frequency domain data use points of the frequency response function directly in the updating procedure. However, this choice of input data produces poor results with this technique. This fact can be understood by analyzing figure (4.2). In this example of a one degree-of-freedom system, the FRF is sampled at ω_1, ω_2, and ω_3. The vectors for each FRF are

$$R = \left\{ \begin{array}{c} R_1 \\ R_2 \\ R_3 \end{array} \right\} \ , \ C = \left\{ \begin{array}{c} C_1 \\ C_2 \\ C_3 \end{array} \right\} \ , \text{and} \ L = \left\{ \begin{array}{c} L_1 \\ L_2 \\ L_3 \end{array} \right\} \ . \tag{4.36}$$

Measuring the distances between R and C and between L and C yields

$$d_{R,C} = (R_1 - C_1)^2 + (R_2 - C_2)^2 + (R_3 - C_3)^2$$

$$d_{L,C} = (L_1 - C_1)^2 + (L_2 - C_2)^2 + (L_3 - C_3)^2$$

Since $L_2 \approx R_2$, $L_1 \approx R_3$, $C_1 \approx C_3$, $R_2 \approx L_2$, and $R_1 \approx L_3$, then $d_{L,C} \approx d_{R,C}$. This means that Euclidean-like distance measures are not capable of distinguishing between the two cases above. This is very significant, since the two cases have very different meanings. This phenomena is also observed with multi-degree-of-freedom systems, making the use of points in the FRF infeasible as input data for the proposed technique.

To solve this problem, the input data was chosen to be the integral of the real and imaginary parts of the FRF over various frequency ranges. These frequency intervals should be such that their limits bound the natural frequencies of the system because these are the regions most sensitive to the parameter changes. figure (4.3) shows the experimental FRF of a cantilevered beam as a solid line and the analytical FRFs corresponding to the lower and upper values of the parameters being updated as dashed lines. The frequency ranges chosen for integration are represented by the shaded areas, and as it can be seen, they are just wide enough to bound the regions of the FRF where there is a significant behavior.

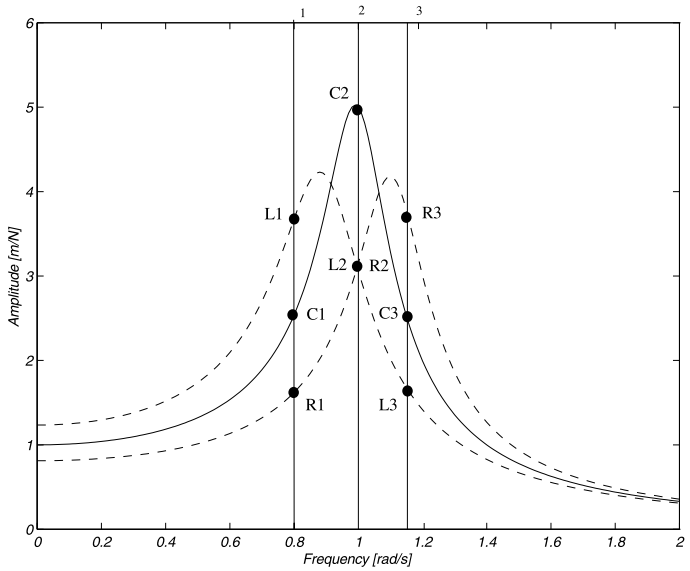

Figure 4.2: Example of possible ambiguity when measuring the distance between FRFs.

4.5.1 Effect of noise in the input data

Noise in the input data can affect the estimation process. Its effect can be studied by calculating the sensitivity of the network output with respect to some small perturbation (noise) in the input vector. The output of the trained network is determined by (eq. 3.27)

$$y = Wh \text{ , or } y_i = \sum_{j=1}^{N_t} w_{i,j} h_j \text{ , } \qquad i = 1, 2, \ldots NO \text{ ,}$$

where y is the network output (the estimated parameters), W is the weight matrix connecting the hidden and output layers of the network, and h is the output of the hidden layer. The vector h is given by

$$h_j = e^{-d_j} \tag{4.37}$$

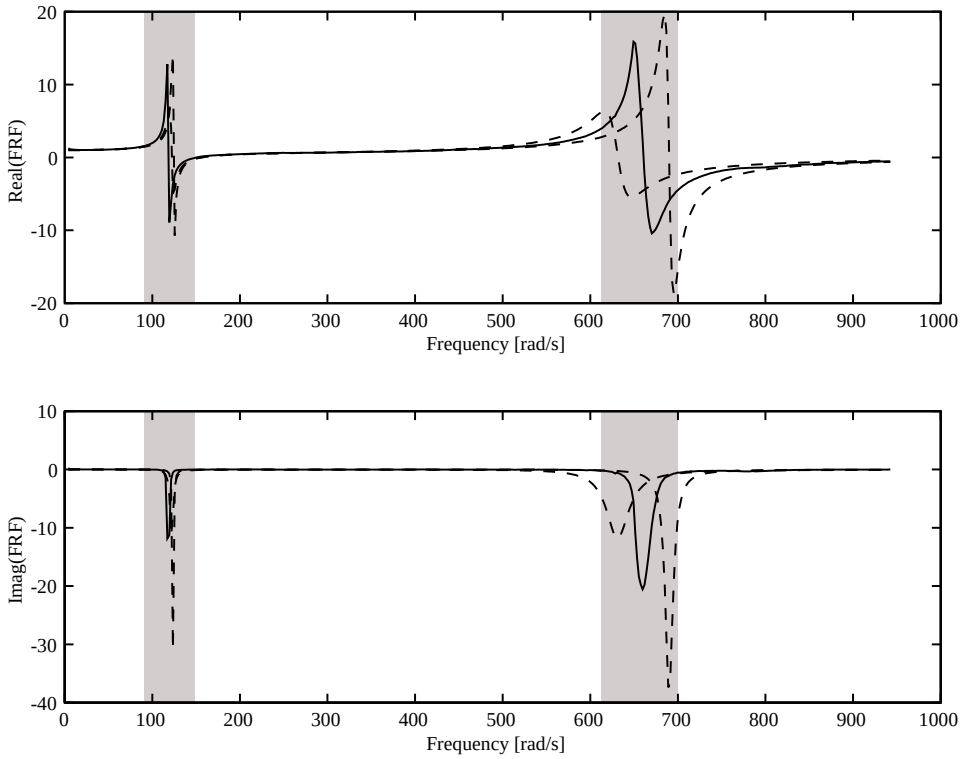

Figure 4.3: Frequency intervals (shaded areas) used to integrate the real and imaginary parts of the frequency response function of a cantilevered beam.

where d_j is the distance between the input vector and the center of the j^{th} neuron. The scalar distance measure d_j is calculated as

$$d_j = \Big(\sum_r \sigma_r^2 (x_r - c_{r,j})^2 \Big)^{1/2} , \tag{4.38}$$

where the column vector c_j is the center of the j^{th} neuron, and σ_r are the spread constants. Using equations (4.2, 4.37, 4.38) one can write the sensitivity of y with respect to a noise vector β as

$$\begin{aligned} \frac{\partial y_i}{\partial \beta_m} = & \quad \frac{\partial \sum_{j=1}^{N_t} w_{i,j} h_j}{\partial \beta_m} = \sum_{j=1}^{N_t} w_{i,j} \frac{\partial h_j}{\partial \beta_m} \\ = & \quad \sum_{j=1}^{N_t} \sum_{k=1}^{N_t} \sum_{l=1}^{NI} w_{i,j} \frac{\partial h_j}{\partial d_k} \frac{\partial d_k}{\partial x_l} \frac{\partial x_l}{\partial \beta_m} \ . \end{aligned} \tag{4.39}$$

Analyzing the expression above term by term yields

$$\frac{\partial h_j}{\partial d_k} = -h_j \delta_{jk} \ , \tag{4.40}$$

a diagonal matrix with entries equal to the entries of the vector h_j.

$$\begin{aligned} \frac{\partial d_k}{\partial x_l} = & \quad \frac{\partial \Big(\sum_{r=1}^{NI} \sigma_r^2 \big(x_r - c_{r,k}\big)^2 \Big)^{1/2}}{\partial x_l} \\ = & \quad \tfrac{1}{2} \Big(\sum_{r=1}^{NI} \sigma_r^2 (x_r - c_{r,k})^2 \Big)^{-1/2} \sum_{r=1}^{NI} \sigma_r^2 \frac{\partial \big(x_r - c_{r,k}\big)^2}{\partial x_l} \\ = & \quad \tfrac{1}{d_k} \sum_{r=1}^{NI} \sigma_r^2 (x_r - c_{r,k}) \delta_{rl} \\ = & \quad \tfrac{1}{d_k} \sigma_l^2 (x_l - c_{l,k}) \ , \qquad \begin{matrix} k = 1, 2, \ldots N_t \\ l = 1, 2, \ldots NI \end{matrix} \ , \end{aligned} \tag{4.41}$$

$$\frac{\partial x_l}{\partial \beta_m} = \frac{\partial (\overline{x}_l + \beta_l)}{\partial \beta_m} = \delta_{lm} \ , \tag{4.42}$$

a identity matrix and $\overline{x}$ is the noise-free input vector.

Putting together the above results yields

$$\frac{\partial y_i}{\partial \beta_m} = - \sum_{j=1}^{N_t} \sum_{k=1}^{N_t} w_{i,j} h_j \delta_{jk} \Big[\frac{1}{d_k} \sigma_m^2 (x_m - c_{m,k}) \Big] \tag{4.43}$$

If x is a vector used to train the network, the k^{th} row of the matrix $\Big[\frac{1}{d_k} \sigma_m^2 (x_m - c_{m,k}) \Big]$ in the equation above should be all ones. To analyze the effect of noise, the input vector $\overline{x}$ used to train the network is corrupted with noise. It is written as

$$x = \overline{x} + \epsilon \ , \tag{4.44}$$

where $\overline{x}$ is one of the vectors used to train the network and ϵ is a noise vector. For the neuron corresponding to the noise-free input vector, the term

$$\sigma_m^2(x_m - c_{m,k}) \tag{4.45}$$

in equation (4.43) is simplified to:

$$\sigma_m^2(\overline{x}_m + \epsilon_m - c_{m,k}) = \sigma_m^2 \epsilon_m \ , \tag{4.46}$$

and the distance term can also simplified to

$$d_k = \Big(\sum_{l=1}^{NI} \sigma_l^2 (x_l - c_{l,k})^2 \Big)^{1/2} = \Big(\sum_{l=1}^{NI} \sigma_l^2 (x_l + \epsilon_l - c_{l,k})^2 \Big)^{1/2} = \Big(\sum_{l=1}^{NI} \sigma_l^2 \epsilon_l^2 \Big)^{1/2} \ . \tag{4.47}$$

Using the two equations above it can be seen that the sensitivity is a function of the spread parameter associated with the input vector x. The larger the variance, the larger the spread constant. This means that, if row m of the input matrix has a large variance due to changes in the parameters being updated, then noise contaminating the m^{th} position of the input vector will significantly affect the network output.

Remembering the results obtained in the preceeding section, the frequency points should be chosen to be near the natural frequencies so the input data is the most sensitive to parameter changes. However, this high sensitivity results in a large variance of the FRFs at those points, and this is not desirable for being highly sensitive to noise.

If the noise present in the FRF is random with a small mean value, the integration of the FRF over a certain frequency range will reduce the effect of the noise on the network output. Integrating the FRF over certain frequency ranges has the positive side effect of reducing the amount of data to be processed by the neural network to $2 \times p$ pieces of information, where p is the number of frequency intervals used. This number is smaller than the number of frequency points normally used by existing techniques.

4.6 Weighting the input data

Often times it is desirable to have higher accuracy over certain frequency ranges, and the use of weight factors allows greater flexibility to the engineer when updating a model. This is made possible with this technique by changing the linear transformation (eq. 4.10). The integrals over the frequency range where a higher accuracy is not necessary is divided by a weight factor, as follows:

$$x_m^{new} = \frac{1}{\rho_m} \frac{x_m^{old} - \min(x_m^{old})}{\max(x_m^{old}) - \min(x_m^{old})} \ , \tag{4.48}$$

where ρ_m is the scaling factor for the m^{th} row of the input matrix. The choice of the weight factors depends on engineering knowledge of the system at hand. The cantilevered beam problem is be used to illustrate the use of and typical results obtained with weight factors. Figure (4.4) shows the worst estimation case (highest error) of the network after it has been trained. As shown, the accuracy in the estimation of the damping of the second mode is lower than the one of the first mode. Choosing $\rho_1 = \rho_2 = 2$, the weights of the integrals of the real and imaginary parts of the first mode result in a better accuracy for the damping of the second mode, as shown in figure (4.5). There is always a trade-off present when using weight factors: a higher accuracy for the second frequency range leads to a lower accuracy for the first frequency range, and there is an optimum point beyond which there is a loss of accuracy in the estimation process as a whole. This happens for $\rho_1 = \rho_2 = 4$ (fig. 4.6).

4.7 Influence of the number of parameters being updated

The response of a system is a function of its parameters. The higher the number of parameters being updated, the more the response of the system will change from the nominal design. An increase in the number of parameters being updated should be followed by a corresponding increase of the number of training samples in order to keep the accuracy at acceptable levels. As an example, a truck suspension is modeled by a 3-degree-of-freedom

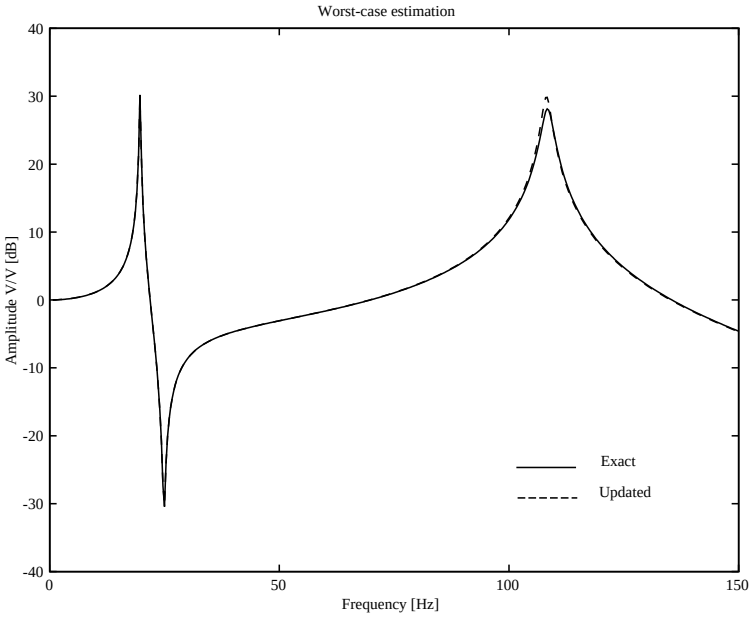

Figure 4.4: Worst case estimation - no weight factors.

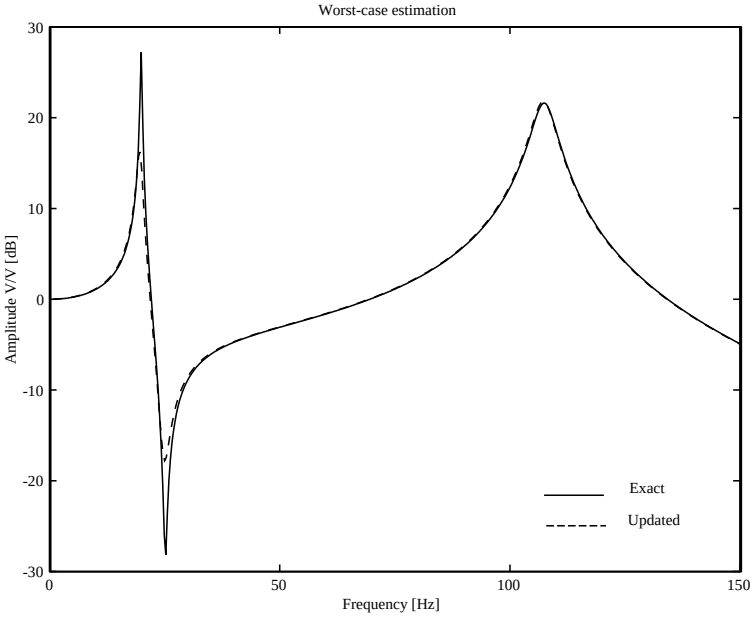

Figure 4.5: Worst case estimation - $\rho_1 = \rho_2 = 2$.

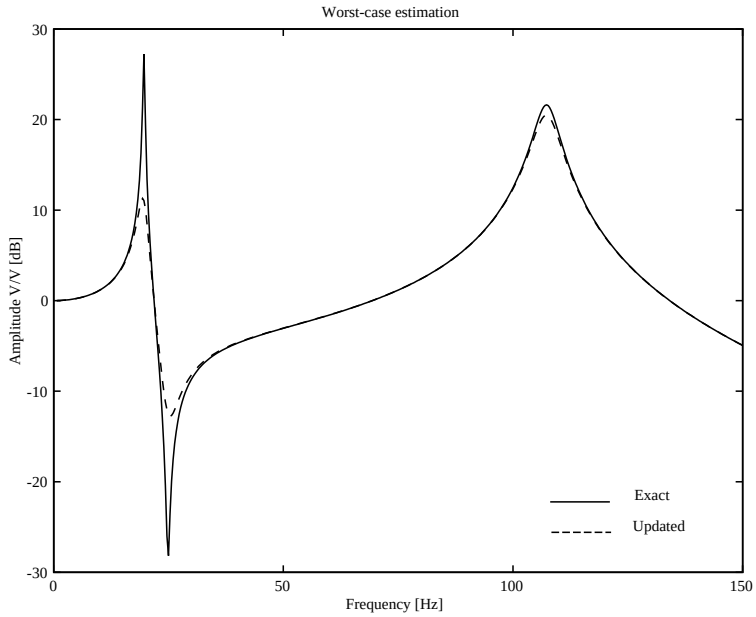

Figure 4.6: Worst case estimation - $\rho_1 = \rho_2 = 4$.

system where the parameters m_1, m_2, and k_1 are constant and equal to 3728 kg, 657 kg, and 21×10^5 N/m, respectively, and the remaining five parameters are known to be bounded as follows

$$100 \text{ kg} \leq m_3 \leq 160 \text{ kg} ,$$

$$2 \times 10^5 \text{ N/m} \leq k_2 \leq 8 \times 10^5 \text{ N/m} ,$$

$$12 \times 10^3 \text{ N/m} \leq k_3 \leq 35 \times 10^3 \text{ N/m} ,$$

$$900 \text{ N s/m} \leq b_2 \leq 1200 \text{ N s/m} , \text{ and}$$

$$500 \text{ N s/m} \leq b_3 \leq 700 \text{ N s/m} .$$

For the following examples the network has been trained with 800 randomly generated samples and their corresponding FRFs. Figure (4.8) shows the generalization error for the

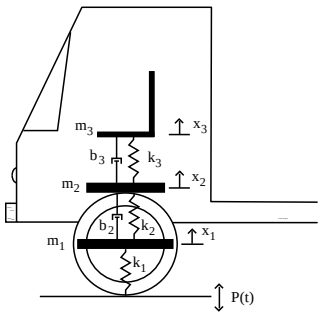

Figure 4.7: Diagram of the truck suspension

case when only m_3 is being updated. The mean absolute error in this case is $2.9 \times 10^{-3}\%$. The next figure (fig. 4.9) shows the generalization error for the estimation of k_2 and k_3. The mean absolute error in this case is $9.3 \times 10^{-2}\%$ and 0.126% for k_2 and k_3 respectively. The last figure (fig. 4.10) shows the generalization error when k_2, k_3, b_2, and b_3 are allowed to vary. The mean absolute error is 0.734%, 0.348%, 4.57%, and 0.824% respectively. The case when all five parameters are allowed to vary is shown in the next section (fig. 4.12). As expected, there is a degradation in performance with the increase in the number of parameters being updated. Therefore, only the most relevant parameters should be updated so that a high accuracy of estimation can be obtained.

4.8 Influence of the number of FRFs used in the updating

As mentioned before, this technique does not require the measurement of one full column (or row) of the frequency response matrix, avoiding the problems associated with mode shape expansion and model reduction. It attempts to find the set of model parameters that results in a close match between estimated and measured FRFs. Changes in the model parameters have different effects over different FRFs in the $H(\omega)$ matrix, *i. e.* sometimes a change in the model has a negligible effect on a particular FRF and a pronounced effect on another. Due to this fact, measuring more FRFs and using them as input to this technique provides more data to the network, resulting in higher accuracy.

