Soft Computing Techniques in a Distributed Environment

A Thesis

Submitted in fulfilment of the requirements for the

award of the degree of

DOCTOR OF PHILOSOPHY

Submitted by

Varinder Pal Singh

(Registration Number - 9000353)

Under the Supervision of

Dr. R. K. Sharma
TU, Patiala

Dr. S. Chakraverty
CBRI, Roorkee

Dr. G.K. Sharma
IIITM, Gwalior

Computer Science and Engineering Department
Thapar University
PATIALA – 147004 INDIA

May 2007

CERTIFICATE

Certified that the thesis entitled "**SOFT COMPUTING TECHNIQUES IN A DISTRIBUTED ENVIRONMENT**", which is being submitted by Mr. Varinder Pal Singh (Registration No. 9000353), to the Computer Science and Engineering Department, Thapar University, Patiala, in fulfilment of the requirements for the award of the degree of **DOCTOR OF PHILOSOPHY**, is a record of bonafide research work carried out by him under our guidance and supervision. The matter presented in this thesis has not been submitted either in part or full to any other university or institute for award of any degree.

(**Dr. R. K. Sharma**)
Thapar University,
Patiala

(**Dr. S. Chakraverty**)
CBRI, Roorkee

(**Dr. G. K. Sharma**)
IIITM, Gwalior

(Supervisors)

Dated: May 04, 2007

ACKNOWLEDGEMENTS

I would like to express my gratitude to my supervisors for helping me to successfully complete this dissertation. I express my greatest appreciation to Professor R. K. Sharma for his patient guidance and encouragement during this doctoral degree. As my supervisor, his insight, observations and suggestions helped me to establish the overall direction of the research and contributed immensely to the success of this research work. I would also express my greatest appreciation to Dr. S. Chakraverty, as my supervisor, his insight, observations and suggestions helped me to establish the overall direction of the research and contributed immensely to the success of the work. His broad knowledge, interest, tenacity, enthusiasm, criticism, and constant encouragement have been essential in my formation as a researcher. I would like to appreciate to Professor G. K. Sharma as my third supervisor whose energy, enthusiasm, insight and vast experience have been a source of inspiration and help me to complete this doctoral degree. Further, I find no word to express my heartfelt thanks to Professor R. K. Sharma and Dr. S. Chakraverty for supervising this work so diligently and delightfully.

I am grateful to the Director, Thapar University, Patiala for providing me university's resources and facilities necessary to carry out this research work at Thapar University. I am also grateful to the Director, Central Building Research Institute, Roorkee for providing me institute's resources and facilities necessary to carry out this research work at CBRI, Roorkee.

I express my gratitude to the Doctoral Committee comprising Dr. Seema Bawa, Head, CSED, Dr. Rajesh Kumar and Dr. Naveen Kwatra and Dean (Research and Sponsored Projects) for monitoring the progress and providing valuable suggestions for improvement of my research work. I am also thankful to the entire faculty and staff of CSED for healthy discussions, suggestions and necessary cooperation to complete this work

I am grateful to the Dr. Seema Bawa, Head, CSED, Thapar University, Patiala for providing me university's resources and facilities necessary to carry out this research work at department and constant encouragement to complete this research work at the earliest.

My hearty thanks to all the anonymous referees of several national and international Journals in the domain of artificial neural network and computer methods in

engineering for their constructive comments and new ideas for improvement of the research work presented in Chapter 3-5 of this thesis.

I would like to render thanks to families of Dr. R. K. Sharma and Dr. S. Chakraverty for their cooperation in sparing time for discussions with my supervisors at odd times.

I would like to express my thanks to Dr. Maninder Singh, Asstt. Professor, Sh. V. P. Singh, System Analyst, CSED, Mr. Karun Verma, System Analyst, SMCA and Mr. Harcharan Singh, system Analyst, CC, for their support, advice and many helps during this doctoral research project. Thanks to Lab. Staff of the CSED Department for their help during the computational setup for part of this investigation. I want to thank all my friends and well wishers for their encouragement to complete this research work. I risk neglecting to mention some of them, so I will trust that they will know I appreciate their friendship.

I must acknowledge the constant cooperation and moral support of my entire family, specially my wife, Mrs. Neena Kaushal along with our sons Dhananjay Kaushal (Gaffi) and Pranav Kaushal (Saffi) throughout this project without which it could not been possible to achieve this target so contentedly.

Above all, I pay my reverence to the almighty God as well as to my esteemed parents (Late) Mrs. Parkash Kaur and Sh C. R. Kaushal, whose blessings bring about this attainment.

Date: May 04, 2007
Place: Thapar University, Patiala

(Varinder Pal Singh)

CONTENTS

LIST OF FIGURES

LIST OF TABLES

$\ddot{x}$	Acceleration measurements
X	An *n*-element input vector
ANN	Artificial neural network
m	Aspect ratio
f(net)	Activation function
BFG	BFGS quasi-Newton algorithm
BCOPs	Boundary characteristic orthogonal polynomials
bc	Boundary condition
CC	Clamped boundary condition
a_{i}	Coefficients of a regression polynomial
b_{i}	Coefficients of second regression polynomial
c_{i}	Coefficients of third regression polynomial
b	Damping factor (In Chapter V)
$[C]$	Damping matrix
$c_{1,}\, c_2$	Damping factor (In Chapter VI)
d_k	Desired output
DSS	Decision support System
x	Displacement measurements
ρ	Density
Err	Error
δ_{lj}	Error at input layer
δ_{ok}	Error at output layer
EBPTA	Error back propagation training algorithm
δ_{pj}	Error signal vector for hidden layer
δ_{ok}	Error signal vector for output layer
$f(t)$	External exciting force
FEM	Finite element method
D	Flexural rigidity
CGF	Fletcher-Reeves update conjugate gradient algorithm

ff	Free boundary condition
FRF	Frequency response function
f	Frequency parameters
α	Gain function for activation function
GDP	Gradient descent algorithm with adaptive learning rate
y_i	Inputs values
η	Learning constant
BR	Levenberg-Marquardt algorithm with Bayesian regularisation
purelin	Linear activation function
NNT	MatLab's neural network tool
m	Mass (In Chapter V)
m_1, m_2	Mass 1 and Mass 2 (In Chapter VI)
$[M]$	Mass matrix
MSE	Mean square error
O_k	Neural network output
R	No. of input patterns
J	No. of nodes at hidden layer
l	No. of nodes at input layer
K	No. of nodes at output layer
Γ	Non-linear sigmoidal function
p_j^i	Output at hidden layer
O_j	Output of the j^{th} neuron
ν	Poisson's ratio
CGB	Powell-Beale restarts conjugate gradient algorithm
CGP	Polak-Ribière update conjugate Gradient algorithm
PCA	Principal component analysis
ω	Radian frequency of vibration
RR	Rayleigh-Ritz method
$r(x, \dot{x})$	Restoring force
RBF	Radial basis function
RN	Recurrent networks

RBNN	Regress based neural network
NRM3	*RBNN* model with regression polynomial degree 3
NRM4	*RBNN* model with regression polynomial degree 4
RK	Runge-Kutta method
RKNN	Runge-Kutta neural network
SRN	Simple recurrent networks
h	Step-size in Runge-Kutta method (In Chapter V & VI)
SS	Simply supported boundary condition
k	Stiffness (In Chapter V)
k_1, k_2	Stiffness (In Chapter VI)
$[K]$	Stiffness matrix
SVM	Support vector machine
tansig	Tan-sigmoid function
$\Re$	The one dimensional real space
h	Uniform thickness (In Chapter II)
$\dot{x}$	Velocity measurements
V_{ji}	Weights at input layer
W_{ki}	Weights at output layer
Y	Young's modulus

ABSTRACT

This present investigation deals with the ability of soft computing techniques, in particular, artificial neural networks in solving vibration and system identification problems. Artificial neural networks are also called parallel distributed systems because they are composed of a series of interconnected processing elements that operate in parallel. New training algorithms have been developed to train artificial neural networks and those are applied to the vibration analysis of structural members as well as system identification problems of structural dynamics for partially known and completely unknown systems. The structural problem is continually acquiring greater importance in modern science and technology and being solved with the help of different analytical and approximate methods. Structural members are often encountered in several engineering applications and their use in machine design, aeronautical engineering, nuclear reactor technology, naval structures, and earthquake resistance structures are very common. The system identification problem is an area of importance in structural engineering and is used to improve dynamic modeling capabilities for civil infrastructure systems such as high-rise building, bridges and dams.

The main contributions of this study are the following.

- New artificial neural network learning algorithms developed which make use of the coefficients of linear and nonlinear regression polynomials, including single and multiple variables, as training weights. These polynomials depend upon the *ANN* architecture to be considered for a particular problem. The proposed algorithm is henceforth abbreviated as *RBNN* (Regression Based Neural Network). In these models, the number of neurons in the hidden layer may be fixed depending upon the required polynomial degree.
- This *RBNN* model is applied to estimate the vibration characteristics for free vibration of elastic plates with different boundary conditions at the edges.
- The proposed algorithm is also applied to partially known and partially unknown system identification problems. Both multi-input single-output and multi-input multi-output cases have been considered for the analysis and simulation.

The whole range of the subject in this thesis is covered in six chapters, which deal with brief discussion of the existing soft computing techniques, vibration and system identification studies and the proposed new algorithms. Then the thesis investigates the

application of these techniques to continuous and discrete systems in terms of vibration and inverse vibration analysis. The reliability, efficiency and powerfulness of the proposed techniques are also discussed by comparing the present with known results in special cases.

Chapter wise summary of the thesis

In the first chapter, basic elements of the soft computing techniques in brief, and the artificial neural network technique, in some details have been presented. The details of the literatures that is surveyed during the study are also covered here. The summary of the present work and further scope of work is also discussed in the end.

In Chapter II, traditional artificial neural network models for plate vibrations *viz.* circular and elliptic plates have been undertaken. The training of network is performed using the patterns computed with the use of powerful method of Boundary Characteristic Orthogonal Polynomials (*BCOPs*) in the Rayleigh-Ritz (*RR*) method. The feedforward Error Back Propagation Training Algorithm (*EBPTA*) with one hidden layer has been considered for training the model. The network is optimized with number of nodes in the hidden layer that are necessary to achieve the desired accuracy in the output of the network.

In Chapter III, a new regression based neural network (*RBNN*) model has been proposed for the training of *ANN*s. Using this algorithm, we are able to provide some initial intelligence to the *ANN* model in the form of the regression coefficients obtained by regression analysis. We have again considered the plate vibration example problem as discussed in Chapter II for the development of the proposed model. Number of neurons in the hidden layer depends upon the degree of the regression polynomial that is used to fit the data between input and output. Performance of this algorithm has been compared with traditional *ANN* model. Proposed *RBNN* model has also been tested for its performance by validating the network with frequency data of various intermediate aspect ratios of the elliptic plate. The network is optimized with degree of the regression polynomial that is used to fit the data between input and output with respect to the number of nodes in the hidden layer that are necessary to achieve the desired accuracy in the output of the network.

In Chapter IV, a regression based neural network with multi input single output model is used for modeling transverse vibration problem of annular circular and annular elliptic plates. In all, nine traditional *ANN* and nine *RBNN* models have been developed in this chapter corresponding to the nine boundary conditions. The output of the network is

computed by regression analysis combined with neural activation function performed at the stage of hidden layer. Number of neurons in the hidden layer depends upon the degree of the multivariable regression polynomial that is used to fit the data between input and output. Different degree of multivariable regression analysis are studied for the investigation. In general, for multi input multi output models, *i.e.*, having more number of nodes the present methods may easily be extended for fixing the number of nodes, architecture *etc*. Performance of this algorithm has been compared with traditional *ANN* model. The developed *RBNN* model has also been tested for its performance by validating the network with frequency data of various intermediate aspect ratios of the annular circular and elliptic plates. The network is optimized with degree of the multivariable regression polynomial that is used to fit the data between input and output with respect to the number of nodes in the hidden layer that are necessary to achieve the desired accuracy in the output of the network.

In Chapter V, traditional *ANN* and proposed *RBNN* models are used for identification of dynamic systems. A single degree of freedom system having equation of motion with/without damping as well as force has been considered to explain the approach. Again the case of *RBNN* is investigated with various degree of polynomials and different cases of the number of nodes in the hidden layer have been considered to facilitate a comparative study on the architecture of the network. This also includes comparison of identification results between the traditional *ANN* and *RBNN*.

In Chapter VI, regression based multi-input multi-output artificial neural network model is used for multi-degree of freedom identification problems. The Multi Degree of Freedom (*MDF*) system having equation of motion with/without damping and force has been considered for modeling and identification. Here two problems of identification *viz.* partially known system and completely unknown system have been investigated. Here, two different multivariable polynomials are regressed first with each of the input data with the output data for determining the initial weights between input and hidden layer in case of partially known system and completely unknown system. These systems are investigated with various degree of polynomials and different cases of the number of nodes in the hidden layer have been considered to facilitate a comparative study on the architecture of the network. This also includes comparison of identification results between the traditional *ANN* and *RBNN* model.

As the conclusion of the work presented in this dissertation, it is worth mentioning that the newly developed method of *RBNN* is computationally efficient, straight forward for computer implementation and simple. This renders the problem to simple by fixing the number of neurons in the hidden layer. Moreover the rate of convergency shown to be fast and the solution are better compared to the traditional *ANN*. The proposed algorithm has been tested by undertaking simulation for a variety of engineering applications *viz.* vibration of plates and system identification. Solution of the example problems show the powerfulness, reliability and efficacy of the newly developed method in *ANN* training. The investigator believes that one can get better results than the traditional *ANN,* if the selection of the initial weights in term of regression coefficients should be done with most care and take the regression coefficients from input to hidden layer and random weights between hidden to output layer. In this way this thesis fills the gap by contributing a new method of training with better accuracy along with application of the method to some challenging problems.

CHAPTER I

INTRODUCTION

CHAPTER I

INTRODUCTION

The complex problems arising in science and engineering systems sometimes may not be accurately described by traditional mathematical models. The conventional approach for understanding and predicting the behavior of such systems based on analytical techniques may prove to be inadequate. These difficulties lead to a number of challenging problems. One of such problems is putting intelligence in machines. This is widely accepted fact that there is a large gap between the human intelligence and the intelligence possessed by the intelligent systems. The intelligent systems can, however, provide human like expertise such as domain knowledge, uncertain reasoning, and adaptation to a noisy and time varying environment. These factors sometimes become important in tackling computational aspects of complex problems. The research in the field of natural intelligence, including human intelligence provides a significant path in the study of soft computing related systems. Soft computing is oriented towards the analysis and design of intelligent systems. The term "soft computing" came into use presumably because it has borrowed many of its terminology from life sciences. Soft computing attempts to solve the class of computationally hard problems that do not seem to lend themselves well for classical algorithmic approaches, whereas humans are particularly proficient in solving them. We can also claim that soft computing is an innovative approach for constructing computationally intelligent systems which possess humanlike expertise within a specific domain, adapt themselves and learn to perform better in changing environments. Soft computing techniques cover the methodologies that tackle real-life problems without finding exact mathematical modeling of the systems. It is also used to construct new generation computationally intelligent hybrid systems.

Soft computing methods consist of several computing paradigms, including artificial neural networks, fuzzy set theory, approximate reasoning, and derivative-free optimization methods such as genetic algorithms. The integration of these constituent methodologies forms the core of soft computing techniques. This integration allows soft computing

techniques to incorporate human knowledge effectively, to deal with imprecision and uncertainty, and to learn how to adapt to unknown or changing environments for better performance. Soft computing is also aimed at formalization of great human ability to make rational decisions in uncertain and imprecise environment. The paradigms of soft computing are complementary to each other rather than competitive. Soft computing can also be termed as an open framework for newly created techniques involving human intelligence. These techniques are appropriate to be added to the soft computing framework. Together with other modern technologies, soft computing and its applications exert unprecedented influence on intelligent systems that mimic human intelligence in thinking, learning, reasoning, and many other aspects. So far, more and more researches and engineering applications show that soft computing can play a key role in the designing of the intelligent systems.

The current studies in intelligent systems, *e.g.*, intelligent control, intelligent agent and intelligent robotics reveal that only partial intelligence were extended or embedded to the system with the utilization of soft computing approaches. For example, artificial neural networks are useful to handle the non-linearities and unknown function approximation problems. Till now, the harmonious body of the intelligent systems has not been understood efficiently and it remains a challenging problem.

Use of soft computing techniques has received much attention in recent years as an alternative approach to cope with real-world problems that are computationally complex or mathematically intractable. It has been used in a wide variety of applications in both, natural and engineering sciences. The application areas of soft computing include decision making, pattern recognition, system control, information processing, natural language processing, optimization, speech recognition, robotics, identification and control of nonlinear systems and many others. The excellent inherent characteristics provided by these methodologies allow them to recognize and model the nonlinear phenomena present in complex processes. Some of the problems that can be dealt with using soft computing techniques include:

- Augmentation and modeling of information systems where basic entities and attributes can only be partially fulfilled, *e.g.*, searching a database for a person who is an expert with respect to specific topics. This attribute is usually fulfilled to a certain degree. A person might have more or less experience concerning a certain topic and the precise boolean values usually cannot be assigned.

- Control of everyday systems such as a heating system, a washing machine, a camera *etc*. The linguistic rules such as 'if it is too hot, regulate the heating system down' can be implemented using the soft computing techniques. These rules can be transformed to fuzzy if-then-rules, which constitute the key part of fuzzy control.
- Visualization and organization of given data, such as identification of typical classes in a data set collection, *e.g.*, a collection of DNA-arrays monitoring the behavior of various bacteria, or extraction of relevant trends from a collection of market data.
- Discrete optimization, particularly in rapidly changing scenario where unforeseeable events can occur, such as railway-timetable scheduling, online routing, or games.
- Discrete optimization for very large systems with large search spaces such as VLSI design.
- Optimization in unclear scenario such as production processes where the time for the production and the costs are not precisely specified.
- Solutions of vibration and inverse vibration problems described by linear or nonlinear processes.

It has been envisaged that soft computing is likely to play an especially important role in science and engineering, but eventually its influence may extend much farther. Soft computing represents a significant paradigm shift in the field of computing. This paradigm shift reflects the fact that the human mind, unlike present day computers, possesses a remarkable ability to store and process information which is pervasively imprecise, uncertain and lacking in categoricity. In the present research work, we have focused specifically on the use of Artificial Neural Networks in the vibration of plates and system identification.

1.2 Artificial Neural Networks

Artificial Neural Network (*ANN*) is one of the main components of the soft computing techniques and is a subject of interest to professionals in many fields. This is being used as a tool in many areas for problem solving. The applications of *ANN*s have spread widely in recent years and a number of researchers from diverse fields are working on the implementation of this technique. This technique has become an alternate to modeling of some physical and non-physical systems with scientific and mathematical basis.

*ANN*s are computational paradigms based on mathematical models that, unlike traditional computing models, have the structures and operations that resemble with mammal brain. Artificial neural networks are also called connectionist systems, parallel distributed systems or adaptive systems, because they are composed of a series of interconnected processing elements that operate in parallel. *ANN* lack centralized control in the classical sense, since all the interconnected processing elements adopt simultaneously with the flow of information and adaptive rules. A generic *ANN* can be described as a computational system consisting of a set of highly interconnected processing elements, called neurons, which process information as a response to external stimuli. An artificial neuron is a simplistic representation that emulates the signal integration and threshold firing behaviour of biological neurons by means of mathematical equations.

An *ANN* is generally considered performing in two different modes: learning (or training) and testing. During learning, a set of examples is presented to the network. At the beginning of the training process, the network guesses the output for each example. However, as training progresses, the network modifies internally until it reaches a stable stage at which the outputs are considered satisfactory. Learning is simply an adaptive process during which the weights associated with all the interconnected neurons change in order to provide the best possible response to all the observed stimuli. When the network has reached the desired performance level, the learning stage is over and the associated weights are frozen. Figure 1.1 depicts a generic model of *ANN*. The input signals to the neurons are modified by assigning weights representing the strengths of synapses associated with each input, for example, a number of inputs $x_1, x_2, ..., x_n$ associated with respective to weights $w_{j1}, w_{j2}, ..., w_{jn}$ form a combined input net_j to the j^{th} neuron, which is expressed as the weighted sum of the inputs:

$$net_j = \sum_{j=1}^{n} w_{ji} x_i, \quad i = 1,2, ..., m \qquad \ldots (1.1)$$

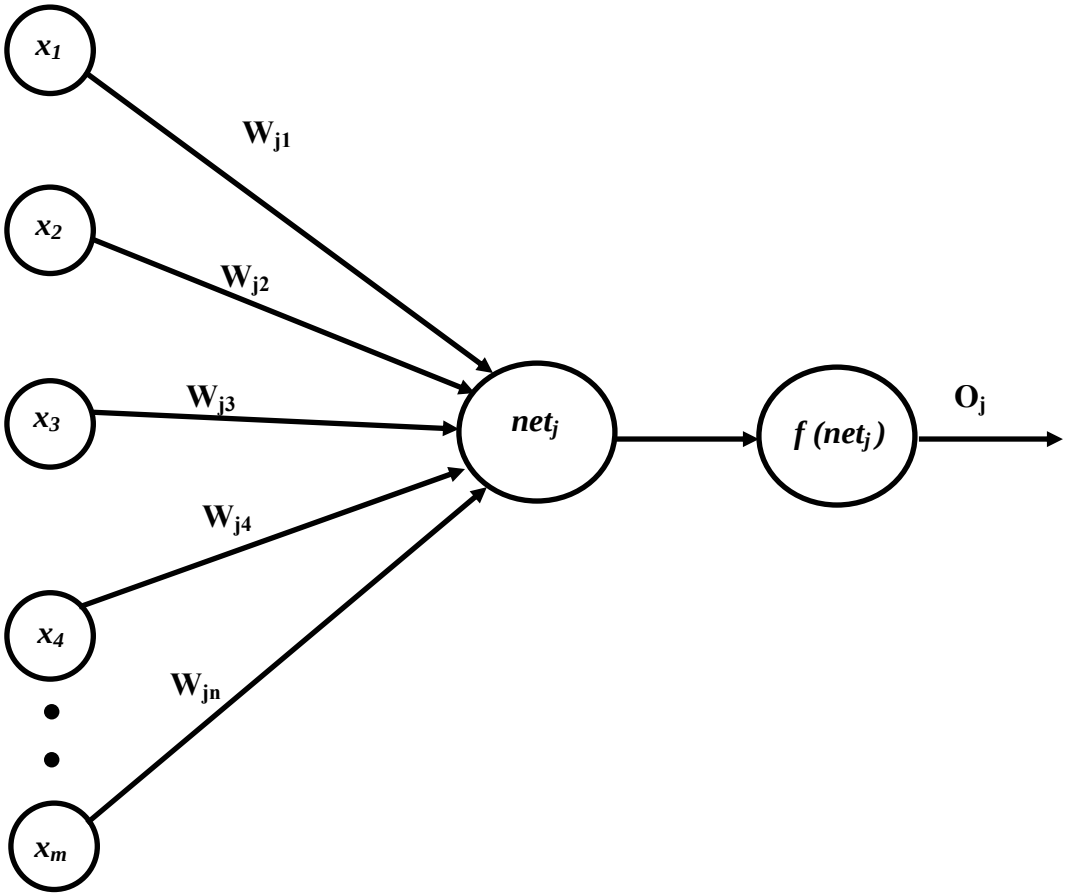


Figure 1.1: Generic Model of *ANN*

Many of times, a bias is added to the net input, that is, we add a new synapse having fixed input $x_0 = +1$ and weight $w_{j0} = bias_j$. This sum of weighted signals, net_j, is transformed into an activation level using a transfer or activation function to produce an output signal O_j only when exceeding a certain threshold as follows:

$$O_j = f(net_j) = f\left(\sum_{j=1}^{n} w_{ji}\, x_i\right),\ i = 1,2,...,m \qquad \ldots (1.2)$$

The output of a neuron is generally determined by the nature of its activation function.

1.2.1 Activation Functions

The activation (or transfer) function determines the output from a summation of the weighted inputs of a neuron. The transfer functions for neurons in the hidden layer are often

nonlinear and they provide the nonlinearities for the network. Some of the commonly used activation functions are given below:

(a) Threshold function

$$f(x)=\begin{cases}1, & x \geq \theta \\ 0, & x < \theta\end{cases} \quad \ldots (1.3)$$

(b) Sigmoid function

$$f(x)=\frac{1}{1+e^{-\alpha x}} \quad \ldots (1.4)$$

(c) Hyperbolic function

$$f(x)=\tanh(\alpha x)=\frac{e^{\alpha x}-e^{-\alpha x}}{e^{\alpha x}+e^{-\alpha x}} \quad \ldots (1.5)$$

(d) Linear threshold

$$f(x)=\begin{cases}1, & x \geq \theta \\ \dfrac{x}{\theta}, & 0 < x < \theta \\ 0, & x \leq 0\end{cases} \quad \ldots (1.6)$$

(e) Gaussian function

$$f(x)=e^{-\alpha x^2} \quad \ldots (1.7)$$

where x, α and θ in the above expressions denote weighted sum of the inputs, network parameter and threshold value, respectively.

At the testing stage, the network receives an input signal, (new or unknown) and processes it to produce an output.

1.2.2 Classification of *ANN*s

As the number of learning algorithms and architectures in the *ANN* literature is large, diverse, and growing, it is impossible to come up with a single classification scheme that can capture all the essential traits of all these paradigms. However, *ANN*s are generally classified based on the architecture of the network, type of learning and the connectivity employed for the neurons. A broad classification of *ANN*s is given below:

Learning Methods: Supervised

Unsupervised

Connection Type:	Static (feed forward)
	Dynamic (feed forward)
Topology:	Single Layer
	Multi Layer
	Recurrent
	Self-Organized
Applications:	Classification
	Clustering
	Function approximation
	Prediction

The different types of *ANN*s are described below, in brief.

1.2.2.1 Supervised Learning Method

In supervised learning, both the inputs and the outputs are provided. The network then processes the inputs and compares its resulting outputs with the desired outputs. Errors are then calculated, causing the system to adjust the weights that control the network. This process occurs again and again and the weights are continuously modified so that error decreases. The supervised learning method is mostly used for accurate classifications and associations.

1.2.2.2 Unsupervised Learning Method

In unsupervised learning, the network is provided with inputs but not with desired outputs. Unsupervised learning algorithms use patterns that are redundant raw data having no labels regarding their class membership or associations. While discovering these, the system itself must then decide what features it will use to group the input data. This is also referred to as self-organization or adoption. This learning technique is also used to perform clustering in the unsupervised classification of objects without providing information about the actual classes.

1.2.2.3 Feed-forward Networks

Feed-forward *ANN*s allow signals to travel one way only from input to output. There is no feedback (loops), *i.e.*, the output of any layer does not affect that layer. Feed-forward *ANN*s tend to be straightforward networks that associate inputs with outputs. These are extensively used in pattern recognition.

1.2.2.4 Feedback Networks

Feedback networks may have signals travelling in both directions by introducing loops in the network. Feedback networks are very powerful and generally have a complicated structure. Feedback networks are dynamic and their state changes continuously until they reach an equilibrium point. They remain at the equilibrium point until the input changes and a new equilibrium needs to be found. Feedback architectures are also referred to as interactive or recurrent, although the latter term is often used to denote feedback connections in single-layer organisations.

1.2.2.5 Single Layer Perceptron

The earliest neural network had been a single-layer perceptron network, which consists of a single layer of output nodes; the inputs are fed directly to the outputs via a series of weights. In this way it is considered as the simplest kind of feedforward network. The sum of the products of the weights and the inputs is calculated at each node, and if the value is above some threshold (typically 0) the neuron fires and takes the value 1; otherwise it takes the value -1. Neurons with this kind of activation function are also called McCulloch-Pitts neurons or threshold neurons. Perceptrons can be trained by a simple learning algorithm that is called the delta-rule. It calculates the errors between calculated output and sample output data, and uses this to create an adjustment to the weights, thus implementing a form of gradient descent. A single layer neural network computes a continuous output instead of a step function by using activation function and this network becomes identical to the logistic regression model, widely used in statistical modeling.

1.2.2.6 Multi Layer Perceptron

The class of these type of networks consists of multiple layers of computational units, usually interconnected in a feedforward way. Each neuron in one layer has directed

connections to the neurons of the subsequent layer. In many applications, the units of these networks apply a sigmoid function as an activation function.

Multi-layer networks use a variety of learning techniques, the most popular being back propagation. The output values of the network are compared with the correct answer to compute the value of some predefined error-function. By various techniques the error is then fed back through the network. Using this information, the algorithm adjusts the weights of each connection in order to reduce the value of the error-function by some small amount. After repeating this process for a sufficiently large number of training cycles the network usually converges to some state where the error of the calculations is small. A gradient descent method is applied to adjust the weights in these types of networks.

1.2.2.7 Recurrent Network

Recurrent Network (*RN*) is a model with bi-directional data flow. While feedforward network propagates data from input to output, *RN* also propagates linearly data from later processing stages to earlier stages.

A Simple Recurrent Network (*SRN*) is a variation on the multi-layer perceptron, sometimes also called an Elman network. A three-layer network is used, with the addition of a set of context units in the input layer. There are connections from the middle hidden layer to these context units fixed with weight 1. At each time step, the input is propagated in a standard feedforward fashion, and then a learning rule (usually backpropagation) is applied. The fixed back connections result in the context units always maintaining a copy of the previous values of the hidden units since they propagate over the connections before the learning rule is applied. Thus the network can maintain a sort of state, allowing it to perform such tasks as sequence-prediction that are beyond the power of a standard multi-layer perceptron.

1.2.2.8 Other Networks

Hopfield Network

The Hopfield network is a recurrent neural network in which all connections are symmetric. Invented by John Hopfield, this network has the property, based on its dynamics that it always converges. If the connections are trained using Hebbian learning then the

Hopfield network can perform robust content-addressable memory, robust to connection alteration.

Support Vector Machine

A Support Vector Machine (*SVM*) is a recently developed form of machine learning algorithm. The training of *SVM*s is based on quadratic programming, a form of optimization that has only one global minimum. The problem of over fitting is considerably addressed in the *SVM*.

Committee of Machines

A Committee of Machines (*CoM*) is a collection of different neural networks that work together on a given example. It has been seen that this gives a much better result. In fact in many cases, starting with the same architecture and training by different initial random weights give vastly different networks.

Spiking Neural Networks

The Spiking (or pulsed) Neural Networks (*SNN*) are models which explicitly take into account timing of inputs. The network input and output are usually represented as series of spikes. *SNN*s have an advantage of being able to continuosly process information and these are often implemented as recurrent networks.

In the next section, we briefly, discuss the problem of vibration of plates.

1.3 Vibration of Plates

The subject of vibration of plates is an extremely important area owing to wide variety of its applications. Since beams, plates and shells form integral parts of structures, a prior knowledge of a first few natural frequencies and the associated modes shapes is essential for an engineer before finalizing the design of a given structure. In particular, plates with different shapes and boundary conditions at the edges are often encountered in serveral engineering applications such as aeronautical engineering, telephone industry, machine design, necular rector technology, naval structures and earthquake resistant structures *etc*. The study of vibration of plates is an old area in which a lot of work has been done. In the earlier periods, results were available for some simple cases only where the

analytical solution could be found. To get the reasonable accurate results even in these cases were impossible due to lack of good computational facilities. With the advent of fast computers, there was a tremendous amount of rise in the research work using approximate and numerical methods. Vibration of plates of various shapes with different boundary conditions becomes easy to solve using the finite element method, boundary integral equation method, finite difference method and the methods of weighted residuals *etc*. It is well-known fact that vibration of plate problems involves the solution of fourth order differential equation and the solution in general turn into a generalized egien value problem which may be complex to solve. As of now, there are plenty of data available for vibration of plates that can be used in artificial neural network training. Here, new training methodologies for *ANNs* have been proposed. The new methodologies are tested and validated by using the data of vibration of elliptic, circular and annular plates.

1.4 System Identification

The dynamic structural identification of mathematical methods of physical structures on the basis of experimental measurements is a problem that has been receiving increasing attention in the recent past. System identification has become an important area of study because of the increasing needs in estimating the behavior of a system with partially known dynamics especially in the areas of control, health monitoring of structures, damages assessment of structures, pattern recognition and even in the realm of stock markets. In these applications, the system of interest needs to be known to some extent. Since a system may have a complicated dynamic behavior, the varying environmental changes make the identification process much more difficult than the cases in which those changes are modeled deterministically. However, with the complexity of the structure and errors in measured data, dynamic structural identification is a challenging task in soft computing.

One can use soft computing techniques to solve computationally complex and mathematically intractable problems. Using soft computing methodologies, one can easily combine the natural system dynamics and an intelligent machine. In this perspective, the intelligence comes through the utilization of an expert's knowledge and massively parallel and adaptive data processing architecture of the computationally intelligent approach.

The most popular constituents of the soft computing methodologies are the *ANNs*, which are considered to provide the mathematical power of the brain. The nonlinear mappings between inputs and outputs of the neural network and the ability of the network to adapt its parameters so as to minimize an error criterion make the use of *ANNs* particularly well suited for the identification of both nonlinear and linear dynamical systems.

1.5 Literature Review

Soft computing techniques have played a major role in many areas of scientific and engineering applications in recent years. Soft computing, being a collection of different techniques that exploit in different ways the tolerance for imprecision and uncertainty, has provided a good option to solve problems that are computationally complex or mathematically intractable *(Zadeh (1994))*. Many researchers are using various soft computing techniques with different computational architectures. *Azvine (1994)* proposed an intelligent multi-modal interface for a large workforce management system called the smart work manager and described the soft computing techniques in the development of interface for temporal reasoning, approximate query matching, learning action sequences and gaze tracking in conjunction with other artificial intelligence based systems.

Bonissone et al. (1999) discussed the use of soft computing methodologies in industrial and commercial applications and proposed a collection of methods and tools that has been used to perform diagnostics, estimation and control in real world applications to the industrial diagnostics, freight train control and residential property valuation. The application of soft computing for information retrieval has been studied extensively by *Bonissone (1997), Crestani* and *Pasi (2000) and Nikravesh et al. (2002). Mitra et al. (2002)* discussed the soft computing techniques that had been applied to data mining and proposed the categorization based on the different soft computing techniques for different functional aspects of data mining. The results of their study show that a categorization can be provided based on the different soft computing tools and the mining function implemented and the preference criterion selected by the model can also be optimized. Hybrid soft computing models used by *Bonissone et al. (2002)* for application in paper making machines and developed a predictive model to estimate time to breakage and web break tendency indicator

in the wet-end part of the paper making machines. Soft computing techniques applied to information retrieval have been described by *Cordon et al. (2003*). *Khan et al. (2003)* have successfully described the comparative study of six soft computing models for the hourly electricity demand forecast of Czech Republic. The soft computing models have been trained and tested using the actual hourly load data for the last seven years and proposed the forecasting for predicting forty eight hourly demands for electricity. The results show that the soft computing techniques are more accurate and effective for the analysis and forecasting of electricity demand.

Chakraverty et al. (2003) has studied the soft computing approach for identification of dynamic systems. Decision support system in tactical air combat has been developed by *Cong et al. (2004)* and discussed several adaptive learning frameworks for constructing intelligent decision support systems. Recently, they have also proposed a framework that could capture imprecision, uncertainty, learn from the data/information and continuously optimize the solution by providing interpretable decision rules. Studies on soft computing techniques for financial classification and financial credit risk management have been carried out by *Compo et al. (2004)*. The applications of soft computing techniques for weather forecasting have been explored by *Abraham et al. (2004)*. They have proposed a method for long-term rainfall prediction. The rainfall data of eighty seven years was analysed and the predicted outputs were compared. It revealed that soft computing techniques are promising and efficient techniques for weather forecasting. *Abraham (2004)* proposed the meta-learning evolutionary artificial neural network, an automatic computational framework for the adaptive optimization of artificial neural networks. Performances of the different learning algorithms were evaluated when the activation functions and architecture were changed. The proposed design of a neural network is smaller, faster and has better performance. Intelligent systems for stock market predictions have been proposed by *Abraham et al. (2004)*. They have investigated the seemingly chaotic behavior of the stock markets using several connectionist paradigms and soft computing techniques. Recently, *Srinivas et al. (2005)* successfully described different soft computing and hard computing techniques for intrusion detection systems and discussed the ensemble of Artificial Neural Networks (*ANNs*), Support Vector Machines (*SVMs*) and Multivariate Adaptive Regression Splines (*MARS*) that are superior to individual approaches for intrusion detection in terms of classification accuracy.

Distributed soft computing intrusion detection system has been proposed by *Abraham et al. (2007)*, in which several intrusion detection systems are implemented over a large network, all of which communicate with each other, or with a central server that facilitates advanced network monitoring. The implementation of the intrusion detection system in a distributed environment has shown that soft computing approach can play a major role for intrusion detection.

Lipmann (1987) discussed the classification tasks using the feed forward neural network with *d* inputs. The nodes of the first hidden layer have been used as functions in hyper-planes that effectively partition the *d*-dimensional space into various regions. Each node in the next layer represents a cluster of points that belong to the same class. The target output of the network is peak ground acceleration normalized and log-sigmoid activation function was adopted in hidden and output layer. Dynamic learning rate of the backpropagation algorithm using derivative information has been discussed by *Yu et al. (1995)*. The algorithm has been explored by deriving the first and second derivatives of the objective function with respect to the learning rate and used the information gathered from the forward and backward propagation instead of calculation of second order derivatives in weight space. *Oh (1997)* has suggested the modification of the error backpropagation algorithm with a modified error function for MultiLayer Perceptrons (*MLPs*) which suffers from slow learning speed. The effectiveness of the method has been tested on handwritten digit recognition data. Iterative pruning algorithm for feedforward neural networks is given in *Castellano et al. (1997)*. This pruning problem has been formulated by solving a system of linear equations. The least square errors have been calculated by using conjugate gradient algorithm. *Younger et al. (1999)* has studied the fixed-weight on-line learning for neural networks. They have suggested a method to design neural networks where recurrent signal loops store the knowledge in a manner analogous to short-term memory of the biological systems. The method has been implemented on four higher order fixed-weight learning networks.

Yamamoto and Nikiforuk (2000) have discussed the learning algorithm for supervised learning of multilayered and interconnected neural networks without using a gradient method. The weight parameters were determined by using exponentially weighted least squares. Single hidden layer feedforward neural networks have been described by *Huang et*

al. (2000). They have suggested that single hidden layer feedforward neural networks with any continuous bounded non constant activation function or any arbitrary bounded (continuous or non-continuous) activation function can form disjoint decision regions with arbitrary shapes in multidimensional cases. *Tsukimoto (2000)* has described the method for extracting rules from trained neural networks. The method has been applied to multilayer neural networks, recurrent neural networks and other networks whose output function is sigmoid function. The network was tested for the results of the votes data, mushroom data and others. Transforming input data for neural networks in order to reduce the prediction error has been discussed by *Shi (2000)* and performance of a neural network has been improved by using the data transformation methods. The cumulative distribution function has been used as data transformation method to transform a stream of random data distributed in any range to data points uniformly distributed in range of 0 -1. Neural Network architecture suitable for learning with complex weights has been discussed by *Igelnik et al. (2001)* and implemented this architecture with Radial Basis Function (*RBF*). De*Nicolao and Ferrari (2001)* described the network as fast weight calculation. The complexity of the network has been reduced by using the spectral factorization and Kalman filtering.

It has been observed that soft computing approach has widely been used in the different aspects of science and engineering. In the present work, this approach has been used to address the problems of plate vibration and system identification, in general. The literature review pertaining to plate vibration is given first in the following paragraphs.

A large amount of research work has been done on the subject of vibration of plates. It is very difficult to give here even a brief summary of the work done by researchers so far. The monograph by *Leissa (1969)* contains huge information regarding the vibration of plates of various shapes under different boundary conditions. It gives a survey of investigations done till that date. It has been observed from the literature survey that large amount of work has been done using classical plate theory with uniform as well as the variable thickness under variety of conditions at the boundary. The geometries include circle, circular annular region, rectangles, rhombus and triangles. Some work has also been reported about elliptic and other geometries. Closed form analytical solution could be obtained in simple situations only. In most of the cases, one has to depend on some approximate analytical or a numerical method. These methods described in the literature are the series solutions by Frobenious

method, Galerkin's method, Rayleigh-Ritz method, use of splines, finite element, boundary element and method based on use of orthogonal polynomials as the basis functions.

The study of the circular plate is one of simplest problems where the solutions are expressible in terms of Bessel's functions. Three different cases arise depending upon whether the boundary is clamped, simply-supported or free. Clamped plate has been studied by *Reid (1962), Gontkevich (1964)* and others given in *Leissa (1969)*. Circular plates with simply-supported boundary have been discussed by *Prescott (1961), Gontkevich (1964)*. *Leissa (1967)* solved the problem involving simply-supported boundary conditions and calculated the fundamental frequency. *Sato (1973)* has presented results for simply-supported and completely free elliptical plates. Besides computing natural frequencies, some authors have also determined stresses in the plate, mode shapes and the nodal radii. *Jones (1975)* and *Johns (1975)* gave approximate methods for computing the fundamental frequencies. A comprehensive set of results for circular plates was presented by *Narita (1980)*. Study of circular plates with mixed boundary conditions has been investigated by *Hamming (1975)* and *Vivoli and Fillippi (1974). Shibaoka (1965)* solved the problem of clamped elliptic plates with small eccentricity for fundamental frequency by taking a series of finite determinants. Other notable reference for elliptic plates is *Neyfeh et al. (1978)*.

In the context of elliptic plates, very little has been reported in the literature. *Leissa (1969)* has cited some references in this regard. The transverse vibrations of circular annular plates have also been extensively studied. Again, the monograph by *Leissa (1969)* is an excellent source of information in this context. Other important references given therein are *Raju (1962)* and *Joga and Vijay (1963). Kim and Dickinson (1989)* have studied the transverse vibration of thin annular circular plates involving certain complicating effects. Other references of interest are *Nagaraja (1985)* and *Gupta et al. (1986, 1987)*. Finite element method has been used by *Pardoen (1975)* to analyze the free vibrations of annular plates. *Sato (1974)* has also studied the annular elliptic plates. He has used the exact solution in terms of Mathien functions when the two boundaries are confocal ellipses which were taken to be either both free or inner clamped and outer free. *Ozkul (1975)* studied the annular elliptic plate having both boundaries clamped. A computationally efficient method of boundary characteristic orthogonal polynomials has been developed by *Bhat (1985)*. He proposed a systematic method to use the boundary characteristics orthogonal polynomials in

the Rayleigh-Ritz method for the study of natural frequencies of rectangular plates. More details about one dimensional orthogonal polynomials may be found in *Chihara (1978)*. Two dimensional boundary characteristics orthogonal polynomials have been generated by *Chakraverty (1992)* and used to study vibration problems of plates for various geometries such as circular, elliptic, circular annular, elliptic annular, triangular and skew plates with all the possible boundary conditions at the edges. These results have been published in a series of papers by *Singh and Chakraverty (1991, 1992a, 1992b, 1994)*. *Yuan and Dickinson (1996)* have studied the vibration of annular, circular and sectorial plates with cut-outs. The results were obtained by using a Ritz solution and satisfied the required continuity conditions. *Romanelli and Laura (1997)* have studied the transverse vibrations of circular, annular plates of non-uniform thickness with free inner boundary. Approximate method was used for analyzing the problem. An interesting paper by *Sahu (2001)* who has developed a simple third order mathematical relation formulated by perturbation technique. He used this technique to determine the effects of small change in pre-twist on the natural frequencies of torsional vibrations of pre-twist beam type blade.

Now, a brief survey of the literature on the applications of neural networks in system identification is presented in the following paragraphs.

Nonparametric identification technique for nonlinear dynamic problems has been studied by *Masri and Chassiakos (1979)*. They have presented the state of variables of nonlinear systems, which expressed the characteristics in term of the orthogonal functions. *Chu et al. (1990)* discussed the utilization of neural networks in identification of dynamical systems. They have used Hopfield network to implement least squares estimation for time-varying and time-invariant systems. They also presented mathematical formulations to construct a dynamic system in terms of its Fourier coefficients. Identification and control of dynamical systems using neural networks have been discussed by *Narendra and Parthasarathy (1990)*. They have discussed static and dynamic backpropagation methods for the adjustment of parameters. Multilayer and recurrent network models have been used by them in novel configurations and their results revealed that the identification and adaptive control schemes practically are feasible. Neural computing strategies to represent the force displacement relationship in the static structural analysis have been presented by *Hajela et al. (1991)*. These models provide computationally efficient capabilities for re-analysis and

appear to be well studied for application in numerical optimum design. *Masri et al.* (*1992*) explored the potential of using parallel distributed processing (neural network) approaches to identify the internal forces of structure-unknown nonlinear dynamic systems. In this approach, the neural network is included as part of the equation of motion and used only to perform static function mapping. Another approach is to involve the dynamic differential equation into each of the neural network processing elements to create a new type of neuron called the dynamic neuron. The same authors also developed two types of approaches for identification of nonlinear systems using neural networks *(Masri et al.* (*1993))*. One of these approaches is first to identify the internal forces from the system responses using neural networks. Then the identified internal forces are put into the equation of motion of the system, solving the equation of motion by a conventional computational scheme gives the solution of the problem.

Wu et al. (1992) used a back propagation network to identify damage in a three story building modeled by two dimensional shear building driven by earthquake excitation. Back propagation network to identify damage in a five-story building has been used by *Elkordy et al. (1993)*. The network was used to map mode shapes to the percent change in member stiffness. The predictions of the damage were generally correct to within 10%. *Worden et al. (1993)* also used a back propagation network to identify damage in a twenty-member framework structure. The neural network used to map static strain data to subjective measure of damage in the scale between 0 and 1. The network was trained on data generated by *FEM* and tested on an experimental model of the same geometry. Back propagation network also used to identify the damage in multistory buildings by *Stephens and VanLuchene (1994)*. The damage was modeled by actual cracks into the concrete model of the structure and network was used to identify the map from three empirical damage indices to a qualitative scale. *Cheng et al. (1994)* also applied back propagation neural network to select earthquake waves in time history approach for dynamic analysis of structure. System identification and fault diagnosis problems in dynamic systems using neural networks has been studied by *Bernieri et al. (1994)*. They have illustrated neural procedure to be used in on-line fault diagnosis applications. *Yen (1994)* studied the identification and control of large structures using neural networks. The radial basis function network has been used as a learning controller to achieve the vibration suppression and trajectory maneuvering. The ability of connectionist systems

has provided an efficient means of modeling, identification and control of complex systems. Stochastic convergence analysis of a two layer perceptron for a system identification model are given in *Vaughn and Bershad (1995)*. The results of the simulations and analysis of the learning behavior of two layer perceptron has been discussed by them for a nonlinear system identification problem.

Kosmatopoulos et al. (1995) studied the approximation and learning properties of the recurrent networks, known as high-order neural networks and applied these architectures to the identification of dynamical systems. In recurrent high-order neural networks the dynamic components has been distributed throughout the network in the form of dynamic neurons. Identification schemes based on high-order network architectures has also been designed and analyzed by them. Expert system using artificial neural networks for the optimal design of RC beams has been developed by *Mukherjee and Deshpande (1995)* and artificial neural network has been used as the preliminary design model. Other interesting paper is by *Chassiakos and Masri (1996)* who have also used the artificial neural networks in the problem of identification for solving the differential equations governing dynamic systems. *Amini et al. (1997)* described a generalized dynamic identification technique using the neural networks based on the understanding of the dynamics of the structure and they verified the generalization of the neural network using different inputs to avoid the difficulties encountered in structural dynamic model identification. Self-organizing neural networks for identification of natural modes have been discussed by *Mukherjee (1997)*. The self-organizing network has been used with unsupervised learning algorithm and applied in classification problems. The network developed by him has been employed to predict the natural mode shapes of building frames with a varying number of stories and with noise tolerance capabilities. Heuristic design knowledge used by structural engineers when performing structural design has been studied by *Biedermann (1997)* and implemented with neural networks. He has developed a methodology to derive an input-output relationship for problems that may be too complex to model mathematically, computationally expensive, or difficult to solve using the traditional procedural computing approaches.

Identification of restoring forces in nonlinear vibration systems based on neural networks has been described by *Liang et al. (1997)*. The methodology has been used effectively for the identification of the restoring forces of multi degree of freedom nonlinear

vibration systems. The effect of nonlinearities in the system has also been estimated from the identified results. *Wang and Lin (1997)* have used the least square method to describe the fast learning algorithm of feedfarward neural networks and the proposed algorithm has less computational cost and is better suited to system identification. *Zhu and Ma (1997)* analysed the relationship between input and output of a dynamic system. They have proposed a new structure of neural network in which a specific layer added in the network had to be the first hidden layer. Each node of this layer receives the input having the same property and produced an integrated signal. The benefits of this network such as speed up the training and convergence of the network has also been discussed. New paradigm called self-recurrent neural network using dynamic back propagation algorithm with adaptive learning rate has been developed by *He et al. (1997)*. *Rao et al. (1998)* presented a methodology employing *ANNs* for active control of structure, under seismic excitation for a target percentage reduction of response. The method was based on the solution of an inverse problem, which attempts to obtain the control force required for a predetermined reduction of response of the structure. Runge-Kutta Neural Networks (*RKNNs*) for identification of unknown dynamical systems using the ordinary differential equations has been studied by *Wang and Lin (1998)*. They have constructed the network according to the Runge-Kutta approximation method. The network has been trained with help of the gradient and nonlinear recursive least squares based algorithms. *Lo and Basar (1998)* described the identification for nonlinear systems in the presence of unknown driving noise using feedforward multilayer neural network and radial basis function network models. They have included several simulation studies to demonstrate the effectiveness of the algorithms.

The methodology of load identification using an artificial neural network is given in *Cao et al. (1998)*. The model has been developed using the load-strain relationship for the structural analysis and used to identify the loads distributed across a cantilevered beam. *Karlik et al. (1998)* investigated the nonlinear vibrations of an Euler-Bernoulli beam with a concentrated mass attached to it. The nonlinear correction coefficients and natural frequencies have been calculated for different boundary conditions, mass ratios and mass locations using the Newton-Raphson method. These parameters were used in training a multi-layer, feed-forward, backpropagation artificial neural network. Identification and control of structures using neural networks has also been studied by *Bani et al. (1999)*. In

their study, the system identification and parameter estimation have been conducted in two experimental methods. The system has been identified in the time domain and the estimated parameters were used in the frequency domain methods and the system was modeled and identified using multiple emulator neural networks with different prediction capabilities. Three controllers were developed and designed in this method. Comparisons between the optimal controller and neurocontrollers have also been presented by them. Solution of inverse vibration problem of cracked structures using general regression neural networks has been studied by *Mahmoud and Abukiefa (1999)*. The first six natural frequencies were used by them as network inputs, with crack size and crack location as the outputs. They have quantitatively evaluated the effect of the number of frequency inputs on prediction accuracy of the network. This methodology has been used on steel cantilever beam with a single edge crack and it has been reported that prediction accuracy increases with larger number of vibration modes. *Tsypkin et al. (1999)* have also studied the identification of a nonlinear dynamic system. They have used a two layer neural network for the solution of the problem. A system disturbed with unmeasurable noise has been considered as the input of the network. Absorption polynomials and nonquadratic loss functions have been used to reduce the effect of the input disturbances on the basis of the optimal memory of the neural network model. *Gupta and Li (2000)* discussed the design optimization with mathematical programming neural networks. The mathematical programming neural network models have been developed and applied to design optimization problems in mechanical motion synthesis and structural design. Stiffness parameters of a complex structural system using a backpropagation neural network have been estimated by *Yun and Bahng (2000)*. Several techniques have been employed to overcome the issues associated with many unknown parameters such as the substructural identification and the submatrix scaling factor in a large structural system. The natural frequencies and mode shapes have been used as input patterns to the neural network for the case with incomplete measurements of the mode shapes. *Zang and Imregun (2001)* investigated the structural damage detection using measured Frequency Response Functions (*FRF*s) as input data to neural networks. The Principal Component Analysis (*PCA*) based data reduction technique has been used to measure *FRF*s data and used as the input to the neural networks. The methodology was applied to the measured *FRF*s of a

railway wheel. The three different networks, each corresponding to a co-ordinate direction, were trained and verified using eighty *PCA* compressed *FRF*s data.

On-line identification of nonlinear system *via* dynamic neural networks has been studied by *Yu and Li (2001)*. The passivity approach has been applied to access several new stable properties of neuro identification. The gradient descent algorithm for weight adjustment has been found stable in an *L*-infinity sense and robust to any bounded uncertainties by them. *Efe and Kaynak (2000)* studied the neural network models in identification of nonlinear systems. *Victor et al. (2005)* discussed implications of the approximation theory dealing with the ability of a class of dynamic neural networks to approximate finite trajectories of nonautonomous systems. They have also introduced an efficient parameterization of a class of dynamic neural networks to adjust the parameters.

1.6 Present Work

The present work deals with soft computing techniques, in particular, *ANNs* in solving vibration and system identification problems. Artificial neural networks are also sometimes called parallel distributed systems because they are composed of a series of interconnected processing elements that operate in parallel. We have developed new training algorithms to train *ANNs* and have applied these to the vibration analysis of structural members as well as system identification problems of structural dynamics for partially known and completely unknown systems. This application is based on input/output information gathered from the literature. These structural problems are continually acquiring greater importance in modern science and technology and being solved with the help of different analytical and approximate methods. The structural members are often encountered in several engineering applications and their use in machine design, aeronautical engineering, nuclear reactor technology, naval structures, and earthquake resistance structures are very common. The system identification problems are having great importance in structural engineering and are used to improve dynamic modeling capabilities for civil infrastructure systems such as high-rise building, bridges and dams.

The main contributions from this study are the following:

- New artificial neural network learning algorithms developed which make use of the coefficients of linear and nonlinear regression polynomials, including single and

multiple variables, as training weights. These polynomials depend upon the *ANN* architecture to be considered for a particular problem. The proposed algorithm is henceforth abbreviated as *RBNN* (Regression Based Neural Network). In this model, the number of neurons in the hidden layer may be fixed depending upon the required polynomial degree.

- This *RBNN* model is applied to estimate the vibration characteristics for free vibration of elastic plates with different boundary conditions at the edges.
- The proposed algorithm is also applied to partially known and partially unknown system identification problems. Both multi-input single-output and multi-input multi-output cases have been considered for the analysis and simulation.

The whole range of the subject in this thesis is covered in six chapters, which deal with brief discussion of the existing soft computing techniques, and the proposed new algorithms. Then the thesis investigates the application of these techniques to continuous and discrete systems in terms of vibration and inverse vibration analysis. The reliability, efficiency and powerfulness of the proposed techniques are also discussed by comparing the present with known results in special cases.

In the first chapter, basic elements of the soft computing techniques, in brief, and that of *ANN* techniques, in some details, have been presented. The brief discussion of the *ANNs,* including their classification and application has also been incorporated here. Basic ideas about the vibration of plates and system identification are also addressed in this Chapter. The details of the literature that is surveyed during the study are also covered here.

In Chapter II, the traditional *ANN* model for plate vibrations *viz*. circular and elliptic plates has been undertaken. A three layer architecture for *ANN* is considered to model the problem. The input layer consists of two inputs. The output layer of the ANN architecture consists of one output as frequency parameter (λ). The network is optimized with number of nodes in the hidden layer that are necessary to achieve the desired accuracy in the output of the network.

Chapter III includes new Regression Based Neural Network (*RBNN*) model that has been proposed for the training of *ANN*s. Using this algorithm, we are able to provide some initial intelligence to the *ANN* model in the form of the regression coefficients obtained by regression analysis. The initial weights are generated for the connections between the nodes of different layers with the help of regression analysis, in the proposed *RBNN* model. Three layer architecture for regression based *ANN* approach is considered in this chapter to understand the proposed model for solving the present problem. The input layer consists of single input as aspect ratio m of the elliptic plate and the output layer consists of one output in the form of the corresponding frequency parameter λ obtained from the Rayleigh-Ritz method using *BCOP*s. Various cases of the number of nodes depending upon the proposed parameter of the methodology have been considered in the hidden layer to facilitate a comparative study on the architecture of the network. The output of the network is computed by regression analysis combined with neural activation function performed at two stages, *i.e.*, the stage of hidden layer and the stage of output layer. Number of neurons in the hidden layer depends upon the degree of the regression polynomial that is used to fit the data between input and output. If a polynomial of degree n is considered, then number of nodes in hidden layer will be $(n+1)$ and the corresponding $(n+1)$ coefficients of this polynomial (say, a_i, $i = 0, 1, ..., n$) are taken as the initial weights from input layer to the hidden layer. Multivariate regression analysis is then used to obtain the initial weights from the hidden layer to output layer. Performance of this algorithm has also been compared with traditional *ANN* model. Proposed *RBNN* model has been tested for its performance by validating the network with frequency data of various intermediate aspect ratios of the elliptic plate. Error analysis between desired and neuron output of the network is also performed by comparing the Mean Square Error (*MSE*) for traditional *ANN* model and proposed *RBNN* model. The network is optimized with degree of the regression polynomial that is used to fit the data between input and output with respect to the number of nodes in the hidden layer that are necessary to achieve the desired accuracy in the output of the network.

In Chapter IV, a *RBNN* multi input single output model is used for modeling transverse vibration problem of annular circular and annular elliptic plates. This study has

mainly focused on the development of *RBNN* model to find the frequency of annular elliptic plates. In all, nine traditional *ANN* and nine *RBNN* models have been developed in this chapter corresponding to the nine boundary conditions. Three layer architecture for *ANN* is again considered to model the annular elliptic and circular plate vibration problems. Input layer consists of two inputs, *m* and *k* designating inner and outer aspect ratios of the annular elliptic plate. Output layer of the *ANN* architecture consists of one output in the form of the corresponding frequency parameters (λ) obtained from the *RR* method using the *BCOP*s. The output of the network is computed by regression analysis combined with neural activation function performed at the stage of hidden layer. Coefficients of the multivariable regression equation have been used as the initial weight matrix between the input and hidden layers. Different degree of multivariable regression analysis are studied for the investigation. Also, initial weights from the hidden layer to output layer are taken by fitting multivariate regression polynomial using the data obtained form hidden layer. Performance of this algorithm has been compared with traditional *ANN* model. The developed *RBNN* model has been tested for its performance by validating the network with frequency data of various intermediate aspect ratios of the annular circular and elliptic plates. Error analysis between desired and neuron output of the network is also performed by comparing the Mean Square Errors (*MSE*) for traditional *ANN* model and proposed *RBNN* model. Network is optimized with degree of the multivariable regression polynomial that is used to fit the data between input and output with respect to the number of nodes in the hidden layer that are necessary to achieve the desired accuracy in the output of the network.

In Chapter V, traditional *ANN* and proposed *RBNN* models have been used for identification of dynamic systems. A single degree of freedom system having equation of motion with/without damping as well as with/without force has been considered to explain the approach. The three layer architecture for neural network is considered to model the system. The input layer consists of two inputs, namely, displacement (x) and velocity ($\dot{x}$) measurements of the single degree of freedom system and the output layer consists of one output in the form of restoring force $r(x, \dot{x})$. With the success of the approach for this problem in this chapter for identification in a single degree of freedom system using the

newly developed *RBNN* model, the next chapter considers the investigation for identification in multi-degree of freedom system.

In chapter VI, regression based multi-input multi-output *ANN* model is used for multi-degree of freedom identification problems. The Multi-Degree of Freedom (*MDF*) system having equation of motion with/without damping and force has been considered for modeling and identification. Here, two problems of identification *viz.* partially known system and completely unknown system have been investigated. In partially known system, the vector of restoring forces $g(x,\dot{x})$ is identified from the input/output data, assuming that the mass vector M is known. On the other hand, in completely unknown system, the masses are not known and the system behaviour is modeled on the basis of input/output data only. Here two different multivariate polynomials are regressed first with each of the input data with the output data for determining the initial weights between input and hidden layer in case of these systems. Comparison of identification results between the traditional *ANN* and *RBNN* models have also been included in this Chapter.

1.7 Further Scope of Work

As mentioned above, the present work is based on soft computing techniques, in particular *ANNs*, in solving vibration and system identification problems. All the problems have been solved by developing and using new algorithm for *ANN* training *viz.*, the regression based neural network *RBNN* model. The initial weights have been generated for the connections between the nodes of different layers with the help of linear and nonlinear single as well as multivariate regression analysis. As such, the architecture of the *ANN* is forced to be fixed depending upon the problem. Moreover the solution is also found to be better and the proposed method turns out to be computationally efficient, simple, straight forward and having faster rate of convergence than the traditional *ANNs*. The work may be extended in many directions. Firstly, other soft computing techniques such as fuzzy set theory, approximate reasoning, and neuro-fuzzy *etc.* can be implemented using the proposed technique. Accordingly many other complex problems may be solved with the evolved methods. Secondly, vibration characteristics for different complex geometries with more complicated effects can be considered with the proposed *RBNN* algorithm. In case of system

identification problems including nonlinear systems such as structures subject to earthquakes or other random loads *etc*. can also be studied. The proposed algorithm is based on the selection of initial weights in the training of the *ANN*, which renders the problem to converge in less time and accuracy in the results are also maintained. The algorithm has been used with unipolar continuous activation function. One can also use other activation functions with other type of connection rules *etc*. to see whether these give a better result and less time of convergence or not. These are the challenges through out the globe to enhance the efficiency of *ANN* computing and the investigator believes that the present thesis fills some gap in this direction.

CHAPTER II

TRADITIONAL ARTIFICIAL NEURAL NETWORKS FOR VIBRATION OF PLATES

CHAPTER II

TRADITIONAL ARTIFICIAL NEURAL NETWORKS FOR VIBRATION OF PLATES

2.1 Introduction

Artificial neural networks have extensively been used in the field of structural engineering. Solution of vibration problems using artificial neural network has, in particular, fascinated the researchers working in the field of structural engineering during the last decade. Some of the characteristics of artificial neural networks appealing to structural engineers are non-linearity, parallelism, and learning capability. Although vibration theories have been well established, they may not be applied directly to actual structures because of the problems such as non-linearity, uncertainty, and time varying properties of structures. These problems necessarily make neural network a promising tool for the study of vibrations in practical applications. Artificial neural network provides the users a technique where one can solve the complex problems based on certain arithmetic operations only and one does not need to employ any analytical method for problem solving.

In this Chapter, artificial neural network model for plate vibrations has been undertaken. The training of network is performed using the patterns calculated with the help of Boundary Characteristic Orthogonal Polynomials (*BCOPs*) in the Rayleigh-Ritz method. The artificial neural network model has also been tested for its performance.

2.2 Vibration of Plates

Vibration analysis of plate shaped structures has been of interest to structural, aeronautical and mechanical designers for several decades. Dynamic behavior of these structures strongly depends on boundary conditions, geometrical shapes, material properties *etc.* Closed-form solutions are possible only for a limited set of simple boundary conditions and geometries. For analysis of plate shaped structures, several numerical methods such as finite element method, finite difference method, boundary element method *etc.* are usually applied. In the present study a computationally efficient method of *BCOPs* developed by *Bhat (1985*), *Singh and Chakraverty (1994*) and *Chakraverty (1998)* and its solution using the Rayleigh-Ritz (*RR*) method has been

considered. Besides providing more accurate results, this method is efficient, simple and easy for computer implementation. Series of papers (more than 100) have been published using this intelligent method and those have been covered in a recent review paper by *Chakraverty (1999)*. In this method, as usual, the expressions for the maximum kinetic and potential energies are obtained in terms of the arbitrary constants in the deflection expression. By equating the maximum potential and kinetic energies, it is possible to obtain an expression for the natural frequency of the structure. Applying the condition of stationarity of the natural frequencies at the natural modes, the variation of natural frequencies with respect to the arbitrary constants is equated to zero to obtain an eigenvalue problem (*Meirovitch 1980*). Solution of this eigenvalue problem provides the vibration characteristics of the system. *BCOP*s as used in *Bhat (1985*), *Singh and Chakraverty (1994*) and *Chakraverty (1998)*, render the problem into a standard eigenvalue problem rather than a generalized one. The elliptical geometry of the plate structure has been considered in this study and the aspect ratio of the ellipse is taken as $m = b/a$, where a and b are semi-major and semi-minor axes of the ellipse, respectively as shown in Figure 2.1. The domain of the plate is defined as:

$$R = \{(x, y), x^2 + y^2 / m^2 \leq 1,\ x, y \in \Re\} \qquad \ldots (2.1)$$

where $\Re$ is the set of real numbers.

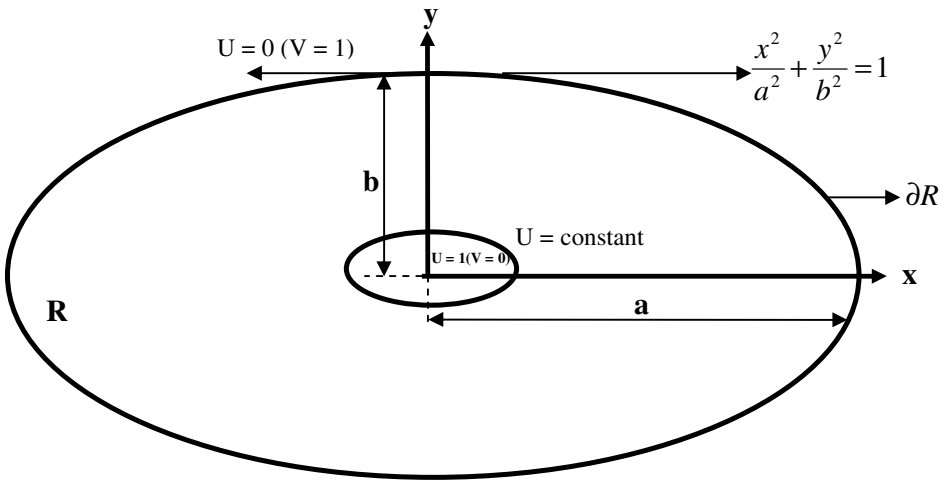


Figure 2.1: Geometry of the Elliptic Plate

The maximum potential and kinetic energies for the above problem can be written as:

$$V_{max} = \frac{D}{2}\iint_R [W_{xx}^2 + 2\nu W_{xx}W_{yy} + W_{yy}^2 + 2(1-\nu)W_{xy}^2]\, dxdy \qquad \ldots (2.2)$$

$$T_{max} = \frac{1}{2}\rho h\omega^2 \iint_R W^2 dxdy \qquad \ldots (2.3)$$

where W_{xx} is the deflection of the plate and subscripts on W denote differentiation with respect to the subscripted variable, $D = Y h^3 /(12(1-\nu^2))$ is flexural rigidity, Y is Young's modulus, ν is the Poisson's ratio, h is uniform thickness, ρ is the density and ω is the radian frequency of vibration. Equating the maximum potential and kinetic energies we can have the Rayleigh quotient for ω^2. By substituting the *N-term* approximation as the *BCOP*s and applying the condition of stationarity in the Rayleigh Ritz method, we will have the standard eigenvalue problem:

$$\sum_{j=1}^{N} (a_{ij} - \lambda^2 \delta_{ij})\, c_j = 0, \quad i = 1, 2, ..., N \qquad \ldots (2.4)$$

where $\delta_{ij} = \begin{cases} 0, & i \neq j \\ 1, & i = j \end{cases}$,

$\lambda^2 = a^4 \rho h \omega^2 / D$,

and a_{ij} are defined as in *Chakraverty (1998*). Solution of the above eigenvalue problem provides the vibration characteristics of the system.

2.3 Training Patterns

The training patterns for training the artificial neural network have been taken from solution of the Rayleigh-Ritz method with *BCOPs* for the vibrating plate. The boundary conditions are taken as clamped, simply supported and free. Solution of the eigenvalue problem of equation 2.4 gives the vibration frequencies and the results have been obtained for aspect ratios $m = 0.1$ *to* 1.0 in step of 0.1 that are reported in *Singh and Chakraverty (1991, 1992a, 1992b*) for various boundary conditions.

The methodology of *BCOPs* in the Rayleigh-Ritz method has been considered here because this is very powerful, simple and straight forward as reported in *Singh and Chakraverty (1994*) and *Chakraverty (1999*). The powerfulness of the method of *BCOPs* proved from the fact that for the special case of circular plate, the results by *BCOPs* method exactly matches with the exact solution by Bessel functions.

2.4 Training Algorithm

The feedforward Error Back Propagation Training Algorithm (*EBPTA*) with one hidden layer has been use in the present study. The typical network proposed in this study is given in Figure 2.2. In this figure Y_i, $i = 1, 2, ..., L$ are the neurons in input layer, P_j, $j = 1, 2, ..., J$ are the neurons in the hidden layer and O_k, $k = 1, 2, ..., K$ are neurons

in the output layer. The weights between input and hidden layers are denoted by V_{ji} and the weights between hidden and output layers denoted by W_{kj}.

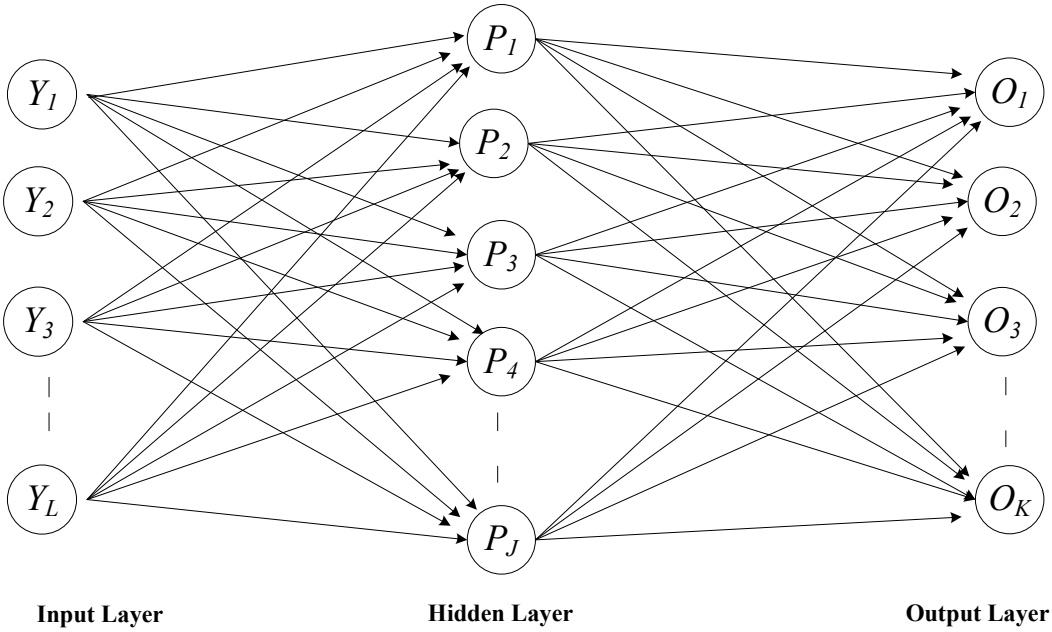


Figure 2.2: Layered feed forward neural network

The neural network architecture for the present problem is shown in Figure 2.3. The training algorithm typically includes the following steps.

Given N training pairs,

$$\{(Y_1, O_1);\ (Y_2, O_2);\ ...;\ (Y_N, O_N)\}$$

where $Y_i(L\times 1)$ is an input vector and $O_i(K\times 1)$ is desired output vector for the given training patterns. We define the error between desired and neural network outputs as,

$$Error = \tfrac{1}{2}(d_k - O_k)^2, \quad k = 1, 2, ..., K$$

where d_k is the output from the network and O_k is the desired output. The error signal terms of the output (δ_{Ok}) and hidden layers (δ_{pj}) can be written as

$$\delta_{Ok} = 0.5 * (d_k - O_k)(1 - O_k^{\ 2}), \quad k = 1, 2, ..., K$$

$$\delta_{pj} = 0.5 * (1 - p_j) \sum_{k=1}^{K} \delta_{Ok}\ W_{kj}, \quad j = 1, 2, ..., J$$

Consequently, output layer weights W_{kj} and hidden layer weights V_{ji} are adjusted as,

$$W_{kj}^{(New)} = W_{kj}^{(old)} + \eta \delta_{Ok} P_j, \quad k = 1, 2, ..., K \text{ and } j = 1, 2, ..., J$$

$$V_{ji}^{(New)} = V_{ji}^{(old)} + \eta \delta_{pj} Y_i, \quad j = 1,2,...,J \text{ and } i = 1, 2, ..., L$$

where η is the learning constant. The training patterns are again propagated in the network using the modified connecting weights and the error is again calculated. The procedure is repeated until one gets a desired predefined level of error.