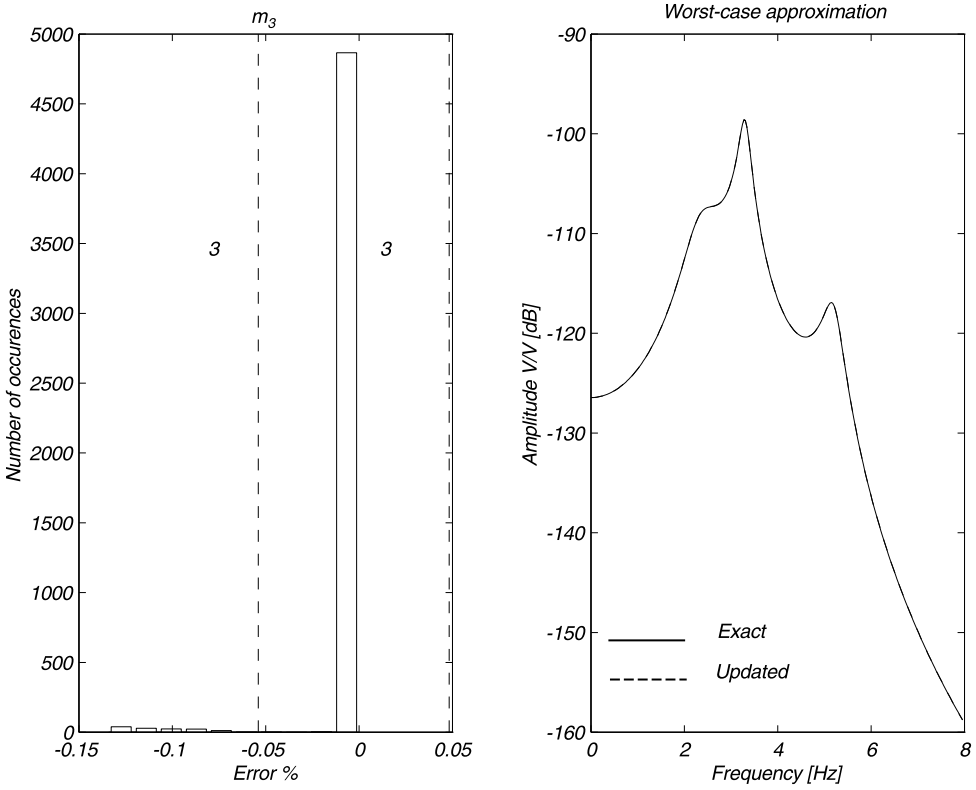

Figure 4.8: Generalization characteristics - One parameter

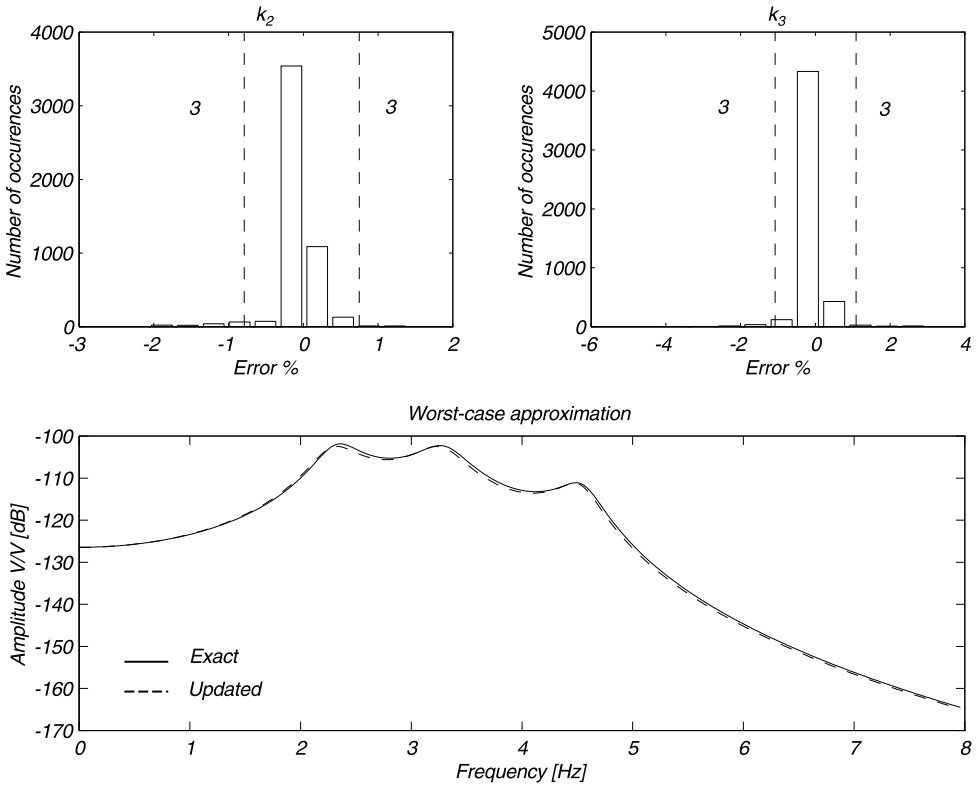

Figure 4.9: Generalization characteristics - Two parameters

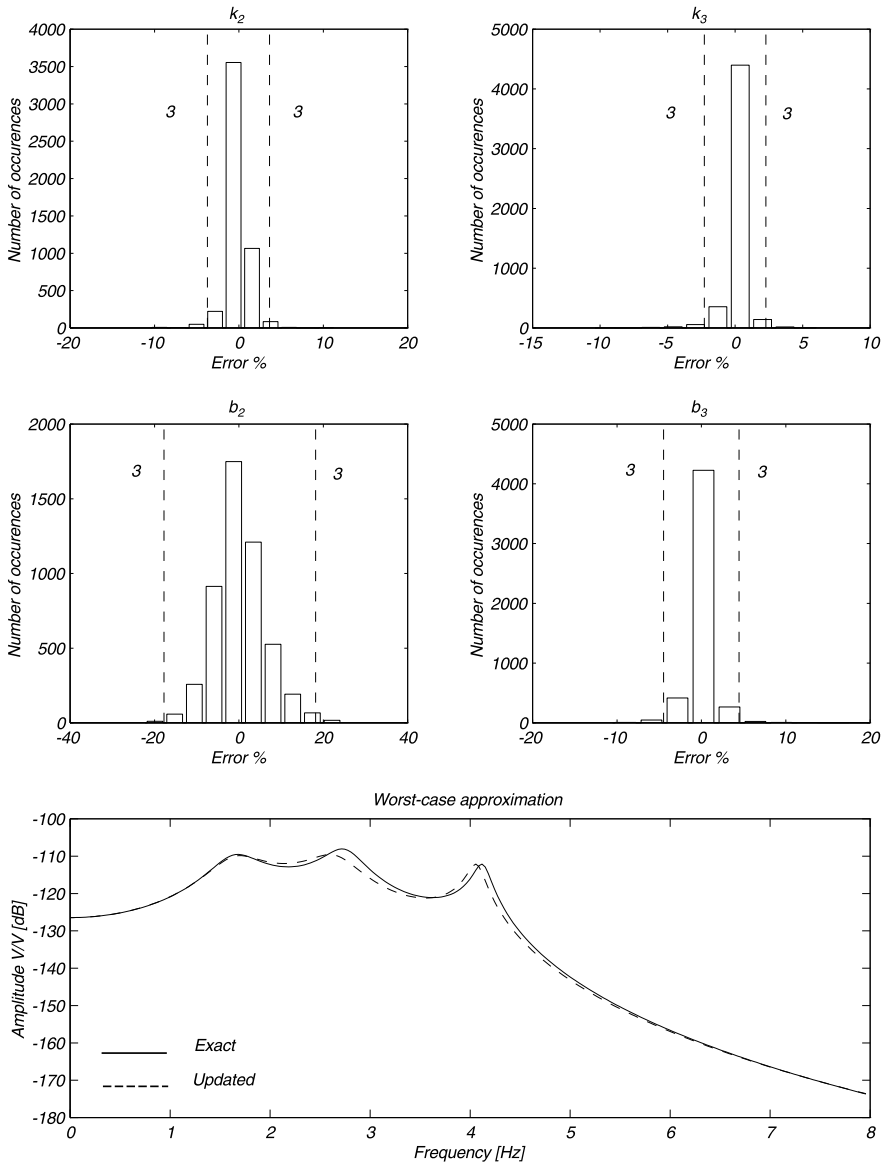

Figure 4.10: Generalization characteristics - Four parameters

Model parameters	One FRF		Two FRFs		Three FRFs	
	e^{avg}	e^{max}	e^{avg}	e^{max}	e^{avg}	e^{max}
m_3	2.86%	21.33%	1.59%	17.75%	1.50%	17.72%
k_2	0.79%	16.11%	0.66%	9.89%	0.79%	10.72%
k_3	2.99%	22.64%	1.63%	19.27%	1.58%	20.47%
b_2	5.30%	27.13%	2.73%	24.37%	2.47%	20.66%
b_3	2.81%	22.78%	1.95%	14.27%	2.03%	13.61%

Table 4.1: Error measures as a function of the number of FRFs used in the updating

The truck suspension example presented in the previous section is used to study the effect of the number of frequency response functions on the accuracy of the network. The network was trained using 800 randomly generated parameter combinations and their corresponding FRFs. The generalization characteristics of the network are shown in figure (4.12). The error between desired and estimated parameters is in the range of $\pm 20\%$ (fig. 4.12), being mostly concentrated in the range of $\pm 8\%$. The error measures for m_3, k_2, k_3, b_2, and b_3 are given in table (4.8) as a function of the number of FRFs used to form the input of the network. The excitation comes from the moving wheel (m_1) and initially only the FRF between f_1 and x_3 is used in the updating procedure. The case where two FRFs are used uses FRFs between f_1 and x_3 and between f_1 and x_2, and the case where three FRFs are used uses the FRF between f_1 and x_1 in addition to the previous two.

A significant improvement can be observed when using more frequency response functions. The use of two FRFs resulted in reduced error measures when compared to the case where only one FRF was used, and the use of three FRFs improves the error measures slightly. The improvement is not as pronounced because after a certain number of input data the network does not gain any new information. Further improvement can only be obtained with a larger training set, in which case more variations of the input space can be learned by the network.

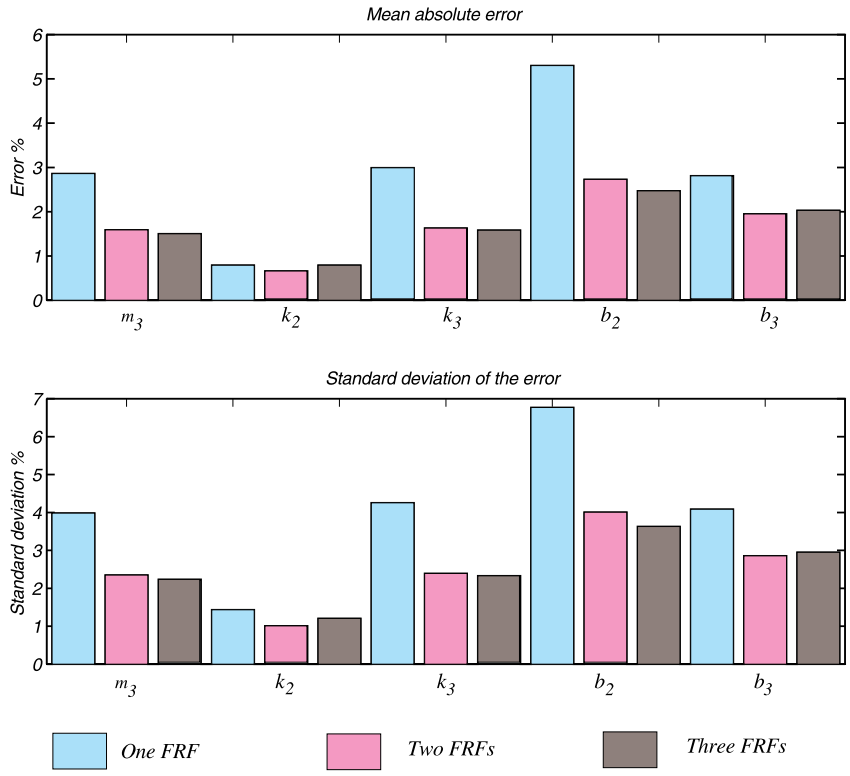

Figure 4.11: Error statistics

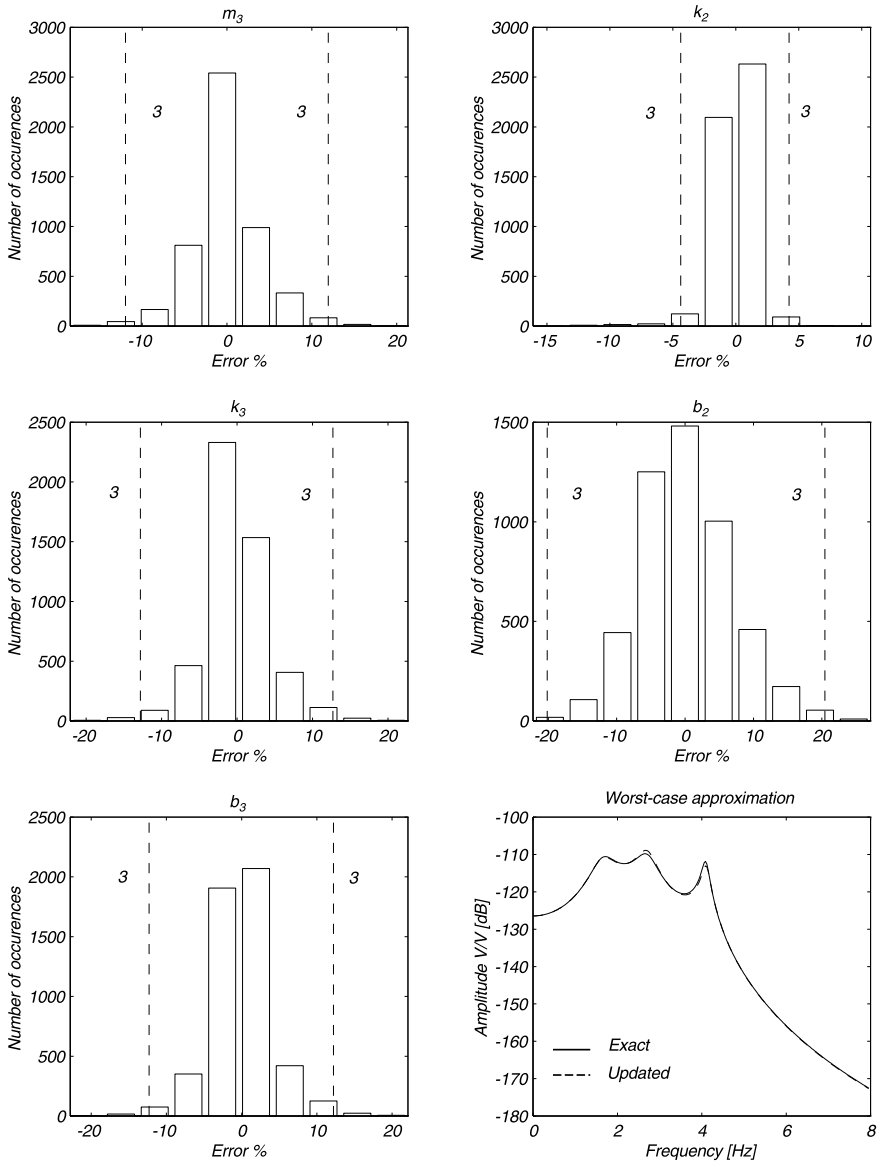

Figure 4.12: Generalization characteristics when trained with one FRF

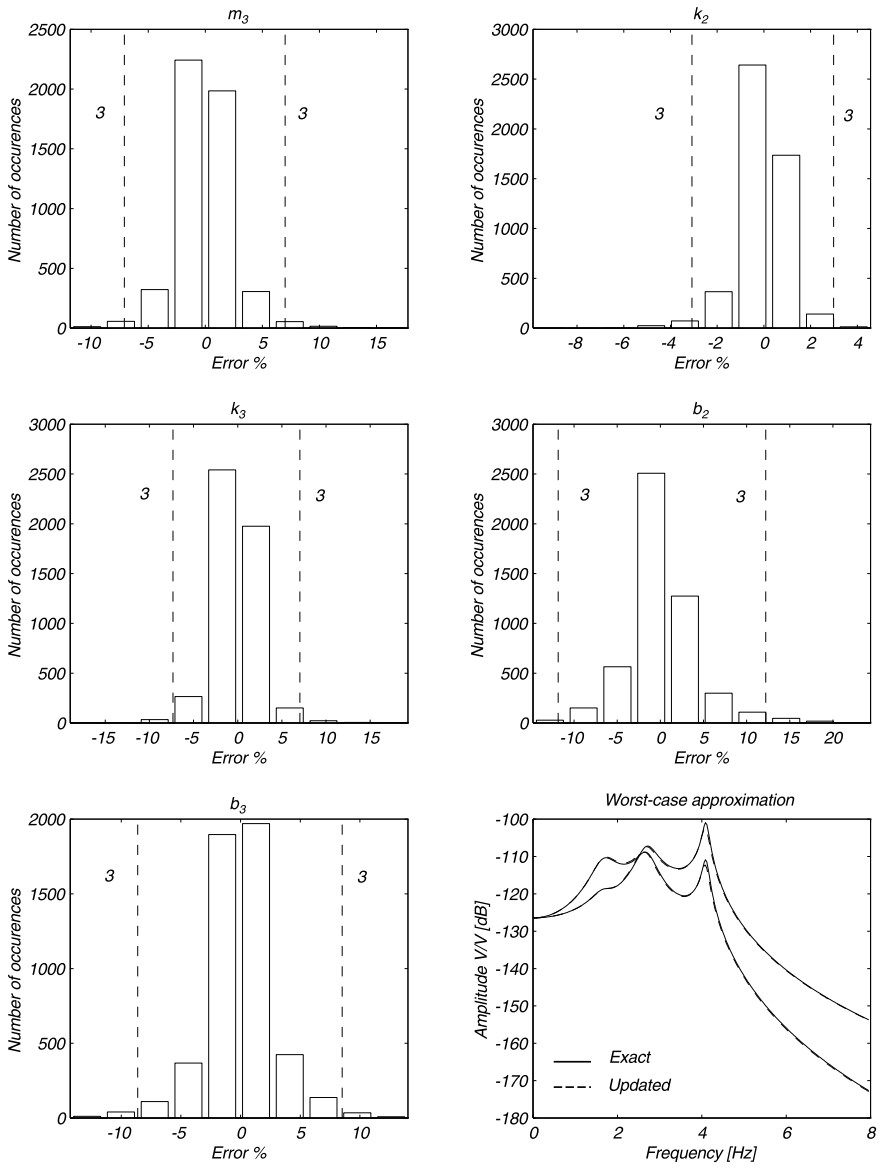

Figure 4.13: Generalization characteristics when trained with two FRFs

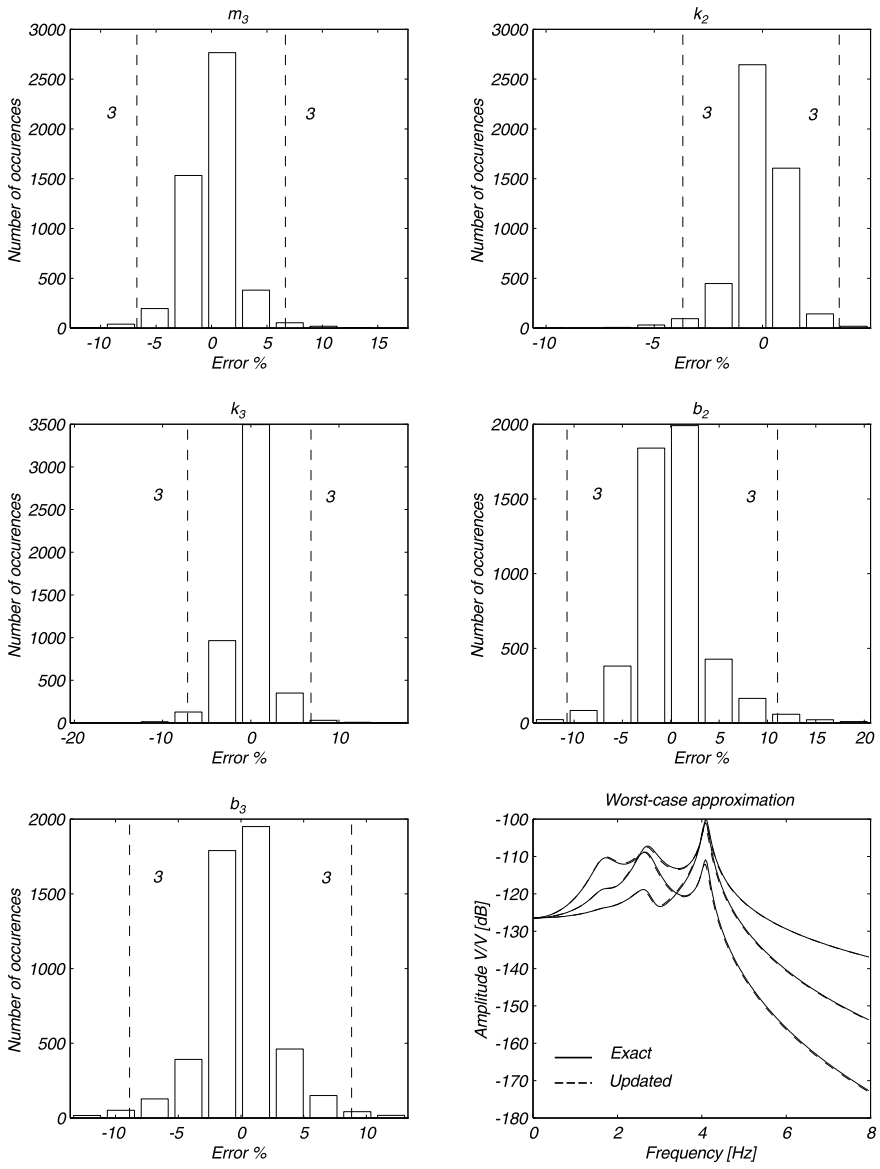

Figure 4.14: Generalization characteristics when trained with three FRFs

4.9 Nonlinear systems

The response of nonlinear systems present many interesting features, such as multiple solutions, sub- and super-harmonic frequencies, dependence on the amplitude of excitation, period-doubling, and chaos, among others [40, 41]. System identification techniques based on time domain data have been developed for weakly nonlinear systems [37]. The NNUM is, to the author's knowledge, the first to use frequency response data to estimate the parameters of the model. So far, nonlinear systems have not been the subject of research in the field of model updating. This may be explained by the fact that such systems cannot be defined by a FRF matrix, and linear algebra cannot be used to predict the response of the system, as it is the case for linear systems. However, this is not the case with the NNUM since it does not rely on linear algebra to predict the response of the system or to update the parameters.