2.5 Identification of the Model

A three-layer architecture for *ANN* is considered to model the plate vibration problem. Figure 2.3 gives the *ANN* architecture for the neural network used in the process. The input layer consists of two inputs, (i) specified boundary condition (*bc*) and (ii) aspect ratio (*m*) of the elliptic plate. The digitized values for the input parameters corresponding to the boundary conditions are fed as 2 for clamped, 1 for simply supported and 0 for free boundary condition. These values of 2, 1 and 0 for the three boundary conditions have been taken because of the fact that while generating the *BCOP*s, these were the powers considered to satisfy the geometrical boundary conditions for clamped, *(Singh and Chakraverty (1992a))*, simply supported, *(Singh and Chakraverty (1992b))* and free, *(Singh and Chakraverty (1991))* boundary conditions. The output layer of the *ANN* architecture consists of one output in the form of the corresponding frequency parameter (λ) obtained from the *RR* method using the *BCOP*s. Several cases of the number of nodes in the hidden layer have been considered to facilitate a comparative study on the architecture of the network. However, the number of nodes in the hidden layer has been reported as 10, 15, 20 and 25 only in this Chapter. The output of the network and the output of the neurons at the intermediate layer are computed using bipolar continuous sigmoidal function.

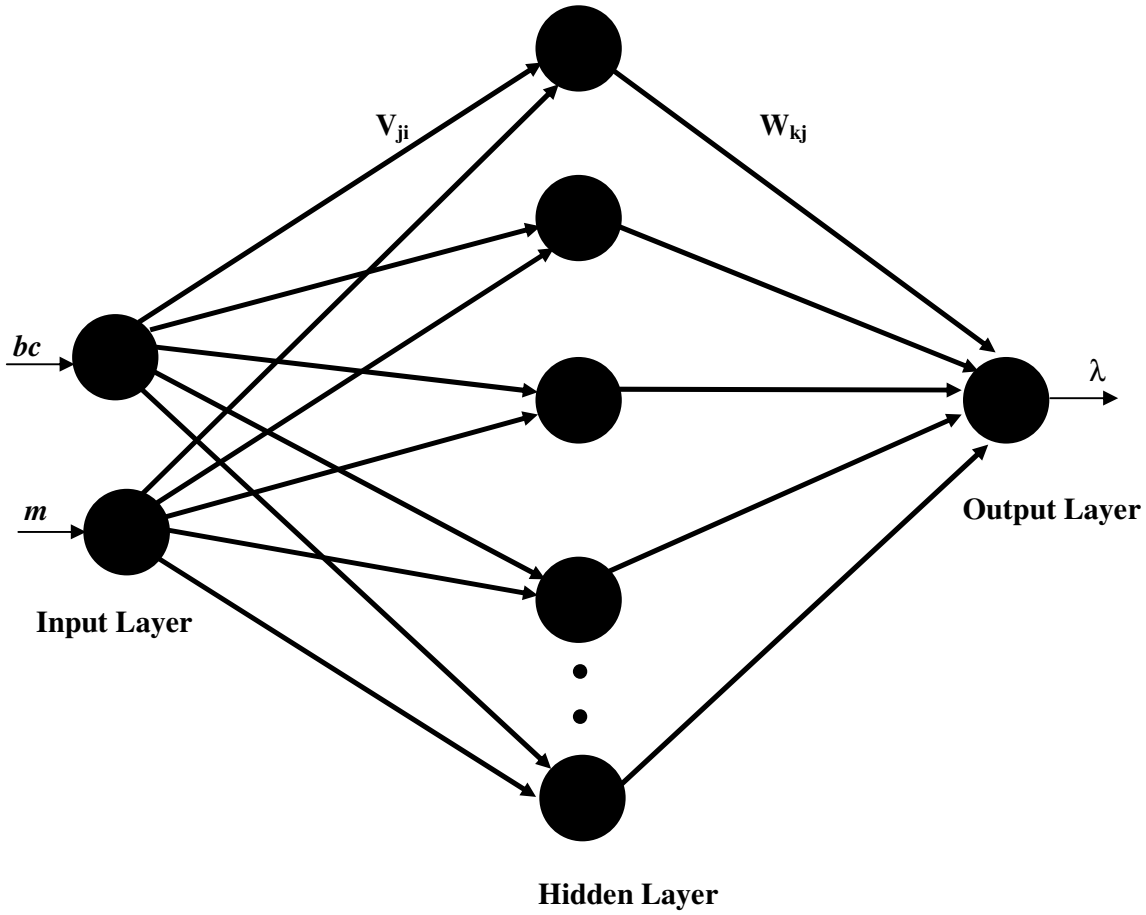


Figure 2.3: ***ANN*** **architecture used for estimation of frequency of the vibrating plate**

2.6 Training Phase

In the training phase, the network is trained with the help of values of input(s) and output(s) as mentioned earlier, called training patterns. The initial weights are generated randomly by initializing the generator with different seed values. Given a set of initial weights, *bc* and *m* these values are propagated in the network to find the output of the *ANN*, *i.e.*, λ and the error between output from *ANN* and desired output (from Rayleigh-Ritz method) is calculated. Based on this error, the quality of training of the neural network is decided. As given in section 2.4, the number of nodes in the hidden layer has been reported as 10, 15, 20 and 25 only in this documentation. The output of the network and the output of the neurons at the intermediate layer are computed by using bipolar continuous sigmoidal function $f(x)=\frac{2}{1+e^{-\alpha x}}-1$ where x and α denote weighted sum of the inputs parameter and threshold value, respectively. A network is considered to be trained if an acceptable level of the error has been achieved; otherwise the weights are adjusted in the direction of decrease in the error and next training pattern

is used to repeat the procedure until an acceptable accuracy has been achieved. When the desired accuracy has been achieved, then the network has learnt the characteristics of the system. The initial weights for the training of the network has been generated randomly by taking different seed values. It has been noted that the training of the network with unipolar continuous activation function not converged as per the desired output values. The effect of learning rate η on the total number of iterations and on the speed of the convergence was examined. The learning rate and activation function have been optimized during the training of the network for given training patterns. The training procedure has been repeated for different initial weights sets generated by different seed values. The input and output patterns are given in Table 2.1(a) to 2.1(c). Here, the first two columns (boundary condition (bc) and aspect ratios (m) are the inputs and the third column (desired frequency (λ)) is the output of the mentioned *ANN* architecture. First these input and output patterns are used to train the network. When the network has learnt the training patterns, then the *ANN* output is calculated for the specified boundary condition and particular aspect ratio. These calculated outputs for three different boundary conditions and various aspect ratios are also given in Table 2.1(a) to 2.1(c). Figures 2.4(a) to 2.4(c) contain the graphical representation of the performance of the *ANN* model using the input pattern for clamped, simply supported and free boundary conditions.

Table 2.1(a): Training results of the ANN using the input pattern for clamped boundary condition

Boundary Condition	***m***	***λ (Desired)***	***λ (Predicted by ANN)***
Clamped	1.00	10.216	10.215
	0.90	11.442	11.452
	0.80	13.229	13.219
	0.70	15.928	15.920
	0.60	20.195	20.228
	0.50	27.377	27.353
	0.40	40.646	40.654
	0.30	69.147	69.144
	0.20	149.66	149.652
	0.10	579.36	579.301

Table 2.1(b): Training results of the ANN using the input pattern for simply-supported boundary condition

Boundary Condition	***m***	***λ (Desired)***	***λ (Predicted by ANN)***
Simply Supported	1.00	4.9351	4.9220
	0.90	5.5282	5.5520
	0.80	6.3935	6.4010
	0.70	7.7007	7.6690
	0.60	9.7620	9.7380
	0.50	13.213	13.252
	0.40	19.514	19.496
	0.30	32.813	32.819
	0.20	69.684	69.684
	0.10	262.98	262.98

Table 2.1(c): Training results of the ANN using the input pattern for free boundary condition

Boundary Condition	***m***	***λ (Desired)***	***λ (Predicted by ANN)***
Free	1.00	5.3583	5.3727
	0.90	5.8381	5.8137
	0.80	6.1861	6.1796
	0.70	6.4185	6.4423
	0.60	6.5712	6.5972
	0.50	6.6705	6.6712
	0.40	6.7321	6.7067
	0.30	6.7654	6.7825
	0.20	6.7778	6.7723
	0.10	6.7781	6.7782

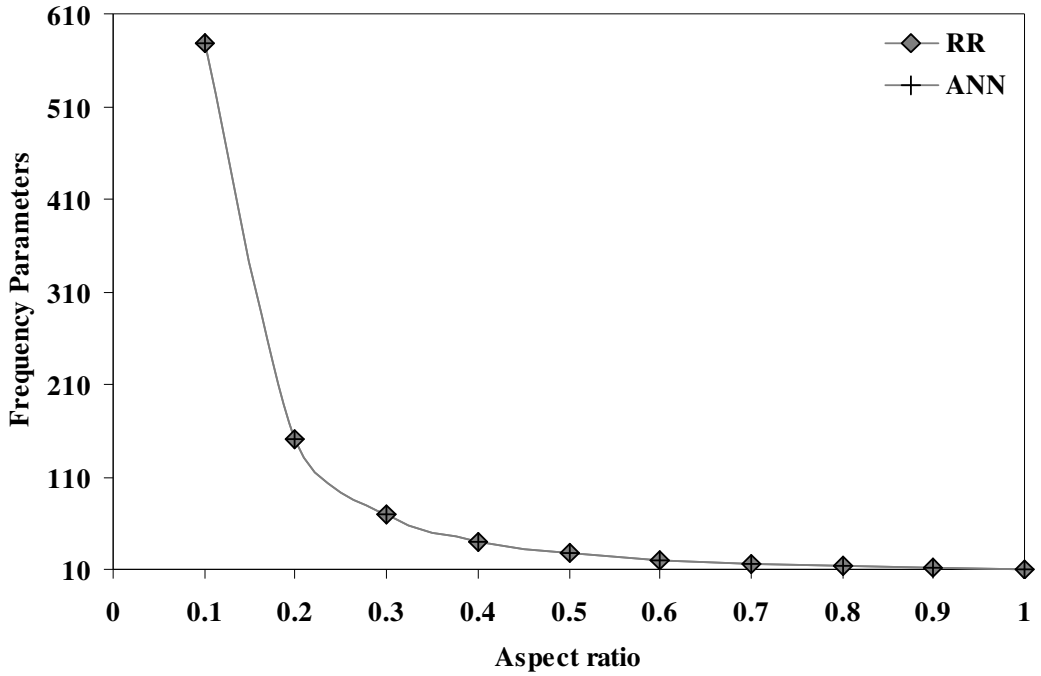


Figure 2.4(a): Results of training of the ANN model using the input pattern for clamped boundary condition

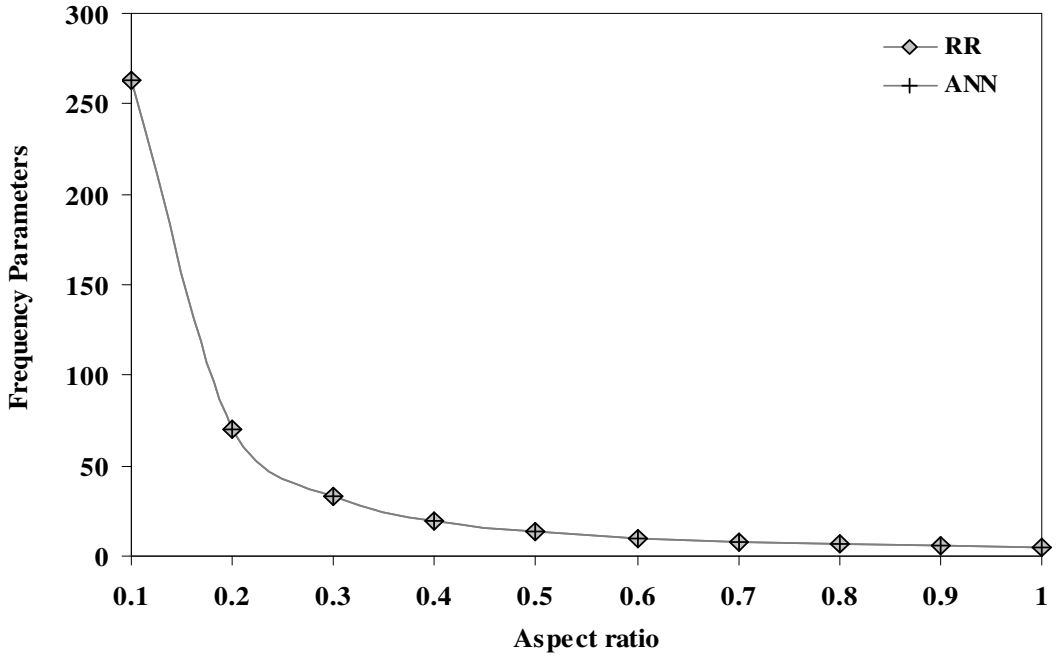


Figure 2.4(b): Results of Training of the ANN model using the input pattern for simply supported boundary

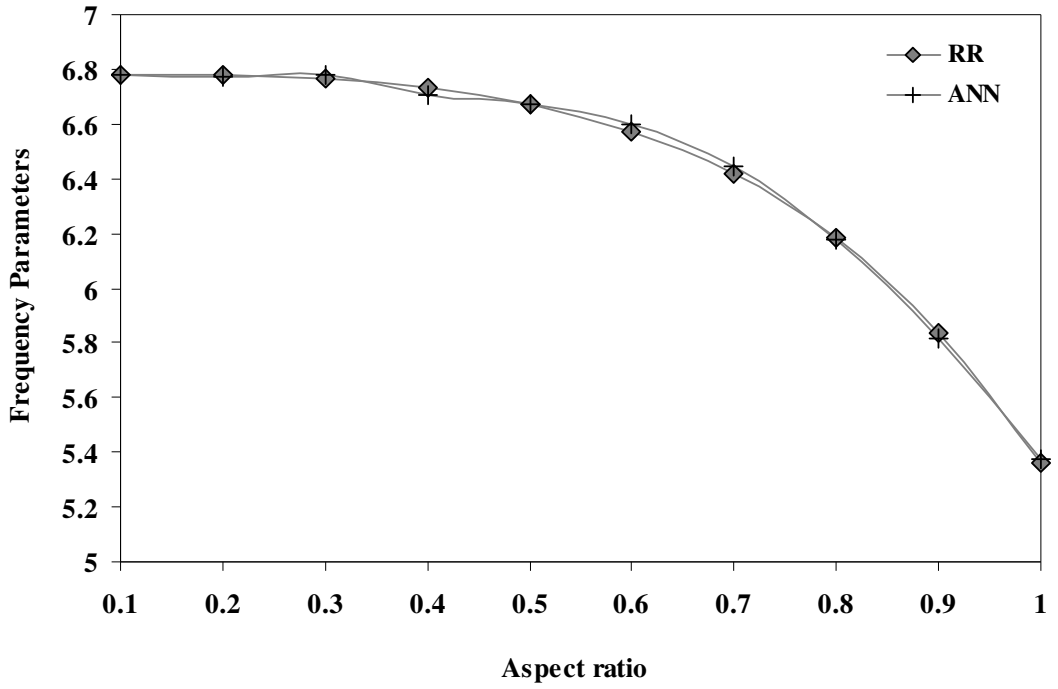


Figure 2.4(c): Results of Training of the ANN model using the input pattern for free boundary condition

2.7 Validation Phase

In order to establish the identifying ability of the network presented in this Chapter, the validation phase has been undertaken. The frequency data of intermediate aspect ratios of the elliptic plate have been taken from *Singh and Chakraverty (1991, 1992a, 1992b*) for various boundary conditions. These are the new data patterns (unknown data) that have not been used during the training of the network. This data for validating the network model is given in Tables 2.2. The comparison of the *ANN* output for the new patterns and the numerical computation by the Rayleigh-Ritz method has also been incorporated in Table 2.2 for all the three types of boundary conditions. It may be seen that the results obtained by the proposed model are in good agreement with the computed results of the problem obtained through the Rayleigh-Ritz method. Figures 2.5(a) to 2.5(c) contains the graphs of the validation of the *ANN* model for clamped, simply supported and free boundary condition.

Table 2.2: Validation of *ANN* Architecture

Boundary Condition	***m***	***λ (Desired)***	***λ (Predicted by ANN)***
Clamped	0.95	10.774	10.784
	0.85	12.148	12.249
	0.65	17.805	17.817
	0.45	32.913	32.829
Simply Supported	0.95	5.2049	5.2176
	0.85	5.9185	5.9403
	0.65	8.6087	8.5702
	0.45	15.853	15.892
Free	0.95	5.6135	5.6004
	0.85	6.0286	6.0082
	0.65	6.5029	6.5322
	0.45	6.7054	6.6907

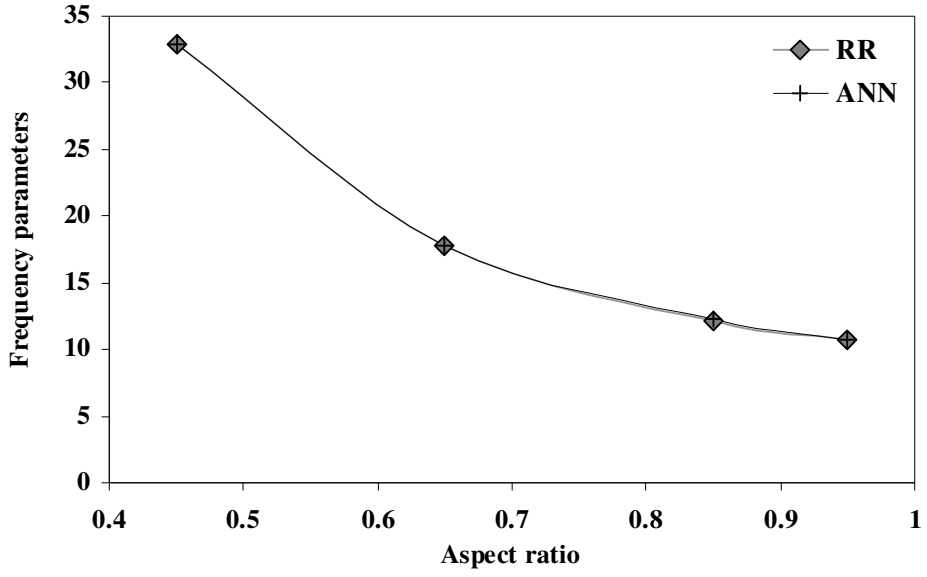


Figure 2.5(a): Validation of the ANN model using the unknown input pattern for clamped boundary condition

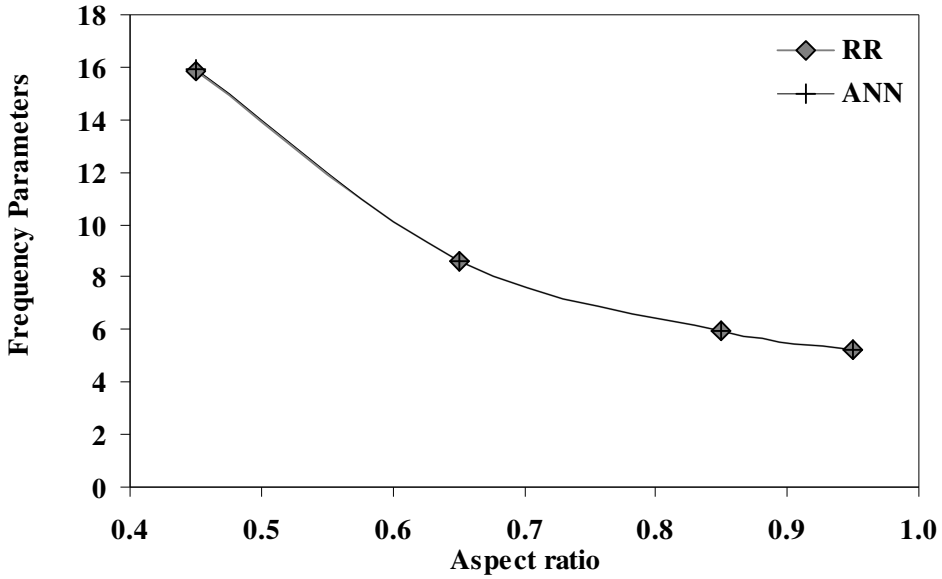


Figure 2.5(b): Validation of the ANN model using the unknown input pattern for simply-supported boundary condition

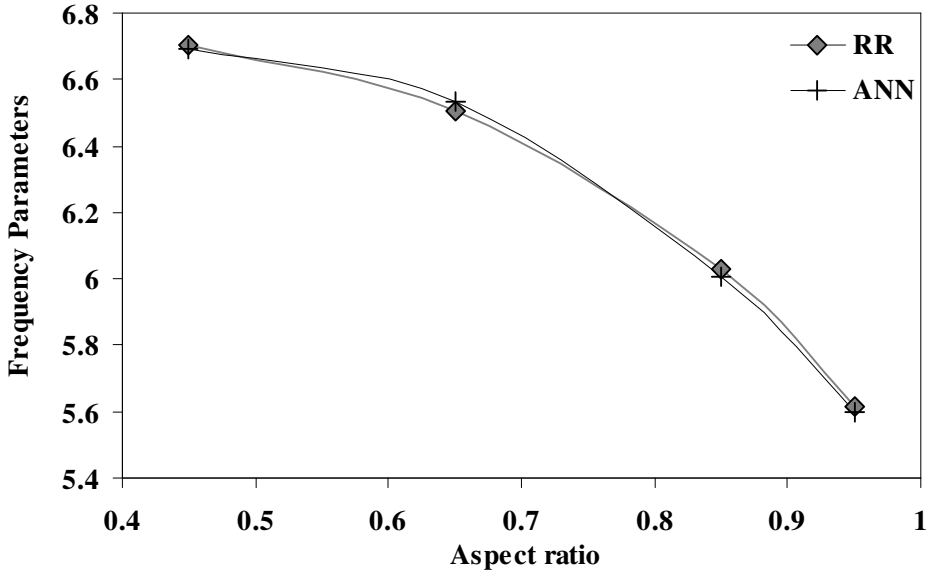


Figure 2.5(c): Validation of the ANN model using the unknown input pattern for free boundary condition

2.8 Error Analysis

In order to use *ANNs* as a reliable tool for vibration analysis of plate shaped structures, one has to take into account the factors that influence its prediction capabilites. As noted by several authors including *Kramer and Leonard (1990), Chryssolouris et al. (1996), Shao et al. (1997), Chinnam and Ding (1998) and Niyogi and Girosi(1999*), the performance of neural networks is influenced by noise corruption, spatial distribution and size of the data used to construct the *ANN* model, and the characteristics of the *ANN*, *i.e.,* number of layers, number of hidden nodes, its architecture, *etc.* The fact that the *ANN* is comprised of hidden layers and then nodes per layer to approximate an unknown function also introduces an error. The magnitude of this error depends on the representational capability of the *ANN*, which may increase due to overfitting as the size of the *ANN* becomes large. Another source of error stems from the fact that only finite data are available for training. As the number of training data sets increases, the error generally supposed to decrease. *Niyogi and Girosi (1999*) however, demonstrated that it is not possible to reduce the upper bounds on errors by increasing the *ANN* size and training data set simultaneously.

In this section, the effect of changing the number of neurons in the hidden layer on the error between desired and neuron output has also been studied. Four cases, namely, number of neurons in the hidden layer equal to 10, 15, 20 and 25 have been considered here. During the training of the network, it has been noted that if the initial random weight sets of the connections are also changed with the same number of nodes, then different number of iterations are needed to have the desired convergence. This is expected because the change in initialization also changes the time and number of iterations to achieve the convergence, which is usual as in other well-known numerical techniques. In order to have a comparison between the iterations that are carried out to achieve desired accuracy, same initial weights (using same seed value) for these cases have been considered. This process is repeated a large number of times with different sets of initial weights with the various chosen seed values. However, in this Chapter, the process is shown only by repeating it four times with four different sets of initial weights with different seed values. Number of iterations that are necessary to achieve the desired accuracy in the output has also been noted. Figures 2.6(i) to 2.6(iv) contain the graphs of error at different iterations as a function of natural logarithm of the number of iterations (required to achieve the accuracy). These graphs depict how the number of iterations changes, in order to achieve the desired accuracy, as we change the number of nodes in

the hidden layer. In Figures 2.6(i) to 2.6(iv), *J* denotes the number of nodes in the hidden layer and *S* denotes the arbitrary seed value considered in the computations for each of the Figures. Tables 2.3(a) to 2.3(c) illustrates the effect of number of nodes in the hidden layer on the *ANN* results for the frequency parameters for the same seed value of the weights that are used to train the network. It may be seen from this Table that the results do not change much if the number of nodes in hidden layer is increased from 10 to 25. But it has been noticed that the hidden layer should have at least 15 nodes for reliable results as shown in Tables 2.3(a) to 2.3(c).

Table 2.3(a): Effect of number of nodes in the hidden layer for clamped boundary condition

Boundary Condition	***m***	***λ (Desired)***	***λ (Predicted by ANN)***			
			No. of nodes in the hidden layer			
			10	15	20	25
Clamped	1.00	10.216	10.219	10.215	10.210	10.209
	0.90	11.442	11.444	11.452	11.459	11.468
	0.80	13.229	13.215	13.219	13.223	13.205
	0.70	15.928	15.930	15.920	15.919	15.932
	0.60	20.195	20.224	20.228	20.221	20.245
	0.50	27.377	27.349	27.353	27.364	27.343
	0.40	40.646	40.658	40.654	40.684	40.654
	0.30	69.147	69.151	69.144	69.158	69.149
	0.20	149.66	149.660	149.659	149.657	149.655
	0.10	579.36	579.291	579.301	579.288	579.298

Table 2.3(b): Effect of number of nodes in the hidden layer for simply supported boundary condition

Boundary Condition	***m***	***λ (Desired)***	***λ (Predicted by ANN)***			
			No. of nodes in the hidden layer			
			10	15	20	25
Simply Supported	1.00	4.9351	4.9246	4.9225	4.9315	4.9307
	0.90	5.5282	5.5502	5.5524	5.5333	5.5378
	0.80	6.3935	6.3934	6.4013	6.3976	6.3939
	0.70	7.7007	7.7104	7.6696	7.7167	7.7193
	0.60	9.7620	9.7582	9.7382	9.7493	9.7552
	0.50	13.213	13.242	13.252	13.238	13.229
	0.40	19.514	19.487	19.496	19.490	19.506
	0.30	32.813	32.820	32.819	32.818	32.921
	0.20	69.684	69.681	69.684	69.682	69.688
	0.10	262.980	262.990	262.982	262.987	262.991

Table 2.3(c): Effect of number of nodes in the hidden layer for free boundary condition

Boundary Condition	***m***	***λ (Desired)***	***λ (Predicted by ANN)***			
			No. of nodes in the hidden layer			
			10	15	20	25
Free	1.00	5.3583	5.3357	5.3827	5.3778	5.3701
	0.90	5.8381	5.8648	5.8137	5.8003	5.8224
	0.80	6.1861	6.2090	6.1796	6.1660	6.1780
	0.70	6.4185	6.4543	6.4423	6.4514	6.4610
	0.60	6.5712	6.5278	6.5712	6.6231	6.6041
	0.50	6.6705	6.6540	6.6712	6.6726	6.6750
	0.40	6.7321	6.7807	6.7067	6.6855	6.6941
	0.30	6.7654	6.7825	6.7723	6.7788	6.7881
	0.20	6.7778	6.7723	6.7661	6.7723	6.7684
	0.10	6.7781	6.7782	6.7782	6.7719	6.7810

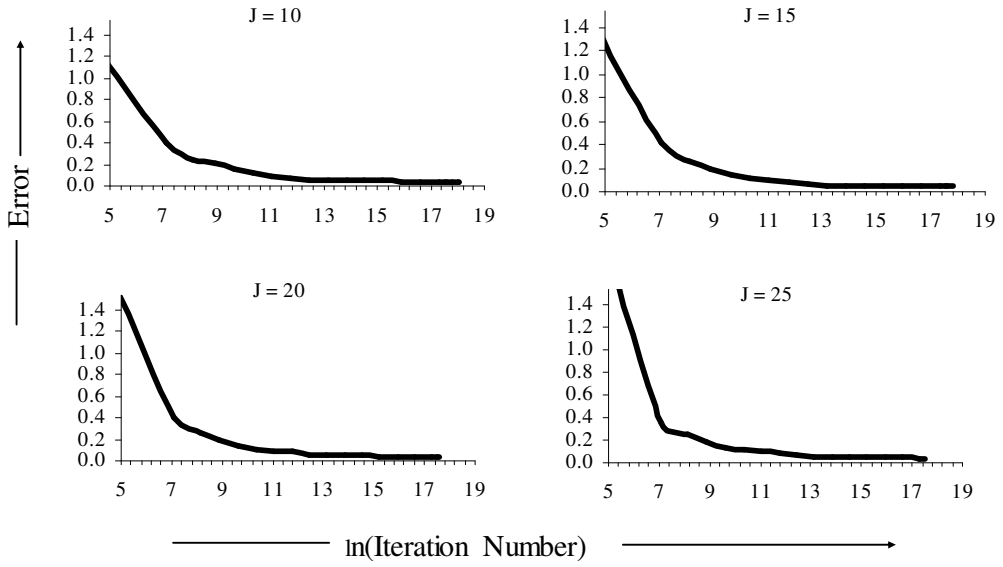


Figure 2.6(i) Effect of number of iterations on Error varying the number of nodes in the hidden layer taking seed value as 2705

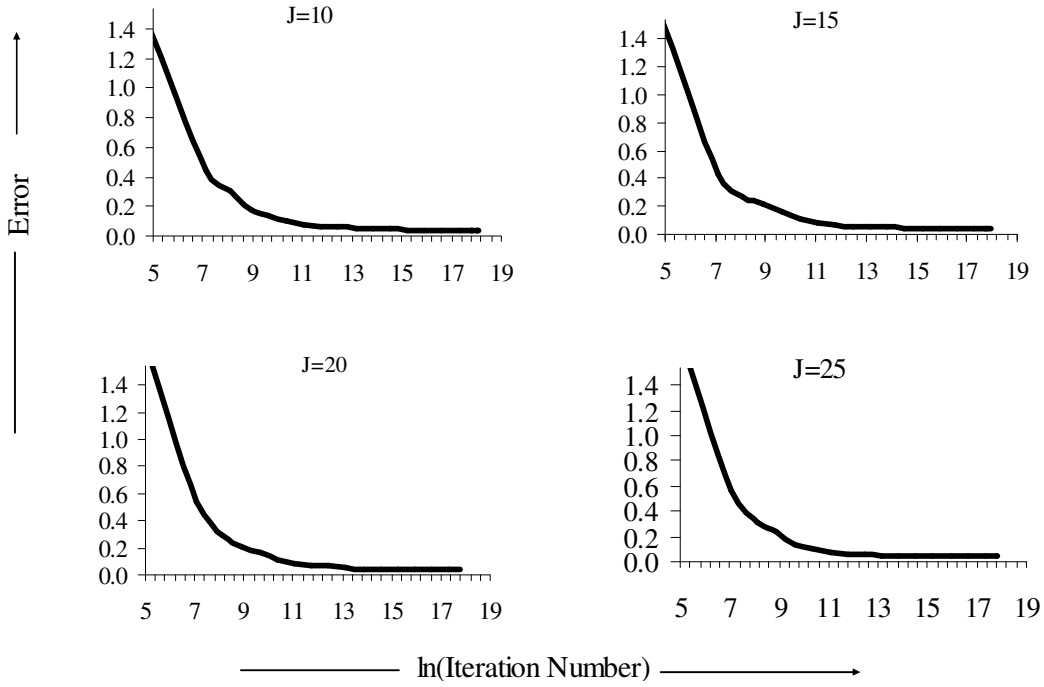


Figure 2.6(ii) Effect of number of iterations on Error varying the number of nodes in the hidden layer taking seed value as 3307

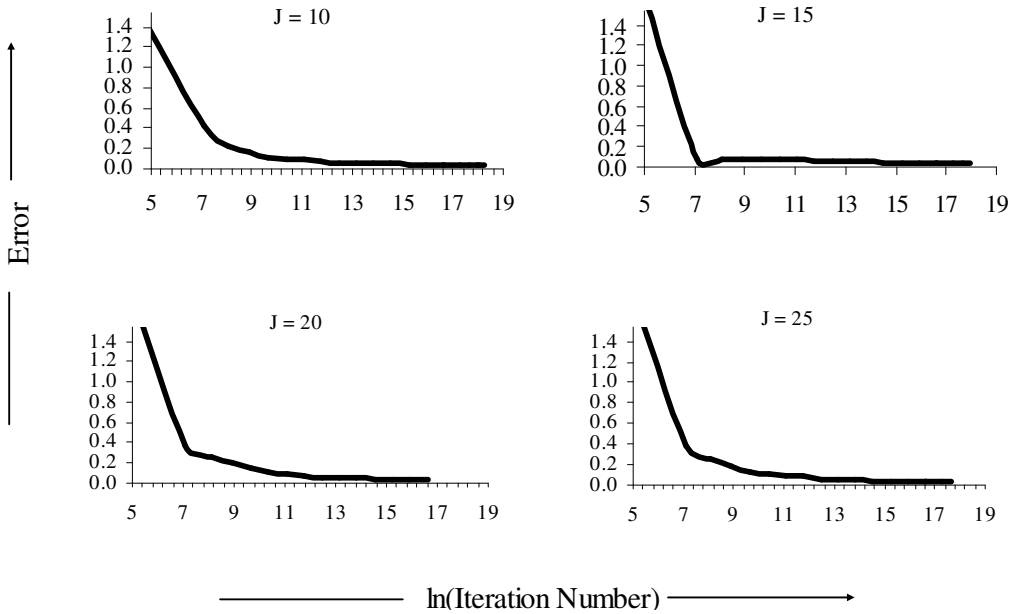


Figure 2.6(iii) Effect of number of iterations on Error varying the number of nodes in the hidden layer taking seed value as 7303.

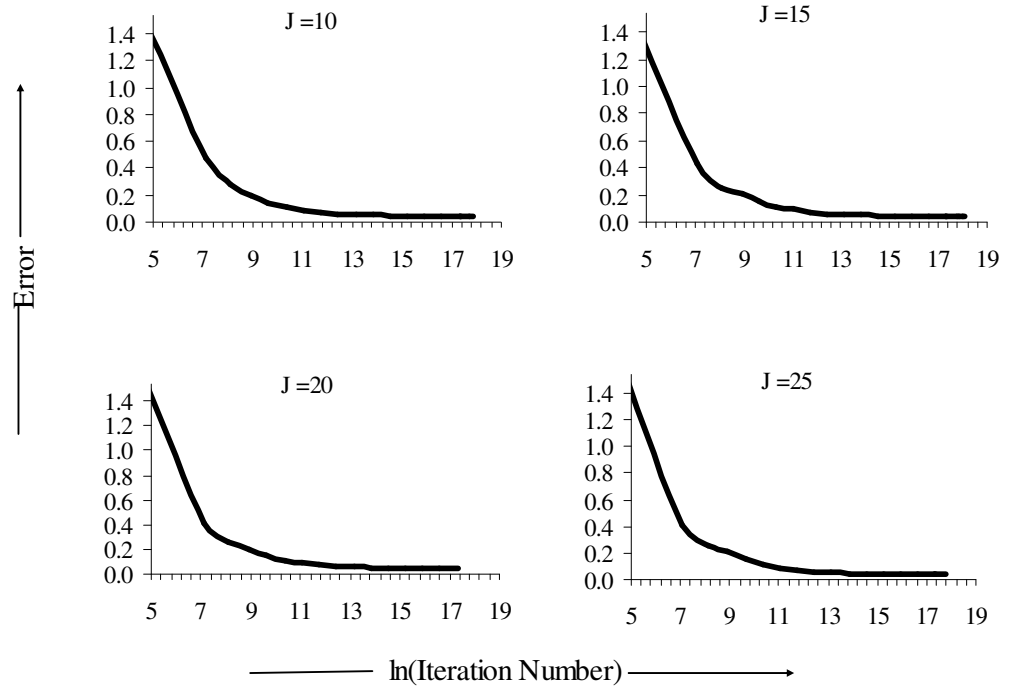


Figure 2.6(iv) Effect of number of iterations on Error varying the number of nodes in the hidden layer taking seed value as 99307

2.9 Conclusion

The methodology proposed in this Chapter gives a direct estimation of the frequencies as compared to the traditional solution procedure by solving fourth order differential equation of motion for the plates. It is well documented now that there are plenty of data and results available in literature for wide variety of problems of vibration. These may be used as discussed in this Chapter to get a well trained *ANN* architecture for future use of the designer to have the results for particular intermediate points (for example in the present problem as discussed, the intermediate points are the values of the aspect ratios). Otherwise the designer has to understand and execute the complete code for his particular problem for a particular parameter (*e.g.,* aspect ratios for the present problem) in order to get the results, which is time consuming and sometimes may not be reliable. Thus, a soft computing method has been used in this Chapter which may be applied for various other vibration problems for the designer's required solution. The powerfulness and reliability of the method has also been discussed and established.