In the present work the weakly nonlinear Duffing oscillator is considered. Its typical frequency response, zoomed around the primary resonance, is shown in figure (4.15.a). When doing a forward sine-sweep from the lower to the higher frequencies, the system follows the path indicated in figure (4.15.b). When doing a backward sine-sweep, the path shown in figure (4.15.c) is the one followed by the system. This response is typical of those observed in many other nonlinear systems.

Existing techniques break down due to the non-unique solutions present in the frequency response and its dependence on the level of excitation. The issue is choosing the input data for the neural network. The integral of the real and imaginary parts of the frequency response during a forward-sweep and during a backward-sweep were used as input data resulting in $4 \times p$ pieces of information, where p is the number of frequency intervals used to integrate the frequency response. This input data is sufficient for the network to construct the appropriate mapping, as demonstrated in the next chapter.

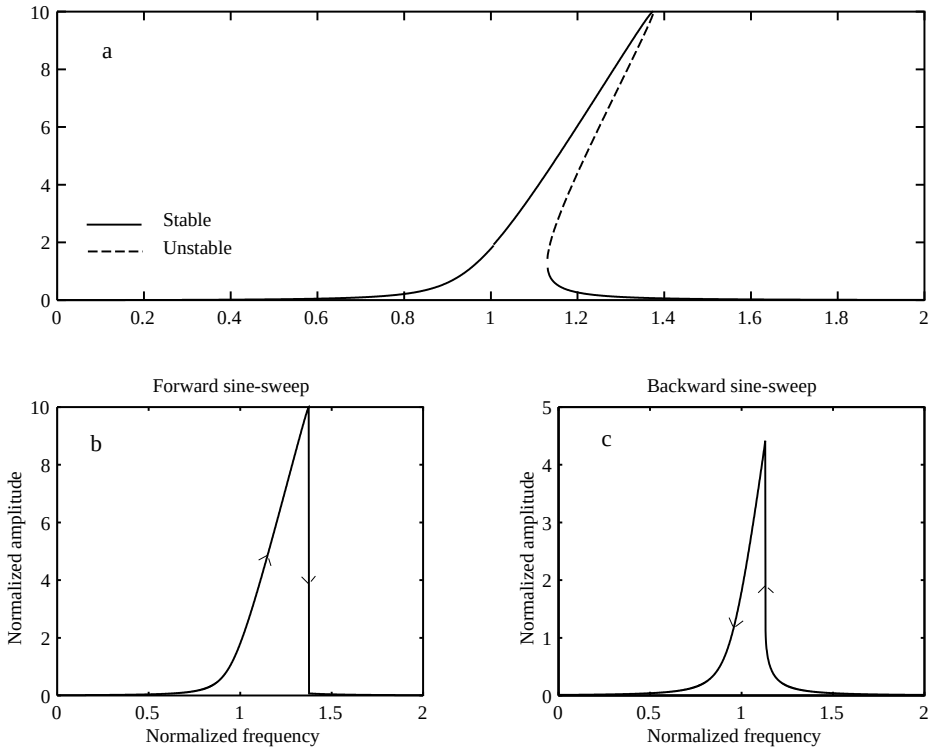

Figure 4.15: Frequency response of a Duffing oscillator around the primary resonance ($F = 2$, $\mu = 0.1$)

4.10 Computational load

Computational load is the amount of operations a computer has to perform when executing a task. This technique transfers most of the computational load of the estimation process to the training phase of the network. It is therefore useful to have an idea of how much computer processing is involved in this phase. When training the network, the most important steps are

i. Generate the data used to train the network

ii. Calculate the matrix of distances

iii. Calculate the output of the hidden layer

iv. Calculate the weight matrix

Step one is problem dependent. Step two has the computational load given approximately by

$$flops_2 = 8 \times NI \times N_t \times N \ , \tag{4.49}$$

where NI is the number of input neurons, N_t is the number of training samples, and N is the number of samples presented to the network. In the training phase $N = N_t$.

Step three involves calculating the exponential of each term of the matrix of distances. The computational load is given by

$$flops_3 = NH \times N \ , \tag{4.50}$$

where NH is the number of neurons in the hidden layer (usually N_t), and N is the number of samples presented to the network. As mentioned above, in the training phase $N = N_t$.

Step four corresponds to calculating the pseudo-inverse of the matrix obtained in step three and multiplying it by the target vectors. Considering the case when $NH = N_t$, the number of operations necessary to invert a matrix is

$$flops_i = 2 \times N_t^3 \ , \tag{4.51}$$

and the number of operations necessary to multiply the two matrices is

$$flops_m = 2 \times NO \times N_t^2 \ , \tag{4.52}$$

therefore, the number of operations necessary in step four is

$$flops_4 = 2 \times N_t^2(NO + N_t) \ . \tag{4.53}$$

Adding the results above, the number of operations necessary to train a network, excluding the generation of the training set, is approximately given by

$$flops_t = N_t^2 \times (2N_t + 8NI + 2NO + 1) \ . \tag{4.54}$$

The number of operations necessary to verify the generalization properties of the network is given by the expression above by replacing N_t by N.

It has been established in this work that the estimation process is fast. The computational load involved in the estimation process, assuming that one sample is input to the trained network and the parameters are estimated based on this input, is

$$flops_e = N_t \times (NI + 2NO + 1) \ . \tag{4.55}$$

For example, a network trained with 800 samples to solve the truck problem which has 6 inputs and 5 outputs, and verified with 5000 samples, takes 1 billion floating point operations

to be trained, 250 billion operations to be verified and 13600 floating point operations to estimate the parameters after being trained. A personal computer nowadays is capable of performing 60×10^6 operations per second, meaning the neural network can be trained in less than a minute, verified in less than an hour, and that parameter estimation parameters takes only a fraction of a second to compute.

4.11 Summary

The major issues and characteristics concerning this technique were explained and illustrated with examples. The neural network maps frequency domain data to model parameters. This map, once learned, allows fast estimation of the parameters being updated, making the technique suitable for use in updating systems that change over time. The estimation involves three steps:

- Measurement of the distance between the centers and the input vector,
- Computation of $e^{-d_{j,i}}$, and
- Computation of $y = Wh$.

The expressions for the estimated computational load show that the estimation of parameters is a fast procedure and that most of the computation is performed when training the network.

Real and imaginary components of the frequency response are integrated over selected frequency intervals and used to provide information about the system to the network. This choice of input data solves the problem of ambiguity when measuring the distance between two vectors. It also reduces the influence of measurement noise over the estimation of parameters.

The issues of normalization and weighting of the input data were analyzed and general guidelines on how to proceed were given. The effect of the number of training samples,

parameters being updated, and number of FRFs used to train the network, on the accuracy of the network were analyzed and illustrated with examples. The network has very good performance when the parameters being updated vary within known bounds, and it can be used for nonlinear systems with modification of input data to include forward and backward frequency responses.

Chapter 5

Examples

5.1 Introduction

Four numerical and two experimental problems representing a broad variety of mechanical systems are presented in this chapter. The numerical problems are chosen to illustrate the use and main characteristics of the proposed technique and to verify its accuracy. The first two problems are taken from the model updating literature and correspond to test cases where model updating is made difficult by characteristics of the systems. The third problem is used to illustrate the use of the technique as part of a control scheme, and the fourth shows its performance when applied to a weakly nonlinear system.

The two experimental problems demonstrate how the NNUM performs when measured data is used to update the model parameters. The first problem is a simple cantilevered beam where two damping parameters and the mass and inertia of an accelerometer mounted on the beam are updated. The second problem is a flexible frame with high modal density and many model uncertainties. Due to simplifying assumptions made in the model, equivalent mass density and Young's modulus must be determined, along with the damping parameters.

In each of the following examples the network is trained to update the model based on

the system's frequency response. Each system has parameters that are assumed to vary causing changes in the system's response. The goal is to train a network that updates the model not for a specific response, but for a collection of responses arising from variations of selected parameters. To validate the network, many possible combinations of the parameters being updated are used to generate new frequency responses. These frequency responses are input to the network and the estimated parameters are compared to those used to calculate the input, generating an error measure. This process is called validation and the characteristics of the estimation error is called generalization. High estimation accuracy requires low generalization errors. The network is trained and validated in each example in this chapter. The results obtained from the validation phase are shown to verify the network's accuracy in estimating each parameter.

5.2 Kabe's example

Kabe's problem, as it is known, was first presented by Alvar Kabe in 1985 [7], and it has become a popular test case for model updating techniques. It was first used as a test structure to be updated by Kabe's Stiffness Matrix Adjustment (KMA) method. The structure (fig. 5.1) is an eight-degree-of-freedom lumped parameter system with large relative difference magnitudes of the stiffness coefficients and very high modal density. All eight natural frequencies are within 27% of each other (the first four are within 5%) making this problem a severe test case for any model updating procedure. The parameters for the original system are given in table (5.1).

The KMA method uses the normal modes of the structure to update the model and

Parameters	m_1	$m_j \quad j = 2, 3, \ldots, 7$	m_8	k_1	k_2	k_3	k_4	k_5	k_6
Nominal Values	0.001	1.0	0.002	1000	10	900	100	1.5	2.0

Table 5.1: Kabe's problem: nominal parameters.

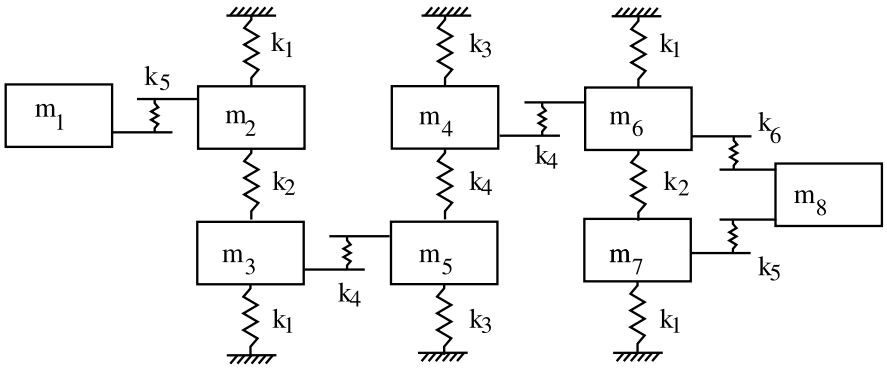

Figure 5.1: Kabe's example.

assumes that damping is not present in the structure. Five of the six stiffness coefficients are allowed to vary, some by as much as 100%, in the original problem. Many authors have modified the original problem to make it more tractable by techniques that do not use mode shapes. The original problem is also modified here: the stiffness coefficients k_1 and k_5 are allowed to vary 20% about their nominal values according to a normal distribution centered at the nominal value with variances $\sigma_1^2 = 180\ (\mathrm{N/m})^2$, and $\sigma_5^2 = 0.25\ (\mathrm{N/m})^2$, respectively. The coefficients k_1 and k_5 are chosen because the large difference in their magnitude makes for a good test case of the neural network's ability to estimate such different quantities accurately. Transfer functions between a disturbance force acting on m_5 and the displacements of the lateral masses m_1 and m_8 are used as input to the network.

The first step is to define the frequency intervals over which the response will be integrated. Figure (5.2) shows the lower (solid line) and upper (dashed line) bounds of the transfer functions zoomed on the frequency range where the natural frequencies occur. The frequency intervals used for updating are shaded. It is clear from this figure that the response is greatly affected by changes in these two stiffness coefficients, requiring high accuracy from the updating technique. No clear separation of modes can be seen in this

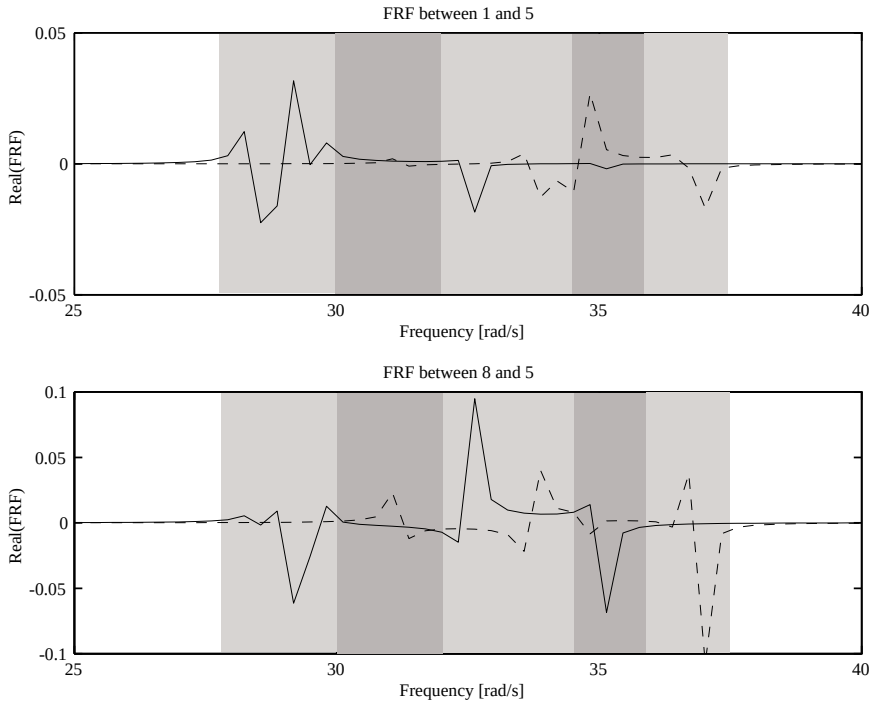

Figure 5.2: Frequency intervals used to update Kabe's problem.

problem, making the definition of the frequency intervals more difficult. In this example the frequency domain is divided into five regions: [27.9 29.8], [30.1 32.3], [32.3 34.5], [34.5 36.4], and [36.7 37.6] rad/s.

The next step is to define the number of sample pairs that will be used to train the network. This number depends on the complexity of the system and on the sensitivity of the response to changes in the parameters being updated. In order to obtain high accuracy, the neural network is trained with 1500 sample pairs. This large amount of training data is required to represent the input space because the characteristics of the system's response change drastically with the parameters. Since damping is not present in this problem, only the integral of the real part of the frequency response is used to provide information to the network. The network's generalization characteristics are shown in figure (5.3), along with

the worst-case estimation, the case where the summed estimation error is the largest. The height of each vertical bar indicates how many parameter combinations fall within the error margin represented by the vertical bar. The vertical dashed lines represent the interval of 99% confidence (3 σ), meaning that the error in 99% of the test cases falls within those lines.

Reasonable overall accuracy is obtained resulting in the following error measures:

$$e^{avg} = 2.3\% \text{ for } k_1 \text{ and } 5.8\% \text{ for } k_5, \text{ and}$$

$$e^{max} = 31.9\% \text{ for } k_1 \text{ and } 46.9\% \text{ for } k_5.$$

The generalization characteristics are considered acceptable even though the maximum error exceeds the expected range of parameter variation. The parameter combinations that result in large errors are very close to the boundaries of expected parameter variation and therefore are not representative of the majority of possible parameter combinations. These large errors are caused by the lack of damping in the system, making the numerical integration of the FRF an unreliable measure. This is explained by a large variance of the rows of the input matrix (an undesirable characteristic explained in chapter 4, section 4.5) resulting from high amplitudes in the FRF at frequency points in the proximity of a natural frequency. A viscous damping coefficient of $c = 0.01$ is added in parallel to each spring to illustrate this fact. This addition results in a noticeable improvement of the network's estimation accuracy as shown in figure (5.4). The new error measures are

$$e^{avg} = 0.4\% \text{ for } k_1 \text{ and } 2.7\% \text{ for } k_5, \text{ and}$$

$$e^{max} = 11.9\% \text{ for } k_1 \text{ and } 40.0\% \text{ for } k_5,$$

proving that the NNUM can successfully solve Kabe's problem, a popular test case for updating techniques.

This example shows that the existence of damping improves the accuracy of the updating procedure. This is a desirable characteristic since all real systems have some form of damping

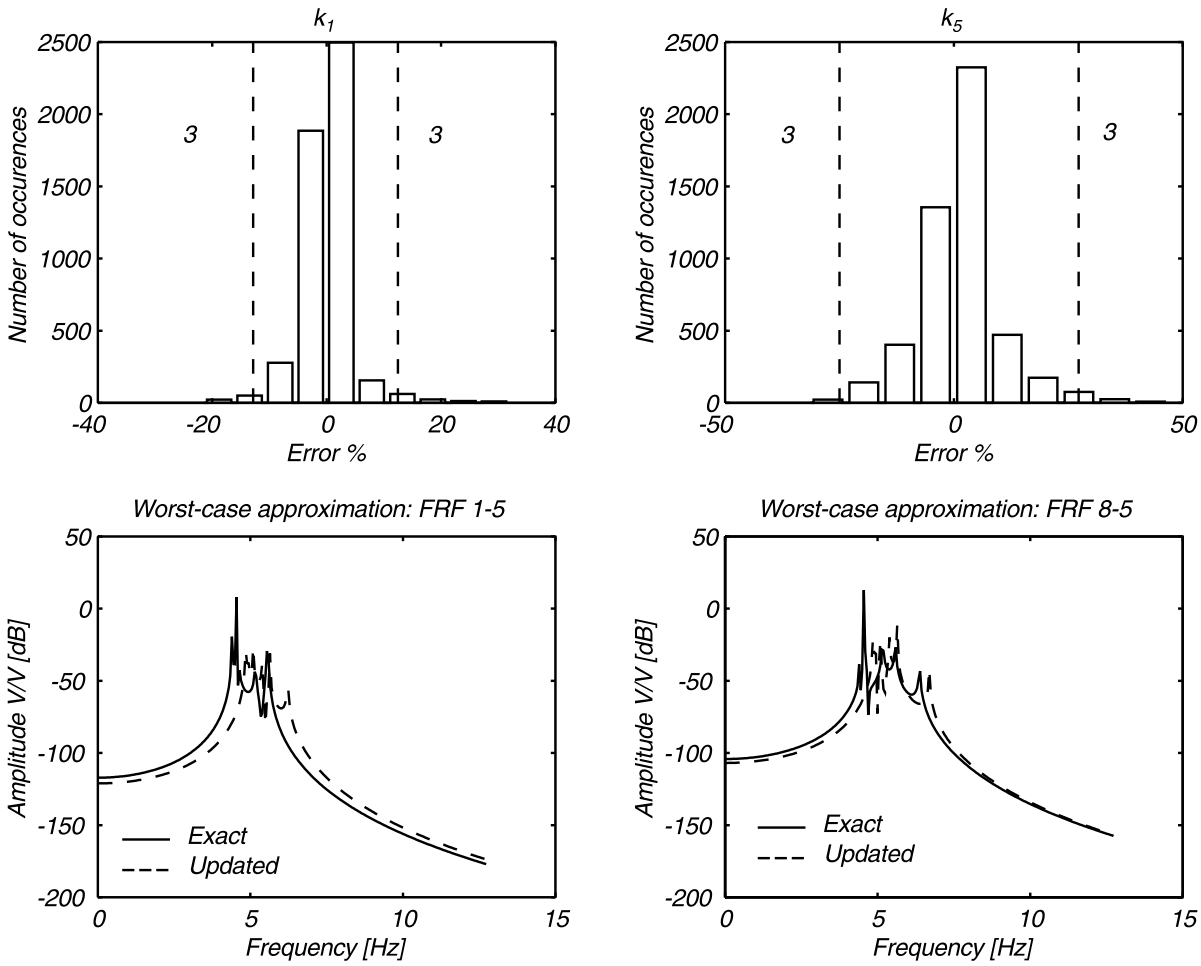

Figure 5.3: Generalization characteristics of the network trained to solve Kabe's problem.