CHAPTER III

REGRESSION BASED NEURAL NETWORK (*RBNN*) FOR TRANSVERSE VIBRATION OF CIRCULAR AND ELLIPTIC PLATES

(The contents of this Chapter are published in Journal of Computer Methods in Applied Mechanics and Engineering, (2006), 195, PP 4194-4202)

CHAPTER III

REGRESSION BASED NEURAL NETWORK (*RBNN*) FOR TRANSVERSE VIBRATION OF CIRCULAR AND ELLIPTIC PLATES

(The contents of this Chapter are published in Journal of Computer Methods in Applied Mechanics and Engineering, (2006), 195, PP 4194-4202)

CHAPTER III

REGRESSION BASED NEURAL NETWORK FOR TRANSVERSE VIBRATION OF CIRCULAR AND ELLIPTIC PLATES

In Chapter II, we have illustrated the use of traditional *ANN* models to solve the problem of vibration of plates. We call this a traditional *ANN* model since we have used the well known traditional error back propagation algorithm in order to train the network. Proposition of new training algorithms and their implementation in different application areas has been the focus of researchers in the field of *ANN* modeling. In this Chapter, we have proposed a new algorithm for the training of *ANN*s. The performance of this algorithm has been compared with traditional *ANN* model that uses the error back propagation algorithm for training. We have again considered the example problem as discussed in Chapter II for the development of the proposed Regression Based Neural Network (*RBNN*) model in this Chapter. As will be evident during the progress of this chapter, we generate the initial weights representing the connections between the nodes of different layers with the help of regression analysis, in the proposed *RBNN* model.

3.1 Transverse Vibrations of Circular and Elliptic Plates

As mentioned in Chapter II, the transverse vibrations of circular and elliptic plates have been studied extensively in the literature and these studies are already surveyed in Chapter II. It is worth to mention again that plate vibration problem involves the solution of fourth order differential equation which is fairly complex to solve. Accordingly, *ANN* model has been used to analyze the problem in Chapter II. It may be noted that we need to define (or fix) a number of parameters including number of hidden layers and number of nodes in each hidden layer for the convergence of the training process. The suitable and optimal value of these parameters is generally obtained using hit and trial method. A training methodology which it defines the parameters completely or partially will certainly have an edge over the other methodologies. Motivated by this factor, we have proposed a new training algorithm in this Chapter. Using this algorithm, we are able to provide with some initial intelligence to the *ANN* model in the form of the regression coefficients obtained by regression analysis.

3.2 The Training Patterns

The training patterns for training the neural network model has been taken from the solution by the Rayleigh-Ritz method with Boundary Characteristic Orthogonal Polynomials for the vibrating plate as discussed in Chapter II. The three boundary conditions have again been considered as clamped, simply supported and free. Solution of the eigenvalue problem (2.3) gives the vibration frequencies and the results have been obtained for aspect ratios $m = 0.1$ *to* 1.0 in step of 0.1 which are reported in *Singh and Chakraverty (1991, 1992a, 1992b*) for various boundary conditions. These patterns are given below for ready reference.

Table 3.1(a): Training patterns for the proposed *RBNN* model for the Clamped boundary conditions

m	λ
0.1	579.36
0.2	149.66
0.3	69.147
0.4	40.646
0.5	27.377
0.6	20.195
0.7	15.928
0.8	13.229
0.9	11.442
1	10.216

Table 3.1(b): Training patterns for the proposed *RBNN* model for the Simply supported boundary conditions

m	λ
0.1	262.98
0.2	69.684
0.3	32.813
0.4	19.514
0.5	13.213
0.6	9.7629
0.7	7.7007
0.8	6.3935
0.9	5.5282
1	4.9351

Table 3.1(c): Training patterns for the proposed *RBNN* model for the free boundary conditions

m	λ
0.1	6.7781
0.2	6.7778
0.3	6.7654
0.4	6.7321
0.5	6.6705
0.6	6.5712
0.7	6.4185
0.8	6.1861
0.9	5.8381
1	5.3583

3.3 The *RBNN* Model

A three-layer architecture for regression based artificial neural network approach is considered in this Chapter to understand the proposed model for solving the present problem. Figure 3.1 shows the neural network used in the process. The input layer consists of single input as aspect ratio m of the elliptic plate and the output layer consists of one output in the form of the corresponding frequency parameter λ obtained from the Rayleigh-Ritz method using *BCOP*s. Two cases of the number of nodes depending upon the proposed parameter of the methodology have been considered in the hidden layer to facilitate a comparative study on the architecture of the network. The output of the network is computed by regression analysis combined with neural activation function performed at two stages, *i.e.*, the stage of hidden layer and the stage of output layer. Number of neurons in the hidden layer depends upon the degree of the regression polynomial that is used to fit the data between input and output. If we consider a polynomial of degree n then number of nodes in hidden layer will be $(n+1)$ and the corresponding $(n+1)$ coefficients of this polynomial (say, a_i, $i = 0, 1, ..., n$) are taken as the initial weights from input layer to the hidden layer. Architecture of the network for a polynomial of degree three is shown in Figure 3.1. As discussed, it will have four nodes in the hidden layer corresponding to four coefficients of the regression polynomial. The example problem has been discussed and solved by taking the polynomials of degree three and four in this Chapter.

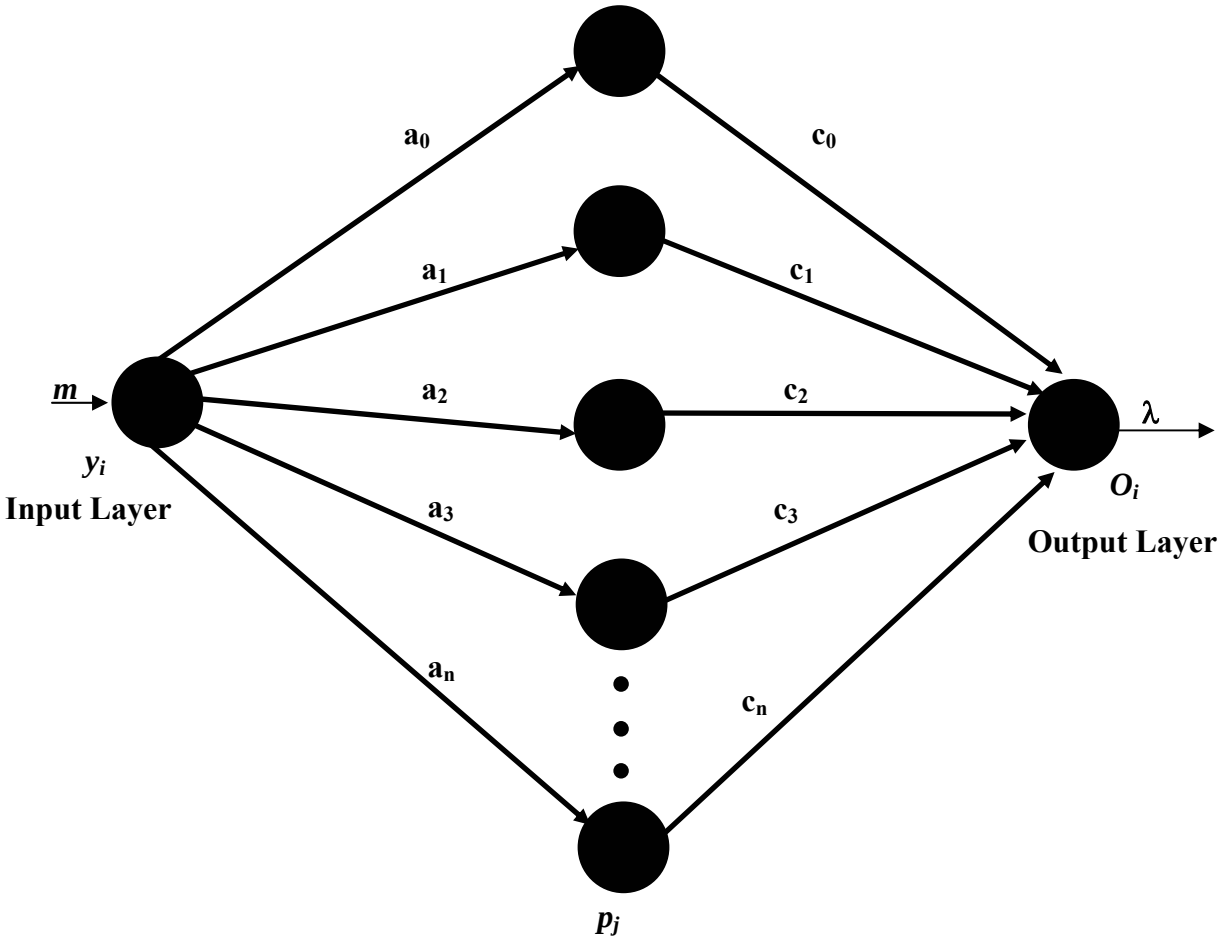


Figure 3.1: Architecture of neural network diagram used for estimation of frequency of the vibrating plate using regression coefficients as weights with four neurons in hidden layer

3.4 Training Algorithm

In this section, we illustrate the training algorithm that has been used to train the *RBNN* model. The training algorithm is described in subsection 3.4.1 and the experiment that has been designed using this proposed algorithm is illustrated in subsection 3.4.2.

3.4.1 The *RBNN* Training Algorithm

Let us take N training patterns as $\{(y_1, O_1); (y_2, O_2); \ ...; \ (y_N, O_N)\}$. These patterns $\{(y_i, O_i), \quad i = 1, 2, ..., N\}$ may, in general, consist of the vector quantities. However, in the present problem, we have considered a single input single output

problem. Let us denote the input and output of the given patterns by **y** and ***O***, respectively. Let us also consider the neural network as given in Figure 3.1 for the development of the training algorithm. Since the hidden layer consists of four neurons, we will have to consider a regression polynomial of degree three in order to find the weights connecting input neurons to hidden neurons. We thus define the weight matrix ***W*** for the input layer as,

$$\boldsymbol{W} = \begin{bmatrix} w_{1,1} \\ w_{2,1} \\ w_{3,1} \\ w_{4,1} \\ . \\ . \\ w_{n,1} \end{bmatrix} = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ . \\ . \\ a_n \end{bmatrix}$$

where a_0, a_1, a_2, a_3, …, a_n are the coefficients of the regression polynomial of degree *n*,

$$p(y) = a_0 + a_1 y + a_2 y^2 + a_3 y^3 + ... + a_n y^n \qquad \ldots (3.1)$$

fitted to the given N training patterns $\{(y_i, O_i),\ i = 1, 2, ..., N\}$.

Now, we calculate the outputs of the neurons of hidden layer using the following activation function.

$$p_j^i = \frac{1}{1 + e^{-\alpha\ a_j\ (y_i)^j}}, \quad j = 0, 1, 2, 3, ..., n; \quad i = 1, 2, ..., N \qquad \ldots (3.2)$$

where α is the training parameter that is to be chosen suitably.

The weight matrix ***V*** for the hidden layer is defined as

$$\boldsymbol{V} = \begin{bmatrix} \boldsymbol{v}_{1,1} \\ \boldsymbol{v}_{2,1} \\ \boldsymbol{v}_{3,1} \\ \boldsymbol{v}_{4,1} \\ . \\ . \\ \boldsymbol{v}_{n,1} \end{bmatrix}$$

where c_0, c_1, c_2, c_3, …., c_n are the coefficients of the multivariate linear regression

polynomial,

$$c_0 p_0^i + c_1 p_1^i + c_2 p_2^i + c_3 p_3^i + ... + c_n p_n^i \quad \text{... (3.3)}$$

fitted to the data set $\{ p_j^i, O_i \}$

The output of the network is now calculated as,

$$o_i = c_0 p_0^i + c_1 p_1^i + c_2 p_2^i + c_3 p_3^i + ... + c_n p_n^i, \quad i = 1, 2, ..., N \quad \text{… (3.4)}$$

The error associated with this *ANN* model is defined as,

$$E = \frac{1}{2} \sum_{i=1}^{N} (o_i - O_i)^2 \quad \text{… (3.5)}$$

This error, as is the case with other *ANN* models is used to decide whether the *ANN* model has learnt from the given training patterns or not. As such, if ***E*** is less than some tolerance level we say that *ANN* model has got trained otherwise we update the elements of ***W*** and ***V*** according to the following process.

Equation 3.5 can be written as,

$$E = \frac{1}{2} \sum_{i=1}^{N} (c_0 p_0^i + c_1 p_1^i + c_2 p_2^i + c_3 p_3^i + ... + c_n p_n^i - O_i)^2$$

$$\text{or, } E = \frac{1}{2} \sum_{i=1}^{N} [\frac{c_0}{1 + e^{-\alpha a_0}} + \frac{c_1}{1 + e^{-\alpha a_1 y_i}} + \frac{c_2}{1 + e^{-\alpha a_2 y_i^2}} + \frac{c_3}{1 + e^{-\alpha a_3 y_i^3}} + \quad ..... + \frac{c_n}{1 + e^{-\alpha a_n y_i^n}} - O_i]^2$$

Thus,

$$\frac{\partial E}{\partial a_0} = \frac{1}{2} \sum_{i=1}^{N} 2(o_i - O_i)(-c_0 (1 + e^{-\alpha a_0})^{-2} e^{-\alpha a_0} (-\alpha)$$

$$\text{or, } \frac{\partial E}{\partial a_0} = \sum_{i=1}^{N} \frac{(o_i - O_i) \alpha c_0 e^{-\alpha a_0}}{(1 + e^{-\alpha a_0})^2}$$

Similarly,

$$\frac{\partial E}{\partial a_1} = \sum_{i=1}^{N} \frac{(o_i - O_i) \alpha y_i c_1 e^{-\alpha a_1 y_i}}{(1 + e^{-\alpha a_1 y_i})^2}$$

$$\frac{\partial E}{\partial a_2} = \sum_{i=1}^{N} \frac{(o_i - O_i) \alpha y_i^2 c_2 e^{-\alpha a_2 y_i^2}}{(1 + e^{-\alpha a_2 y_i^2})^2}$$

$$\frac{\partial E}{\partial a_3} = \sum_{i=1}^{N} \frac{(o_i - O_i) \alpha y_i^3 c_3 e^{-\alpha a_3 y_i^3}}{(1 + e^{-\alpha a_3 y_i^3})^2}$$

.
.
.

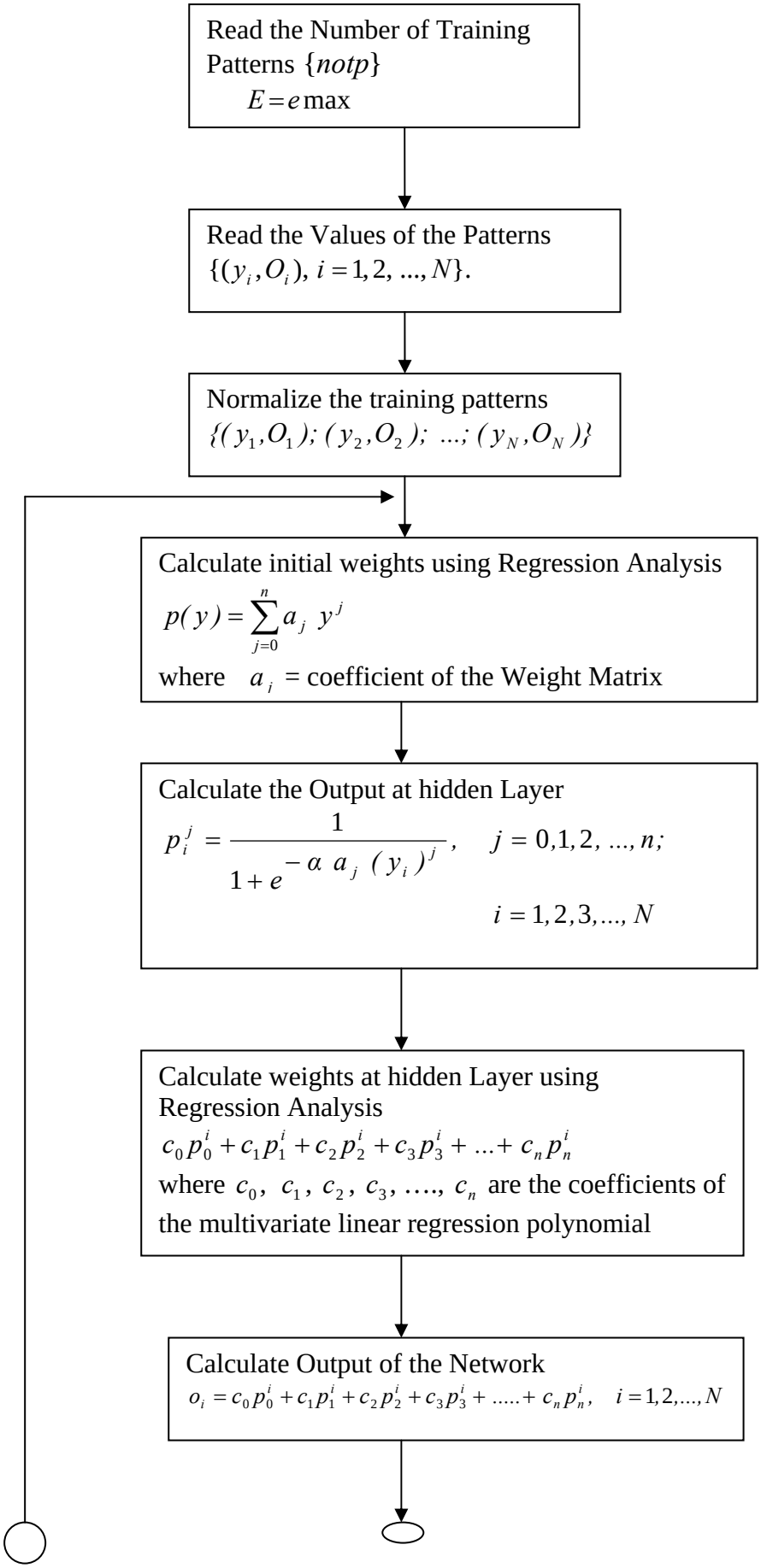
$$\frac{\partial E}{\partial a_n} = \sum_{i=1}^{N} \frac{(o_i - O_i)\alpha\, y_i^n\, c_n\, e^{-\alpha\, a_n\, y_i^n}}{(1 + e^{-\alpha\, a_3 y_i^n})^2}$$

The weights matrix $\boldsymbol{W}$ is redefined as,

$$\boldsymbol{W} = \begin{bmatrix} a_0 - \frac{\partial E}{\partial a_0} \\ a_1 - \frac{\partial E}{\partial a_1} \\ a_2 - \frac{\partial E}{\partial a_2} \\ a_3 - \frac{\partial E}{\partial a_3} \\ \vdots \\ a_n - \frac{\partial E}{\partial a_n} \end{bmatrix} \quad \ldots (3.6)$$

Using the matrix $\boldsymbol{W}$ as changed above, we can calculate the new outputs of the hidden neurons using equation 3.2.

Now, the weight matrix $\boldsymbol{V}$ is calculated by fitting the multivariate linear regression polynomial as given in equation 3.3 using the new outputs of the hidden layer. Using this new $\boldsymbol{V}$, the output of the network can again be calculated using equation 3.4. This training procedure is repeated until the desired tolerance level of error E is achieved and thus we may say that the *RBNN* model has learnt from the given training patterns. A flow chart based on above mentioned algorithm is given in Figure 3.2.


Read the Number of Training Patterns {notp}
E = emax
Read the Values of the Patterns
{(y_i, O_i), i = 1, 2, ..., N}.
Normalize the training patterns
{(y_1, O_1); (y_2, O_2); ...; (y_N, O_N)}
Calculate initial weights using Regression Analysis
p(y) = Σ_{j=0}^{n} a_j y^j
where a_i = coefficient of the Weight Matrix
Calculate the Output at hidden Layer
p_i^j = 1 / (1 + e^{− α a_j (y_i)^j}), j = 0, 1, 2, ..., n;
i = 1, 2, 3, ..., N
Calculate weights at hidden Layer using Regression Analysis
c_0 p_0^i + c_1 p_1^i + c_2 p_2^i + c_3 p_3^i + ... + c_n p_n^i
where c_0, c_1, c_2, c_3, …., c_n are the coefficients of the multivariate linear regression polynomial
Calculate Output of the Network
o_i = c_0 p_0^i + c_1 p_1^i + c_2 p_2^i + c_3 p_3^i + ... + c_n p_n^i, i = 1, 2, ..., N

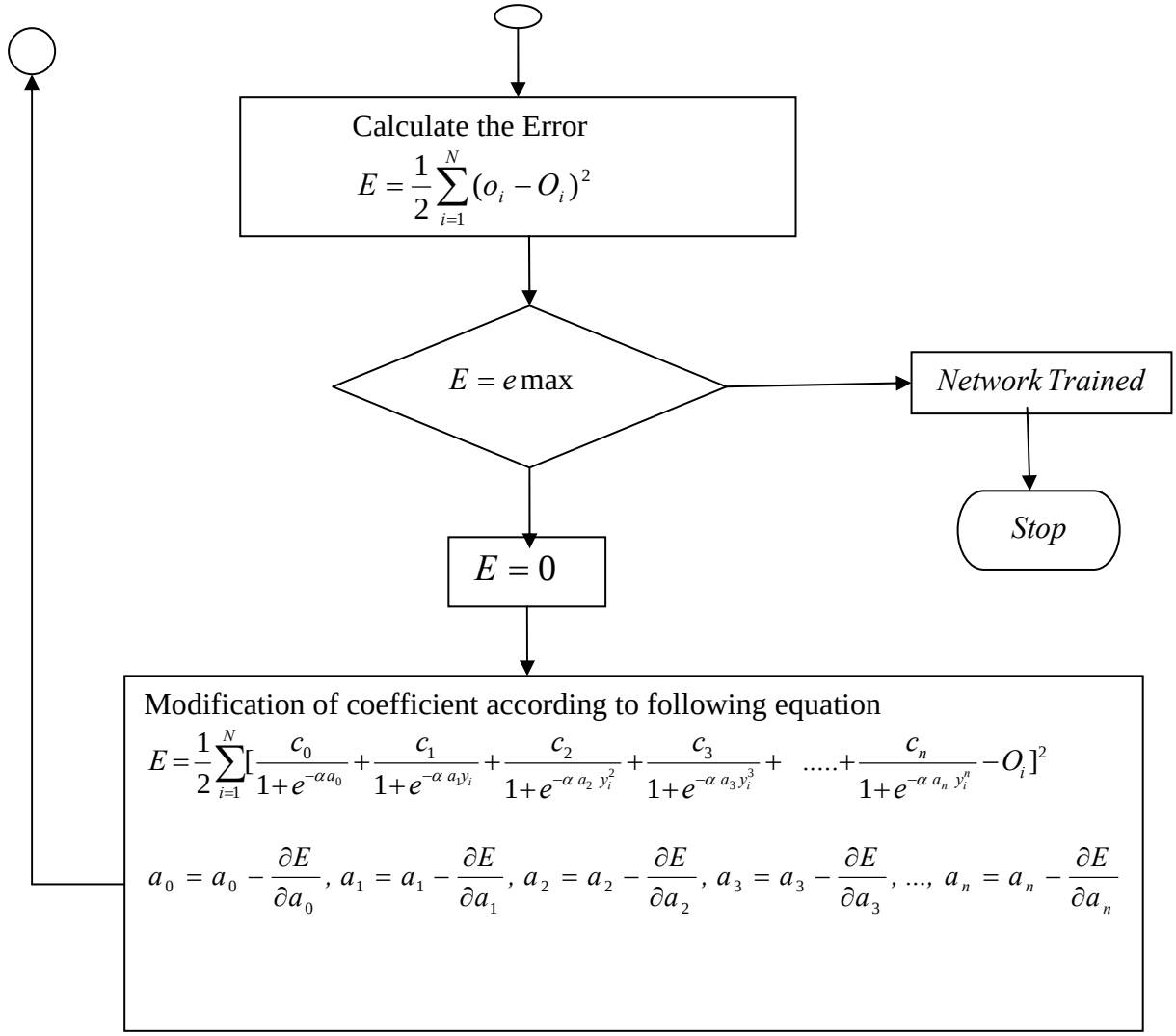


Figure 3.2: Flow diagram of *RBNN* approach

3.4.2 Experiment I

The procedure described in subsection 3.4.1 for training the proposed *RBNN* model has been implemented in the MatLab environment. In the first phase, regression polynomials of degree three are fitted to the training patterns given in Table 3.1(a), 3.1(b) and 3.1(c). These polynomials are:

$$p_1(y) = 1.5756 - 8.0210\,y + 12.8336\,y^2 - 6.424\ y^3 \quad \ldots (3.7)$$

$$p_2(y) = 1.5725 - 9.1220\,y + 6.2969\,y^2 - 6.3578\,y^3 \quad \ldots (3.8)$$

$$p_3(y) = 1.0078 - 0.9101\,y + 0.3138\,y^2 - 0.4381\,y^3 \quad \ldots (3.9)$$

The coefficients of these polynomials are taken as the connecting weights between input and hidden layer. Using these weights and equation 3.2, the outputs of the neurons in the hidden layer are calculated. These values are used to fit a regression polynomial of the

form of equation 3.3 and now we can calculate the output of the network using equation 3.4. At this stage, the error of the *RBNN* model is calculated and a decision is taken whether the network has been trained or not. If the tolerance level of the error is not achieved, we redefine the connection weights ***W*** according to equation 3.6 and repeat the procedure otherwise we say that the network has got trained. As reported earlier, we have used the regression polynomials of degree 3 and degree 4 to implement the proposed *RBNN* model. The regression polynomials corresponding to the polynomial of degree 4 for the training patterns given in Tables 3.1(a), 3.1(b) and 3.1(c) are:

$$p_1(y) = 2.153 - 15.427y + 39.933y^2 - 43.455y^3 + 16.832y^4 \quad \ldots (3.10)$$

$$p_2(y) = 2.142 - 15.268y + 39.442y^2 - 42.878y^3 + 16.601y^4 \quad \ldots (3.11)$$

$$p_3(y) = 0.997 + 0.047y - 0.199y^2 + 0.256y^3 - 0.315y^4 \quad \ldots (3.12)$$

For the sake of presentation the *RBNN* model with the regression polynomial of degree 3 is named as NRM3 and that with regression polynomial of degree 4 is named as NRM4 in this Chapter. The results of training of the proposed models NRM3 and NRM4 are given in Figures 3.3(a) to 3.3(c) and Figures 3.4(a) to 3.4(c), respectively.

3.4.3 Experiment II

In order to report the benefits of using the *RBNN* modeling approach as explained in Experiment I over the traditional *ANN* modeling, we have performed this experiment. In this experiment, two traditional *ANN* models are used. The architecture of these models is similar to the *RBNN* model as proposed in Experiment I. This experiment has again been performed in the MatLab environment using built-in function of the Neural Network Toolbox (*NNT*). The network is trained with the help of values of input(s) and output(s) taken as training patterns as given in Tables 3.1(a) to 3.1(c). The training procedure in this experiment is based on the feedforward backpropagation algorithm. The built-in transfer functions, namely, tan-sigmoid (*tansig*) and linear (*purelin*) of the *NNT* have been used in the input layer and the output layer, respectively, in this experiment. The neural network based on the feedforward backpropagation algorithm has been trained with conjugate gradient, quasi-Newton and Levenberg-Marquardt built-in training functions of the *NNT*. The connection weights interconnecting the neurons between different layers are taken through a random number generator built-in function in the *NNT*. The built-in transfer functions *tansig* and *purelin* along with the different training functions have been adjusted interchangeably between input and output layers to get the network converged at

the desired level of accuracy as set in Experiment I. The learning rate parameter has been varied from 0.01 to 0.05 with a step of 0.01 in the training of the network. The training of the network using conjugate gradient Fletcher-Reeves algorithm (*traincgf*), Polak-Ribiere conjugate gradient algorithm (*traincgp*), Powell-Beale conjugate gradient algorithm (*traincgb*), scaled conjugate gradient (*trainscg*) and one-step secant (*trainoss*) algorithms could not converge and the performance goal not met for the same data and tolerance of accuracy as set in Experiment I.

In this experiment, the network could get trained with the quasi-Newton training algorithm (*trainlm*) by meeting the desired performance goal, but the mean square error in this case is higher than the mean square error obtained in Experiment I. Tables 3.2(a) and 3.2(b) depict the values of *MSE* for NRM3 and NRM4 architectures. It has been observed that average *MSE* value for the *RBNN* model is 92.47% less as compared to *ANN* model in clamped boundary conditions, 61.87% less in simply supported boundary conditions and 84.44% less in free boundary conditions for NRM3 architecture. Also, average *MSE* value for *RBNN* model are 90% less as compared to *ANN* model in case of clamped boundary conditions, 89% less as compared to *ANN* model in case of simply supported boundary conditions and 98% less as compared to *ANN* model in case of free boundary conditions for NRM4 architecture. The pattern characteristics of the traditional *ANN* model are also incorporated in Figures 3.3(a) to 3.3(c) for NRM3 architecture and Figures 3.4(a) to 3.4(c) for NRM4 architecture for clamped, simply supported and freely boundary conditions of plate geometry, respectively. Figures 3.3(a) to 3.3(c) also compare the desired output, *ANN*'s output and *RBNN*'s output of the NRM3 architecture for clamped, simply supported and free boundary conditions. Similar results are incorporated in Figures 3.4(a) to 3.4(c) for NRM4 architecture. It has also been noted that the proposed *RBNN* model reduces the average convergence time by 80% for clamped and simply supported boundary conditions and by 63% for free boundary conditions

Table 3.2(a): Comparison of Mean Square Error (*MSE*) of the *ANN* model and *RBNN* model for NRM3 architecture

Boundary Conditions	***MSE* (*ANN* Model)**	***MSE* (*RBNN* Model)**	**Relative decrease in *MSE* of *RBNN* model over *ANN* model**
Clamped	3.544	0.267	92.47%
Simply Supported	0.139	*0.053*	61.87%
Free	0.055	0.00006	99.89%

Table 3.2(b): Comparison of Mean Square Error (*MSE*) of the *ANN* model and *RBNN* model for NRM4 architecture

Boundary Conditions	*MSE* (*ANN* Model)	*MSE* (*RBNN* Model)	Relative decrease in *MSE* of *RBNN* Model over *ANN* Model
Clamped	6.566	0.655	90.00%
Simply Supported	1.365	0.146	89.30%
Free	0.002	0.00004	98%

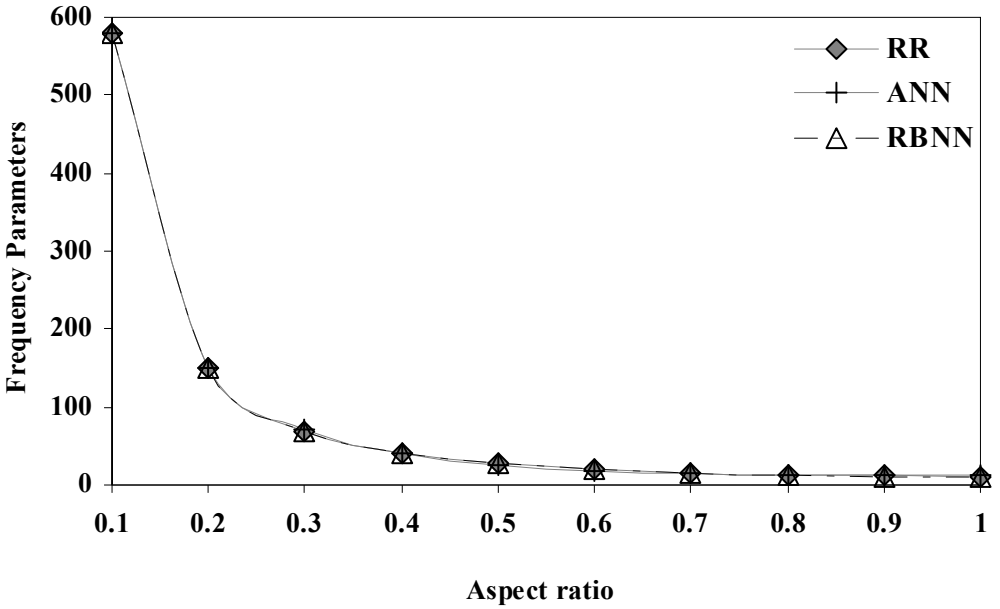


Figure 3.3(a): Comparison among *RR*, *ANN* and *RBNN* approach for Clamped boundary (NRM3 – Architecture)

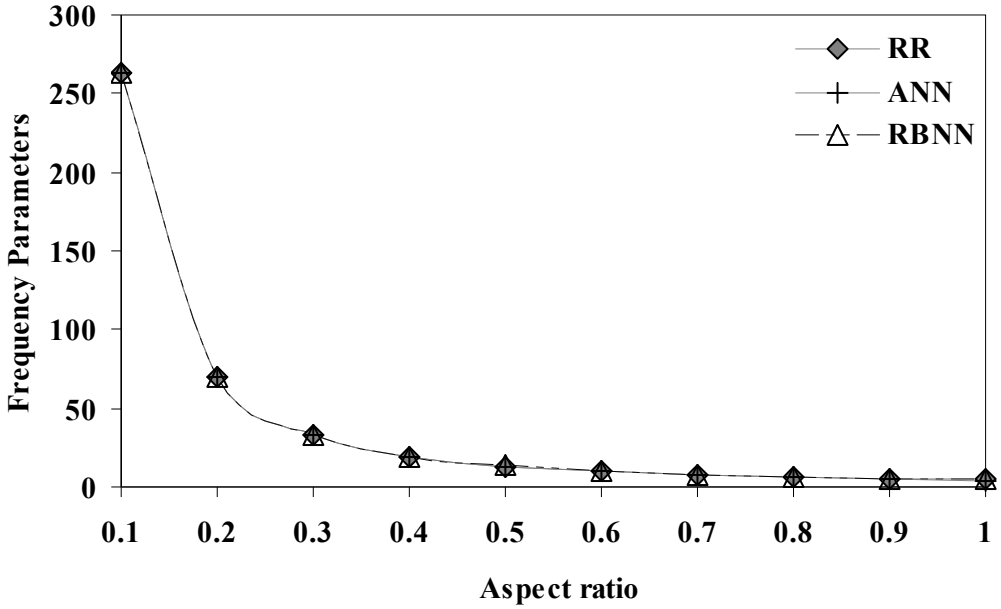


Figure 3.3(b): Comparison among *RR, ANN* and *RBNN* approach for Simply Supported boundary (NRM3 – Architecture)

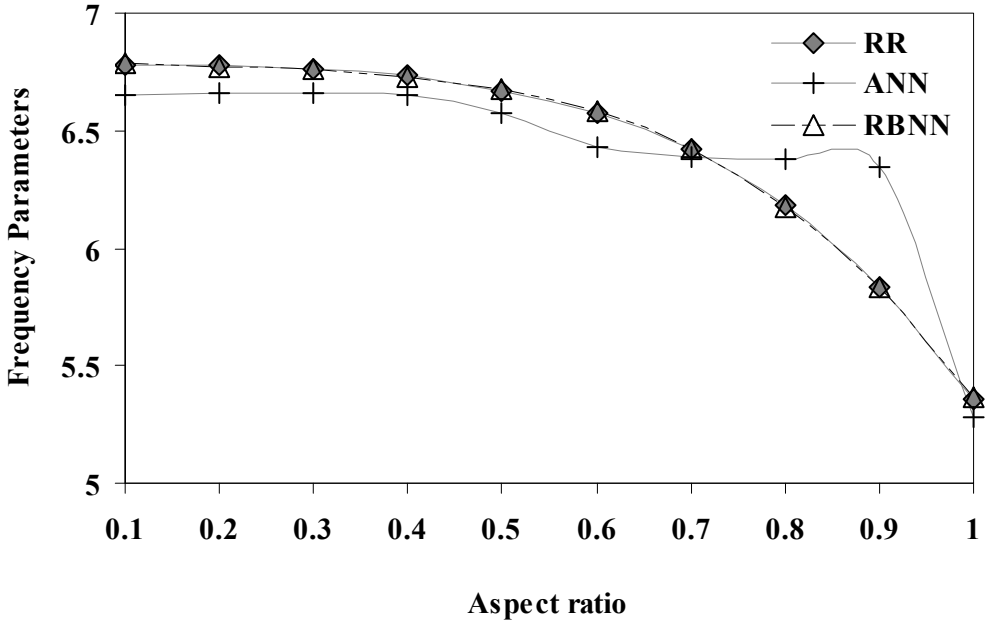


Figure 3.3(c): Comparison among *RR*, *ANN* and *RBNN* approach for Free boundary (NRM3 - Architecture)

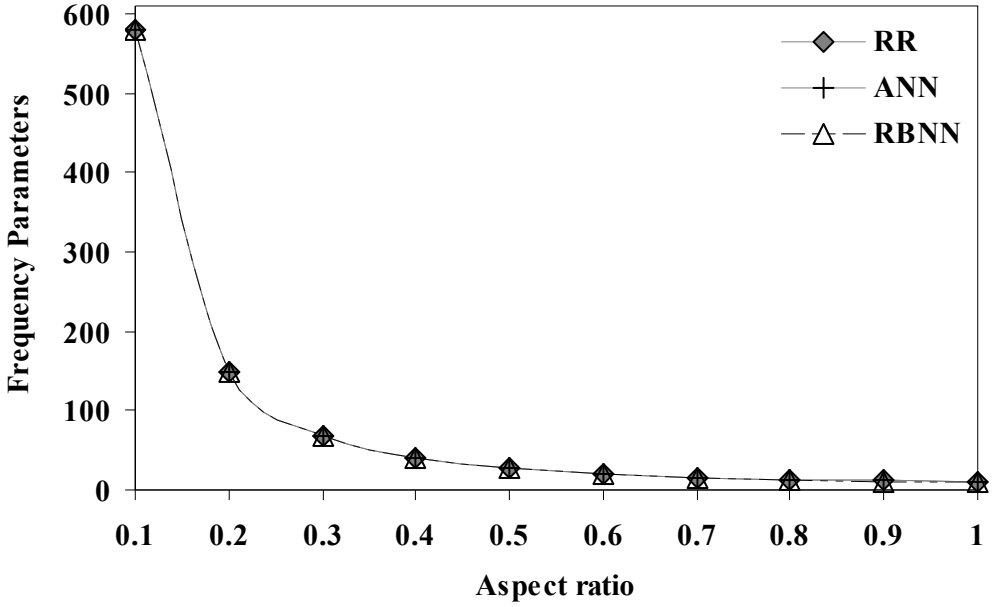


Figure 3.4(a): Comparison among *RR*, *ANN* and *RBNN* approach for Clamped boundary (NRM4 – Architecture)

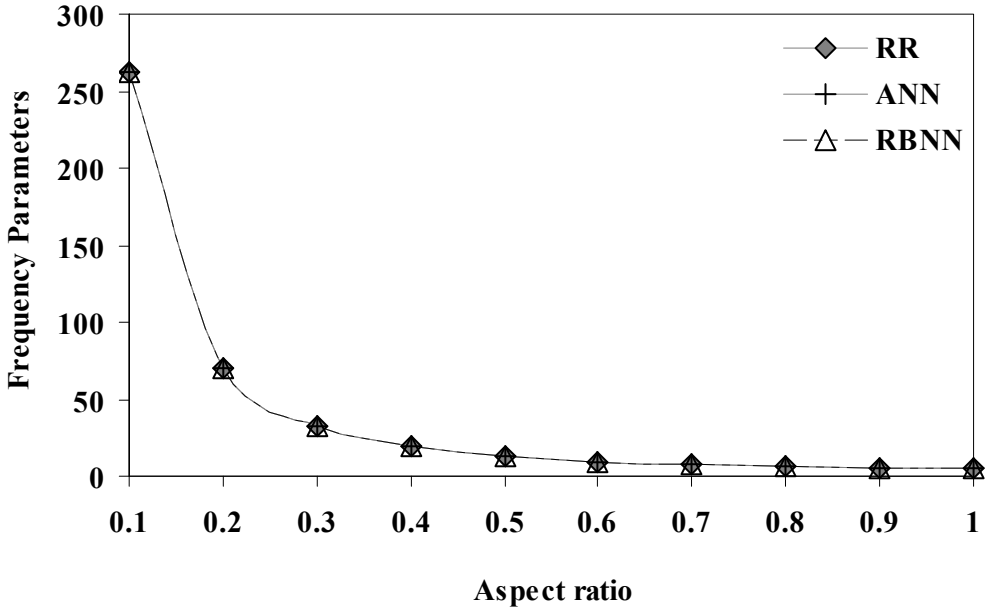


Figure 3.4(b): Comparison among *RR*, *ANN* and *RBNN* approach for Simply-Supported boundary (NRM4 – Architecture)

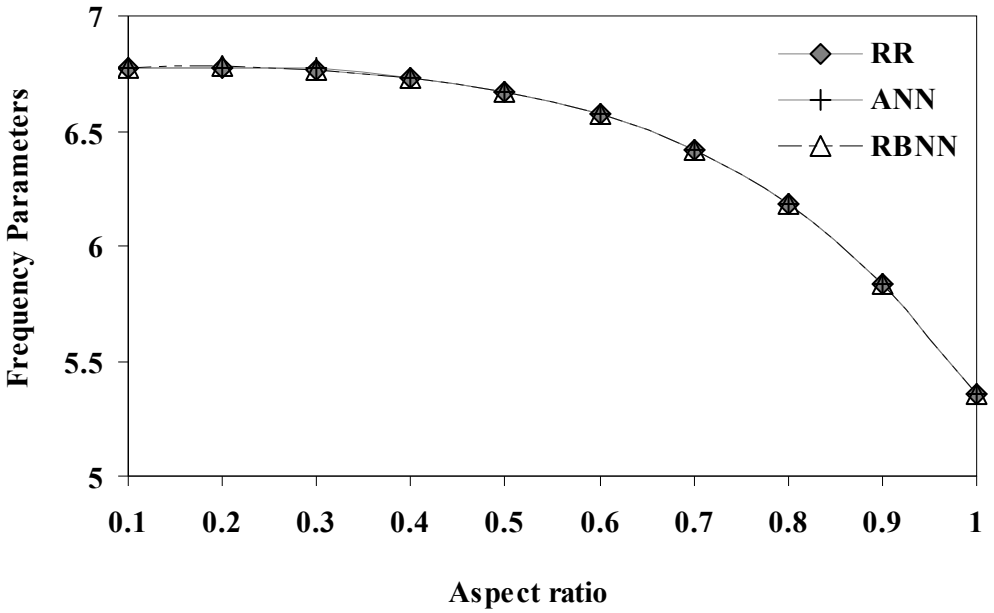


Figure 3.4(c): Comparison among *RR*, *ANN* and *RBNN* approach for Free boundary (NRM4 - Architecture)

3.5 Validation Phase

In this phase, the identifying ability of the network has been analyzed. The frequency data of various intermediate aspect ratios of the plate (new patterns) have again been taken from *Singh and Chakraverty (1991, 1992a, 1992b*) for various boundary conditions. This data for validating the proposed model is given in Tables 3.3(a) to 3.3(c). The results of the validation phase are reported in Figures 3.5(a) to 3.5(c) for NRM3 architecture and in Figures 3.6(a) to 3.6(c) for NRM4 architecture. In these Figures, *RR* represents the numerical values of the frequency taken from the literature, *ANN* represents the values of frequency as obtained by traditional *ANN* models and *RBNN* represents frequency obtained from the proposed model. It has been noted that the network for NRM3 and NRM4 architectures could converge using Quesi-Newton training function with the learning rate value of 0.05. The mean square error of the output value in this case is higher in comparison with the proposed *RBNN* model. It can be inferred from the Figures 3.5(a) to 3.5(c) and Figures 3.6(a) to 3.6(c) that NRM3 and NRM4 architectures based on proposed *RBNN* model give better results when compared with traditional *ANN* model.

Table 3.3(a): Testing patterns for the proposed *RBNN* model for the Clamped boundary conditions

m	λ
0.25	97.599
0.35	51.894
0.45	32.913
0.55	23.292
0.65	17.805
0.75	14.433
0.85	12.248
0.95	10.774

Table 3.3(b): Testing patterns for the proposed *RBNN* model for the Simply supported boundary conditions

m	λ
0.25	45.917
0.35	24.793
0.45	15.853
0.55	11.253
0.65	8.6087
0.75	6.9796
0.85	5.9185
0.95	5.2049

Table 3.3(c): Testing patterns for the proposed *RBNN* model for the Free boundary conditions.

m	λ
0.25	6.7737
0.35	6.7518
0.45	6.7054
0.55	6.6264
0.65	6.5029
0.75	6.3144
0.85	6.0286
0.95	5.6135

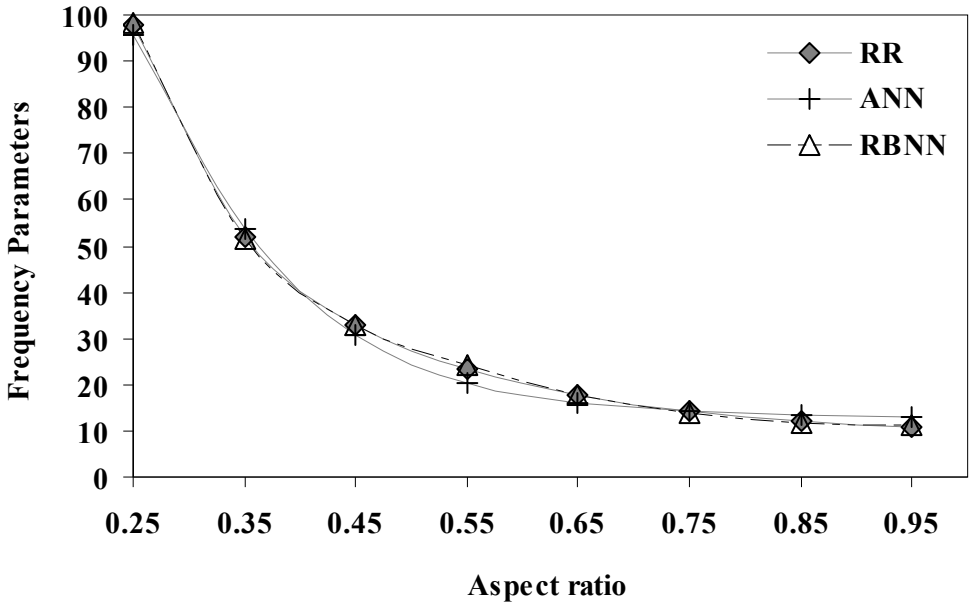


Figure 3.5(a): Comparison among *RR*, *ANN* and *RBNN* approach for Validation of Clamped boundary conditions (NRM3 - Architecture)

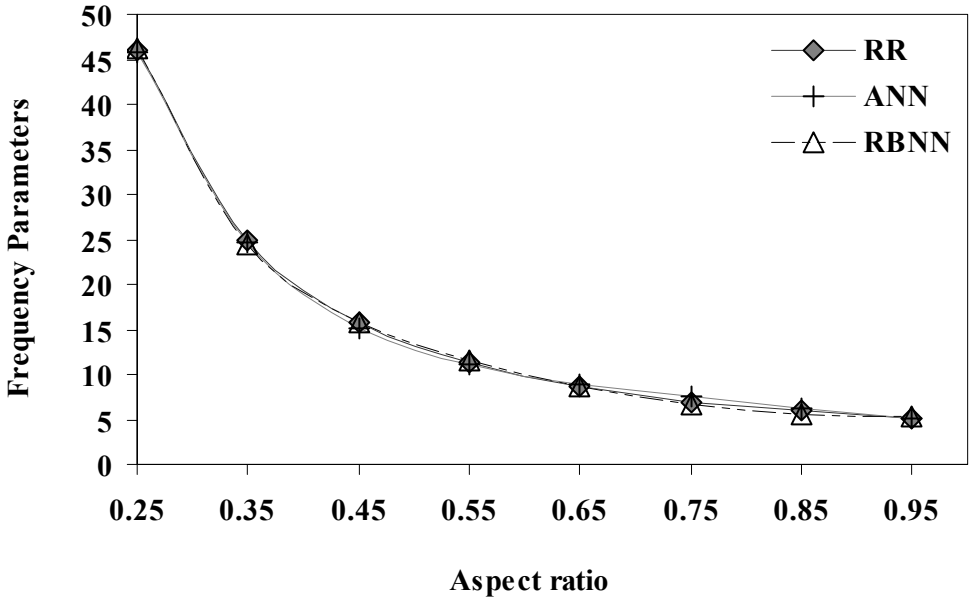


Figure 3.5(b): Comparison among *RR*, *ANN* and *RBNN* approach for Validation of Simply Supported boundary conditions (NRM3 - Architecture)

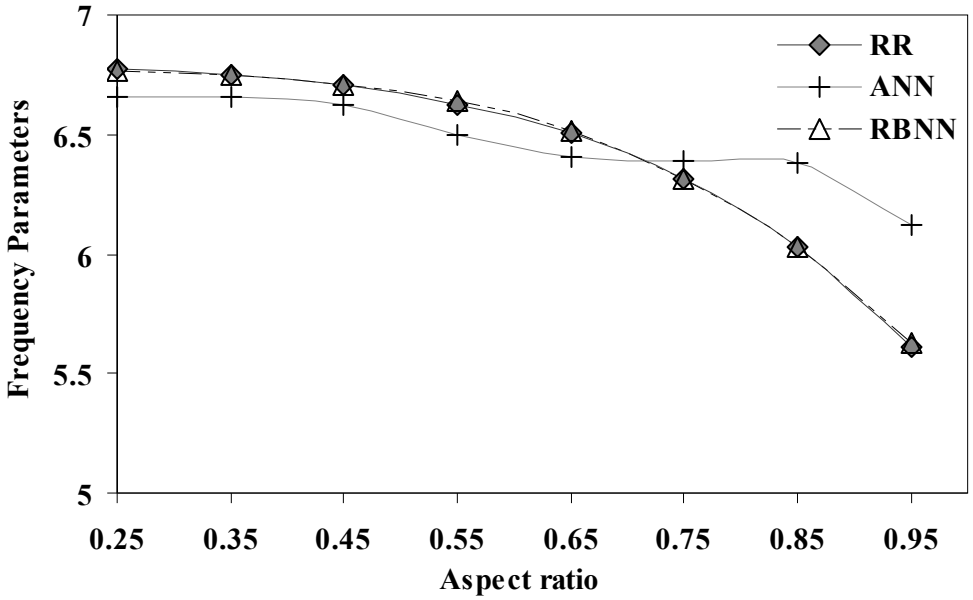


Figure 3.5(c): Comparison among *RR*, *ANN* and *RBNN* approach for Validation of free boundary conditions (NRM3 - Architecture)

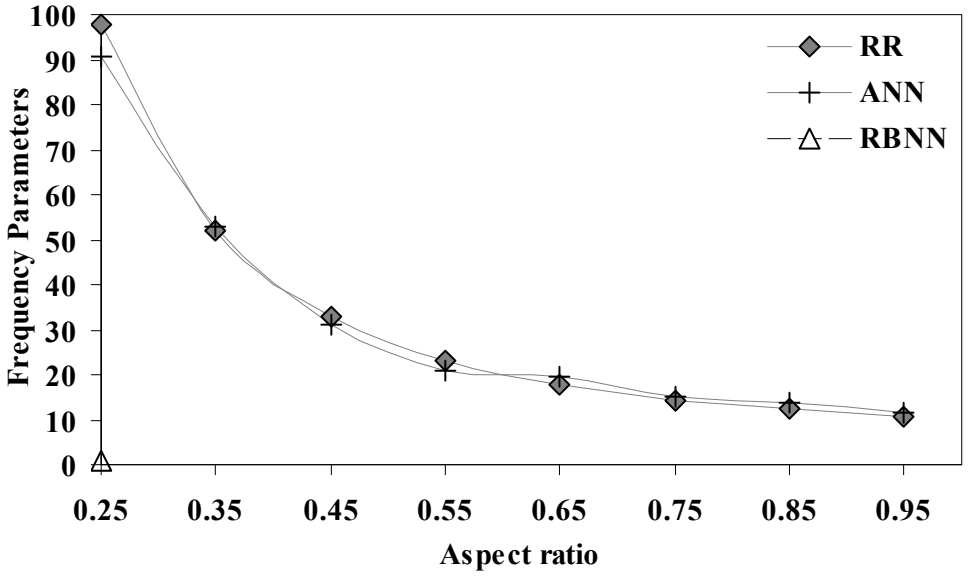


Figure 3.6(a): Comparison among *RR*, *ANN* and *RBNN* approach for Validation of Clamped boundary conditions (NRM4 - Architecture)

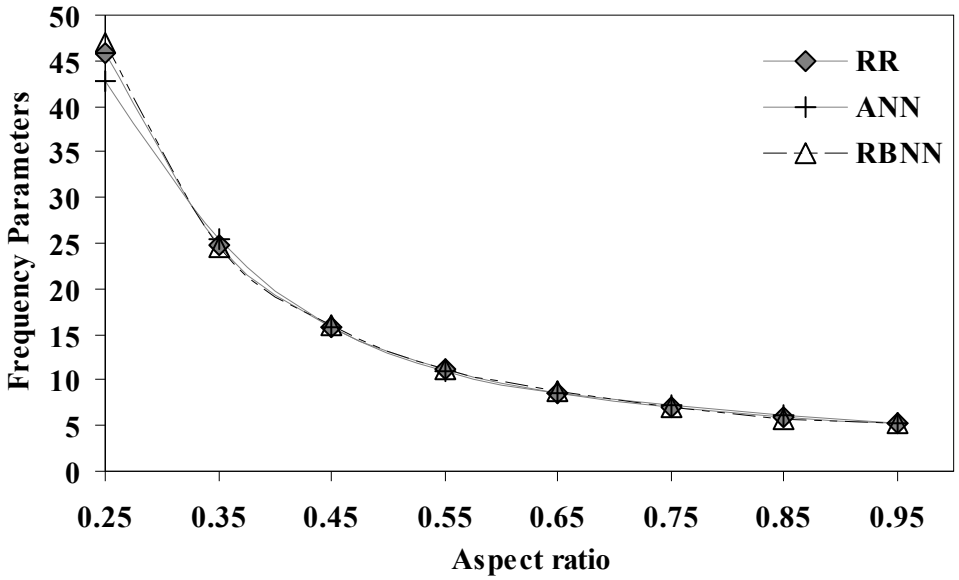


Figure 3.6(b): Comparison among *RR*, *ANN* and *RBNN* approach for Validation of simply-supported boundary conditions (NRM4 - Architecture)

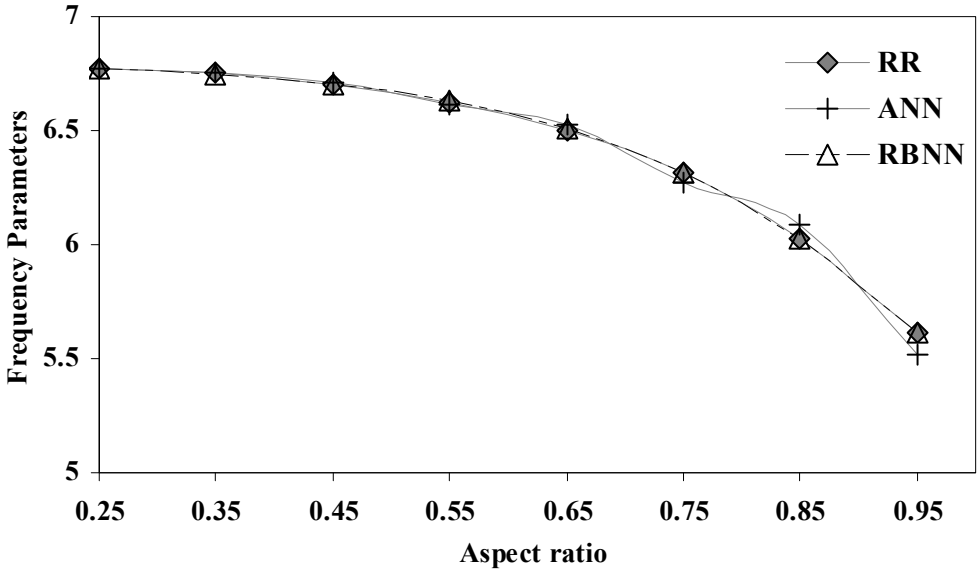


Figure 3.6(c): Comparison among *RR*, *ANN* and *RBNN* approach for Validation of Free boundary conditions (NRM4 - Architecture)

3.6 Error Analysis

As mentioned in Chapter II, in order to use *ANN*s as a reliable tool for vibration analysis of plate shaped structures, one has to take into account the factors that influence its predictions. In this section, we present the *ln*(*MSE*) as function of iteration numbers for

the various models in order to include the trends of the errors for these models. It is, however, very difficult to account for each and every factor that influences the prediction capability of a network. One of the popular measurements for the prediction capability of a network is the mean square error. The graph for *ln*(*MSE*) as a function of iteration numbers is shown in Figure 3.7(a) for regression based NRM3 architecture for the clamped boundary condition. This Figure also gives the comparison of *MSEs* for traditional *ANN* model and *RBNN* model. The comparison of *MSEs* for simply supported and free boundary conditions data for the regression based NRM3 architecture is shown in Figures 3.7(b) to 3.7(c). The Figures 3.8(a) to 3.8(c) contain the similar graphs for the regression based NRM4 architecture with all these boundary conditions. It has been observed that *RBNN* models takes less time in training and have less *MSE* when compared with the conventional *ANN* models.

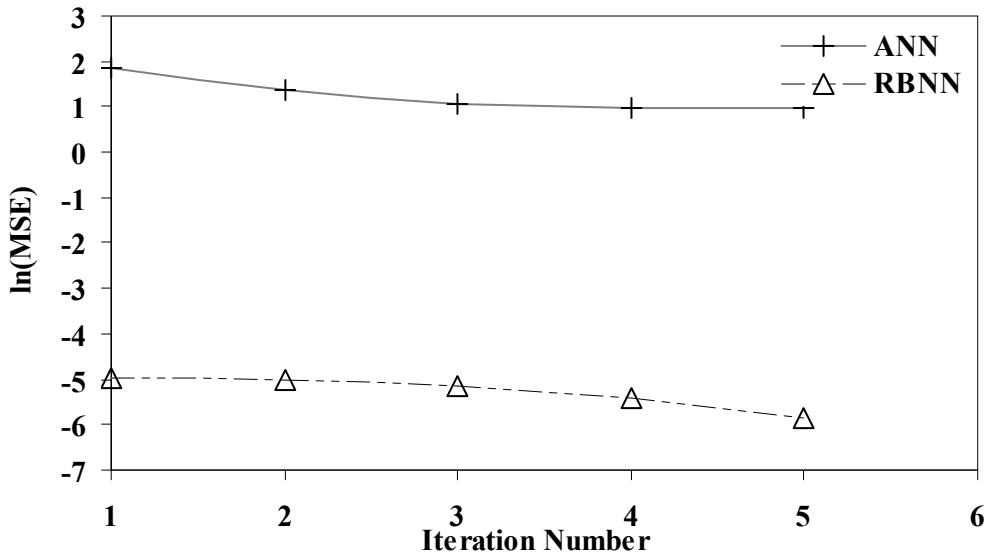


Figure 3.7(a): Error Analysis for Clamped boundary (NRM3 - Architecture)

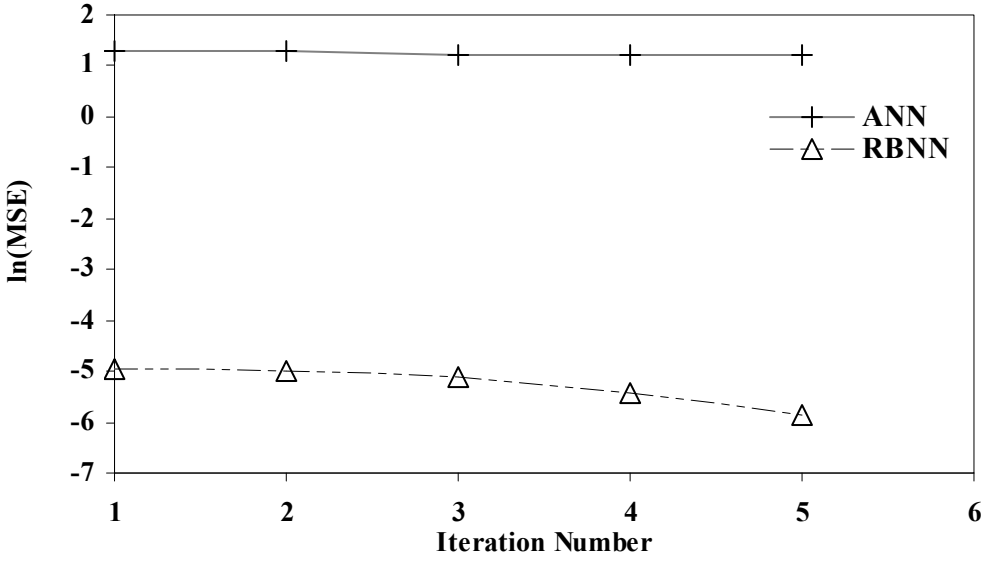
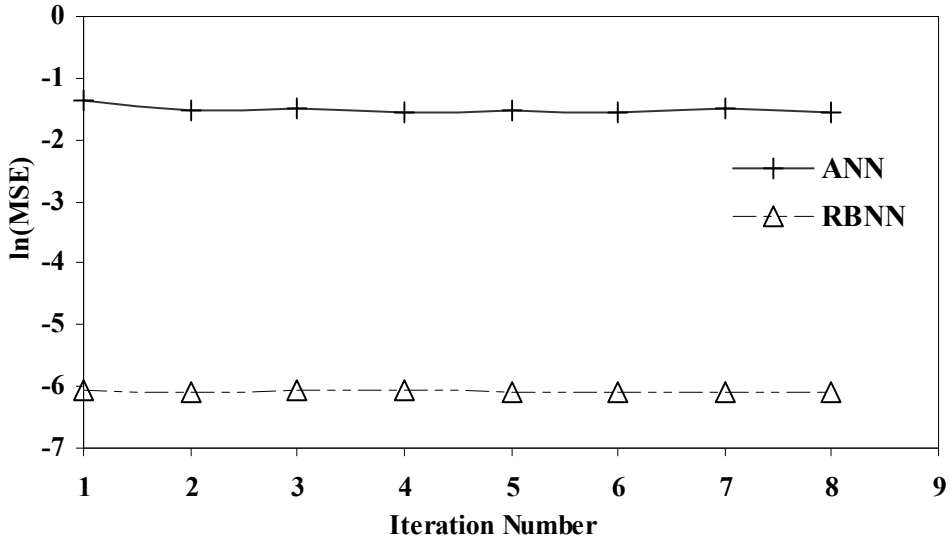


Figure 3.7(b): Error Analysis for Simply Supported boundary (NRM3 - Architecture)



.

7(c): Error Analysis for Free boundary (NRM3 - Architecture)

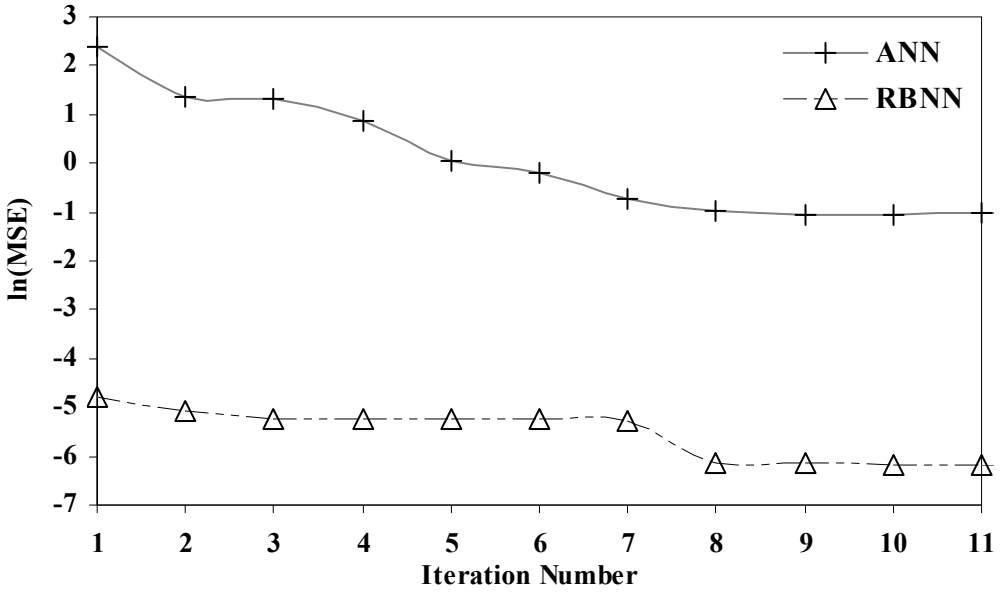


Figure 3.8(a): Error Analysis for Clamped boundary (NRM4 - Architecture)

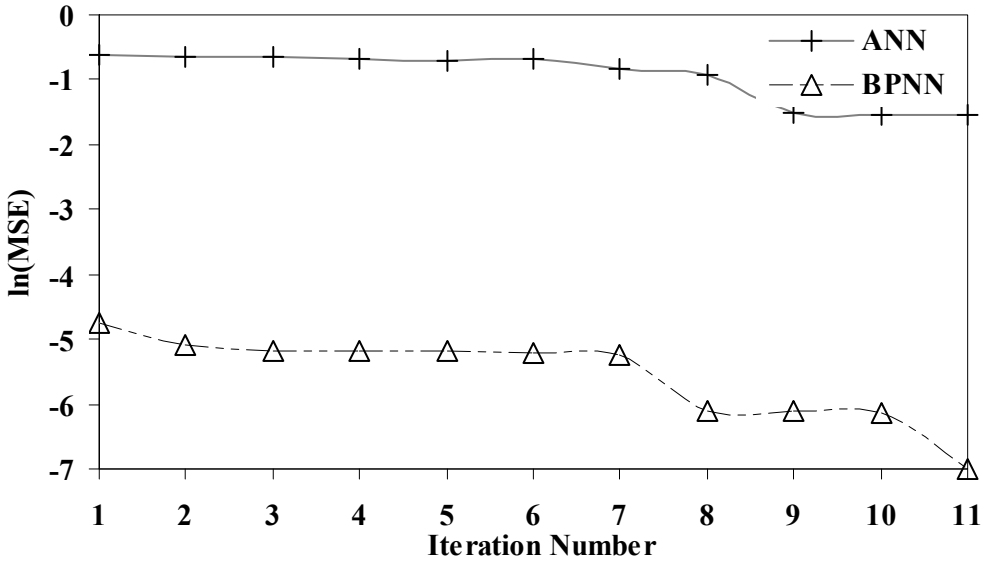


Figure 3.8(b): Error Analysis for Simply Supported boundary (NRM4 - Architecture)

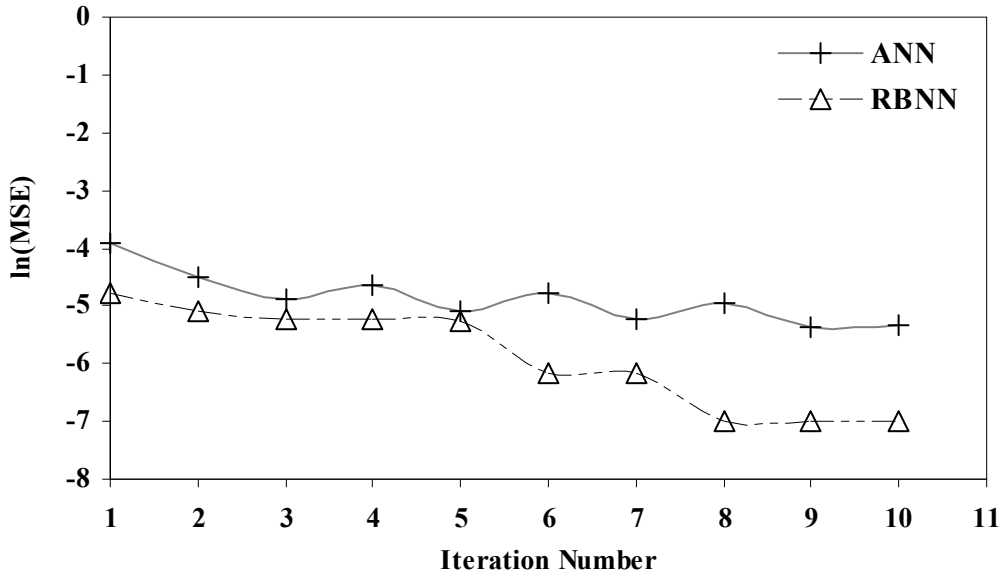


Figure 3.8(c): Error Analysis for Free boundary (NRM4 - Architecture)

3.7 Conclusion

Herein, a regression based neural network model has been proposed and it has been simulated for the problem of plate vibrations. The testing of this model with new data shows the efficacy and reliability of this new model. In this model, the number of neurons in the hidden layer can be fixed depending upon the degree of regression polynomial fitted to the training data. Comparison of the present model with the traditional *ANN* model establishes an edge to the proposed regression based model for the example problem that has been studied. It has been noted that time taken for training the neural network with the present model is also less. Moreover, the present *RBNN* model when trained can give the frequency parameters directly for a particular aspect ratio and boundary condition without going into the solution of fourth order differential equation associated with plate structure.

CHAPTER IV

REGRESSION BASED MULTI INPUT SINGLE OUTPUT (MISO) ANN FOR TRANSVERSE VIBRATION OF ANNULAR CIRCULAR AND ELLIPTIC PLATES

(The contents of this Chapter have been communicated for publication)

CHAPTER IV

REGRESSION BASED MULTI INPUT SINGLE OUTPUT (MISO) ANN MODELS FOR TRANSVERSE VIBRATION OF ANNULAR CIRCULAR AND ELLIPTIC PLATES

4.1 Introduction

Transverse vibrations of annular circular plate have extensively been studied in the literature. These studies generally rely on analytical solutions based on Bessel functions. The review work by *Leissa(1969)* is an excellent piece of work for information in this context. *Joga Rao and Vijay kumar (1963)* and *Gontkevich(1964)* are the important references cited in this review paper. These references cover various types of boundary conditions on the inner and outer boundaries of the plate. The annular plates give rise to nine case of boundary conditions for which the frequency determinant is available in closed form. *Kim and Dickinson (1989)* have studied the transverse vibration of thin annular circular plates involving certain complicating effects. Some of the other interesting references are *Gupta et al. (1986,1987)* and *Nagaraja (1985).* It is worth to mention here that elliptic annular plates have not been studied much. *Sato (1974)* used Mathieu functions in order to give the exact solutions for the vibrations of annular elliptic plates. He has considered the boundaries of confocal ellipses to be either both free or inner one clamped and outer one free. *Ozkul (1975)* studied this problem by taking both boundaries of confocal ellipses as clamped. The literature for the study of annular elliptic plate is not plenty because unlike Bessel functions, the Mathieu functions that appear in geometry of elliptic plates are difficult to handle analytically as well as numerically. It is worth noting that the use of characteristic orthogonal polynomials to this problem reduces these complexities to some extent. *Bhat (1985)* proposed the use of these characteristic orthogonal polynomials in one variable when the plate deflection could be described by separable functions as in case of rectangular and circular plates. This method was successfully used by *Kim and Dickinson (1989)* and *Bhat et al. (1990).* When the plate deflection could not be expressed by separable functions as in triangular, trapezoidal or polygonal plates, *Bhat (1987)* proposed the use of characteristic orthogonal polynomials

in two variables. This method was employed by *Laura et al. (1989) and Liew et al. (1990)*. Encouraged by the success of these methods, *Chakraverty (1992, 1998)* used them for elliptic plates with clamped, simply supported and free boundaries. The integrals involved in generating the final equation has been evaluated exactly by using the formulae given in *Singh and Tyagi (1985)* and *Singh and Goel (1985)*.