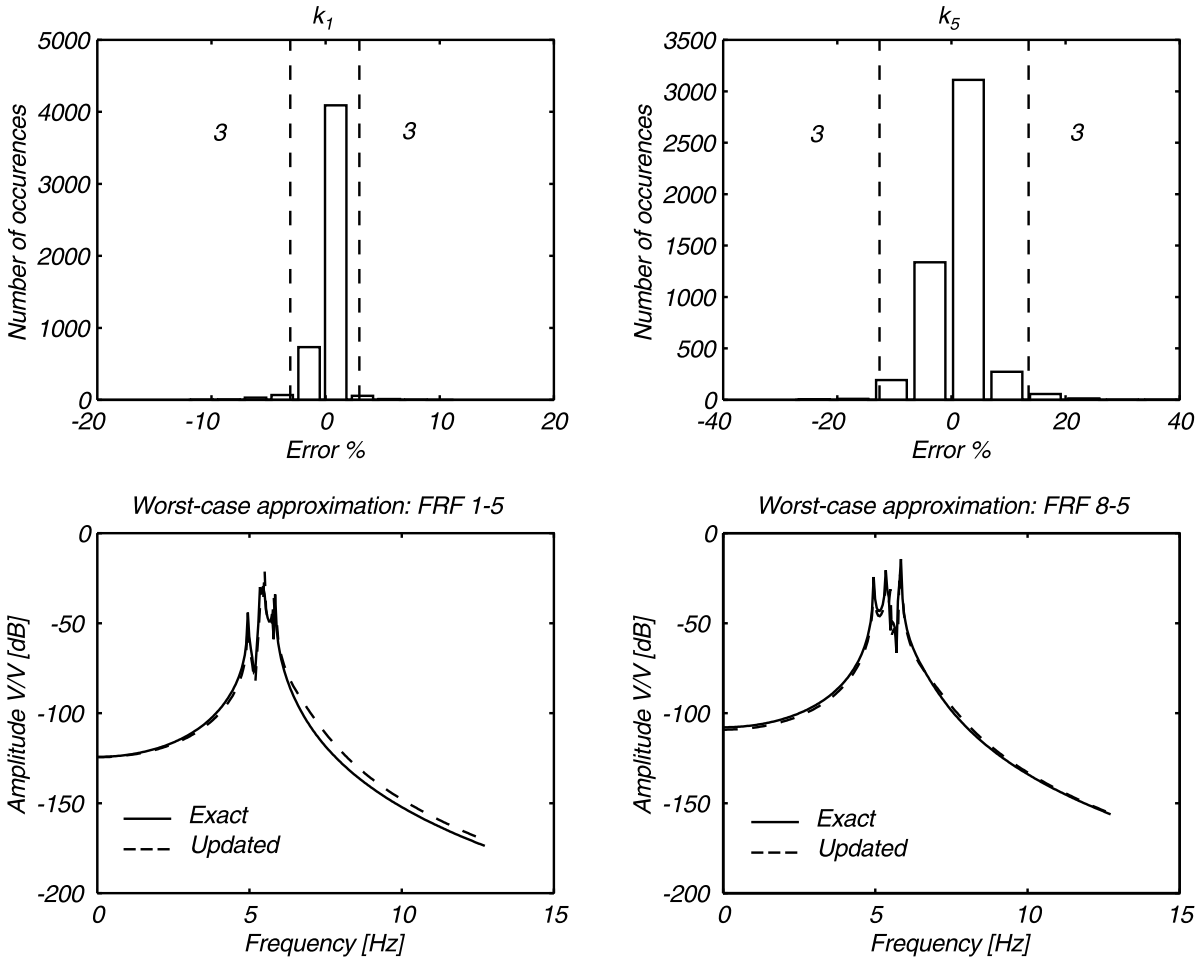

Figure 5.4: Generalization characteristics of the network trained to solve the damped version of Kabe's problem.

mechanism. The estimation/generalization error of mostly $\pm 5\%$ observed in figure (5.4) is acceptable for most engineering applications. The accuracy of the NNUM is verified by the worst-case approximation shown in figure (5.4), where exact and updated responses are sufficiently similar for most engineering applications.

5.3 Yang's example

Recently, Yang & Brown [8] presented a technique for handling damping when updating a model and used an interesting and challenging example to test their technique. Their example, shown in figure (5.5), is a more realistic problem than the one proposed by Kabe because it has bands of high modal density and the modes are not extremely close to each other. This system is a fifteen-degree-of-freedom lumped parameter system with high modal density. The original paper updates the parameters m_{11} through m_{15} and k_1 and k_6 for two different damping configurations: low and high damping. The high damping case corresponds to damping coefficients seven times higher than the ones used for the low damping case. The response of a nominal system is calculated and used as the "measured" response. The parameters being updated are then modified, simulating a mismodeling of the structure, and a new response is calculated. The goal is to use the "measured" response to update the mismodeled system. The parameters obtained by the updating procedure are then compared to the ones used to generate the "measured" response and a good agreement is verified. The nominal parameters for this model are shown in table (5.2).

In this example, the mass parameters are assumed to vary 20% about the nominal

Parameters	m_1 to m_{10}	m_{11} to m_{15}	k_1 to k_{18}	k_{19}	k_{20}
Nominal Values	0.0259 lbs	0.001295 lbs	1000 lbs/in	1100 lbs/in	900 lbs/in
Parameters	k_{21}	k_{22}	c_1 to c_{12}	c_{13} to c_{22}	
Nominal Values	1200 lbs/in	800 lbs/in	0.1 lbs.s/in	0.01 lbs.s/in	

Table 5.2: Yang's problem: nominal parameters.

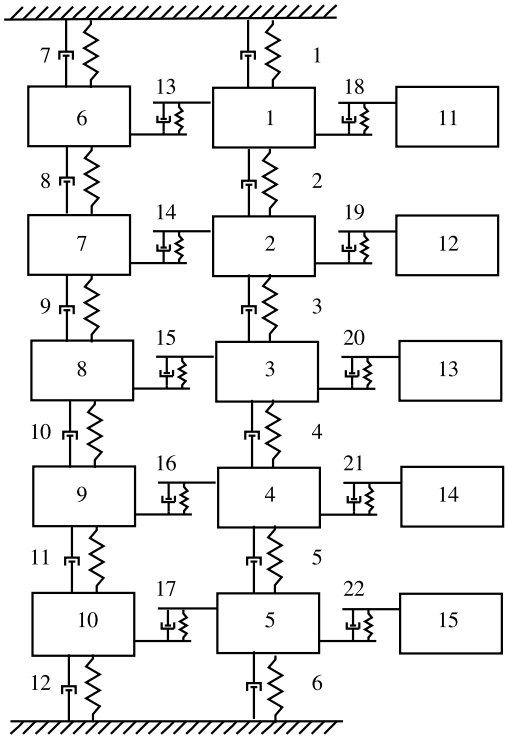

Figure 5.5: Yangs's example

Case	Measure	m_{11}	m_{12}	m_{13}	m_{14}	m_{15}	k_1	k_6
$N = 528$	e^{avg}	1.25%	0.79%	1.07%	1.31%	1.95%	0.31%	0.56%
low c	e^{max}	5.74%	5.68%	6.07%	6.74%	13.25%	1.52%	1.94%
$N = 828$	e^{avg}	1.04%	0.75%	1.01%	1.29%	1.72%	0.26%	0.34%
low c	e^{max}	4.33%	4.85%	4.37%	6.32%	8.07%	1.20%	1.40%
$N = 528$	e^{avg}	1.38%	1.01%	1.45%	1.40%	1.31%	1.05%	1.16%
high c	e^{max}	4.85%	3.16%	4.54%	5.77%	5.06%	2.47%	3.12%
$N = 828$	e^{avg}	1.13%	0.69%	1.32%	1.25%	1.12%	0.74%	0.88%
high c	e^{max}	4.57%	2.74%	3.96%	4.84%	4.29%	1.91%	2.39%

Table 5.3: Error measures for the four different variations of Yang's problem.

values according to a normal distribution centered at the nominal value with variance $\sigma_m^2 = 1.5 \times 10^{-4}$ lb^2. The stiffness parameters are assumed to vary 30% about the nominal values also according to a normal distribution centered at the nominal values with variance $\sigma_k^2 = 300$ (lb/in)^2. Following the suggestion given in the original paper, transfer functions between inputs at coordinates 1 and 2 and the displacements of masses 1, 6, 11, 12 13, 14, and 15 are used to update the model. Figure (5.6) shows the lower (solid line) and upper (dashed line) bounds of the transfer function between coordinates 2 and 11, and the frequency intervals selected for updating are represented by the shaded area. In this example it is possible to distinguish the frequency ranges where the changes occur. The chosen frequency intervals are [82.1 138.9], [189.4 227.3], [233.6 328.4], and [328.4 524.1] rad/s. Note that some high frequency modes are not considered in the updating procedure. The accuracy of the updated response of the high frequency modes depends on the accuracy of the structural model and on the accuracy of the parameter estimate produced by the updating procedure.

To verify the increase in accuracy with the increase in the number of training samples, the model is updated using two different neural networks: one trained with 528 and another with 828 sample pairs. The networks' generalization characteristics for both damping cases are shown in figures (5.7, 5.8, 5.9, and 5.10). The error measures for the four cases are

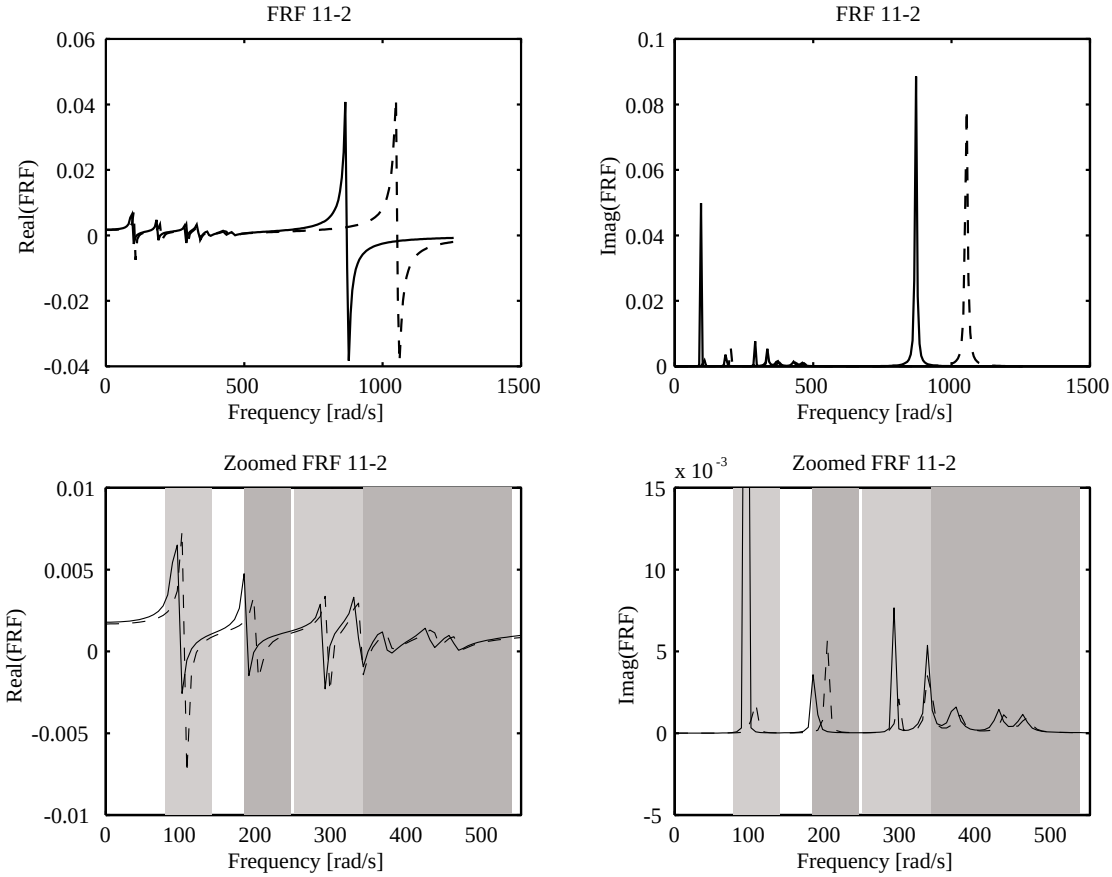

Figure 5.6: Frequency intervals used to update Yang's problem.

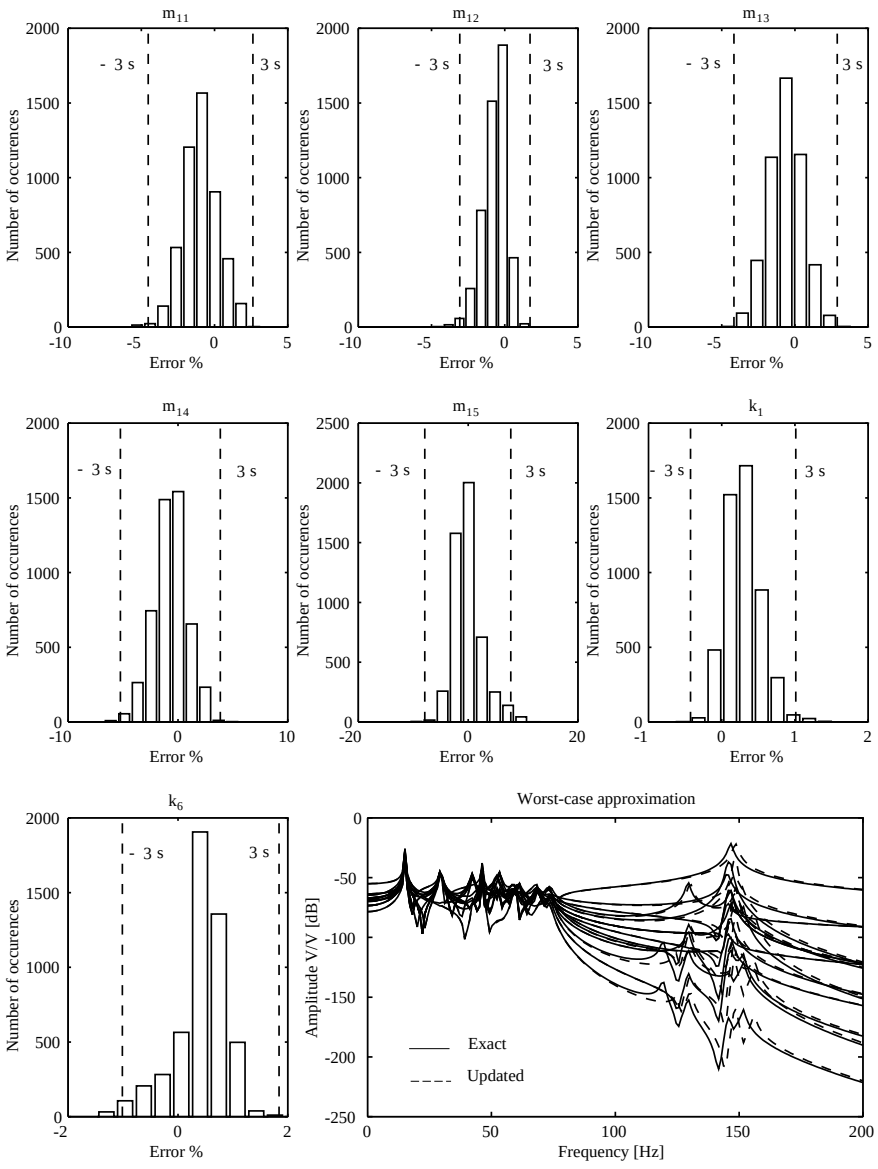

Figure 5.7: Generalization characteristics for the network trained with 528 sample pairs to solve the low damping variation of Yang's problem.

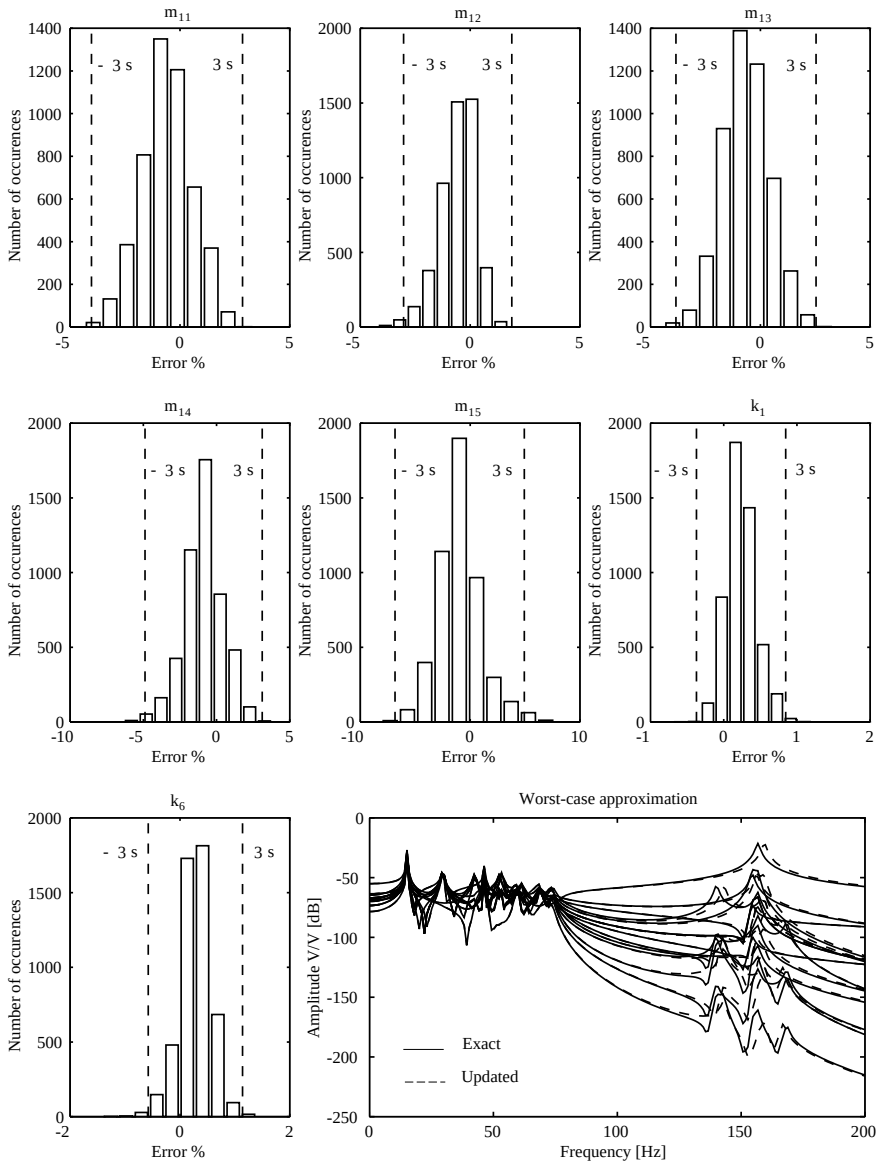

Figure 5.8: Generalization characteristics for the network trained with 828 sample pairs to solve the low damping variation of Yang's problem.

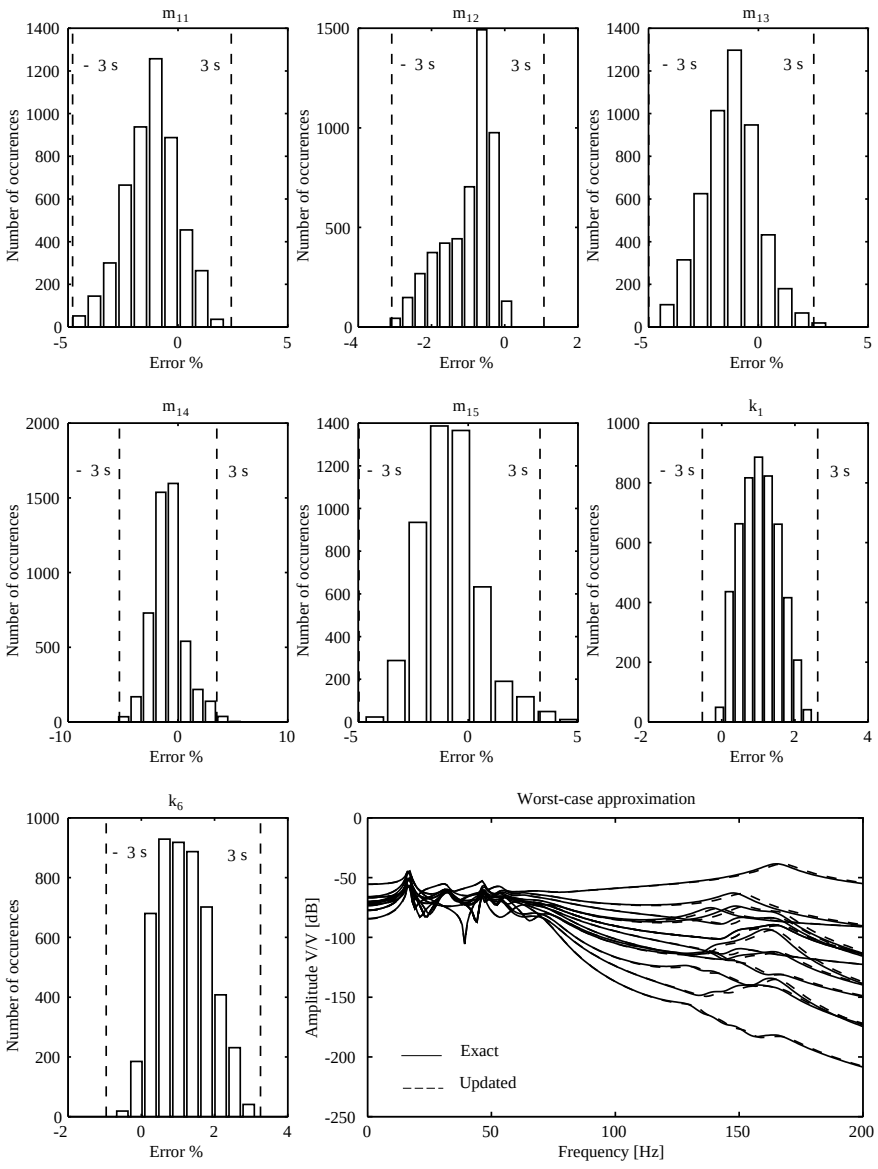

Figure 5.9: Generalization characteristics for the network trained with 528 sample pairs to solve the high damping variation of Yang's problem.