The study mainly focuses here on the development of traditional *ANN* model and *RBNN* model (as proposed in Chapter III) to find the frequency of annular elliptic and circular plates. In all, nine traditional *ANN* and nine *RBNN* models have been developed in this Chapter corresponding to the nine boundary conditions as given below.

i) Outer clamped and inner free (*CF*)

ii) Outer clamped and inner simply supported (*CS*)

iii) Outer and inner both clamped (*CC*)

iv) Outer simply supported and inner free (*SF*)

v) Outer and inner both simply supported (*SS*)

vi) Outer simply supported and inner clamped (*SC*)

vii) Outer free and inner clamped (*FC*)

viii) Outer free and inner simply supported (*FS*)

ix) Outer and inner both free (*FF*).

In the next section, we present a brief overview of the theoretical concepts governing the annular elliptic plate structures.

4.2 Governing Equations

We assume that the displacement can be represented as,

$$w = W(x, y)\cos\omega t \qquad \ldots (4.1)$$

where ω is the natural frequency and *x, y, t* are space and time coordinates. The Rayleigh quotient is,

$$\omega^2 = \frac{D}{h\rho}\frac{\iint[(\nabla^2 W)^2 + 2(1-\nu)(W_{xy}^2 + W_{xx}W_{yy})]dxdy}{\iint W^2 dxdy} \qquad \ldots(4.2)$$

where $D = Yh^3/(12(1-\nu^2))$, (called as the flexural rigidity) … (4.3)

and Y, h, ν and ρ are the young's modulus, thickness of plate, poisson ratio and the mass per unit volume, respectively. The integrals are to be evaluated over the domain occupied by the plate.

Now the function $W(x, y)$ is approximated by

$$W = \sum_{J=1}^{N} c_j \ \phi_j(x,y) \qquad \ldots (4.4)$$

where $\phi_j(x, y)$ are suitable functions satisfying the essential boundary conditions and c_j's are constants to be determined such that ω^2 is extremum. Substituting (4.4) in (4.2), extremizing ω^2 as a function of $c_j's$ and simplifying we get

$$\sum_{j=1}^{N} (a_{ij} - \lambda^2 b_{ij}) c_j = 0 \quad i = 1, 2, ..., N \qquad \ldots (4.5)$$

where

$$a_{ij} = \iint [\phi_i^{xx} \phi_j^{xx} + \phi_i^{yy} \phi_j^{yy} + \nu(\phi_i^{xx} \phi_j^{yy} + \phi_i^{yy} \phi_j^{xx}) + 2(1-\nu)\phi_i^{xy} \phi_j^{xy}] dxdy \ \ldots (4.6)$$

$$b_{ij} = \iint \phi_i \phi_j dxdy \qquad \ldots (4.7)$$

and $\lambda^2 = a^4 \omega^2 \rho h / D$ …(4.8)

4.3 Generation of Orthogonal Polynomials Over the Annular Region for *ANN* Data Patterns

The outer boundary of the elliptic plate is given by the equation

$$x^2/a^2 + y^2/b^2 = 1 \qquad \ldots (4.9)$$

where a and b are the semi-major and semi-minor axes of the plate, respectively. The family of ellipses concentric to (4.9) is considered by introducing the variables u as below

$$x^2/a^2 + y^2/b^2 = 1-u, \ 0 \le u \le u_0 < 1 \qquad \ldots (4.10)$$

where the inner boundary is given by $u = u_0$. The eccentricity and the relative sizes of the two boundaries are defined by taking parameters k and m as:

$$k = (1-u_0)^{1/2} \text{ and } m = b/a \qquad \ldots (4.11)$$

To generate orthogonal polynomials over the annular region $0 \le u \le u_0$, the linearly independent set of functions is defined by,

$$F_i(x,y) = u^r (u_0 - u)^s \ f_i(x,y), \ \ i = 1,2,3,... \qquad \ldots (4.12)$$

where $f_i(x, y)$ are suitably chosen functions and $r = 0, 1$ or 2 according to the condition that outer boundary is free, simply supported or clamped. Similarly, $s = 0, 1 \text{ or } 2$ depending upon whether the inner boundary is free, simply-supported or clamped. One can note that the functions defined by (4.12) satisfy all the essential boundary conditions. The Gram-Schmidt process is used to generate an orthogonal set from (4.12) and the inner product of two functions f and g and the norm of f over the annular region are respectively given as,

$$\langle f, g \rangle = \iint f(x,y)\, g(x,y)\, dxdy, \quad \| f \| = \langle f, f \rangle^{1/2} \qquad \ldots (4.13)$$

Numerical computations for the following values of the four parameters *r, s, m, k* and ν

(i) $r = 0,1 \text{ or } 2$ (*F*, *S* and *C* outer boundary)

(ii) $s = 0,1 \text{ or } 2$ (*F*, *S* and *C* inner boundary)

(iii) $m = 0.2, 0.4, 0.5, 0.6, 0.8 \text{ and } 1.0$

(iv) $k = 0.2, 0.4, 0.5, 0.6 \text{ and } 0.8$

(v) $\nu = 1/3$

have been carried out by *Chakraverty (1992)* and *Singh and Chakraverty (1993)*and their results for various values of *m* and *k* as input and their corresponding frequencies parameters as output are used as the training patterns to train the *ANN* model and *RBNN* model.

4.4 Training Patterns

The training patterns for training the neural network have been taken from solution of the Rayleigh-Ritz method with Boundary Characteristic Orthogonal Polynomials (*BCOPs*) for various values of $m = b/a$ and k of annular elliptic and circular plates. Solution of the equation (4.2) gives the vibration frequencies and the results have been obtained for aspect ratio *m = 0.2, 0.4, 0.5, 0.6, 0.8, and 1.0* and *k = 0.2, 0.4, 0.5, 0.6 and 0.8*. The training patterns corresponding to nine boundary conditions, namely, outer clamped and inner free (*CF*), outer clamped and inner simply supported (*CS*), outer and inner both clamped (*CC*), outer simply supported and inner free (*SF*), outer and inner both simply supported (*SS*), outer simply supported and inner clamped (*SC*), outer free and inner clamped (*FC*), outer free and inner simply supported (*FS*), outer and inner free both (*FF*) are given in Table 4.1 for ready reference.

Table 4.1: Training patterns for the proposed model for the nine boundary conditions

m	*k*	*CF*	*CS*	*CC*	*SF*	*SS*	*SC*	*FF*	*FS*	*FC*
0.2	0.2	155.9	155.9	288.5	66.43	142.0	177.2	6.558	12.45	20.28
0.2	0.4	198.6	198.6	355.1	52.24	179.5	221.5	6.079	17.86	27.97
0.2	0.5	231.7	231.7	423.7	52.07	217.6	267.8	5.754	23.65	36.33
0.2	0.6	283.2	283.2	542.1	56.13	282.0	349.1	5.39	33.76	51.78
0.2	0.8	576.6	576.6	1362	91.33	711.1	930.8	5.232	85.80	175.3
0.4	0.2	42.19	78.82	95.18	19.18	49.67	63.16	6.496	7.833	11.33
0.4	0.4	54.42	104.2	125.6	17.79	66.05	83.89	5.979	10.35	15.65
0.4	0.5	66.67	130.2	158.7	18.62	83.11	106.7	5.654	12.21	20.56
0.4	0.6	84.76	177.1	219.9	20.75	113.6	149.0	5.297	15.01	29.84
0.4	0.8	196.6	531.3	707.1	34.58	346.7	490.0	4.511	29.66	106.9
0.6	0.2	20.92	46.87	59.33	9.703	30.28	40.78	6.305	5.996	8.709
0.6	0.4	26.85	66.67	84.87	9.312	43.09	58.18	5.754	6.647	12.34
0.6	0.5	34.21	87.88	113.4	9.824	56.78	77.70	5.421	7.364	16.62
0.6	0.6	46.36	126.7	167.0	10.99	81.90	114.5	5.057	8.701	24.89
0.6	0.8	128.5	439.6	608.8	18.27	284.2	419.9	4.27	16.36	94.47
0.8	0.2	13.69	33.95	45.17	6.383	27.74	30.83	5.882	4.577	7.066
0.8	0.4	17.52	53.29	71.10	6.177	34.08	48.72	5.308	4.692	10.75
0.8	0.5	22.72	73.19	98.65	6.527	46.86	67.46	4.957	5.197	14.85
0.8	0.6	32.43	109.7	150.1	7.313	70.39	102.7	4.579	6.114	22.73
0.8	0.8	105.8	409.3	578.1	12.19	262.6	397.1	3.810	11.23	89.54
1	0.2	10.56	26.78	35.87	4.930	17.03	24.18	5.065	3.592	5.593
1	0.4	13.51	44.93	61.91	4.777	28.08	41.37	4.541	3.654	9.118
1	0.5	17.60	63.85	89.25	5.051	40.01	59.91	4.214	4.074	13.10
1	0.6	25.24	98.79	139.6	5.664	62.12	94.25	3.871	4.809	20.60
1	0.8	92.81	389.5	559.1	9.455	247.0	381.6	3.197	8.782	84.67

4.5 Identification of the Model

A three-layer architecture for *RBNN* approach is considered to model the annular elliptic and annular circular plates vibration problem. Figures 4.1 and 4.2 contain the neural network models used in the process. The input layer consists of two inputs, m and k associated with the annular elliptic plate. The output layer of the *ANN* architecture consists of one output in the form of the corresponding frequency parameters (λ) obtained from the *RR* method using the *BCOPs* (*Chakraverty (1992)*). Here, again two cases of the number of nodes depending upon the proposed parameter of the methodology have been considered in the hidden layer to facilitate a comparative study on the architecture of the network. The output of the network is computed by regression analysis combined with neural activation function performed at two stages, *i.e.*, the stage of hidden layer and the stage of output layer.

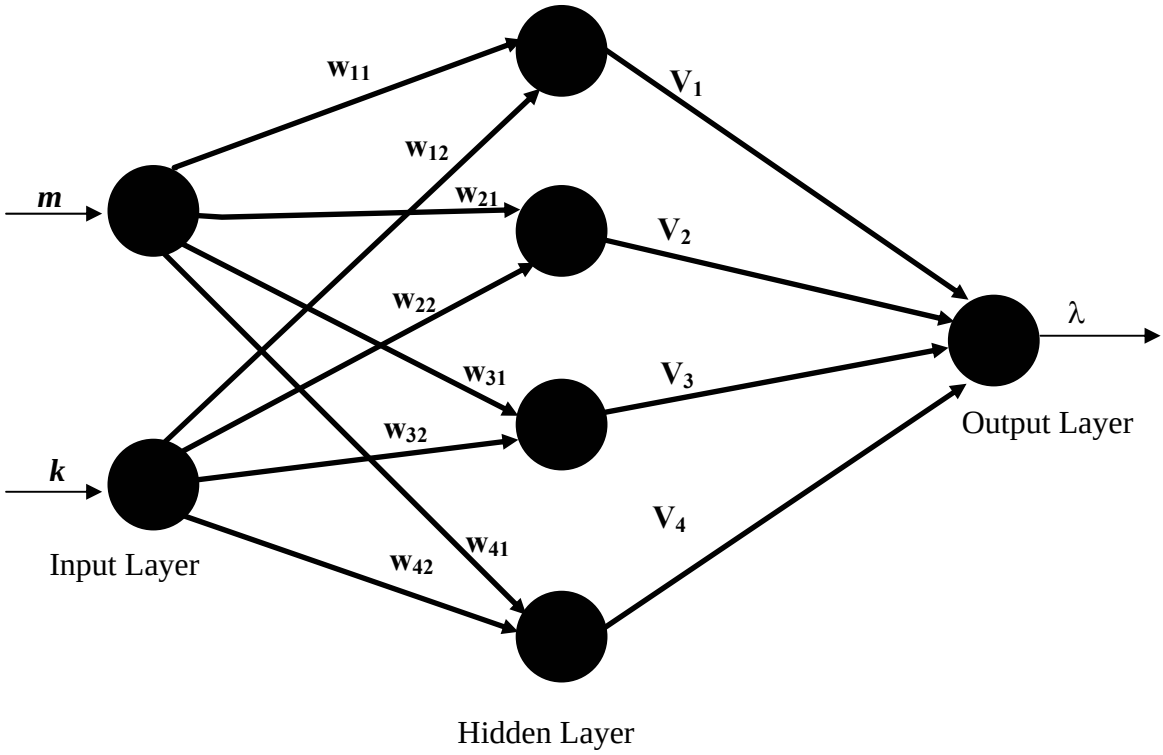


Figure 4.1: *ANN* architecture used for estimation of frequency of the vibration of annular elliptic and annular circular plates

4.6 Training Algorithm

In this section, we illustrate the training algorithm that has been used to train the *RBNN* model.

4.6.1 The *RBNN* Training Algorithm

Let us take N training patterns as $\{(x_1, y_1, O_1); (x_2, y_2, O_2); ...; (x_N, y_N, O_N)\}$. These patterns $\{(x_i, y_i, O_i), i = 1,2,...,N\}$ may, in general, consist of the vector quantities. However, in the present Chapter, we have considered a two input single output problem. We denote the input and output of the given patterns by ***x*, y** and ***O***, respectively. The neural network as given in Figure 4.1 is considered here for the development of the training algorithm. We define the weight matrix ***W*** for the input layer as,

$$\boldsymbol{W} = \begin{bmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \\ w_{3,1} & w_{3,2} \\ w_{4,1} & w_{4,2} \end{bmatrix} = \begin{bmatrix} \frac{a_0}{2} & \frac{a_0}{2} \\ a_1 & a_2 \\ \frac{a_3}{2} & \frac{a_3}{2} \\ a_4 & a_5 \end{bmatrix}$$

where a_0, a_1, a_2, a_3, a_4 and a_5 are the coefficients of the regression polynomial,

$$p(x, y) = a_0 + a_1 x + a_2 y + a_3 xy + a_4 x^2 + a_5 y^2 \quad \ldots (4.14)$$

fitted to the given *N* training patterns.

Now, we calculate the outputs of the neurons of hidden layer using the following activation functions, motivated by the regression polynomial of equation 4.14 by taking $x = m$ and $y = k$ but we will write the further equations in terms of *x* and *y* only in general.

$$p_1^i = \frac{1}{1 + e^{-\alpha\ a_o(x_i + y_i)}}, \quad i = 1, 2, 3, ..., N \quad \ldots (4.15)$$

$$p_2^i = \frac{1}{1 + e^{-\alpha\ (a_1 x_i + a_2 y_i)}}, \quad i = 1, 2, 3, ..., N \quad \ldots (4.16)$$

$$p_3^i = \frac{1}{1 + e^{-\alpha\ a_3(x_i y_i)}}, \quad i = 1, 2, 3, ..., N \quad \ldots (4.17)$$

$$p_4^i = \frac{1}{1 + e^{-\alpha\ (a_4 x_i^2 + a_5 y_i^2)}}, \quad i = 1, 2, 3, ..., N \quad \ldots (4.18)$$

where α is the training parameter that is to be chosen suitably.

The Weight matrix ***V*** for the hidden layer is defined as,

$$\boldsymbol{V} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix}$$

where c_1, c_2, c_3 and c_4 are the coefficients of the multivariate linear regression polynomial,

$$c_1 p_1^i + c_2 p_2^i + c_3 p_3^i + c_4 p_4^i \quad \ldots (4.19)$$

fitted to the data set $\{(p_1^i, p_2^i, p_3^i, p_4^i, O_i): \quad i = 1, 2, ..., N\}$

The output of the network is now calculated as,

$$o = c_1 p_1^i + c_2 p_2^i + c_3 p_3^i + c_4 p_4^i \quad \ldots (4.20)$$

The error associated with this *RBNN* model is defined as,

$$E = \frac{1}{2}\sum_{i=1}^{N}(o_i - O_i)^2 \quad \ldots (4.21)$$

This error, as is the case with other *ANN* models is used to decide whether the *ANN* model has learnt from the given training patterns or not. As such, if *E* is less than some tolerance level we say that *ANN* model has got trained otherwise we update the elements of $\boldsymbol{W}$ and $\boldsymbol{V}$ according to the following process.

Using regression polynomials given in 4.14 and 4.19, equation 4.21 can be written as,

$$E = \frac{1}{2}\sum_{i=1}^{N}(c_1 p_1^i + c_2 p_2^i + c_3 p_3^i + c_4 p_4^i - O_i)^2$$

or

$$E = \frac{1}{2}\sum_{i=1}^{N}\left[\frac{c_1}{1+e^{-\alpha a_o(x_i+y_i)}} + \frac{c_2}{1+e^{-\alpha(a_1 x_i + a_2 y_i)}} + \frac{c_3}{1+e^{-\alpha a_3(x_i y_i)}} + \frac{c_4}{1+e^{-\alpha(a_4 x_i^2 + a_5 y_i^2)}} - O_i\right]^2$$

Thus,

$$\frac{\partial E}{\partial a_0} = \frac{1}{2}\sum_{i=1}^{N} 2(o_i - O_i)(-c_1 \alpha(x_i + y_i)(1+e^{-\alpha\, a_o(x_i+y_i)})^{-2} e^{-\alpha\, a_o(x_i+y_i)}$$

or,

$$\frac{\partial E}{\partial a_0} = \sum_{i=1}^{N}\frac{(o_i - O_i)\alpha c_1(x_i+y_i)e^{-\alpha\, a_o(x_i+y_i)}}{(1+e^{-\alpha\, a_o(x_i+y_i)})^2}$$

Similarly,

$$\frac{\partial E}{\partial a_1} = \sum_{i=1}^{N} \frac{(o_i - O_i)\alpha c_2 x_i e^{-\alpha (a_1 x_i + a_2 y_i)}}{(1+e^{-\alpha a_o (x_i + y_i)})^2}$$

$$\frac{\partial E}{\partial a_2} = \sum_{i=1}^{N} \frac{(o_i - O_i)\alpha c_2 y_i e^{-\alpha (a_1 x_i + a_2 y_i)}}{(1+e^{-\alpha a_o (x_i + y_i)})^2}$$

$$\frac{\partial E}{\partial a_3} = \sum_{i=1}^{N} \frac{(o_i - O_i)\alpha c_3 x_i y_i e^{-\alpha a_3 x_i y_i}}{(1+e^{-\alpha a_3 x_i y_i})^2}$$

$$\frac{\partial E}{\partial a_4} = \sum_{i=1}^{N} \frac{(o_i - O_i)\alpha c_4 x_i^2 e^{-\alpha (a_4 x_i^2 + a_5 y_i^2)}}{(1+e^{-\alpha (a_4 x_i^2 + a_5 y_i^2)})^2}$$

$$\frac{\partial E}{\partial a_5} = \sum_{i=1}^{N} \frac{(o_i - O_i)\alpha c_4 y_i^2 - \alpha (a_4 x_i^2 + a_5 y_i^2)}{(1+e^{-\alpha (a_4 x_i^2 + a_5 y_i^2)})^2}$$

The weights matrix $\boldsymbol{W}$ for the next iteration is redefined as,

$$\boldsymbol{W} = \begin{bmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \\ w_{3,1} & w_{3,2} \\ w_{4,1} & w_{4,2} \end{bmatrix} = \begin{bmatrix} \frac{a_0}{2} - \frac{2\partial E}{\partial a_0} & \frac{a_0}{2} - \frac{2\partial E}{\partial a_0} \\ a_1 - \frac{\partial E}{\partial a_1} & a_2 - \frac{\partial E}{\partial a_2} \\ \frac{a_3}{2} - \frac{2\partial E}{\partial a_3} & \frac{a_3}{2} - \frac{2\partial E}{\partial a_3} \\ a_4 - \frac{\partial E}{\partial a_4} & a_5 - \frac{\partial E}{\partial a_5} \end{bmatrix} \quad \ldots (4.22)$$

Using the matrix $\boldsymbol{W}$ as changed above, we can calculate the new outputs of the hidden neurons using equations 4.15 to 4.18.

Now, the weight matrix $\boldsymbol{V}$ is calculated by fitting the multivariate linear regression polynomial as given in equation 4.19 using the new outputs of the hidden layer. Now, using this new $\boldsymbol{V}$, the output of the network can again be calculated using equation 4.20.

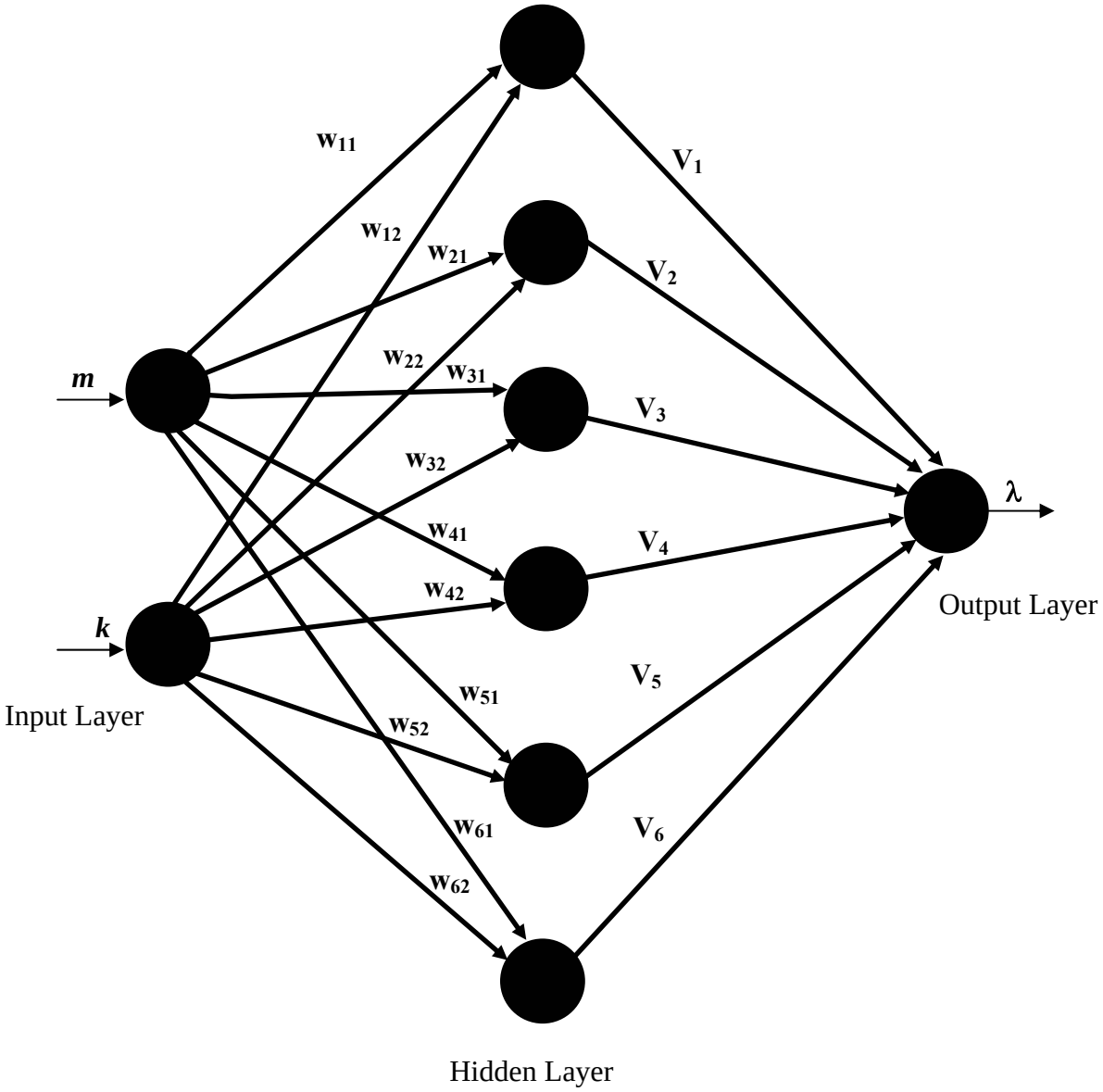


Figure 4.2: *ANN* architecture used for estimation of frequency of the vibration of annular elliptic and annular circular plates

This training procedure is repeated until we get the desired tolerance level of error *E* and thus claim that the *RBNN* model has learnt from the given training patterns.

The training algorithm for the case when we have six neurons in the hidden layer (Figure 4.2) can be explained on similar lines as outlined below. For this case, we define the weight matrix ***W*** for the input layer as,

$$\boldsymbol{W} = \begin{bmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \\ w_{3,1} & w_{3,2} \\ w_{4,1} & w_{4,2} \\ w_{5,1} & w_{5,2} \\ w_{6,1} & w_{6,2} \end{bmatrix} = \begin{bmatrix} \frac{a_0}{2} & \frac{a_0}{2} \\ a_1 & a_2 \\ \frac{a_3}{2} & \frac{a_3}{2} \\ a_4 & a_5 \\ a_6 & a_7 \\ a_8 & a_9 \end{bmatrix}$$

where a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8 and a_9 are the coefficients of the regression polynomial,

$$p(x,y) = a_0 + a_1x + a_2y + a_3xy + a_4x^2 + a_5y^2 + a_6x^3 + a_7y^3 + a_8x^2y + a_9xy^2 \quad \ldots (4.23)$$

fitted to the given *N* training patterns.

Now, we calculate the outputs of the neurons of hidden layer using the following activation functions, motivated by the regression polynomial of equation 4.23 by taking $x = m$ and $y = k$ but we will write the further equations in terms of *x* and *y* only in general.

$$p_1^i = \frac{1}{1+e^{-\alpha\ a_o(x_i + y_i)}}, \quad i = 1, 2, 3, ..., N \quad \ldots (4.24)$$

$$p_2^i = \frac{1}{1+e^{-\alpha\ (a_1x_i + a_2y_i)}}, \quad i = 1, 2, 3, ..., N \quad \ldots (4.25)$$

$$p_3^i = \frac{1}{1+e^{-\alpha\, a_3(x_i y_i)}}, \quad i = 1, 2, 3, ..., N \quad \ldots (4.26)$$

$$p_4^i = \frac{1}{1+e^{-\alpha\ (a_4x_i^2 + a_5y_i^2)}}, \quad i = 1, 2, 3, ..., N \quad ... (4.27)$$

$$p_5^i = \frac{1}{1+e^{-\alpha\ (a_6x_i^3 + a_7y_i^3)}}, \quad i = 1, 2, 3, ..., N \quad ... (4.28)$$

$$p_6^i = \frac{1}{1+e^{-\alpha\ (a_8x_i^2y_i + a_9x_iy_i^2)}}, \quad i = 1, 2, 3, ..., N \quad ... (4.29)$$

where α is again the training parameter to be chosen suitably. Now, the Weight matrix $\boldsymbol{V}$ for the hidden layer is defined as

$$\boldsymbol{V} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \end{bmatrix}$$

where c_1, c_2, c_3, c_4, c_5 and c_6 are the coefficients of the multivariate linear regression polynomial,

$$c_1 p_1^i + c_2 p_2^i + c_3 p_3^i + c_4 p_4^i + c_5 p_5^i + c_6 p_6^i \quad \ldots (4.30)$$

fitted to the data set $\{(p_1^i, p_2^i, p_3^i, p_4^i, p_5^i, p_6^i, \quad O_i): \quad i = 1, \ 2, \ \ldots, \ N\}$

The output of the network is now calculated as,

$$o = c_1 p_1^i + c_2 p_2^i + c_3 p_3^i + c_4 p_4^i + c_5 p_5^i + c_6 p_6^i \quad \ldots (4.31)$$

The error associated with this *RBNN* model is defined as,

$$E = \frac{1}{2}\sum_{i=1}^{N} (o_i - O_i)^2 \quad \ldots (4.32)$$

This error E is again used to train the network as described below by redefining the elements of $\boldsymbol{W}$ and $\boldsymbol{V}$ according to the following process.

$$E = \frac{1}{2}\sum_{i=1}^{N} (c_1 p_1^i + c_2 p_2^i + c_3 p_3^i + c_4 p_4^i + c_5 p_5^i + c_6 p_6^i - O_i)^2$$

or

$$E = \frac{1}{2}\sum_{i=1}^{N} \left[\frac{c_1}{1+e^{-\alpha a_o(x_i+y_i)}} + \frac{c_2}{1+e^{-\alpha(a_1 x_i + a_2 y_i)}} + \frac{c_3}{1+e^{-\alpha a_3(x_i y_i)}} + \frac{c_4}{1+e^{-\alpha(a_4 x_i^2 + a_5 y_i^2)}} + \frac{c_5}{1+e^{-\alpha(a_6 x_i^3 + a_7 y_i^3)}} + \frac{c_6}{1+e^{-\alpha(a_8 x^2 y + a_9 x y^2)}} - O_i\right]^2$$

Thus,

$$\frac{\partial E}{\partial a_0} = \frac{1}{2}\sum_{i=1}^{N} 2(o_i - O_i)(-c_1 \alpha(x_i + y_i)(1 + e^{-\alpha\, a_o(x_i + y_i)})^{-2} e^{-\alpha\, a_o(x_i + y_i)}$$

or,

$$\frac{\partial E}{\partial a_0} = \sum_{i=1}^{N} \frac{(o_i - O_i)\alpha c_1(x_i + y_i) e^{-\alpha\, a_o(x_i + y_i)}}{(1 + e^{-\alpha\, a_o(x_i + y_i)})^2}$$

Similarly,

$$\frac{\partial E}{\partial a_1} = \sum_{i=1}^{N} \frac{(o_i - O_i)\alpha c_2 x_i e^{-\alpha (a_1 x_i + a_2 y_i)}}{(1+e^{-\alpha a_o (x_i + y_i)})^2}$$

$$\frac{\partial E}{\partial a_2} = \sum_{i=1}^{N} \frac{(o_i - O_i)\alpha c_2 y_i e^{-\alpha (a_1 x_i + a_2 y_i)}}{(1+e^{-\alpha a_o (x_i + y_i)})^2}$$

$$\frac{\partial E}{\partial a_3} = \sum_{i=1}^{N} \frac{(o_i - O_i)\alpha c_3 x_i y_i e^{-\alpha a_3 x_i y_i}}{(1+e^{-\alpha a_3 x_i y_i})^2}$$

$$\frac{\partial E}{\partial a_4} = \sum_{i=1}^{N} \frac{(o_i - O_i)\alpha c_4 x_i^2 e^{-\alpha (a_4 x_i^2 + a_5 y_i^2)}}{(1+e^{-\alpha (a_4 x_i^2 + a_5 y_i^2)})^2}$$

$$\frac{\partial E}{\partial a_5} = \sum_{i=1}^{N} \frac{(o_i - O_i)\alpha c_4 y_i^2 - \alpha (a_4 x_i^2 + a_5 y_i^2)}{(1+e^{-\alpha (a_4 x_i^2 + a_5 y_i^2)})^2}$$

$$\frac{\partial E}{\partial a_6} = \sum_{i=1}^{N} \frac{(o_i - O_i)\alpha c_5 x_i^3 - \alpha (a_6 x_i^3 + a_7 y_i^3)}{(1+e^{-\alpha (a_6 x_i^3 + a_7 y_i^3)})^2}$$

$$\frac{\partial E}{\partial a_7} = \sum_{i=1}^{N} \frac{(o_i - O_i)\alpha c_5 y_i^3 - \alpha (a_6 x_i^3 + a_7 y_i^3)}{(1+e^{-\alpha (a_6 x_i^3 + a_7 y_i^3)})^2}$$

$$\frac{\partial E}{\partial a_8} = \sum_{i=1}^{N} \frac{(o_i - O_i)\alpha c_6 x_i^2 y_i - \alpha (a_8 x_i^2 y_i + a_9 x_i y_i^2)}{(1+e^{-\alpha (a_8 x_i^2 y_i + a_9 x_i y_i^2)})^2}$$

and

$$\frac{\partial E}{\partial a_9} = \sum_{i=1}^{N} \frac{(o_i - O_i)\alpha c_6 x_i y_i^2 - \alpha (a_8 x_i^2 y_i + a_9 x_i y_i^2)}{(1+e^{-\alpha (a_8 x_i^2 y_i + a_9 x_i y_i^2)})^2}$$

Using these partial derivates, the weights matrix $\boldsymbol{W}$ is redefined as,

$$\boldsymbol{W} = \begin{bmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \\ w_{3,1} & w_{3,2} \\ w_{4,1} & w_{4,2} \\ w_{5,1} & w_{5,2} \\ w_{6,1} & w_{6,2} \end{bmatrix} = \begin{bmatrix} \frac{a_0}{2} - \frac{2\partial E}{\partial a_0} & \frac{a_0}{2} - \frac{2\partial E}{\partial a_0} \\ a_1 - \frac{\partial E}{\partial a_1} & a_2 - \frac{\partial E}{\partial a_2} \\ \frac{a_3}{2} - \frac{2\partial E}{\partial a_3} & \frac{a_3}{2} - \frac{2\partial E}{\partial a_3} \\ a_4 - \frac{\partial E}{\partial a_4} & a_5 - \frac{\partial E}{\partial a_5} \\ a_6 - \frac{\partial E}{\partial a_6} & a_7 - \frac{\partial E}{\partial a_7} \\ a_8 - \frac{\partial E}{\partial a_8} & a_9 - \frac{\partial E}{\partial a_9} \end{bmatrix} \quad \ldots (4.33)$$

Using the matrix $\boldsymbol{W}$ as changed above, we can calculate the new outputs of the hidden neurons using equations 4.24 to 4.29.

Now, the weight matrix $\boldsymbol{V}$ is calculated by again fitting the multivariate linear regression polynomial as given in equation 4.30 using the new outputs of the hidden layer. Utilizing present new $\boldsymbol{V}$, the output of the network can again be calculated with the help of equation 4.31. This training procedure is repeated until we get the desired tolerance level of error *E* and thus the *RBNN* model is said to be learnt from the given training patterns.

4.6.2 The Experiment

The procedure described in subsection 4.6.1 for training the proposed *RBNN* model has been implemented in the MatLab environment. In this phase, complete regression polynomials of degree two and three are fitted to the training patterns given in Table 4.1. The coefficients of these polynomials are taken as the connecting weights for the hidden layer in the form of $\boldsymbol{W}$. Using these weights and equations 4.15 to 4.18 (for 4 hidden neurons) and equations 4.24 to 4.29 (for 6 hidden neurons) the outputs of the neurons in the hidden layer are calculated. These values are used to fit a regression polynomial of the form equations 4.19 and 4.29 and we can calculate the output of the network using equations 4.20 and 4.31, for four and six neurons in the hidden layer, respectively. At this stage, the error of the *RBNN* model is calculated and a decision is taken whether the network has been trained or not. If the tolerance level of the error is not achieved, we redefine the connection weights $\boldsymbol{W}$ according to equations 4.22 and 4.33 and repeat the procedure otherwise we say that the network has got trained. As such, we have used the regression polynomials of degree 2 and degree 3 to implement the proposed *RBNN* model.

In order to report the benefits of using the *RBNN* modeling approach as explained above, over the traditional *ANN* modeling, we have performed this experiment again using traditional *ANN* models. The architecture of these models has been taken similar to the *RBNN* model. This experiment has again been performed in the MatLab environment using built-in functions of the *NNT*. The network is trained with the help of values of input(s) and output(s) as training patterns that are given in Table 4.1. The training procedure in this experiment is based on the feedforward backpropagation algorithm. The built-in transfer functions, namely, tan-sigmoid (*tansig*) and linear (*purelin*) of the *NNT* have been used in the input layer and the output layer, respectively, in this experiment. The neural network based on the feedforward backpropagation algorithm has been trained

with conjugate gradient, quasi-Newton and Levenberg-Marquardt training algorithms. The connection weights interconnecting the neurons between different layers are taken through a random number generator built-in function in the *NNT*. The built-in transfer functions *tansig* and *purelin* along with the different training functions have been adjusted between input and output layers to get the network converged at the level of desired accuracy as set in experiment for *RBNN* model. The learning rate parameter has been varied from 0.01 to 0.05 with a step of 0.01 in the training of the network for all above said transfer functions. The network training process with conjugate gradient Fletcher-Reeves algorithm (*traincgf*), Polak-Ribiere conjugate gradient algorithm (*traincgp*), Powell-Beale conjugate gradient algorithm (*traincgb*), scaled conjugate gradient (*trainscg*) and one-step secant (*trainoss*) algorithms could not converge and the performance goal could not meet for the same tolerance level as set in experiment for *RBNN* model.