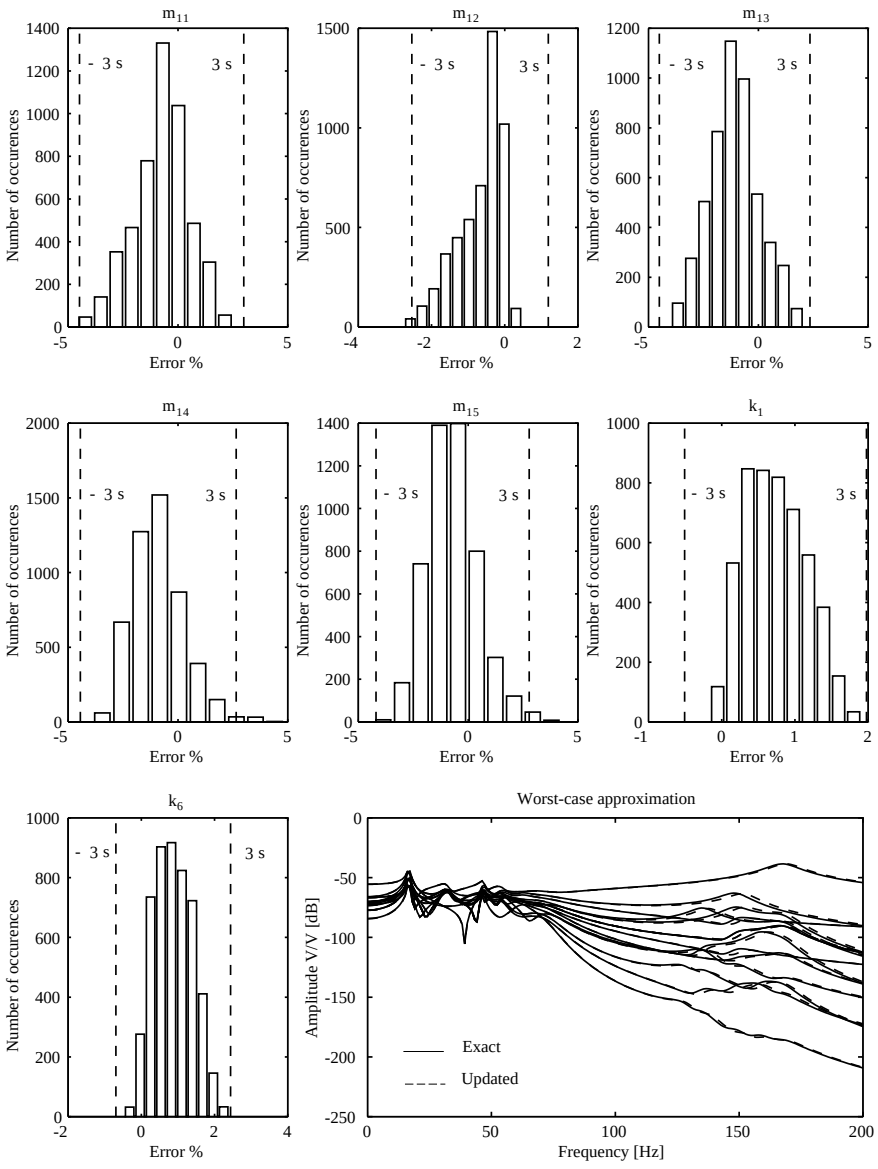

Figure 5.10: Generalization characteristics for the network trained with 828 sample pairs to solve the high damping variation of Yang's problem.

summarized in table (5.3). The increase in the number of training samples reduces the estimation error as demonstrated in chapter 4. In this case, 528 training samples are enough to generate a network with acceptable estimation accuracy.

Contrary to most existing techniques, the increase in damping does not pose any extra difficulty to the proposed technique. The accuracy of the NNUM is further corroborated by the estimation error of mostly $\pm 5\%$, verified in figures (5.7, 5.8, 5.9, 5.10). Furthermore, these figures show that the response of the high frequency modes not used in the updating procedure are accurate even for the worst-case estimation.

Based on the figures presented in [8], the method proposed by the authors needs about 8 iterations to reach an error of less than 5%. For this problem, NNUM becomes advantageous in two situations:

- when a fast estimate of the parameters is necessary after the parameter varies, or
- when the parameters are expected to vary more than $\frac{N_t}{8} = \frac{528}{8} = 66$ times during the lifetime of the structure.

The accuracy achieved by the NNUM is similar to the one achieved by Yang and Brown, and has the advantage of being able to update the model many times in a quick manner.

5.4 Truck Suspension with a LQR Controller

The truck suspension problem introduced in chapter 4 is used to test the performance of the proposed technique in a closed-loop control system. This problem is chosen because many parameters of the system are allowed to vary, and the bounds on some of the parameters are fairly wide. The suspension is modeled as shown in section (4.7), and its diagram is repeated in figure (5.11) for convenience.

The mass of the driver and the stiffness and damping coefficients of the suspension are expected to vary during the lifetime of the vehicle. An initial model is derived based on nominal values of these parameters. The objective of the control law is to reduce the

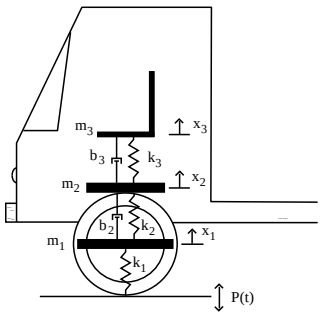

Figure 5.11: Diagram of the truck suspension

vibrations of the driver's seat (m_3) due to the excitation coming from the wheel (**P**). The frequency response between **P** and x_3 is used to train the network since it is usually measured to verify the performance of the control law. The parameters are allowed to vary according to a normal distribution with the following characteristics:

i. m_3 varies as 130 kg $\pm$ 30% with $\sigma^2 = 15$ kg^2

ii. k_2 varies as 5×10^5 N/m $\pm$ 60% with $\sigma^2 = 1.6 \times 10^5$ (N/m)^2

iii. k_3 varies as 23.5×10^3 N/m $\pm$ 49% with $\sigma^2 = 7 \times 10^3$ (N/m)^2

iv. b_2 varies as 1050 N m/s $\pm$ 14% with $\sigma^2 = 100$ (N m/s)^2

v. b_3 varies as 600 N m/s $\pm$ 17% with $\sigma^2 = 70$ (N m/s)^2

The network is trained with 832 training samples and its performance is verified with 5000 parameter combinations. Each of these combinations is used to generate a FRF that is then fed to the neural network. The neural network then generates an estimate of the parameters used in the calculation of the FRF. The generalization characteristics have been shown in figure (4.12).

This example is used to investigate the effect of the estimation error in the performance of the closed-loop system. In this example the objective is to control the vibration of the mass m_3, the drive-seat subsystem, by placing an actuator between m_2 and m_3.

The Linear Quadratic Regulator (LQR) control design technique is known to provide high performance controllers [38, 39], however, variations in the system's parameters can lead to degraded performance or even unstable closed-loop responses. The goal of this example is to use the NNUM to update the model as the system changes, recalculating the gain matrix using the LQR design technique, and verify the stability and performance of the closed-loop system.

Three possibilities are considered:

i. A constant gain matrix K_0 calculated for the nominal model of the system is always used,

ii. A gain matrix K_e is calculated for the estimated model of the system every time the system changes, and

iii. A gain matrix K_i is calculated for the ideal situation, when the model is exact for each combination of parameters.

Case iii will yield the best results and case i the worst, while case ii should fall somewhere in between. The question is: how close are the performances of cases ii and iii? If they are sufficiently close, then the proposed technique successfully estimates the model as it changes, yielding stable, high-performance controllers. Three measures are used to compare the performance of the different cases: settling time, peak amplitude, and the value of the functional J used in the LQR design. The results are shown for the situations of expensive ($R/Q = 1 \times 10^{-2}$) and cheap ($R/Q = 1 \times 10^{-6}$) control effort, where Q is the cost associated with the states and R is the cost associated with the control force in the LQR design. Figures (5.12) and (5.13) show the results for the two situations. The behavior of case ii compared to case iii is very similar. The time response characteristics are similar and the functional is only slightly larger than the ideal case. Case i resulted in very different time responses, higher functionals, and unstable closed-loop responses for many parameter combinations. No instability was observed for case ii.

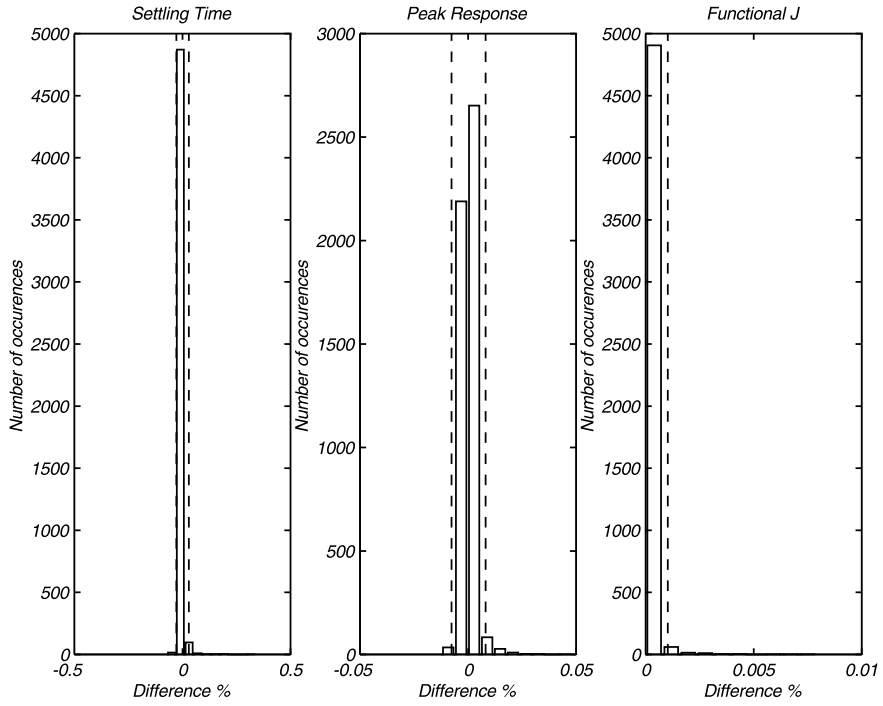

Figure 5.12: Performance comparison for the expensive control effort case ($R/Q = 1\times10^{-2}$).

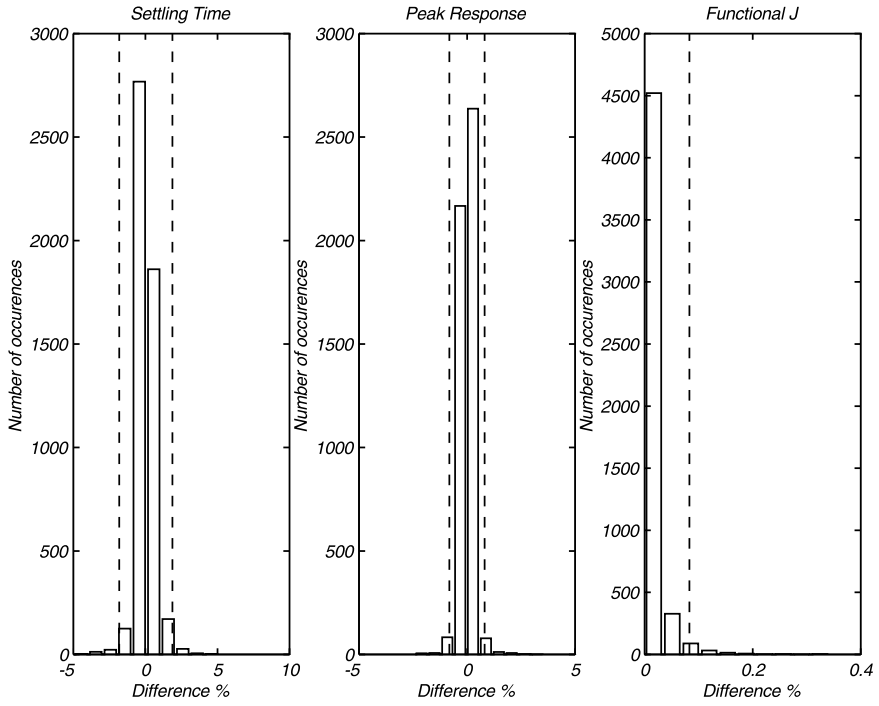

Figure 5.13: Performance comparison for the cheap control effort case ($R/Q = 1 \times 10^{-6}$).

These results show that the error in the estimation procedure does not compromise the behavior of the closed-loop system. In fact, the gain matrix calculated based on the updated model results in a closed-loop system that performs similarly to the ideal case. By using this technique, an engineer would design a control law robust to 10% to 20% of peak parameter variation (0.7% to 5.3% on average), instead of the original 60% of parameter uncertainty. Accuracy can be further increased by using more FRFs and more training samples, as explained in chapter 4.

The advantage of using this technique over others is that each time a parameter changes a new, physically plausible model is generated and used to recalculate the closed-loop system. This updated model can also be used to safely increase the rate of convergence of control schemes that estimate all the states of the system based on the measurement of a few states.

5.5 Duffing oscillator

The Duffing oscillator is an one-degree-of-freedom damped system (fig. 5.14) with a nonlinear spring described by

$$m\ddot{x}(t^*) + c\dot{x}(t^*) + k_L x(t^*) + k_N x^3(t^*) = f\cos\omega t^*, \tag{5.1}$$

where m is the mass, c is the damping coefficient, k_L is the linear spring constant, k_N is the nonlinear spring constant, and f is the amplitude of the force applied to the system. The parameters that are assumed to vary, and therefore are targeted for updating, are the nonlinear spring constant k_N and the damping coefficient c. The neural network is trained to produce estimates of k_N and c based on the frequency response of the Duffing oscillator about its primary resonance.

It is normal practice to convert the equation (5.1) to nondimensional form via the transformations

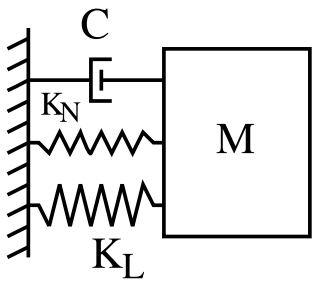

Figure 5.14: The Duffing oscillator

$$t = \frac{t^*}{T} \text{ and } \; u = \frac{x}{X} \; , \tag{5.2}$$

where T and X are characteristic time and space constants respectively. Substituting these transformations into equation (5.1) yields

$$\ddot{u} + \frac{c}{m}T\dot{u} + \frac{k_L}{m}T^2 u + \frac{k_N}{m}T^2 X^2 u^3 = \frac{fT^2}{mX}\cos\omega T t \; . \tag{5.3}$$

Choosing $T = \frac{1}{\omega_0}$ where $\omega_0 = \sqrt{\frac{k}{m}}$ results in

$$\ddot{u} + 2\mu\dot{u} + u + \frac{k_N}{k_L}X^2 u^3 = \frac{f}{k_L X}\cos\Omega t \; , \tag{5.4}$$

where $\mu = \frac{c}{2m\omega_0}$ and $\Omega = \frac{\omega}{\omega_0}$. This equation can also be written as

$$\ddot{u} + \epsilon 2\mu\dot{u} + u + \epsilon\alpha u^3 = F\cos\Omega t \; , \tag{5.5}$$

where α indicates the strength of the nonlinearity, and ϵ is a scaling factor that indicates that the damping and nonlinear springs term are orders of magnitude smaller than the other terms.

This system possesses interesting dynamics that are discussed in the literature [40, 41]. The frequency response in the region of primary resonance is considered here. For primary resonance, equation (5.5) is rewritten as

$$\ddot{u} + \epsilon 2\mu\dot{u} + u + \epsilon\alpha u^3 = \epsilon F \cos\Omega t \ , \tag{5.6}$$

to scale the force acting on the system. The frequency response is determined by the following equations [40, 41]

$$\begin{aligned} \mu a &= \frac{1}{2}\frac{F}{\omega_0}\sin\gamma \ , \text{ and} \\ a\sigma - \frac{3}{8}\frac{\alpha}{\omega_p}a^3 &= -\frac{1}{2}\frac{F}{\omega_0}\cos\gamma \ , \end{aligned} \tag{5.7}$$

where a is the amplitude of the response, γ is the phase angle, and σ is a detuning parameter related to the frequency by $\Omega = \omega_0 + \epsilon\sigma$. Solving the equation above for a yields

$$\left[\mu^2 + \left(\sigma - \frac{3}{8}\frac{\alpha}{\omega_0}a^2\right)^2\right] a^2 = \frac{F^2}{4\omega_0^2} \ , \tag{5.8}$$

a cubic equation in a^2. A typical response is presented in chapter 4, figure (4.15).

The NNUM uses the real and imaginary components of the FRF to produce parameter estimates. These quantities are obtained by further manipulation of equation (5.7), yielding

$$\begin{aligned} \Re(FRF) &= a\left(\frac{3}{8}\frac{\alpha}{\omega_0}a^2 - \sigma\right) \ , \text{ and} \\ \Im(FRF) &= a\mu \ . \end{aligned} \tag{5.9}$$

A typical behavior of the real and imaginary components of the FRF during forward and backward sweeps is shown in figure (5.15).

The amplitude of excitation F and the linear natural frequency ω_0 are considered to be constant and equal to 0.6 and 1.0, respectively. The damping and nonlinear spring parameters are allowed to vary about their nominal values of 1.5 and 0.25, respectively, according to a normal distribution with the following characteristics

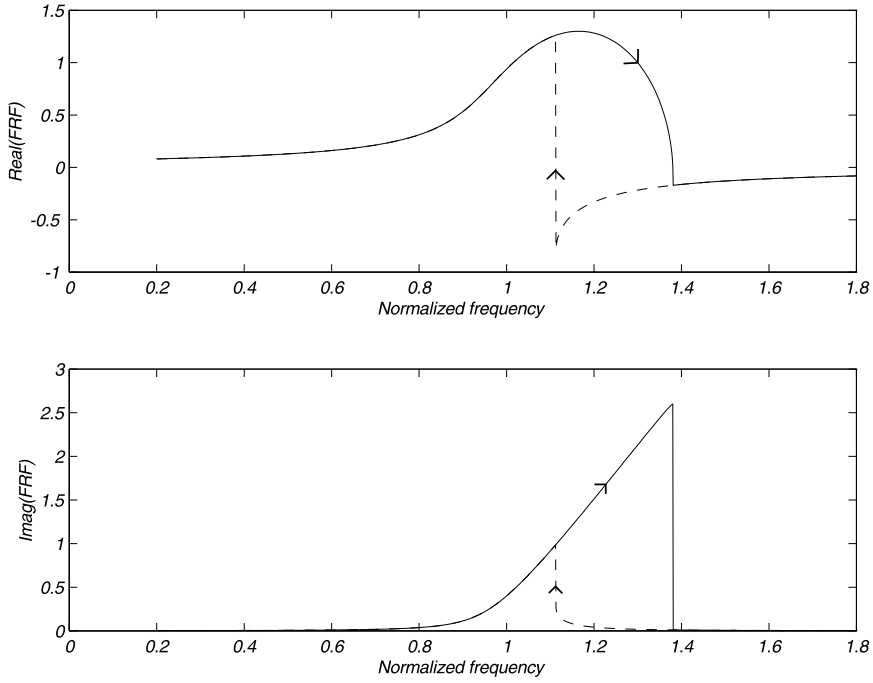

Figure 5.15: Real and imaginary components of the frequency response function of the Duffing oscillator.