The network could get trained in this experiment with the Quasi-Newton and Levenberg Marquardt training algorithm (*trainlm*) by meeting the performance goal, but the mean square error of the output values in this case are higher than the output values obtained in the experiment for *RBNN* model. The training results of *RBNN* model and *ANN* model experiments for nine different boundary conditions are given in Table 4.2 to 4.10. In these tables, *m* and *k* are the input parameters for the network model; *RR* represents the theoretical frequency corresponding to the values of *m* and *k*; *RBNN4* represents the values of frequency obtained by *RBNN* model with four neurons in the hidden layer; *RBNN5* corresponds to the values of frequency obtained by *RBNN* model with five neurons in the hidden layer and *ANN* represents the frequency calculated by traditional *ANN* model, as described above in this experiment. The pattern characteristics for these experiments are also incorporated in Figure 4.3(a) for both inner and outer clamped boundary conditions, Figure 4.3(b) for outer simply supported boundary conditions and Figures 4.3(c) for outer free boundary conditions. The Mean Square Error (*MSE*) for the *RBNN* and *ANN* models has been tabulated in table 4.11. It has been noted from this that the *MSE* for the *RBNN* model has the higher accuracy then the *ANN* model.

Table 4.2: Training results of *RBNN* and *ANN* models for *CF* conditions of the annular plate

m	*k*	*RR*	*RBNN*	*ANN*
0.2	0.2	155.9	155.9	156.9
0.2	0.4	198.6	198.8	197.5
0.2	0.5	231.7	231.4	230.7
0.2	0.6	283.2	283.3	283.9
0.2	0.8	576.6	576.6	577.6
0.4	0.2	42.19	41.09	42.72
0.4	0.4	54.42	55.11	55.91
0.4	0.5	66.67	66.34	65.25
0.4	0.6	84.76	84.92	85.95
0.4	0.8	196.6	196.6	197.8
0.6	0.2	20.92	20.58	21.80
0.6	0.4	26.85	27.12	27.48
0.6	0.5	34.21	33.80	33.10
0.6	0.6	46.36	46.21	46.87
0.6	0.8	128.5	128.6	129.0
0.8	0.2	13.69	14.02	13.29
0.8	0.4	17.52	18.04	18.49
0.8	0.5	22.72	22.80	22.28
0.8	0.6	32.43	32.37	32.88
0.8	0.8	105.8	105.7	106.8
1	0.2	10.56	10.60	10.12
1	0.4	13.51	13.37	13.79
1	0.5	17.60	17.45	18.09
1	0.6	25.24	25.34	25.98
1	0.8	92.81	92.90	93.87

Table 4.3: Training results of *RBNN* and *ANN* models for CS conditions of the annular plate

m	*K*	*RR*	*RBNN*	*ANN*
0.2	0.2	155.9	155.9	156.8
0.2	0.4	198.6	198.2	197.1
0.2	0.5	231.7	232.3	230.2
0.2	0.6	283.2	282.9	284.1
0.2	0.8	576.6	576.6	577.6
0.4	0.2	78.82	79.01	77.13
0.4	0.4	104.2	104.0	104.9
0.4	0.5	130.2	130.0	131.9
0.4	0.6	177.1	177.4	176.1
0.4	0.8	531.3	531.3	530.7
0.6	0.2	46.87	46.79	47.95
0.6	0.4	66.67	67.3	67.98
0.6	0.5	87.88	87.90	88.48
0.6	0.6	126.7	126.4	125.9
0.6	0.8	439.6	439.6	440.6
0.8	0.2	33.95	33.67	34.94
0.8	0.4	53.29	53.42	53.97
0.8	0.5	73.19	72.9	72.32
0.8	0.6	109.7	109.7	109.7
0.8	0.8	409.3	409.4	408.0
1	0.2	26.78	27.02	26.18
1	0.4	44.93	45.01	44.37
1	0.5	63.85	63.56	64.97
1	0.6	98.79	99.19	99.88
1	0.8	389.5	389.4	390.8

Table 4.4: Training results of *RBNN* and *ANN* models for *CC* conditions of the annular plate

m	*k*	*RR*	*RBNN*	*ANN*
0.2	0.2	288.5	288.2	287
0.2	0.4	355.1	356.8	357.9
0.2	0.5	423.7	421.5	420.4
0.2	0.6	542.1	542.9	543.9
0.2	0.8	1362	1362	1361
0.4	0.2	95.20	94.88	96.89
0.4	0.4	125.6	126.8	126.9
0.4	0.5	158.7	157.8	156.0
0.4	0.6	219.9	220.3	219.9
0.4	0.8	707.1	707.2	707.9
0.6	0.2	59.33	58.60	57.10
0.6	0.4	84.87	85.78	85.98
0.6	0.5	113.4	112.3	114.9
0.6	0.6	167.0	166.9	168.9
0.6	0.8	608.8	608.3	607.1
0.8	0.2	45.17	45.60	45.97
0.8	0.4	71.10	72.26	73.46
0.8	0.5	98.65	97.98	98.88
0.8	0.6	150.1	150.8	150.8
0.8	0.8	578.1	579.0	577.7
1	0.2	35.87	35.78	36.87
1	0.4	61.91	62.50	63.77
1	0.5	89.25	87.98	85.36
1	0.6	139.6	139.9	140.9
1	0.8	559.1	558.7	559.8

Table 4.5: Training results of *RBNN* and *ANN* models for *SF* conditions of the annular plate

m	*k*	*RR*	*RBNN*	*ANN*
0.2	0.2	66.43	66.38	65.37
0.2	0.4	52.24	52.39	53.48
0.2	0.5	52.07	51.70	51.25
0.2	0.6	56.13	56.35	56.39
0.2	0.8	91.33	91.28	92.30
0.4	0.2	19.18	19.22	19.93
0.4	0.4	17.79	17.89	16.57
0.4	0.5	18.62	18.48	17.38
0.4	0.6	20.75	20.73	21.86
0.4	0.8	34.58	34.68	34.96
0.6	0.2	9.703	9.532	9.061
0.6	0.4	9.312	9.317	9.923
0.6	0.5	9.824	9.746	10.807
0.6	0.6	10.99	10.95	11.706
0.6	0.8	18.27	18.12	18.68
0.8	0.2	6.383	6.474	6.006
0.8	0.4	6.177	6.404	6.963
0.8	0.5	6.527	6.679	6.946
0.8	0.6	7.313	7.422	7.567
0.8	0.8	12.19	12.30	12.97
1	0.2	4.930	4.854	5.938
1	0.4	4.777	4.775	5.919
1	0.5	5.051	4.970	4.096
1	0.6	5.664	5.529	6.662
1	0.8	9.455	9.444	10.44

Table 4.6: Training results of *RBNN* and *ANN* models for *SS* conditions of the annular plate

m	*k*	*RR*	*RBNN*	*ANN*
0.2	0.2	142.0	141.9	141.0
0.2	0.4	179.5	179.6	187.8
0.2	0.5	217.6	217.3	212.8
0.2	0.6	282.0	282.1	278.9
0.2	0.8	711.1	711.7	710.9
0.4	0.2	49.67	49.41	50.13
0.4	0.4	66.05	65.92	65.03
0.4	0.5	83.11	83.88	79.18
0.4	0.6	113.6	113.3	118.8
0.4	0.8	346.7	346.7	347.8
0.6	0.2	30.28	31.35	28.80
0.6	0.4	43.09	42.16	37.95
0.6	0.5	56.78	56.33	57.95
0.6	0.6	81.90	81.67	86.76
0.6	0.8	284.2	283.9	281.3
0.8	0.2	27.74	26.58	24.89
0.8	0.4	34.08	35.07	31.61
0.8	0.5	46.86	47.03	49.96
0.8	0.6	70.39	70.91	69.81
0.8	0.8	262.6	263.6	266.5
1	0.2	17.03	17.09	21.35
1	0.4	28.08	28.95	28.08
1	0.5	40.01	39.46	43.38
1	0.6	62.12	62.02	57.59
1	0.8	247.0	246.7	244.7

Table 4.7: Training results of *RBNN* and *ANN* models for *SC* conditions of the annular plate

m	*k*	*RR*	*RBNN*	*ANN*
0.2	0.2	177.2	176.4	175.4
0.2	0.4	221.5	223.5	223.6
0.2	0.5	267.8	265.8	265.0
0.2	0.6	349.1	349.6	350.9
0.2	0.8	930.8	930.7	931.6
0.4	0.2	63.16	63.76	64.96
0.4	0.4	83.89	85.27	84.80
0.4	0.5	106.7	105.2	104.2
0.4	0.6	149.0	149.3	148.4
0.4	0.8	490.0	490.02	490.5
0.6	0.2	40.78	40.41	41.03
0.6	0.4	58.18	58.84	58.87
0.6	0.5	77.70	75.70	77.95
0.6	0.6	114.5	114.0	115.7
0.6	0.8	419.9	419.6	420.4
0.8	0.2	30.83	31.38	30.04
0.8	0.4	48.72	49.56	50.04
0.8	0.5	67.46	66.67	66.08
0.8	0.6	102.7	103.2	101.5
0.8	0.8	397.1	397.79	398.3
1	0.2	24.18	23.78	25.21
1	0.4	41.37	41.02	42.97
1	0.5	59.91	59.19	57.08
1	0.6	94.25	95.67	94.09
1	0.8	381.6	380.9	381.9

Table 4.8: Training results of *RBNN* and *ANN* models for *FF* conditions of the annular plate

m	*k*	*RR*	*RBNN*	*ANN*
0.2	0.2	6.558	6.561	6.760
0.2	0.4	6.079	6.073	6.173
0.2	0.5	5.754	5.754	5.956
0.2	0.6	5.390	5.391	5.490
0.2	0.8	5.232	5.232	5.532
0.4	0.2	6.496	6.493	6.403
0.4	0.4	5.979	5.983	5.782
0.4	0.5	5.654	5.658	5.757
0.4	0.6	5.297	5.293	5.293
0.4	0.8	4.511	4.511	4.511
0.6	0.2	6.305	6.302	6.392
0.6	0.4	5.754	5.756	5.797
0.6	0.5	5.421	5.420	5.490
0.6	0.6	5.057	5.054	5.156
0.6	0.8	4.270	4.271	4.279
0.8	0.2	5.882	5.883	5.894
0.8	0.4	5.308	5.307	5.406
0.8	0.5	4.957	4.955	4.995
0.8	0.6	4.579	4.581	4.679
0.8	0.8	3.810	3.808	3.891
1	0.2	5.065	5.064	5.095
1	0.4	4.541	4.541	4.938
1	0.5	4.214	4.213	4.995
1	0.6	3.871	3.871	3.892
1	0.8	3.197	3.197	3.296

Table 4.9: Training results of *RBNN* and *ANN* models for *FS* conditions of the annular plate

m	*k*	*RR*	*RBNN*	*ANN*
0.2	0.2	12.45	12.45	12.87
0.2	0.4	17.86	17.88	17.95
0.2	0.5	23.65	23.62	23.56
0.2	0.6	33.76	33.77	33.78
0.2	0.8	85.80	85.79	85.79
0.4	0.2	7.833	7.823	8.918
0.4	0.4	10.35	10.40	10.02
0.4	0.5	12.21	12.14	12.09
0.4	0.6	15.01	14.99	14.09
0.4	0.8	29.66	29.67	29.95
0.6	0.2	5.996	6.012	5.554
0.6	0.4	6.647	6.680	6.629
0.6	0.5	7.364	7.385	7.444
0.6	0.6	8.701	8.768	8.748
0.6	0.8	16.36	16.26	16.29
0.8	0.2	4.577	4.472	4.558
0.8	0.4	4.692	4.681	4.942
0.8	0.5	5.197	5.131	5.263
0.8	0.6	6.114	6.090	6.075
0.8	0.8	11.23	11.37	11.27
1	0.2	3.592	3.637	3.554
1	0.4	3.654	3.742	3.699
1	0.5	4.074	4.066	4.061
1	0.6	4.809	4.773	4.820
1	0.8	8.782	8.715	8.770

Table 4.10: Training results of *RBNN* and *ANN* models for *FC* conditions of the annular plate

m	*k*	*RR*	*RBNN*	*ANN*
0.2	0.2	20.28	20.19	21.17
0.2	0.4	27.97	28.30	28.85
0.2	0.5	36.33	35.99	37.29
0.2	0.6	51.78	51.88	51.75
0.2	0.8	175.3	175.2	175.9
0.4	0.2	11.33	11.28	11.85
0.4	0.4	15.65	15.81	15.05
0.4	0.5	20.56	20.28	20.05
0.4	0.6	29.84	29.95	30.80
0.4	0.8	106.9	106.8	107.9
0.6	0.2	8.709	8.698	8.266
0.6	0.4	12.34	12.58	12.73
0.6	0.5	16.62	16.44	16.99
0.6	0.6	24.89	24.98	26.24
0.6	0.8	94.47	94.58	94.95
0.8	0.2	7.066	7.082	6.067
0.8	0.4	10.75	10.89	10.01
0.8	0.5	14.85	14.59	14.39
0.8	0.6	22.73	22.72	22.20
0.8	0.8	89.54	89.33	89.20
1	0.2	5.593	5.508	5.705
1	0.4	9.118	9.312	9.203
1	0.5	13.10	12.91	13.98
1	0.6	20.60	20.71	20.92
1	0.8	84.67	84.77	84.64

Table 4.11: Comparison of *MSE*s for *RBNN* and *ANN* models

Boundary Condition	*MSE for ANN Model*	*MSE for RBNN* Model	*Relative decrease in MSE of RBNN Model over ANN Model*
CF	3.337	0.167	94%
CS	27.060*	0.103	99.61%
CC	3.540	1.309	63.02%
SF	2.396*	0.024	98.99%
SS	31.387*	0.565	98.19%
SC	3.975	0.147	95.37%
FF	0.050	0.00008	99.84%
FS	0.369*	0.0036	99.04%
FF	0.145	0.029	80%

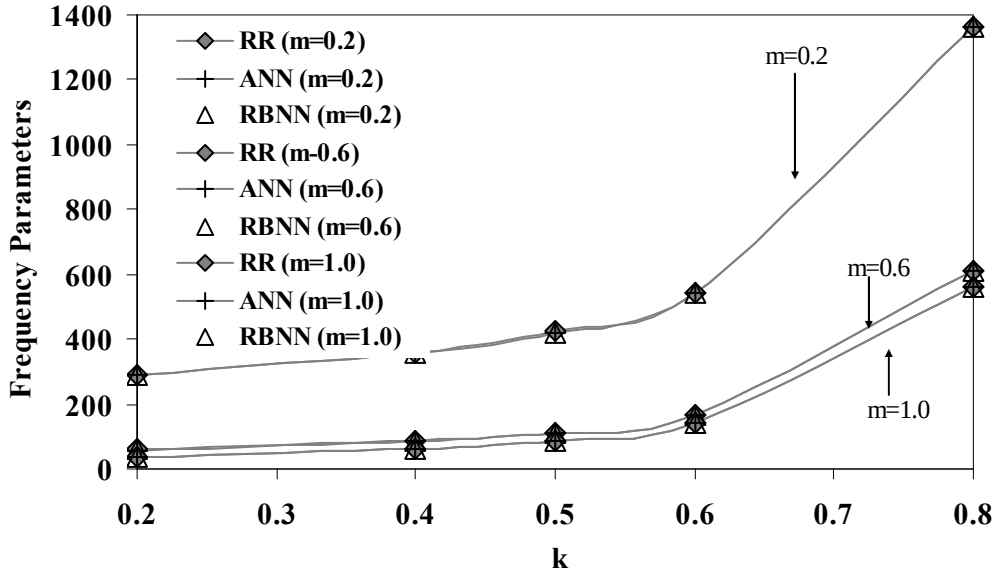


Figure 4.3(a): Results of training of *RBNN* and *ANN* models for Frequency parameters with (both inner and outer Clamped) boundary conditions of the annular plate for m =0.2, 0.6 and 1.0

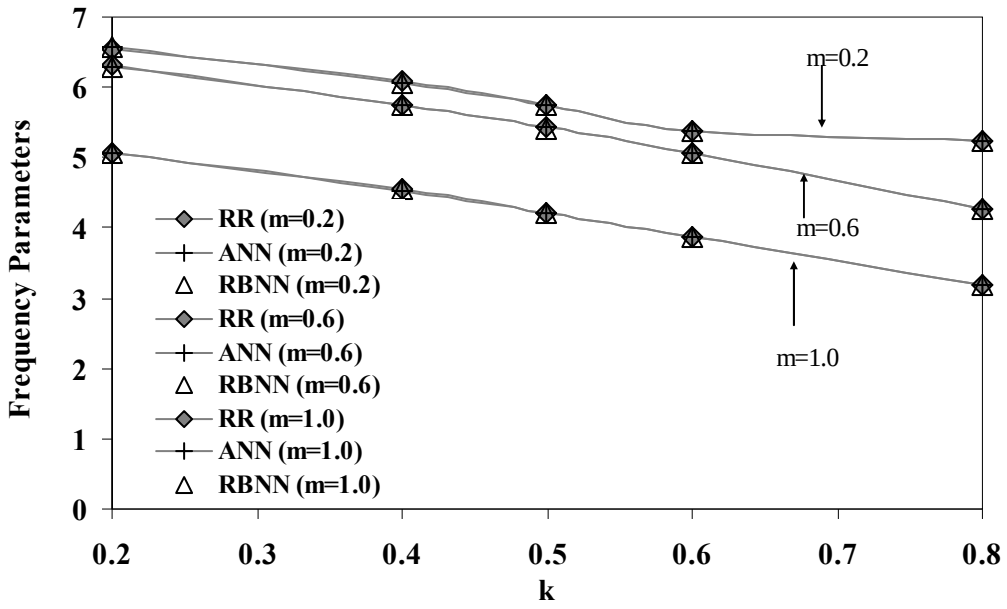


Figure 4.3(b): Results of training of *RBNN* and *ANN* models for Frequency parameters with (both inner and outer simply supported) boundary conditions of the annular plate for m =0.2, 0.6 and 1.0

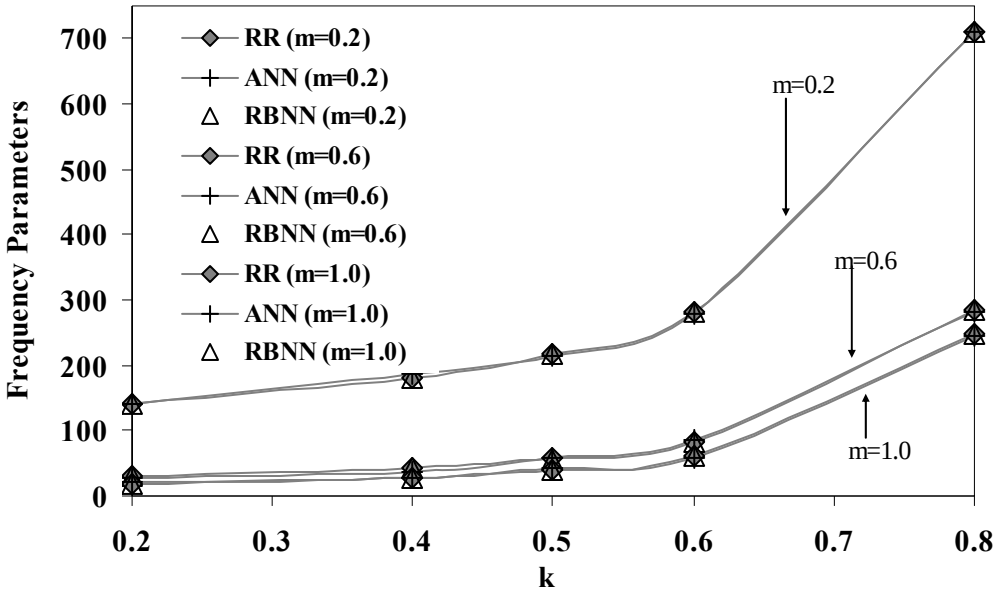


Figure 4.3(c): Results of training of *RBNN* and *ANN* models for Frequency parameters with (both inner and outer Free) boundary conditions of the annular plate for m =0.2, 0.6 and 1.0

4.7 Validation Phase

The new patterns for validating the proposed models for nine boundary conditions have again been taken from *Chakraverty (1992).* The training patterns for validating the proposed model are given in Table 4.12 for a ready reference. The results of the validation phase for the comparison of *RBNN* and *ANN* models along with the numerical computation by the Rayleigh-Ritz method have been tabulated in the table 4.13 for nine boundary conditions. In this table, *RR* represents the numerical value of the frequency taken from the literature, *ANN* represents the value of frequency as obtained by traditional *ANN* model and *RBNN* represents frequency obtained from the proposed model with five neurons in the hidden layer. The performance of the proposed model is given in Figures 4.4(a) to 4.4(c). It can be noted from Figures 4.4(a) to 4.4(c) that the *RBNN* model gives better results when compared with traditional *ANN* model.

Table 4.12: Testing patterns for the proposed *RBNN* model for the nine boundary conditions

m	*k*	*CF*	*CS*	*CC*	*SF*	*SS*	*SC*	*FF*	*FS*	*FC*
0.2	0.2	155.9	155.9	288.5	66.43	142.0	177.2	6.558	12.45	20.28
0.2	0.4	198.6	198.6	355.1	52.24	179.5	221.5	6.079	17.86	27.97
0.2	0.5	231.7	231.7	423.7	52.07	217.6	267.8	5.754	23.65	36.33
0.2	0.6	283.2	283.2	542.1	56.13	282.0	349.1	5.390	33.76	51.78
0.2	0.8	576.6	576.6	1362	91.33	711.1	930.8	5.232	85.80	175.3
0.5	0.2	28.38	58.40	71.90	13.07	37.51	48.91	6.421	6.823	9.750
0.5	0.4	36.52	79.91	98.94	12.40	51.33	67.21	5.886	8.215	13.60
0.5	0.5	45.76	102.6	128.9	13.06	66.21	87.84	5.559	9.269	18.11
0.5	0.6	59.99	144.1	184.8	14.59	93.06	126.4	5.201	11.08	26.73
0.5	0.8	152.0	470.8	641.7	24.27	305.8	443.7	4.417	21.24	98.93
0.6	0.2	20.92	46.87	59.33	9.703	30.28	40.78	6.305	5.996	8.709
0.6	0.4	26.85	66.67	84.87	9.312	43.09	58.18	5.754	6.647	12.34
0.6	0.5	34.21	87.88	113.4	9.824	56.78	77.70	5.421	7.364	16.62
0.6	0.6	46.36	126.7	167.0	10.99	81.90	114.5	5.057	8.701	24.89
0.6	0.8	128.5	439.6	608.8	18.27	284.2	419.9	4.270	16.36	94.47
0.8	0.2	13.69	33.95	45.17	6.383	27.74	30.83	5.882	4.577	7.066
0.8	0.4	17.52	53.29	71.10	6.177	34.08	48.72	5.308	4.692	10.75
0.8	0.5	22.72	73.19	98.70	6.527	46.86	67.46	4.957	5.197	14.85
0.8	0.6	32.43	109.7	150.1	7.313	70.39	102.7	4.579	6.114	22.73
0.8	0.8	105.8	409.3	578.1	12.19	262.6	397.1	3.810	11.23	89.54
1	0.2	10.56	26.78	35.87	4.930	17.03	24.18	5.065	3.592	5.593
1	0.4	13.51	44.93	61.91	4.777	28.08	41.37	4.541	3.654	9.118
1	0.5	17.60	63.85	89.25	5.051	40.01	59.91	4.214	4.074	13.10
1	0.6	25.24	98.79	139.6	5.664	62.12	94.25	3.871	4.809	20.60
1	0.8	92.81	389.5	559.1	9.455	247.0	381.6	3.197	8.782	84.67

Table 4.13: Validation results for *RBNN* trained model

bc	***m* = 0.5**					
CF	***k***	0.2	0.4	0.5	0.6	0.8
	RR	28.38	36.52	45.76	59.99	152.0
	RBNN	28.50	36.51	44.87	59.49	151.9
	ANN	29.97	37.98	46.64	64.47	158.9
SS	*RR*	58.40	79.91	102.6	144.1	470.8
	RBNN	59.00	80.60	102.6	143.8	471.0
	ANN	57.50	82.02	103.9	145.1	445.0
CC	*RR*	71.99	98.94	128.9	184.8	641.7
	RBNN	70.60	99.10	126.8	183.7	638.8
	ANN	69.80	97.90	126.8	184.2	637.6
SF	*RR*	13.07	12.40	13.06	14.59	24.27
	RBNN	12.82	12.30	12.83	14.42	24.16
	ANN	12.04	11.15	13.85	14.48	21.88
SS	*RR*	37.51	51.33	66.21	93.06	305.8
	RBNN	36.85	49.85	65.46	91.96	305.8
	ANN	18.48	41.29	61.67	97.61	300.1
SC	*RR*	48.91	67.21	87.84	126.4	442.9
	RBNN	48.89	67.32	87.91	126.9	441.2
	ANN	48.12	66.87	86.25	124.3	439.8
FF	*RR*	6.421	5.886	5.559	5.201	4.417
	RBNN	6.418	5.891	5.561	5.197	4.409
	ANN	6.498	5.886	5.653	5.987	4.375
FS	*RR*	6.823	8.215	9.269	11.08	21.24
	RBNN	6.936	8.297	9.334	11.18	20.98
	ANN	6.058	8.018	9.773	11.46	20.10
FC	*RR*	9.750	13.60	18.11	26.73	98.93
	RBNN	9.720	13.68	18.22	26.73	98.74
	ANN	9.930	13.90	17.29	27.06	98.91

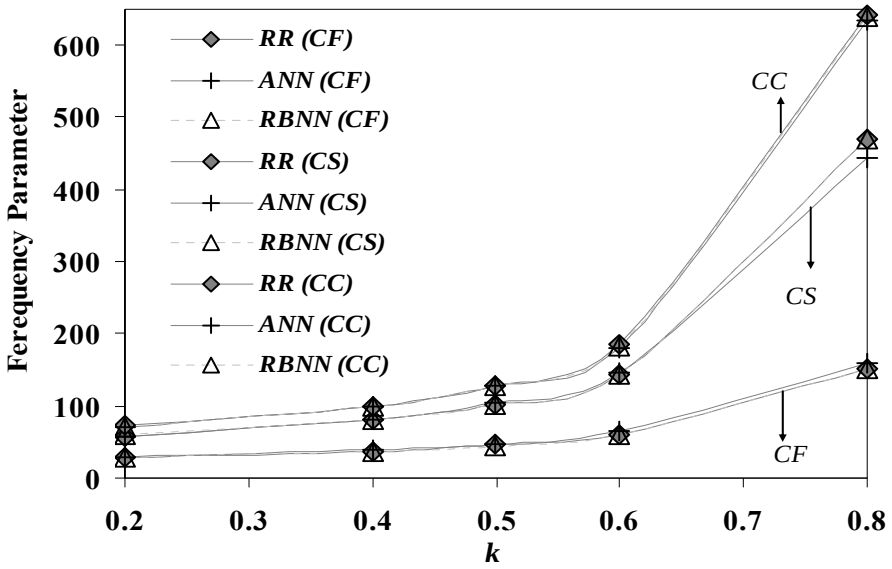


Figure 4.4(a): Performance of the proposed Regression based Neural Network for Frequency parameters with *CF*, *CS* and *CC* conditions (m = 0.5) of the annular plate

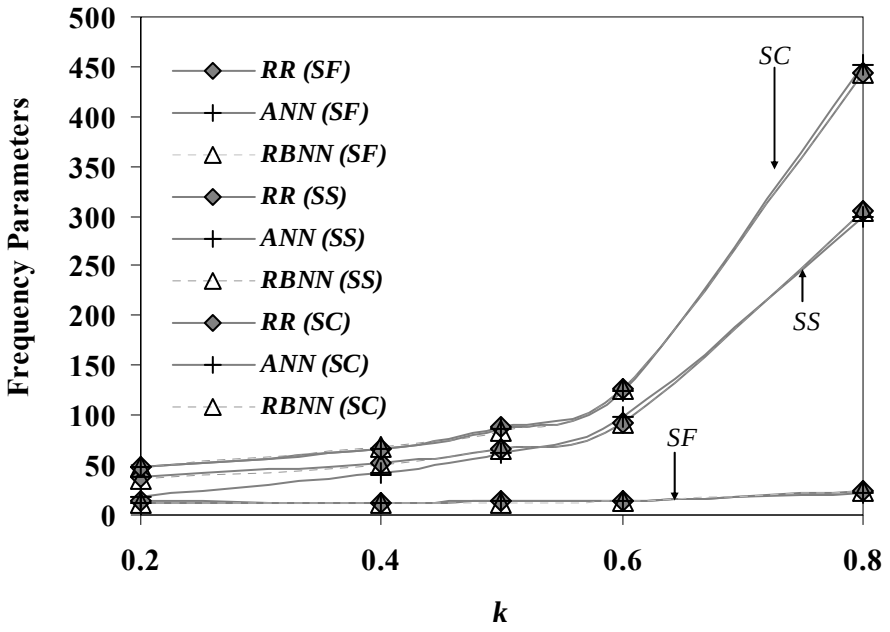


Figure 4.4(b): Performance of the proposed Regression based Neural Network for Frequency parameters with *SF*, *SS* and *SC* conditions (m = 0.5) of the annular plate

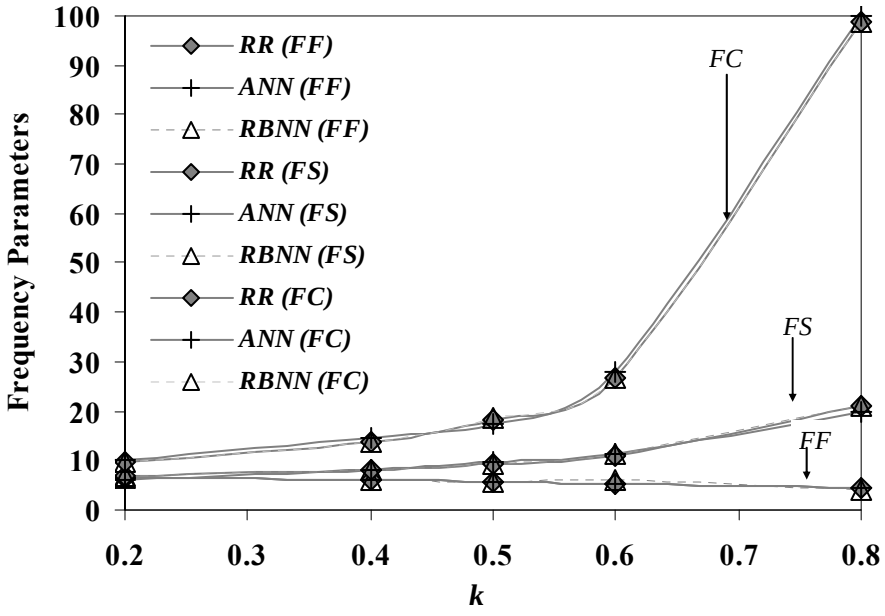


Figure 4.4(c): Performance of the proposed Regression Based Neural Network for Frequency parameters with *FF*, *FS* and *FC* conditions (m = 0.5) of the annular plate

4.8 Error Analysis

In order to use *ANN*s as a reliable tool for vibration analysis of annular circular and annular elliptic plates, one has to take into account the factors that influence its predictions. In this section, we present the *ln*(*MSE*) as function of iteration numbers for all boundary conditions in order to include the trends of the errors for the model used in training the network. It is, however, very difficult to account for each and every factor that influences the prediction capability of a network. One of the popular measurements as explained in Chapter III, for the prediction capability of a network is the mean square error. The graph for *ln*(*MSE*) as a function of iteration number is shown in Figure 4.5(a) for regression based model for *CF*, *CS* and *CC* boundary conditions, in Figure 4.5(b) for *SF*, *SS* and *SC* boundary conditions and in Figure 4.5(c) for *FF*, *FS* and *FC* boundary conditions. These Figures also give the comparison of *MSEs* for traditional *ANN* model and *RBNN* model. The comparison of *MSEs* in the case of outer clamped and inner simply supported (*CS*) (Figure 4.5(a)) shows that the traditional *ANN* model have not converged for the same accuracy as set in the proposed *RBNN* model. The similar trend of *MSEs* are also depicted in Figure 4.5(b) for outer simply supported and inner free (*SF*), outer and inner both simply supported (*SS*) and in Figure 4.5(c) for outer free and inner simply supported (*FS*) boundary conditions, *i.e.*, the traditional *ANN* models do not converge in these cases. It has also been observed that *RBNN* models takes less time in training and have less *MSE* when compared with the traditional *ANN* models in all the nine cases of boundary conditions considered in the present Chapter.

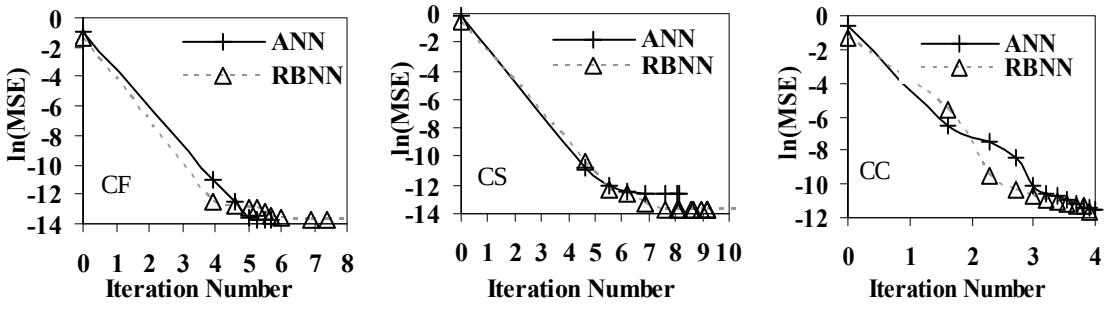


Figure 4.5(a): Error Analysis for *CF*, *CS* and *CC* boundary conditions

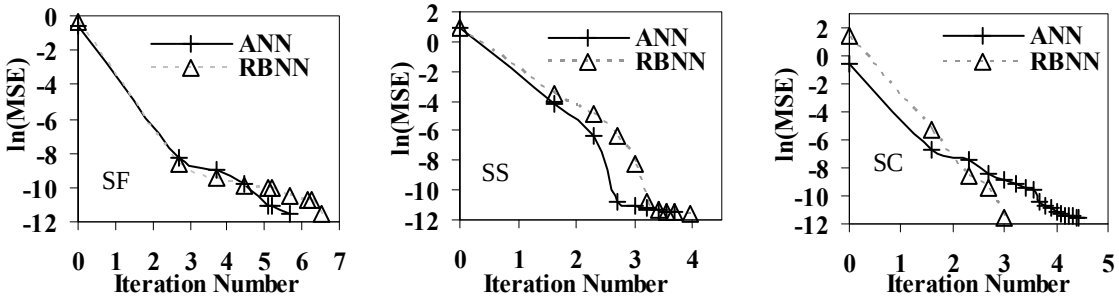


Figure 4.5(b): Error Analysis for *SF, SS* and *SC* boundary conditions

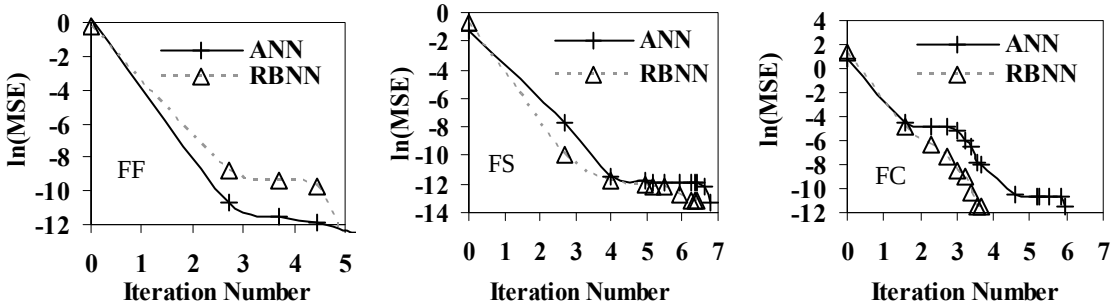


Figure 4.5(c): Error Analysis for *FF, FS* and *FC* boundary conditions

4.9 Conclusion

In this Chapter, a regression based neural network model has been undertaken and it has been simulated for the problem of transverse vibrations of circular and elliptic annular plates for nine different boundary conditions. The training and testing of the proposed model with new data show the efficacy and reliability of this new model. In this model, the number of neurons in the hidden layer can be fixed by choosing a regression polynomial of desired degree. Comparison of the proposed model with the *ANN* model implemented using *NNT* module of MatLab, gives an edge to the proposed regression based model for the example problem considered in this Chapter. It has also been noted that the time devoted in training of the proposed *RBNN* model is considerably less in comparison with the traditional *ANN* model. Tables 4.2 to 4.10 also reveal that the prediction accuracy of the proposed *RBNN* model is higher than the traditional *ANN* model. We also gain in terms of *MSE* when we use the proposed *RBNN* model as is evident from Table 4.13. As such, we may claim that the proposed *RBNN* model is a potential choice for solution of a variety of problems. In particular here the use of *RBNN* model in vibration of plate problems is found to be efficient and better than other traditional methods.