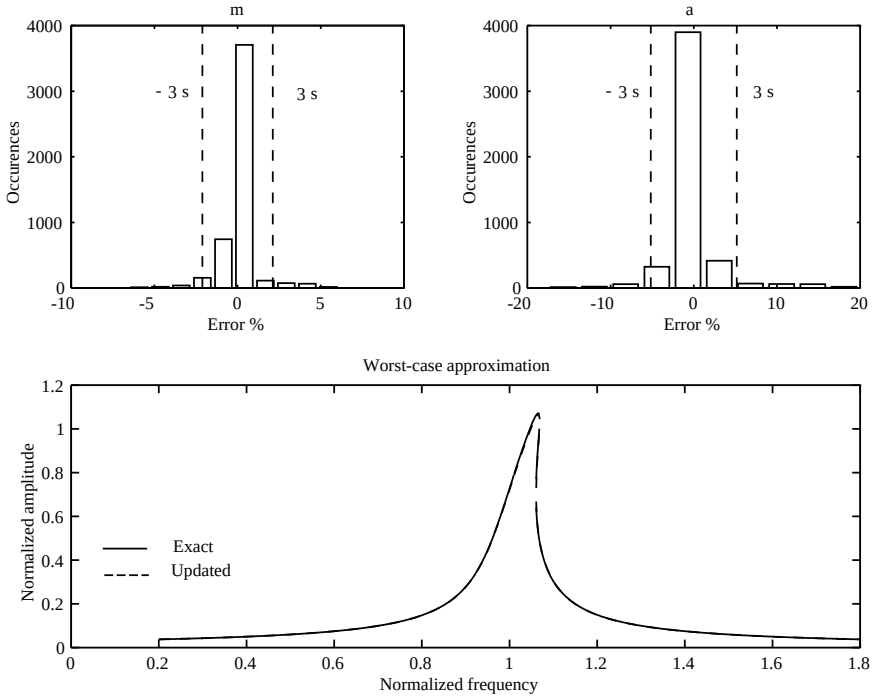

Figure 5.16: Generalization of the network trained to solve the Duffing oscillator problem.

i. μ varies as $0.25 \pm 30\%$ with $\sigma^2 = 0.04$,

ii. α varies as $1.5 \pm 30\%$ with $\sigma^2 = 0.25$.

Variations in α and μ are equivalent to variations in c and k_N, since the relationship between these parameters is linear.

The network is trained with 300 training samples resulting in the generalization characteristics shown in figure (5.16). The error measures are

$$e^{avg} = 0.47\% \text{ for } \mu \text{ and } 1.43\% \text{ for } \alpha \text{ ,and}$$

$$e^{max} = 6.53\% \text{ for } \mu \text{ and } 19.9\% \text{ for } \alpha.$$

This example shows that the NNUM, unlike existing updating techniques, successfully updates the model of a weakly nonlinear system based on its frequency response. The

solution is obtained very much like for linear systems, the only difference being that weakly nonlinear systems require information about the forward and backward sweep responses. Since the response of a weakly nonlinear system depends strongly on the system parameters and forcing level, large parameter uncertainties can lead to very large training set sizes that adequately represent the input space.

5.6 Cantilevered Beam Experiment

A cantilevered beam, with characteristics shown in table (5.4), is used to verify this technique using measured data. The difficulty in this example comes from the fact that the response is very sensitive to the parameters being updated. The beam is modeled with six beam elements (12 active DOF) and proportional damping ($C = \alpha M + \beta K$) is assumed since the structure is very lightly damped. The beam is mounted on a magnetic shaker that provides the beam with base excitation, and accelerometers are placed at the base (node 1) and on node 3, as shown in figure (5.17). The beam is excited with a 5–1000 Hz, 0.4 Vrms random signal and analytical transfer functions between the two accelerometers are used in the training process. The details of the experimental setup are described in appendix A. The coefficients α and β and the mass and inertia of the accelerometer mounted on the beam are selected for updating because they strongly affect the response of the beam. These parameters are unknown, but assumed to be bounded by

i. m_{acc} varies as 10 g $\pm$ 30% with variance $\sigma^2 = 2$ g^2

ii. I_{acc} varies as 1.5×10^{-6} N m $\pm$ 50% with variance $\sigma^2 = 3.5 \times 10^{-7}$ (N m)^2

iii. α varies as $1.5 \pm 66.6\%$ with variance $\sigma^2 = 0.70$

iv. β varies as $2.0 \times 10^{-5} \pm 75\%$ with variance $\sigma^2 = 9 \times 10^{-6}$.

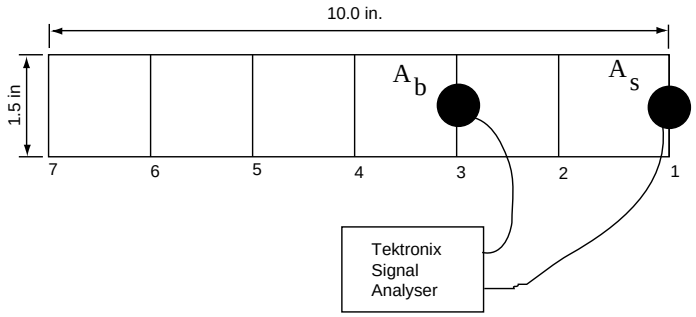

Figure 5.17: Cantilevered beam: experimental setup

Property	Symbol	Value
Young's modulus	E	1×10^7 psi
Mass density	ρ	0.1 lb/in^3
Length	L	10.0 in.
Thickness	t	0.0625 in.
Width	w	1.5 in.

Table 5.4: Physical characteristics of the beam

The frequency response function between the two accelerometers can be calculated analytically by

$$\frac{A_b(\omega)}{A_s} = \omega^2 B(\omega)^{-1} M + 1, \tag{5.10}$$

where A_b is the acceleration of node 3, A_s is the acceleration of the base, ω is the frequency, and $B(\omega) = -\omega^2 M + \jmath\omega C + K$ is the dynamic stiffness matrix.

The frequency intervals used to integrate the FRF are [86.4 141], [636 699], [1932 2058], and [3770 4131] rad/s (fig. 5.18). Using the integral of the real and imaginary parts of every interval yields poor results because the first and fourth modes have very low sensitivity to changes in the parameter α. To overcome this problem, only the integral of the real part of the first and fourth modes are used in the updating. The network is trained with 1000 samples and no weighting of the input matrix. The updated parameters are $m_{acc} = 12.80$ grams, $I_{acc} = 1.45 \times 10^{-6}$ N m, $\alpha = 1.60$, and $\beta = 2.33 \times 10^{-6}$. Since the actual value of

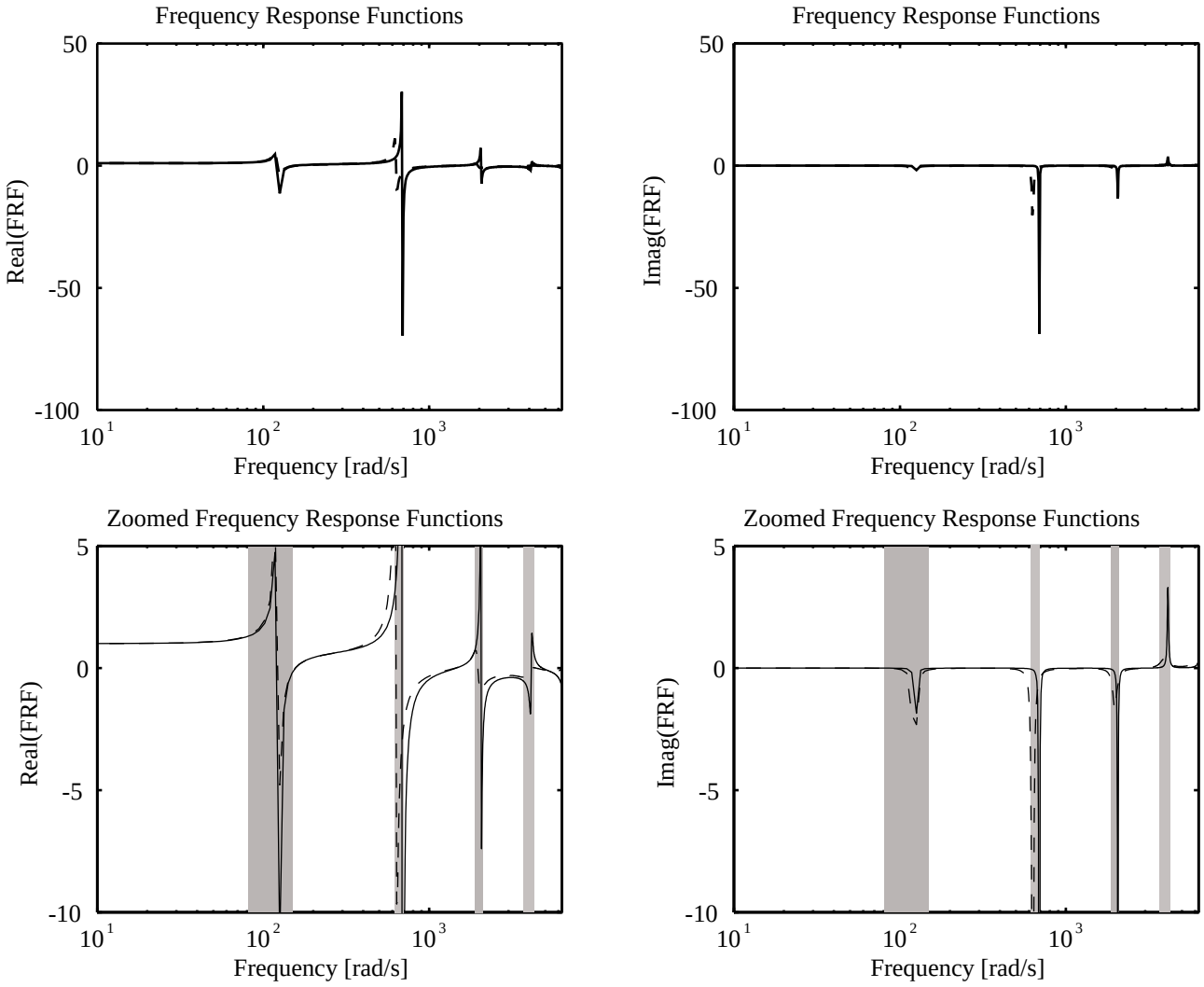

Figure 5.18: Frequency intervals used in the cantilevered beam experiment.

Measure	m_{acc}	I_{acc}	α	β
e^{avg}	0.76%	1.22%	3.52%	2.80%
e^{max}	11.9%	35.3%	82.6%	86.1%

Table 5.5: Error measures for the cantilevered beam experiment.

the mass is 10.44 grams, the network is trained once again, this time using weights for the real part of the third mode and for the imaginary parts of the first and third modes. These rows of the input matrix are multiplied by 0.4. This is done because the second mode is the one with the highest approximation error. The new updated values obtained with these weights are $m_{acc} = 10.20$ grams, $I_{acc} = 1.08 \times 10^{-6}$ N m, $\alpha = 1.36$, and $\beta = 2.99 \times 10^{-6}$. The results for both unweighted and weighted updating are shown in figure (5.19).

Figure (5.20) shows the generalization error for the network trained with weighted inputs. Five thousand possible combinations are used to verify the generalization of the network resulting in the following error measures shown in table (5.5).

Because the NNUM uses intervals in the frequency domain to update the model, it is possible to exclude frequency intervals that are not reliable. This is the case here, where the first torsion mode (≈ 160 Hz) is excited by the accelerometer wiring. Another characteristic is the versatility of the method allowing the engineer to use only the necessary information when updating the model. In this case the integral over the first and fourth modes of the imaginary part of the FRF are discarded because they are nearly constant, and therefore do not provide any information to the network. The updated model, despite some error in estimating the mass, closely matches the measured data, making for a great improvement over the initial model.

5.7 Flexible Frame Experiment

The last example is a flexible frame designed to study and actively control the vibrations of solar panels (fig. 5.21) [42, 43]. The frame consists of rectangular and circular thin-

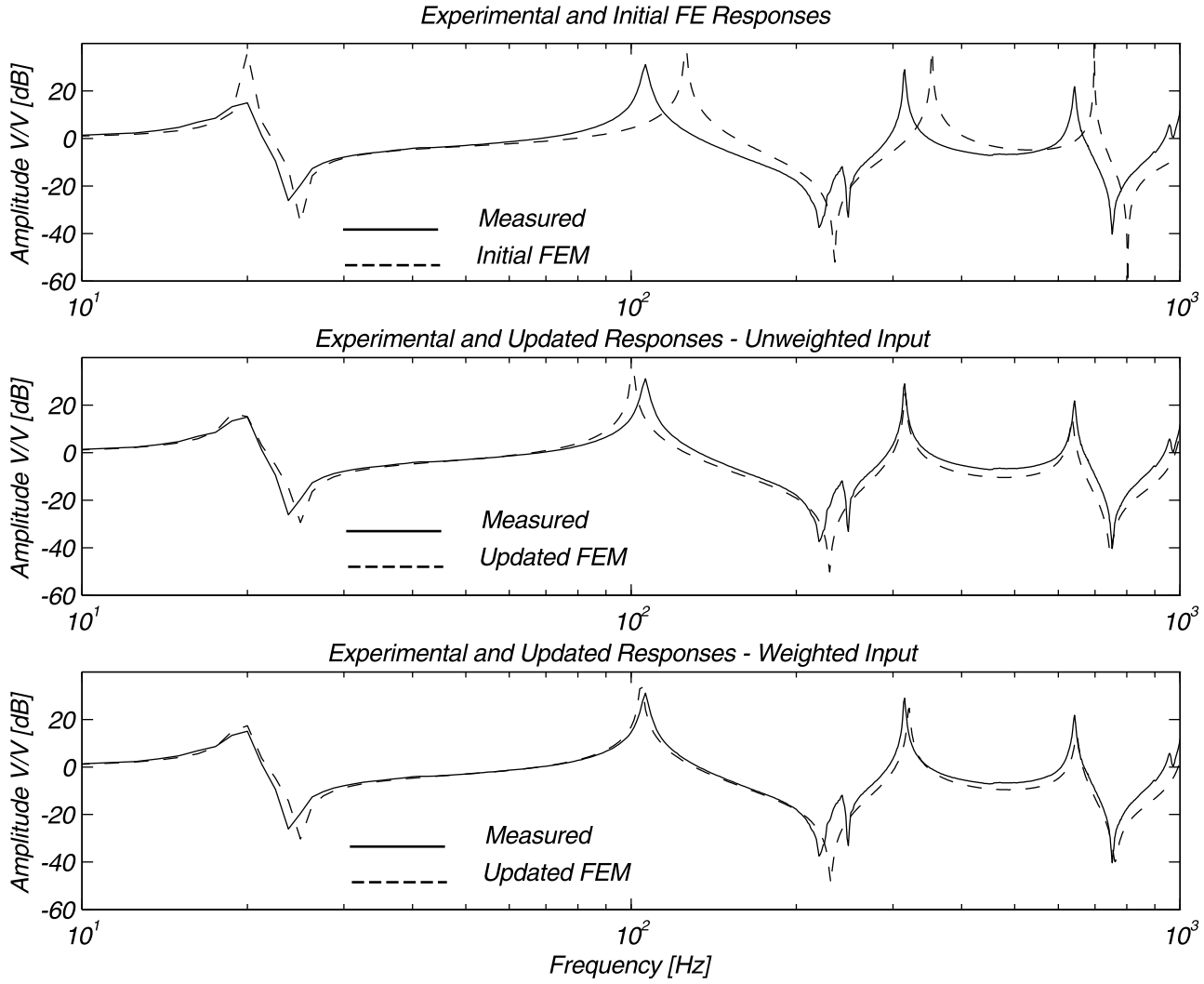

Figure 5.19: Measured, initial and updated frequency responses of the cantilevered beam.

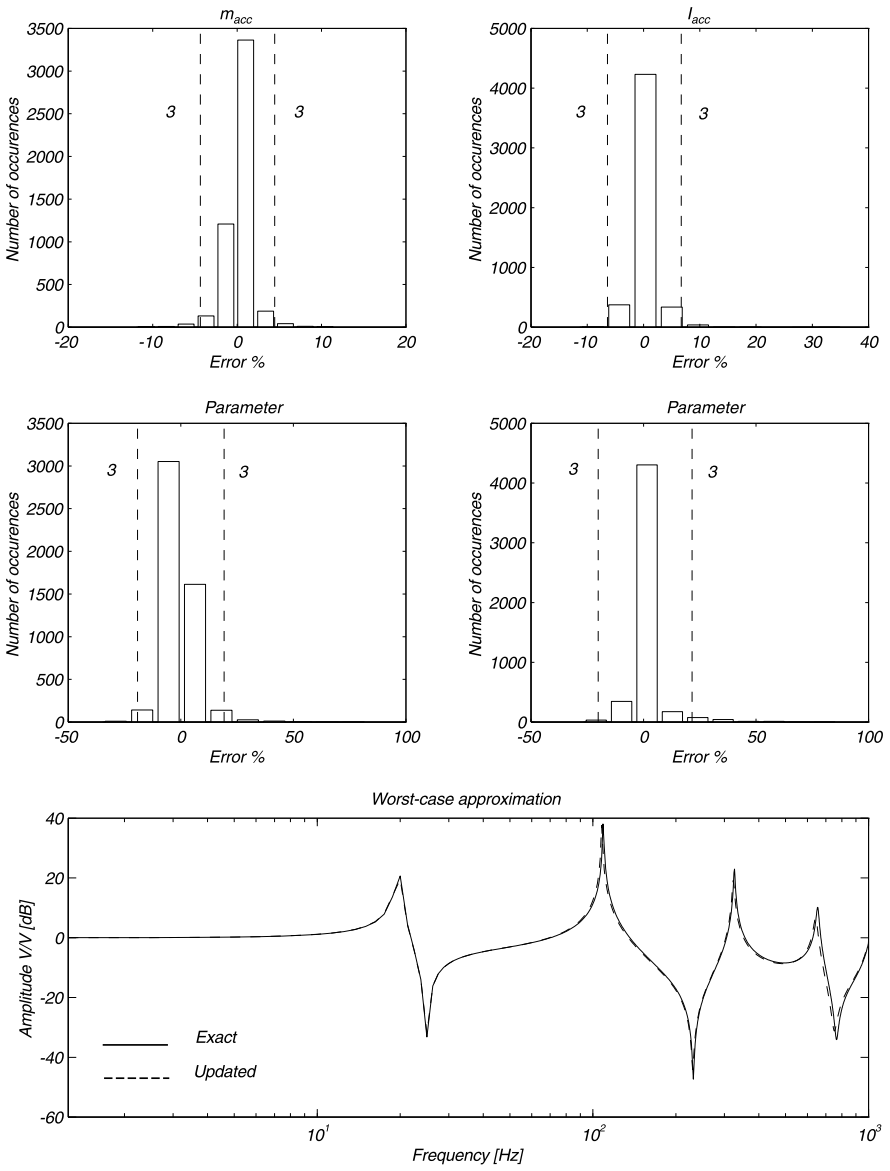

Figure 5.20: Generalization characteristics of the network trained to solve the cantilevered beam experiment.

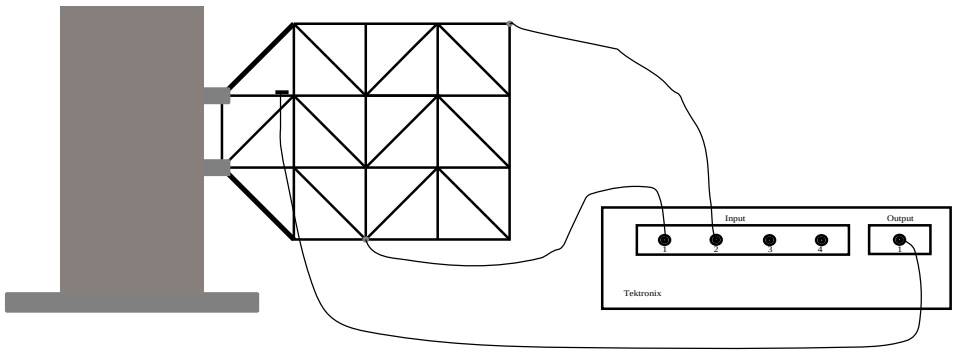

Figure 5.21: Flexible frame diagram.

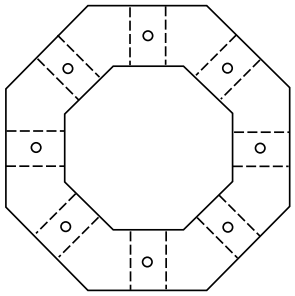

Figure 5.22: Diagram of the joints used in the flexible frame.

walled aluminum tubes connected by octagonal aluminum elements (fig. 5.22). Each tube is pinned and bolted into the connection element to avoid slippage. The frame is attached to a concrete block that serves as ground, as long as the level of excitation is low so its natural frequencies are not excited.

The structure is modeled with beam and single-point mass elements, resulting in a model with 96 active degrees-of-freedom (5.23). The assumptions in the model are that the octagonal joints behave as single-point masses, implying that the joint is a rigid member and that its rotatory inertia is negligible, and that the beams meet at the center of the octagonal joints, resulting in longer beams in the FE model. This causes an increase of mass in the FE model and a reduction of its stiffness, since the coefficients of the stiffness matrix are inversely proportional to the length of the beam. These assumptions lead to a

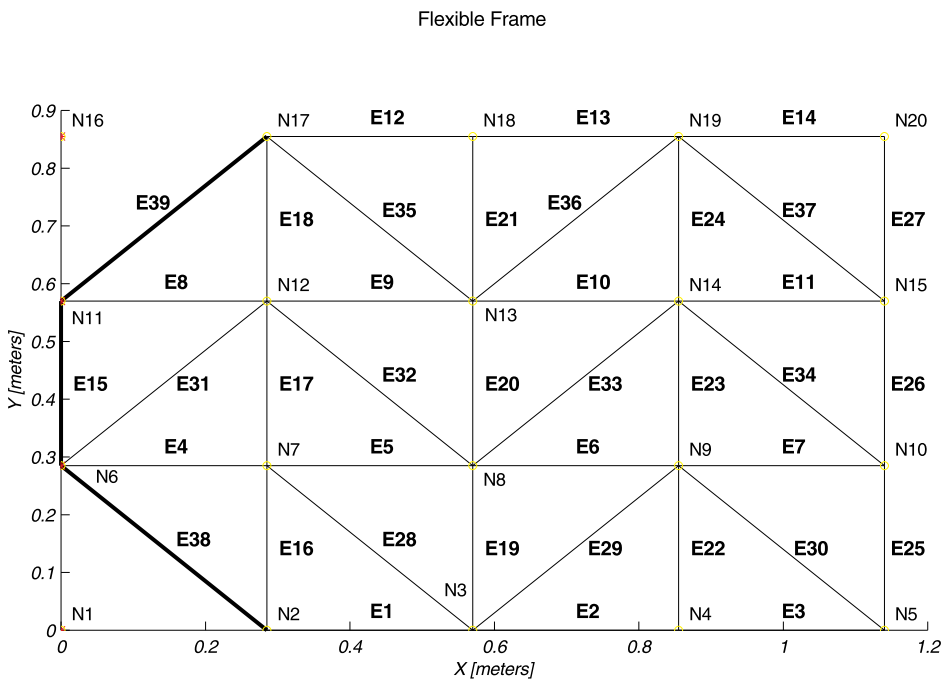

Figure 5.23: Finite element mesh for the flexible frame experiment.

decrease of the natural frequencies of the FE model. The rotatory inertia and applied torque to the bolts are neglected in the model. The masses in the FE model are determined by measurement. One joint was disassembled and its components (pins, bolts, and octagonal connection element) weighed. The average values obtained for this joint are used for all remaining joints. The mass and moment of inertia of the accelerometers were taken to be the values estimated in the cantilevered beam experiment. Nodes N1, N6, N11, and N16 of the finite element model have all degrees of freedom constrained, and elements E38 and E39 in figure (5.23) are rectangular aluminum beams used in previous research to actively control the frame's vibration. Since the structure is very lightly damped, proportional damping is assumed ($C = \alpha M + \beta K$). The parameters of the frame are listed in table (5.6).

In this example the possibilities of sensor failure and parameter drift during the lifetime of the structure are considered. Finding an equivalent length (L) for the beams allows the adjustment of the mass and stiffness parameters of the FE model and the detection

Circular beams	Rectangular beams	Joints	Pin+Bolt
E = 69×10^9 Pa	E = 69×10^9 Pa	m $\approx$ 9.7 g	m $\approx$ 3.2 g
$\rho = 2710$ kg/m^3	$\rho = 2710$ kg/m^3		
R = 3.2 mm	b = 2.86 cm		
r = 2.54 mm	h = 3.2 mm		

Table 5.6: Physical parameters of the components of the flexible frame.

of changes in the mass or stiffness of the structure. Since the equivalent length is used to correct all beams of the structure this detection occurs in a global sense, not allowing for the location of the element that caused the change. It is also possible that a sensor fails and must be replaced. However, it is experimentally verified that the response of this structure is not very sensitive to changes in mass in the positions where the accelerometers are mounted. A 2.3-gram mass was added to the accelerometer resulting in a change of at most 0.25 Hz in the natural frequencies. Since the structure is not very sensitive to changes in the accelerometer parameters they are not included in the updating procedure.

Frequency responses are experimentally obtained by exciting the frame at node N12 with an impact hammer and measuring accelerations at nodes N3 and N20 with two accelerometers (fig. 5.21). The details of the experimental setup are presented in appendix A. The measured and predicted undamped FE responses of the accelerometer located at node N20 are shown in figure (5.24), along with its coherence information. Data collected from the accelerometer located at node N20 is used because its signal has the highest quality. As expected, the finite element model has natural frequencies lower than those measured experimentally.

The FE model needs to be updated in order to predict the behavior of the structure accurately. The model is adjusted by updating the α and β damping coefficients and by finding an equivalent length (L) for the beams to compensate for the added mass and reduced stiffness. In addition to these three parameters, the mass of the bolt+pin combination is also updated. This is necessary because the pins are of different lengths and the bolts

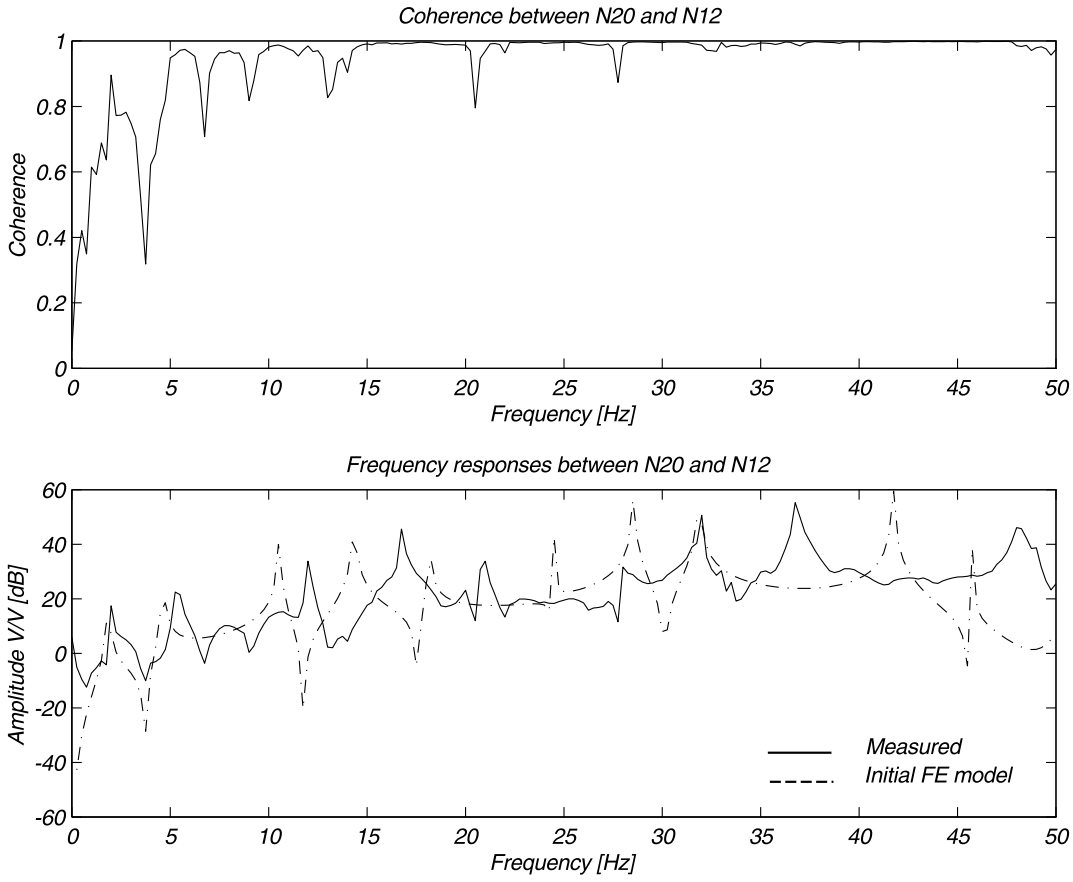

Figure 5.24: Experimental data obtained for the flexible frame experiment.

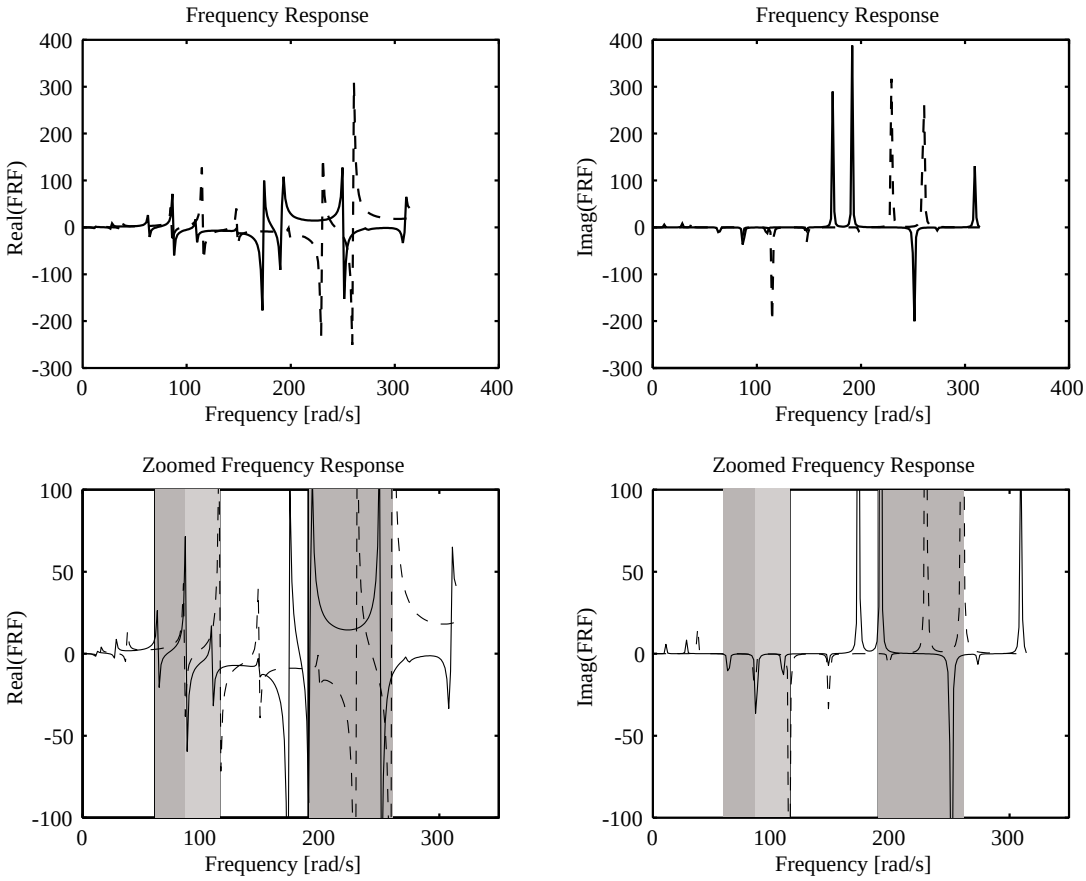

Figure 5.25: Frequency intervals used in updating the flexible frame model.

are not identical. The frequency intervals selected to be used are [65 86], [86 114], and [190 260] rad/s (fig. 5.25). The low frequency modes were not used because of low measured coherence in these modes.

The equivalent length is used to scale the mass and stiffness matrices of the beam elements as follows. The coefficients of the mass matrix are of the form $CA\rho L_n$, where C is a number, ρ is the mass density, and L_n is the length of the beam element. Based on this, the mass matrix of each beam element is multiplied by $\frac{L}{L_n}$, where L is the equivalent length used to correct the FE model. The coefficients of the stiffness matrix are inversely

proportional to the length of the element. For scaling purpose, the coefficient $12\frac{EI}{L_n^3}$ is used because the most important modes of the beams in the frequency range being considered are primarily bending. The stiffness matrix of each beam element is multiplied by $\left(\frac{L_n}{L}\right)^3$. Since the mass of the initial FE model is higher than the real mass and since the the stiffness coefficients are lower than the real ones, it is expected, based on the expressions above, that the equivalent length will be shorter than its nominal value. This results in a reduction of the mass density and in an increase of the Young's modulus, making the updated model closer to the real structure.

The parameters being updated are allowed to vary as follows

i. The equivalent length L varies as 27 cm $\pm$ 2 cm with variance $\sigma^2 = 1.3$ cm^2,

ii. The mass of the combination bolt+pin, m_{bp}, varies as 3.2 g $\pm$ 1.2 g with variance $\sigma^2 = 0.9$ g^2,

iii. α varies as $1.0 \pm 80\%$ with variance $\sigma^2 = 0.55$, and

iv. β varies as $8.0 \times 10^{-6} \pm 75\%$ with variance $\sigma^2 = 4 \times 10^{-6}$.

The network is trained with 1200 sample pairs and verified with 5000 samples. The imaginary components of the input matrix are multiplied by a weighting factor of 0.7 to increase the accuracy in estimating the equivalent length. The results in figure (5.26) show that the NNUM updates the parameters L and m_{bp} accurately, but not the damping parameters α and β. This is explained by the low sensitivity of the response with respect to these parameters. This also means that an error of 50% in these parameters is not a problem because they do not affect the response significantly. Figure (5.27) shows the response of the model updated using the measured data. The updated parameters are

- $L = 27.29$ cm,
- $m_{bp} = 2.63$ g,

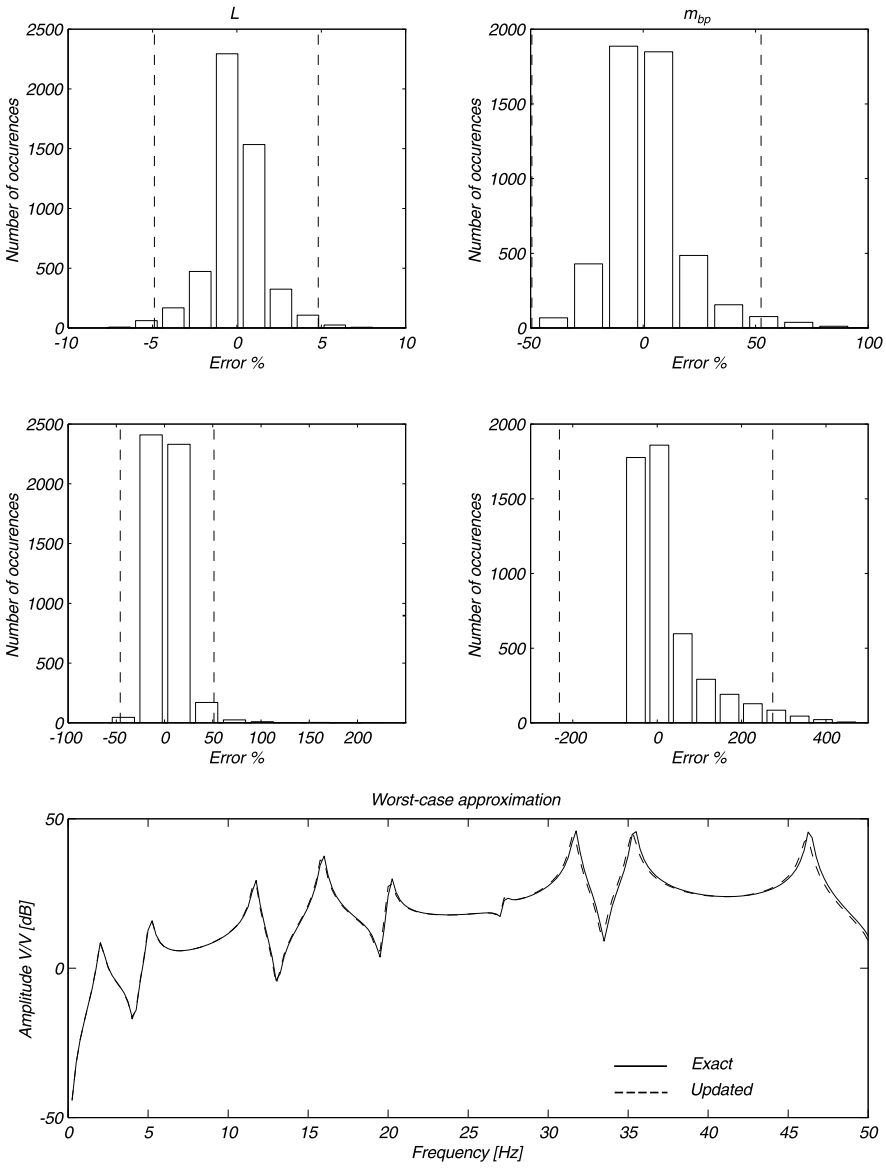

Figure 5.26: Generalization of the network trained to update the flexible frame model.

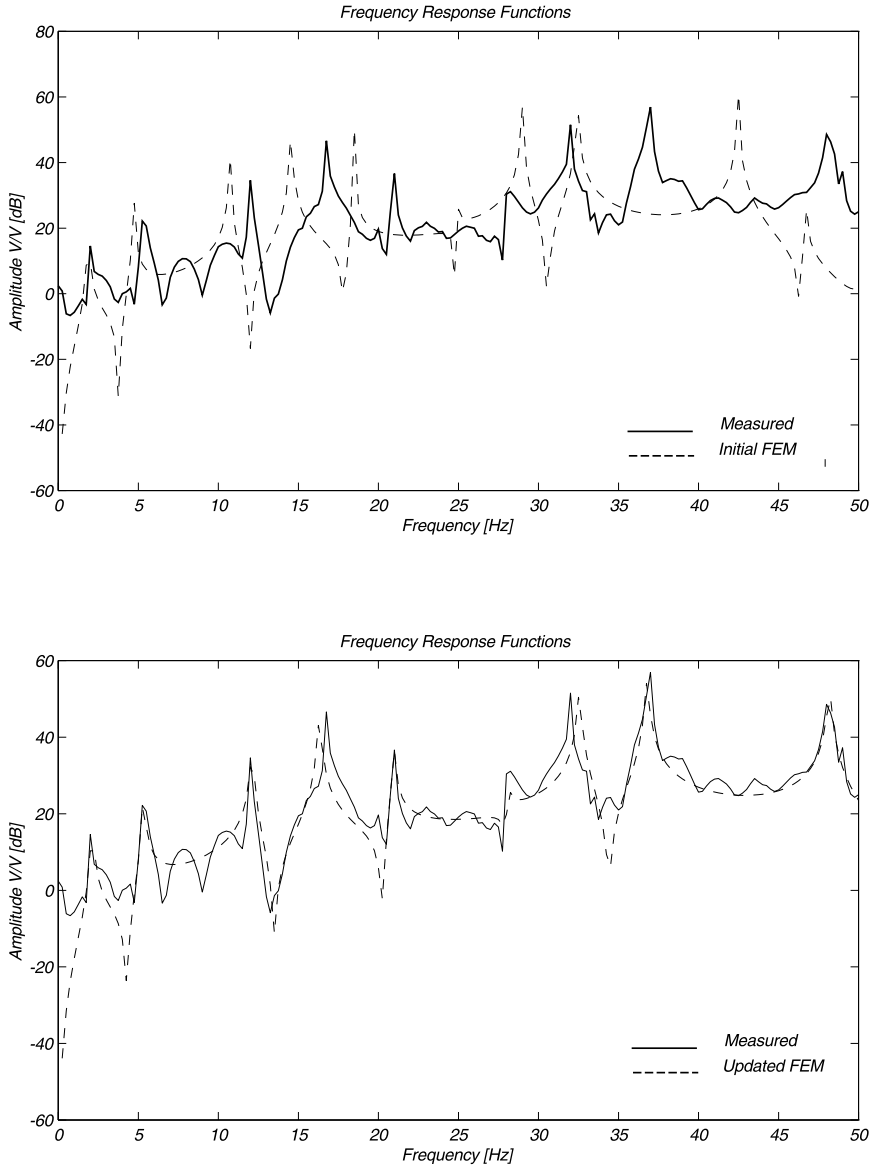

Figure 5.27: Measured, initial and updated responses of the flexible frame experiment.

Measure	L	m_{bp}	α	β
e^{avg}	1.11%	11.7%	10.5%	54.9%
e^{max}	8.19%	108%	231%	476%

Table 5.7: Error measures for the flexible frame experiment.

- $\alpha = 0.77$, and
- $\beta = 5.1 \times 10^{-6}$.

The error measures are listed in table (5.7). As mentioned in previous sections, the parameter combinations that result in large errors are the ones very close to the boundaries of parameter variations and therefore not representative of the majority of cases.

This example demonstrates the versatility and accuracy of the NNUM when used to update a complex structure using experimental data. Once again, use is made of weighting factors, incorporating engineering knowledge into the procedure. This example uses global equivalent parameters to update the structure due to the large amount of uncertainties, not allowing for localized updating of parameters. However, this is a more complex case since all the effects are averaged, making it more difficult for the NNUM to update the model.

5.8 Summary

The numerical simulations and experiments presented in this chapter show that the NNUM successfully solves the model updating problem of a variety of systems. Kabe's and Yang's examples illustrate its accuracy in updating systems with high modal density. The NNUM handles damping very well, as illustrated in the second variation of Kabe's problem. The truck suspension problem shows that the NNUM can be successfully used as the estimation technique of an adaptive LQR controller that has its gain matrix updated based on the estimates produced by this technique, while the Duffing oscillator example shows NNUM's accuracy in estimating the parameters of a weakly nonlinear system. The

two experiments demonstrate the NNUM's robustness in dealing with noise and unmodelled dynamics, as it is the case of the first torsion mode of the cantilevered beam, and its versatility in allowing the exclusion of unreliable data, as is the case in the frame experiment.

The computational effort of solving the model updating problem is concentrated in the training phase of the neural network. Once the network is trained, solutions to the model updating problem are quickly and accurately obtained. All these characteristics make this technique a valuable tool to update models of structures that change over time.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

The neural network updating method developed in this dissertation has been shown to be an effective tool to solve the model updating problem. The main advantage of using radial basis function neural networks is fast updating of a model once it has been trained. Choosing to integrate the real and imaginary components of the frequency response functions allows for a uniform approach to updating linear and weakly nonlinear systems. The main drawback of this approach is the intensive computational load required to train the neural network. Another drawback is the lack of mathematical tools to assure convergence and accuracy of the network's estimation as a function of the number of inputs and training pairs. These drawbacks, however, did not prevent the solution of difficult numerical and experimental problems. In studying these problems, trade-offs between accuracy and computational load have been analyzed, leading to general guidelines to be followed when using this technique.

The existing model updating literature is very rich, but lacks in tools to quickly update systems once they are in use. The NNUM solves this problem by using a neural network to represent the mapping between the system's response and the system's parameters. The

fidelity of this mapping is a function of many variables. The power and limitations of this technique are explained during its development in chapter 4. The data used to train the neural network must reflect the expectations of how the parameters vary. All the problems solved in this dissertation assume a normal distribution with known mean and variance, however, any probability density distribution can be used.

Due to the nature of the distance measures, it was necessary to use input data other than the frequency response functions themselves. This comes from the inherent ambiguity of Minkowski-like distance measures. To circumvent this problem, a new form of input data was defined to be the integral of the real and imaginary components of the frequency response function over selected frequency intervals. This choice of input data reduces the effect of zero-mean noise, improving the estimation of parameters. The drawback is that the data is smoothed by the integration, possibly leading to inaccuracy in estimating parameters. This is avoided by the proper selection of the frequency intervals, as explained in chapter 4.

Normalization and scaling of input data have also been shown to be very important issues. It is essential to normalize the input data so it lies between known bounds, preferably [-1, 1] or [0, 1]. Once the input is normalized it becomes necessary to scale the rows of the input matrix to improve the discerning capabilities of the hidden neurons. This is successfully done by using the standard deviation of the rows of the input matrix as scaling factors. This choice assures good accuracy in most of the cases, however, in critical cases, it can be refined by using the gradient-descent-based formulation presented in chapter 4. The formulation developed to deal with the normalization and scaling of input data also corroborates the choice of input data. Integrating the response leads to lower variance values for the rows of the input matrix, resulting in better accuracy and noise handling capabilities.

Engineering knowledge is used in many steps of the process. It starts with selecting the bounds and probability density distribution of the parameters being updated, continues with the selection of frequency intervals used to integrate the frequency response functions, and

ends with the selection of weighting factors. Weighting factors are made available so that important features of the input data can be enhanced to yield more accurate parameter estimates. This feature was used in both experimental problems solved in chapter 5 to enhance the estimation of mass parameters over damping coefficients. This is strictly a personal choice that depends on the design requirements of the system being updated.

The number of training samples directly influences the accuracy of the procedure. The more training samples available, the more accurate the estimated parameter. There is, however, a trade-off between accuracy and computational load. The more training samples, the higher the computational cost of training the network. The accuracy of the procedure is also a function of how many parameters are updated. In general, for a fixed number of training samples, more parameters means lower accuracy. This leads to another general rule: only the most relevant parameters should be updated.

The influence of the number of different frequency response functions used to compose the input matrix was also investigated. Feeding the network with frequency response functions between different points of the structure leads to an improvement in the estimation of parameters. There is, however, a point beyond which very little improvement is obtained because the network saturates its ability to recognize features in the input data. Further improvement can only be achieved by using more training samples.

The computational cost to train the neural network is a cubic polynomial in the number of training samples. The *flops* measure of computational load allows the proper choice of training set size based on the available computer hardware. The process of estimating the parameters is fast once the neural network has been trained. It is a linear function of the training set size and it takes tenths of a second to be computed in modern computer hardware.

A broad variety of systems has been used to test the NNUM. Kabe's example served to point a strong aspect of the proposed technique: its ability to handle damping. The existence of damping improves the accuracy of the NNUM because it reduces the variance of the rows of the input matrix. This example also shows that systems with very high modal

density and sensitivity to parameter changes require a large training set, leading to longer training periods.

The 15-degree-of-freedom system presented the NNUM with the challenge of updating a system with high modal density, repeated modes, and high damping. This example illustrates well the power of the proposed technique. All parameters can be estimated within 5% accuracy in the case of high damping with roughly 500 training samples. The updated model accurately estimates the response of out-of-band modes.

Being able to quickly update the system's model allows the use of this technique as part of an adaptive control scheme. This is verified by updating an LQR controller based on a model updated by the NNUM. The truck suspension problem solved in chapter 5 compares the ideal case with the results obtained by using this technique and shows that both cases yield very similar results.

Weakly nonlinear systems can also be updated by the proposed technique. This is verified in a simple example, the Duffing oscillator. For this class of systems, the input data is slightly modified to include information about the forward and backward frequency responses of the system. Good results are obtained for the case when the damping and nonlinear stiffness parameters are updated. However, the response of such systems can be very complex, making the use of frequency-domain-based techniques infeasible. The response is also very sensitive to changes in the system parameters, and, as mentioned before, this leads to very large training sets.

Experiments have been performed to verify the behavior of the NNUM when measured data is used as input. The cantilevered beam problem shows that the technique is accurate in updating relevant parameters of the system, such as the accelerometer mass and inertia and the damping coefficients. This experiment also illustrates the ability to exclude data known to be inaccurate or to represent unmodeled dynamics. The flexible frame is a complex structure with high modal density and model uncertainties. The NNUM successfully updates the model yielding a updated response that closely matches the measured data. A nonphysical parameter is used to correct the mass density and Young's modulus

of the structure. The use of a global parameter like this, one that corrects all elements of the structure, improves the accuracy of the NNUM by reducing the number of parameters being updated, but it increases the difficulty in finding the value that results in a good updated model. The NNUM successfully estimates this parameter.

The proposed technique adds to the current model updating literature by dealing with systems that can change in time in an efficient manner. The use of neural networks, an increasingly popular technique, has been used for the first time to solve the model updating problem. Using the integral of the real and imaginary parts of the frequency response solves the problem of ambiguity when measuring distances and also minimizes the effect of noise in the input data. Integrating the response also offers the ability to deal with linear and weakly nonlinear systems in a similar way. However, despite the good performance of the proposed technique, one should answer the following questions before using it:

- Is the system expected to change while in operation?
- Is the computational cost of training ($O(N_t^3)$) and validating ($O(N^3)$) the network feasible?
- Is the error margin of the network acceptable?

If the answer to these questions is affirmative, then the technique should be used. The numerical simulations and experimental results show that the proposed technique is an efficient tool in solving the model updating problem.

6.2 Future Work

A next step would be to implement the NNUM as part of a real-time control scheme and verify the practical implications of the estimation error in the stability of the closed-loop response.

This dissertation uses analytical frequency response functions to train the network and measured transfer functions to update a model. Since model uncertainties and noise are

often present in measured data, it may be advantageous to train the network with analytical frequency response functions corrupted with noise. This requires some model for the noise present in the measurement, a piece of information often available.

The high computational cost of generating the training samples and training the network is a limiting factor of this technique. Another area not yet explored by the model updating community is the use of time-domain-based estimation techniques. It may be possible to modify existing time-domain estimation techniques to yield estimates of model parameters. In the field of neural networks, the use of recurrent neural networks, networks that possess dynamics and are time-dependent, may provide a solution to this problem. These networks are able to emulate the dynamics of a system, like a "black-box" between the system's input and output. The issue is: can this network can be modified in a way that allows for the estimation of model parameters?

Bibliography

[1] M. I. Friswell and J. E. Mottershead. *Finite Element Model Updating in Structural Dynamics*. Kluwer Academic Publishers, 1995.

[2] Ching I. Chen, Marcello R. Napolitano, and Chih-Liang Chen. A new learning algorithm for neural network estimation in active vibration control. *Smart Materials and Structures*, 1:250–257, 1992.

[3] Vittal Rao, Rajendra Damie, Chris Tebbe, and Frank Kern. The adaptive control of smart structures using neural networks. *Smart Materials and Structures*, 3:354–366, 1994.

[4] Z. P. Szewczyk and Prabhat Hajela. Neural network based selection of dynamic system parameters. *Transactions of the Canadian Society of Mechanical Engineers*, 17(4A):567–584, 1993.

[5] Shankar Sastry and Marc Bodson. *Adaptive Control - Stability, Convergence, and Robustness*. Prentice Hall, 1989.

[6] Karl Johan Åström and Björn Wittenmark. *Adaptive Control*. Addison Wesley, 1989.

[7] Alvar M. Kabe. Stiffness matrix adjustment using modal data. *AIAA Journal*, 23(9):1431–1436, 1985.

[8] Michael Yang and David Brown. An improved procedure for handling damping during finite element model updating. In 14^{th} *International Modal Analysis Conference*, pages 576–584, Dearborn, Michigan, 1996.

[9] R. J. Allemang and D. L. Brown. A correlation coefficient for modal vector analysis. In *First International Modal Analysis Conference*, pages 110–116, Orlando, Florida, 1982.

[10] N. Niedbal. Analytical determination of real normal modes from measured complex responses. In 25^{th} *Structures, Structural Dynamics and Materials Conference*, pages 292–295, Palm Springs, May 1984.

[11] R. J. Guyan. Reduction of stiffness and mass matrices. *AIAA Journal*, 3(2):380–, 1965.

[12] J. O'Callahan, P. Avitabile, and R. Riemer. System equivalent reduction expansion process. In 7^{th} *International Modal Analysis Conference*, pages 29–37, Las Vegas, 1989.

[13] J. E. T. Penny, M. I. Friswell, and S. D. Garvey. The automatic choice of measurement locations for dynamic tests. In 10^{th} *International Modal Analysis Conference*, pages 30–36, San Diego, February 1992.

[14] D. C. Kammer. Sensor placements for on-orbit modal identification and correlation of large space structures. *Journal of Guidance, Control and Dynamics*, 14(2):251–259, 1991.

[15] D. C. Kammer. Effect of noise on sensor placement for on-orbit modal identification and correlation for large space structures. *Journal of Dynamical Systems, Measurement and Control - Transactions of ASME*, 114(3):436–443, 1992.

[16] G. Lallement and J. Piranda. Localisation methods for parameter updating of finite element models in elastodynamics. In 8^{th} *International Modal Analysis Conference*, pages 579–585, Orlando, Florida, 1990.

[17] H. G. Natke. Updating computational models in the frequency domain based on measured data: a survey. *Probabilistic Engineering Mechanics*, 3(1):28–35, 1988.

[18] C. Minas and D. J. Inman. Matching finite element models to modal data. *Transactions of the ASME, Journal of Vibration and Acoustics*, 112(1):84–92, 1990.

[19] S. W. Smith and C. A. Beattie. Secant-method adjustment to structural models. *AIAA Journal*, 29(1):119–126, 1991.

[20] M. J. Lam and D. J. Inman. Methods of preserving symmetry in model updating. *Journal of Vibration and Acoustics*, 117:349–354, 1995.

[21] Ladislav Starek and Daniel J. Inman. A symmetric positive definite inverse vibration problem with underdamped modes. In *1995 Design Engineering Technical Conferences*, volume 3C, pages 1089–1094, Boston, September 1995.

[22] M. I. Friswell and J. E. T. Penny. Updating model parameters from frequency domain data via reduced order methods. *Mechanical Systems and Signal Processing*, 4(5):377–391, 1990.

[23] P. Eykhoff. *System Identification: Parameter and State Estimation.* John Wiley & Sons, 1974.

[24] L. Ljung. *System Identification - Theory for the User.* Prentice Hall, 1987.

[25] R. M. Lin and D. J. Ewins. Model updating using frf data. In 15^{th} *International Modal Analysis Seminar*, pages 141–163, K. U. Leuven, Belgium, Sept 1990.

[26] S. S. Simonian. Inverse problems in structural dynamics i - theory. *International Journal of Numerical Methods in Engineering*, 17(3):357–365, 1981.

[27] S. S. Simonian. Inverse problems in structural dynamics ii - applications. *International Journal of Numerical Methods in Engineering*, 17(3):367–386, 1981.

[28] J. E. Mottershead and R. Stanway. Identification of structural vibration parameters by using a frequency domain filter. *Journal of Sound and Vibration*, 109(3):495–506, 1986.

[29] J. E. Mottershead. A unified theory of recursive, frequency domain filters with application to system identification in structural dynamics. *Journal of Vibration, Acoustics, Stress and Reliability in Design - Transactions of ASME*, 110(3):360–365, 1988.

[30] D. E. Rumelhart and J. L. McCleland. *Parallel Distributed Processing.* M.I.T. Press, 1986.

[31] K. Funahashi. On the approximate realization of continuous mappings by neural networks. *Neural Networks*, 2:183–192, 1989.

[32] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

[33] D. S. Broomhead and D. Lowe. Multivariate functional interpolation and adaptive networks. *Complex Systems*, 2:321–355, 1988.

[34] J. Park and I. W. Sandberg. Universal approximation using radial-basis-function networks. *Neural Computation*, 3(2):246–257, 1991.

[35] A. Cichocki and R. Unbehauen. *Neural Networks for Optimization and Signal Processing.* John Wiley & Sons, 1993.

[36] James A. Leonard and Mark A. Kramer. Radial basis function networks for classifying process faults. *IEEE Control Systems*, pages 31–38, Apr 1991.

[37] N. van De Wouw, G. Verbeek, and D. H. van Campen. Nonlinear parametric identification using chaotic data. *Journal of Vibration and Control*, 1(3):291–305, 1995.

[38] Brian D. O. Anderson and John B. Moore. *Optimal Control - Linear Quadratic Methods.* Prentice Hall, 1990.

[39] D. Brett Ridgely and Siva S. Banda. *Introduction to Multivariable Control - Tecnhical Report AFWAL-TR-85-3102.* Wright-Patterson Air Force Base, 1986.

[40] Ali H. Nayfeh and Dean T. Mook. *Nonlinear Oscillations.* John Wiley & Sons, 1979.

[41] Ali H. Nayfeh. *Introduction to Perturbation Techniques.* John Wiley & Sons, 1980.

[42] Donald J. Leo and Daniel J. Inman. Modeling and control simulations of a slewing frame containing active members. *Smart Materials and Structures*, 2:82–95, 1992.

[43] J. Dosch, D. J. Inman, and E. Garcia. A self-sensing piezoelectric actuator for collocated control. *Journal of Intelligent Materials, Systems and Structures*, 3:166–185, 1992.

Appendix A

Experimental Hardware

A.1 The beam experiment

The beam is clamped to a plate attached to the shaker's armature. Data is collected from the accelerometers mounted on the moving armature and on the beam, and input to a Tektronix Fourier analyzer. The Fourier analyzer filters the signal above 1 kHz and calculates the frequency response function between the two accelerometers (fig. A.1). The software used is the Instrument Program (IP) provided with the Fourier analyzer, and a Hanning window is used when acquiring the data. The hardware used in this experiment is

- Two accelerometers with the following characteristics

  Brand: Kistler

  Model: 8630B50

  Range: $\pm$ 50 g[1]

  Sensitivity at 100 Hz: 99.3 mV/g[1]

  Resonant frequency: 22 kHz

[1] g = 9.8065 m/s^2

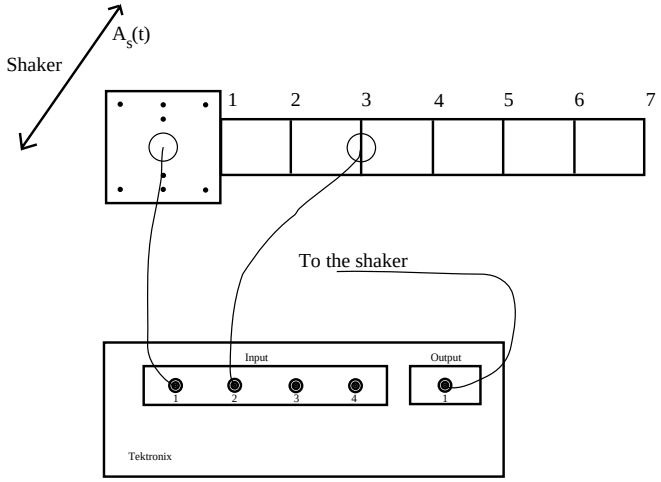

Figure A.1: Cantilevered beam: experimental setup.

Weight: 7.5 g[2]

- One Fourier analyzer with the following characteristics

  Brand: Tektronix

  Model: 2630

  A/D converter accuracy: 12 bits

- One magnetic shaker with the following characteristics

  Brand: Acoustic Power Systems, Inc.

  Model: 113

  Frequency range: 0 to 200 Hz

  Maximum stroke: 158 mm

  Velocity: 762 mm/s

  Weight: 36 kg

[2] g=grams

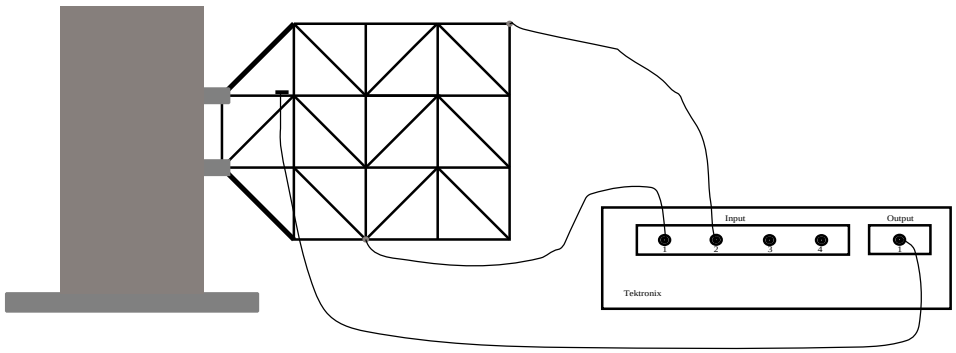

Figure A.2: Flexible frame: experimental setup.

A.2 The flexible frame experiment

The flexible frame is clamped to a concrete block that serves as ground for the experiment. The structure is excited with an impact hammer at node N12 and two accelerometers collect data at nodes N3 and N20 (fig. A.2). The Fourier analyzer collects the data from the hammer and accelerometers, filtering them above 50 Hz, and calculates the frequency response function between the signal of each accelerometers and the signal from the impact hammer. The accelerometers used in this experiment are the same as those used in the beam experiment. The window used when acquiring the data is the Boxcar window. The information about the impact hammer is

- One impact hammer with the following characteristics

  Brand: Kistler

  Model: 9722A500

  Threshold: 0.02 lbs rms

  Sensitivity: 10 mV/N

  Resonant frequency: 70 kHz

Vita

Mauro Jorge Atalla was born on July 19, 1968 in Belo Horizonte, Minas Gerais, Brazil, the oldest of three children of Shirley and Mauro Atalla. He attended the State University of Campinas, in the state of São Paulo, where he received a Bachelor of Science degree in Mechanical Engineering in January 5, 1990. During his undergraduate studies he spent the last semester of his senior year at the Bergishe Universität and Gesamthochshule Wuppertal, in Wuppertal, Germany, where he developed his senior project under the tutelage of Prof. Peter C. Müller and Prof. Hans I. Weber. After graduating he joined the graduate school of the State University of Campinas, receiving the degree of Masters of Science on July 30, 1992, under the tutelage of Prof. Hans I. Weber. He soon joined the graduate school of the State University of New York at Buffalo, under the tutelage of Prof. Daniel J. Inman. One semester later he transferred to the Virginia Polytechnic Institute and State University, in Blacksburg, Virginia, following his advisor. He received the degree of PhD in Engineering Science on May 10, 1996. Upon graduating, he will join the Mechanical Engineering Department at the State University of Campinas as an assistant professor.

Mauro J. Atalla