CHAPTER V

ARTIFICIAL NEURAL NETWORK APPROACH FOR IDENTIFICATION OF SINGLE DEGREE OF FREEDOM (SDOF) DYNAMIC SYSTEMS

(The contents of this Chapter are published in Journal of New Building Materials and Construction World (Software Issue), (2003), 8, PP 50-56)

CHAPTER V

ARTIFICIAL NEURAL NETWORK APPROACH FOR IDENTIFICATION OF SINGLE DEGREE OF FREEDOM (SDOF) DYNAMIC SYSTEMS

5.1 Introduction

The dynamic structural identification of mathematical methods of physical structures on the basis of experimental measurements is a problem that has been receiving increasing attention in the past. System identification has become an important area of study because of the increasing needs in estimating the behavior of a system with partially known dynamics especially in the areas of control, health monitoring of structures, damages assessment of structures, pattern recognition (*Natke 1982*). In these applications, the system of interest needs to be known to some extent. Since a system may have a complicated dynamic behavior, the varying environmental changes make the identification process much more difficult than the cases in which those changes are modeled deterministically. However, with the complexity of the structure and errors in measured data, dynamic structural identification is a challenging task in soft computing.

Soft computing techniques are being widely used these days in the field of applied science and engineering. These techniques provide efficient solution of a computationally complex and mathematically intractable problem. One can use soft computing techniques to solve these problems. The most popular constituents of the soft computing methodologies are the *ANNs*, which are considered to provide the mathematical power of the brain. Artificial neural networks provide users a technique where one can solve the complex problems based on certain arithmetic operations only and one does not need to employ any analytical method for problem solving. As such, neural networks have an inherent edge over analytical methods *Narendra and Parthasarathy (1990), Zurada (1994).*

In various fields of study such as structural dynamics, control Engineering, pattern recognition *etc.*, one has to estimate the complete behaviour of the system with a little knowledge about the dynamics of the system. The systems under these fields of study are

described with the help of multiple variables and these are sometimes subjected to disturbances. These disturbances in the parameters, generally, lead to the complexities in the modeling process of systems. As such, it becomes an important problem to estimate the behaviour of these systems.

Mathematical model of a dynamic system, which is based on measured/empirical data, is known as system identification. In the system identification problems, a set of inputs and resulting outputs for a system is known and we desire to find a mathematical description or model of the system. The problems in system identification are of two types, direct and indirect problems. Direct problems are defined by the equations governing the system and the parameters of the system are known. These parameters are used to find the response of the system to a specific input. Indirect problems or inverse problems are defined by the output response to a given input, which is known, but either the governing equation or some of the parameters of the physical process are unknown. *Olivera (1997)* studied identification of dynamic system using neural network and *Wang and Lin (1998)* used Runge-Kutta Neural network for the identification problems. Identification of nonlinear dynamic systems using neural network have also been studied by *Masri et al. (1992, 1993 and 1997)*. Other interesting papers are by *Chassiakos and Masri (1991, 1996)* who have also used artificial neural network to the problem of identification. Detail literature review has already been incorporated in Chapter I. In this Chapter, we have explored the proposed *RBNN* neural network technique for solving system identification of dynamical systems. The fundamental equation of motion (vibration) has been used to explain the approach.

5.2 Identification of the Model

In order to describe the procedure, a single degree of freedom system with/without damping and with/without force has been considered. The corresponding differential equation may be written as: *Warbuton (1964)* and *Biggs (1964)*.

$$m\ddot{x} + b\dot{x} + k\,x = f(t) \qquad \ldots (5.1)$$

For the structural dynamic model, m is the mass, k is the stiffness, b is the damping factor and $f(t)$ is the external exciting force applied to the system. Equation 5.1 can be written as $b\dot{x} + k\,x = f(t) - m\ddot{x} = r(x, \dot{x})$, say

Thus, $r(x,\dot{x})$ represents the unknown restoring force of the system. The neural network identification technique has been used to identify the restoring force of the system

governed by equation 5.1. Figure 5.1 gives the *ANN* architecture used in the identification process.

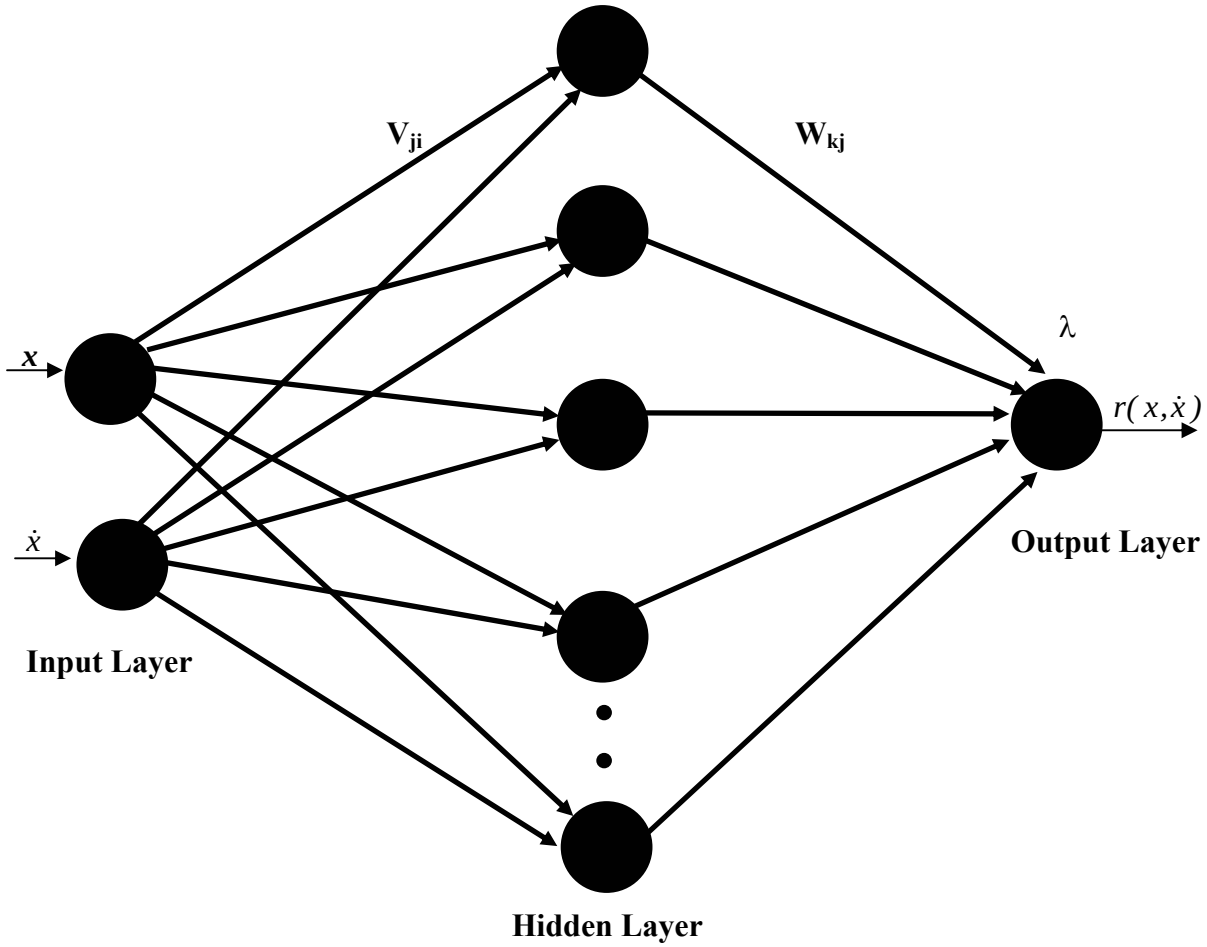


Figure 5.1: *ANN* architecture used to identify the system.

5.3 Training Patterns

In this section, the details of the generation of the training patterns are given. As stated above, fourth order Runge-Kutta method has been used to generate these patterns. Equation 5.1 has been solved by this method to give the values of $x(t)$, $\dot{x}(t)$ and $r(x,\dot{x})$ with specified initial conditions. The patterns that are actually used for training are corresponding to the discrete values of t, starting at $t=0$, with a step-size of 0.1 and up to $t=100$. As such, in all, 1000 patterns have been used to train the network. These patterns can be considered as the approximations to the parameters of the system due to the inherent approximation structure of Runge-Kutta method. Training patterns have been generated for the following four problems with the corresponding differential equations from (5.1) as given below.

$$m\ddot{x} + k\,x = 0 \quad \ldots (5.2)$$

$$m\,\ddot{x} + b\,\dot{x} + k\,x = 0 \quad \ldots (5.3)$$

$$m\,\ddot{x} + k\,x = f(t) \quad \ldots (5.4)$$

$$m\,\ddot{x} + b\,\dot{x} + k\,x = f(t) \quad \ldots (5.5)$$

Equation 5.2 describes a system with without damping and without force. System without and with damping is governed by equation 5.3. Similarly equation 5.4 is the differential equation for without damping and with force and lastly dynamical system with force and damping is described by equation 5.5.

For the empirical experimentation, values of the parameters associated with these equations are taken for an example as: $m = 10$, $k = 54$, $b = 1$ and $f(t) = f_0 \cos(2\,\pi\,t)$, with $f_0 = 1$. All the units of these parameters are in their consistent units. Using these values of the parameters, training patterns (x, $\dot{x}$, $r(x,\dot{x})$) have been generated for each of the cases given in equations 5.2 to 5.5. For the completeness of the study Figures 5.2(a) to 5.2(d) gives the time history plots for the single degree of freedom system for the equations 5.2 to 5.5. The network is then trained with the help of these patterns and applied to model the unknown restoring force of the system.

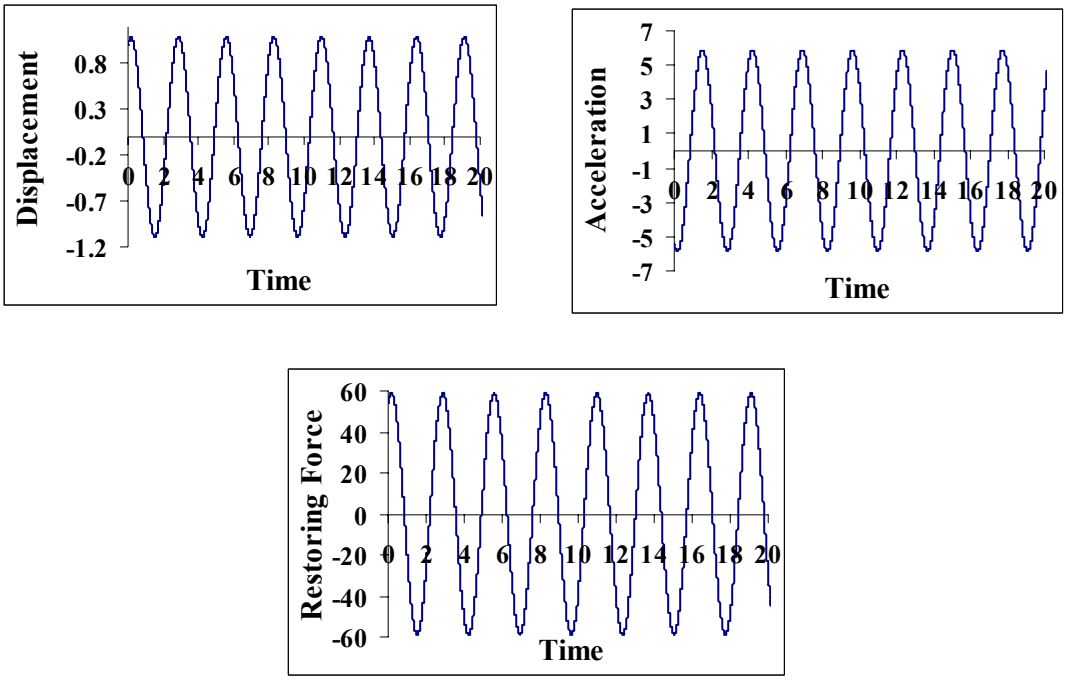


Figure 5.2(a): Time history plots for equation 5.2

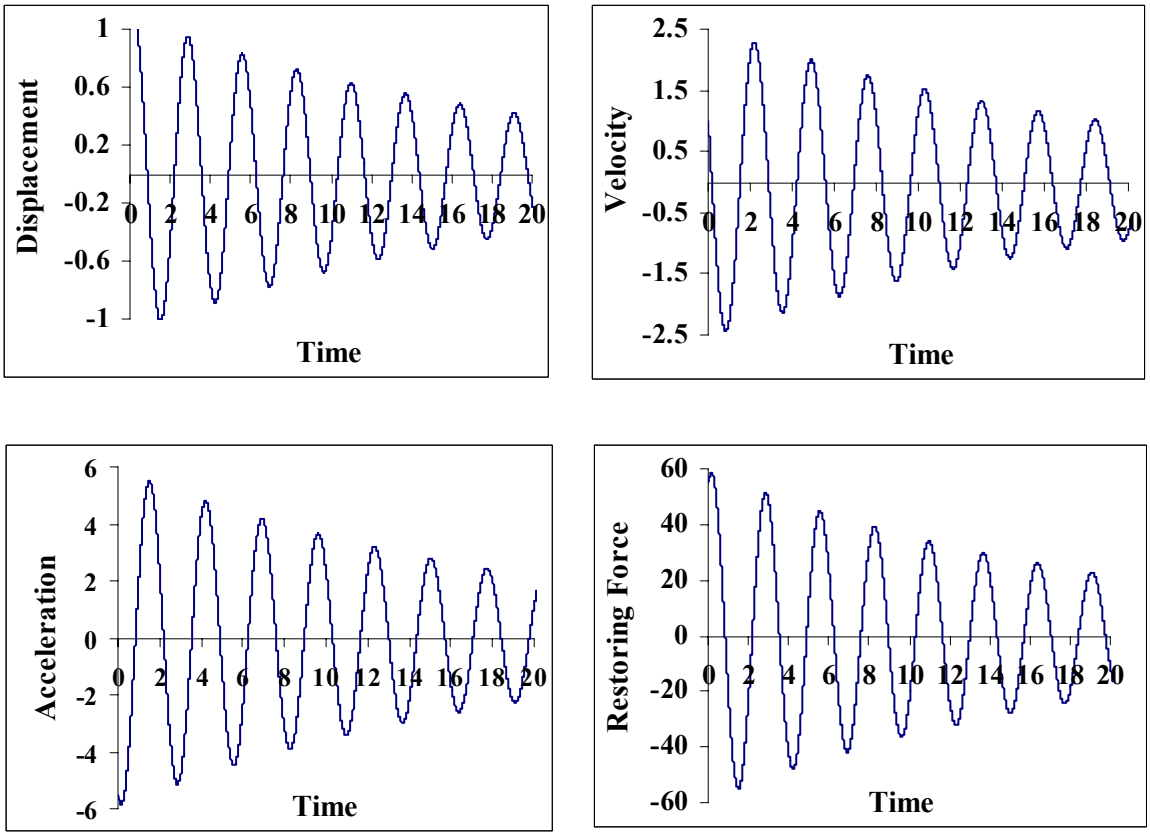


Figure 5.2(b): Time history plots for equation 5.3

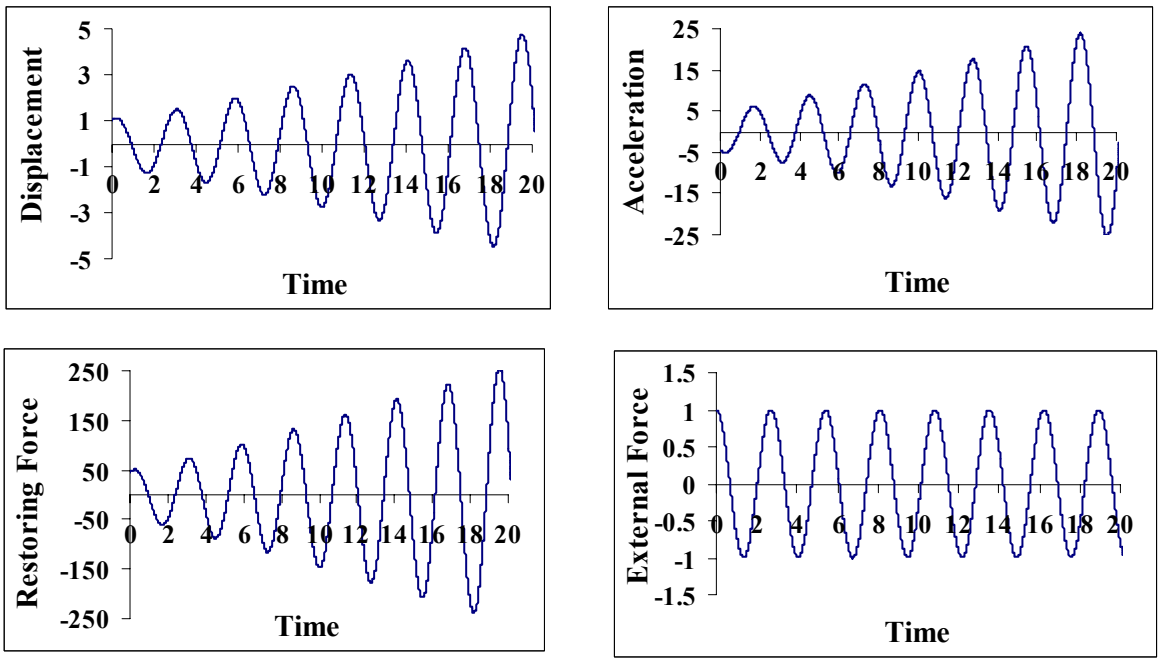


Figure 5.2(c): Time history plots for equation 5.4

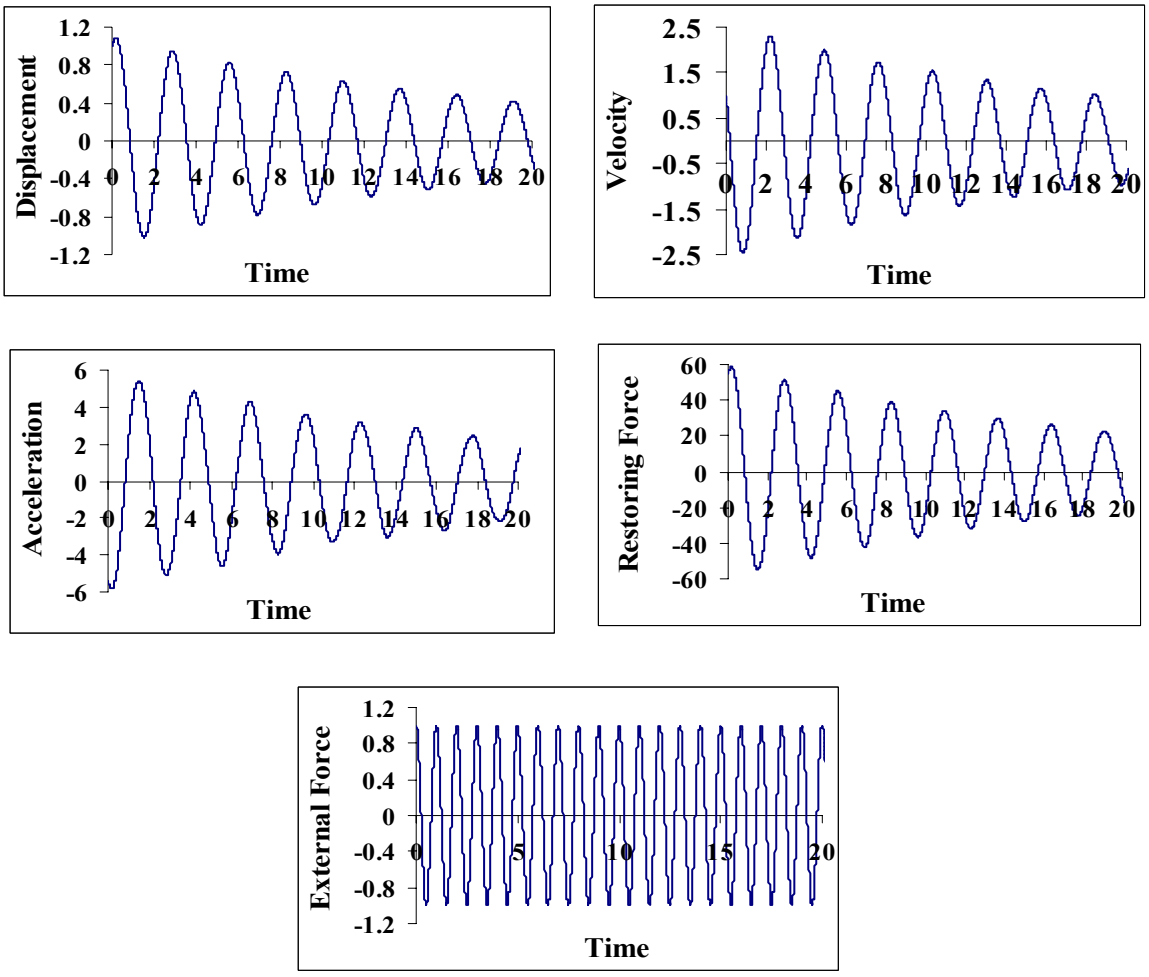


Figure 5.2(d): Time history plots for equation 5.5

5.4 Training Algorithm

In this section, we illustrate the training algorithm that has been used for the *ANN* architecture as shown in Figure 5.1.

5.4.1 The Traditional *ANN* Model

A three-layer architecture for traditional *ANN* approach is considered to model the system as described in Figure 5.1. In this figure, the input layer consists of two inputs, namely displacement (x) and velocity ($\dot{x}$) measurements of the system. The output layer of the this architecture consists of one output in the form of restoring force $r(x, \dot{x})$, which can be considered as an approximation to the actual restoring force of the system. The training algorithm based on the feedforward backpropagation algorithm has been used to train this network. The connection weights interconnecting the neurons between

different layers are taken through a random number generator built-in function in the *NNT*.

5.4.2 The *RBNN* Model

The architecture of this model *Viz.* the number of layer, nodes *etc.* has been taken similar to the traditional *ANN* model. Here, the methodology applied on this *RBNN* model is similar to the methodology described in Chapter IV. In this model, regression polynomials of degree three are fitted to the training patterns and five numbers of nodes have been considered in the hidden layer to facilitate a comparative study on the two models. The output of the network is computed by regression analysis combined with neural activation function as done previously and performed at two stages, *i.e.,* the stage of hidden layer and the stage of output layer. The identification process consists of two essential phases, namely, training phase and validation phase and these have also been investigated here. The training phase has been explained by performing the experiment on these models as discussed in next section.

5.4.3 The Experiment

The procedure described in subsection 5.4.1 and 5.4.2 for training the traditional *ANN* model and proposed *RBNN* model has been implemented in the MatLab environment. In the traditional model, the output of the network is computed by built-in transfer functions, namely, tan-sigmoid (tansig) and linear (purelin) of the *NNT* performed at two stages, *i.e.,* the stage of hidden layer and the stage of output layer. Four cases of the number of nodes in the hidden layer have been considered to facilitate a comparative study on the architecture of the network. The number of nodes in the hidden layer has been considered as 5, 10, 15 and 20 in this for the traditional model. The neural model with feedforward backpropagation algorithm has been trained with Levenberg-Marquardt training function of the *NNT*.

In proposed *RBNN* model, regression polynomial of degree three is fitted to the training patterns. The coefficients of this polynomial are taken as the connecting weights between input and hidden layer. The output of the neurons in the hidden layer is calculated using activation function defined in equations 4.24 to 4.28. These values are used to fit a regression polynomial of the form given in equation 4.29. Finally, the output of the network is computed using equation 4.30 for five neurons in the hidden layer. At this stage, the error of the *RBNN* model is found and a decision is taken whether the network has been trained or not. If the tolerance level of the error is not achieved, we

redefine the connection weights ***W*** according to equation 4.32 and repeat the procedure otherwise we say that the network has been converged or saturated.

In this case the network has been converged with the desired accuracy as shown in the figures 5.3(a) to 5.3(d). The output of the network, $r(x,\dot{x})$ and the *MSE* between neural and desired output is the computed. In these Figures, *RK* represents the values of the displacement, velocity, acceleration and restoring force computed by numerical methods, *ANN* represents these values obtained by the traditional *ANN* models, *RBNN* represents the values of these parameters obtained from the proposed model with five nodes in the hidden layer. The performance of the proposed model is given in the Figures 5.3(a) to 5.3(d) for single degree of freedom system with/without damping and with/without force.

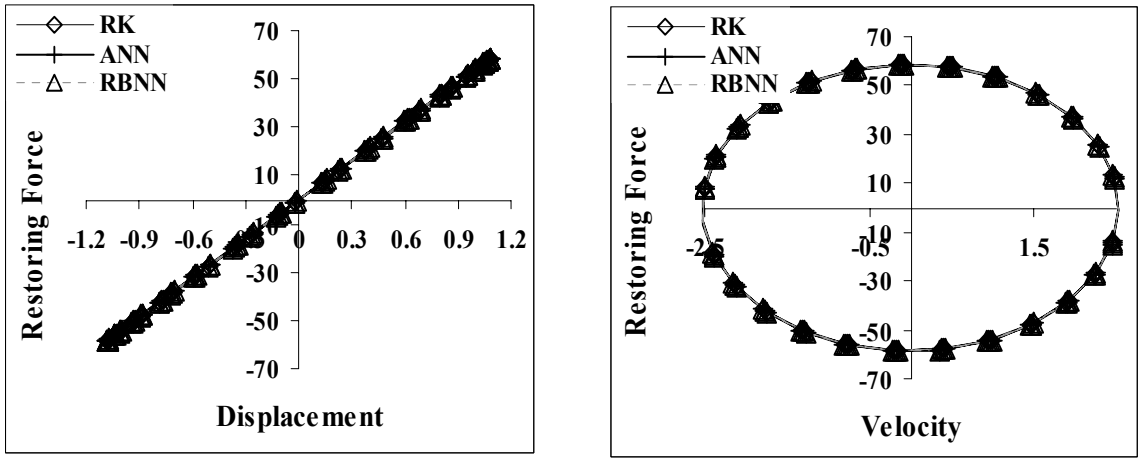


Figure 5.3(a): Results of training of *RBNN* and *ANN* model for equation 5.2

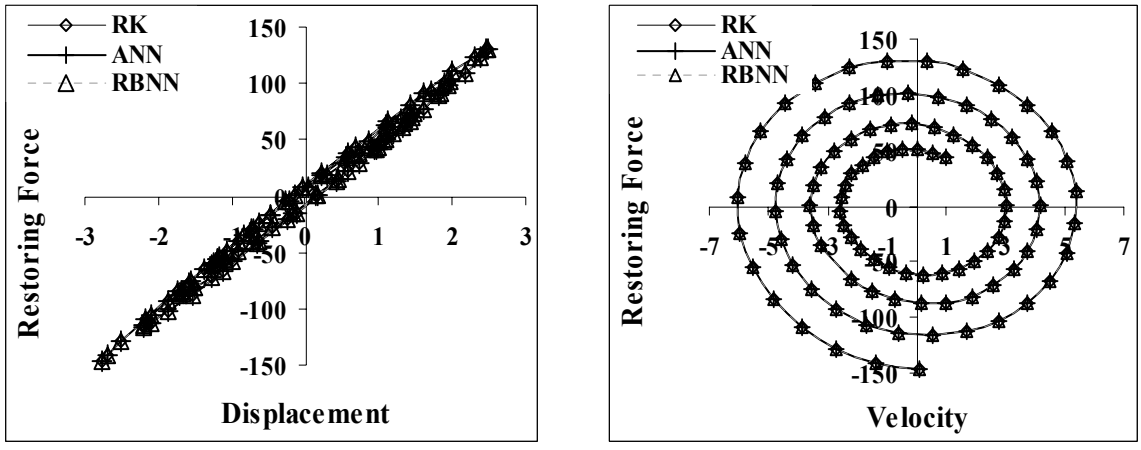


Figure 5.3(b): Results of training of *RBNN* and *ANN* model for equation 5.3

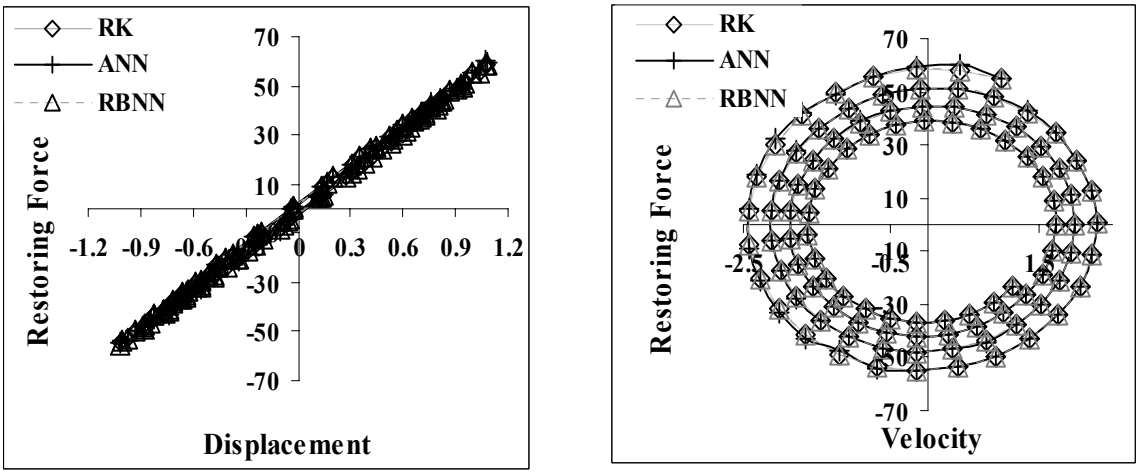


Figure 5.3(c): Results of training of *RBNN* and *ANN* model for equation 5.4

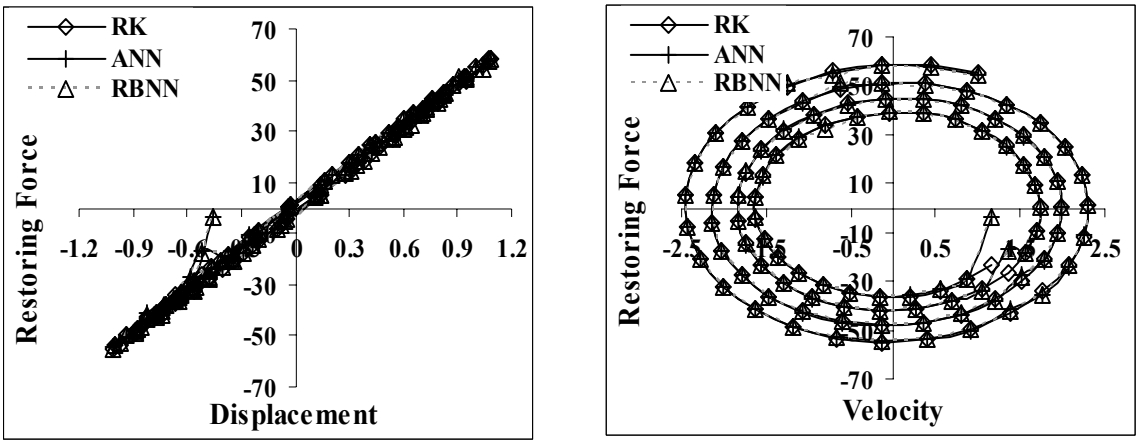


Figure 5.3(d): Results of training of *RBNN* and *ANN* model for equation 5.5

5.6 Validation Phase

In the validation phase, the identifying or generalization ability of the network has been analysed. The patterns for testing, *i.e.*, x, $\dot{x}$ and $r(x, \dot{x})$ are again generated by the fourth order Runge-Kutta method with same initial conditions but with different step-size h. The state variable plots of these records obtained by Runge-Kutta method and that by the developed neural network are presented in Figures 5.4(a) to 5.4(d) for the problems defined by the equations 5.2 to 5.5.

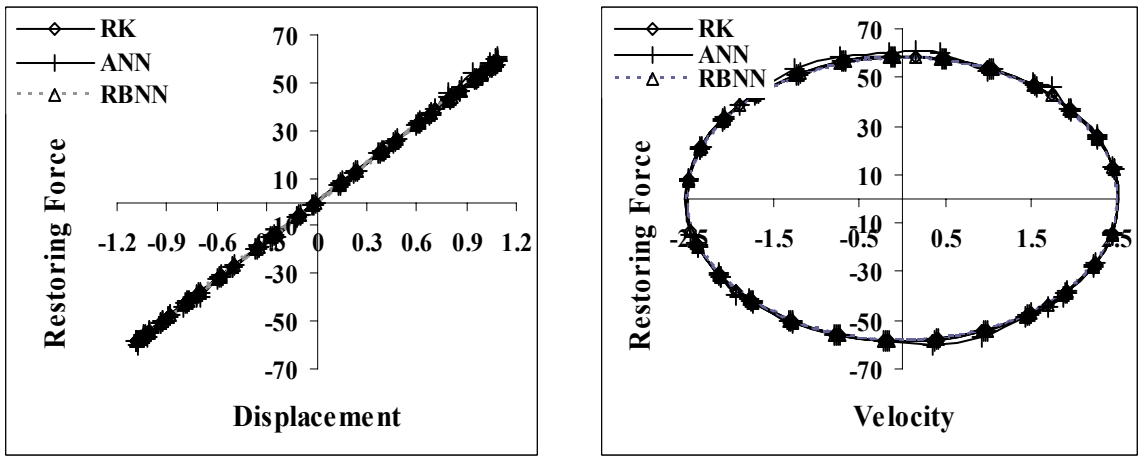


Figure 5.4(a): Validation of *RBNN* and *ANN* model for equation 5.2

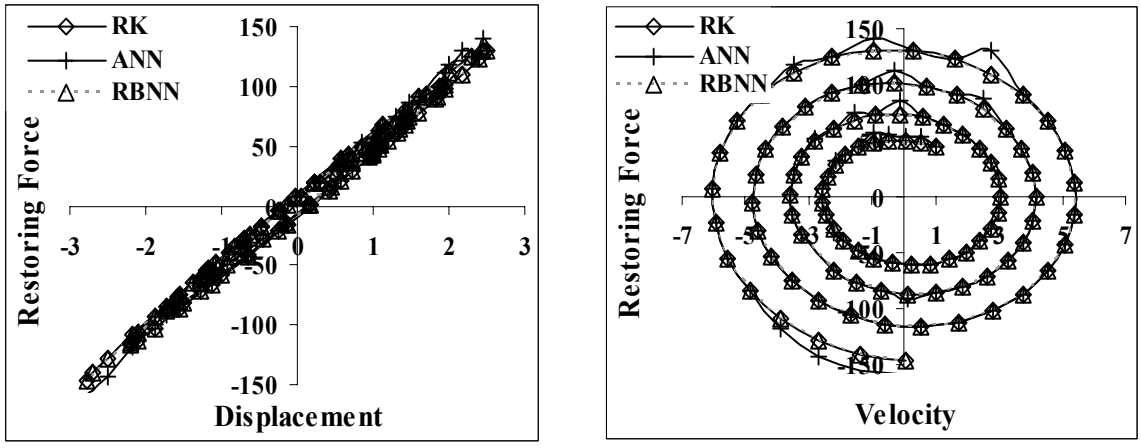


Figure 5.4(b): Validation of *RBNN* and *ANN* model for equation 5.3

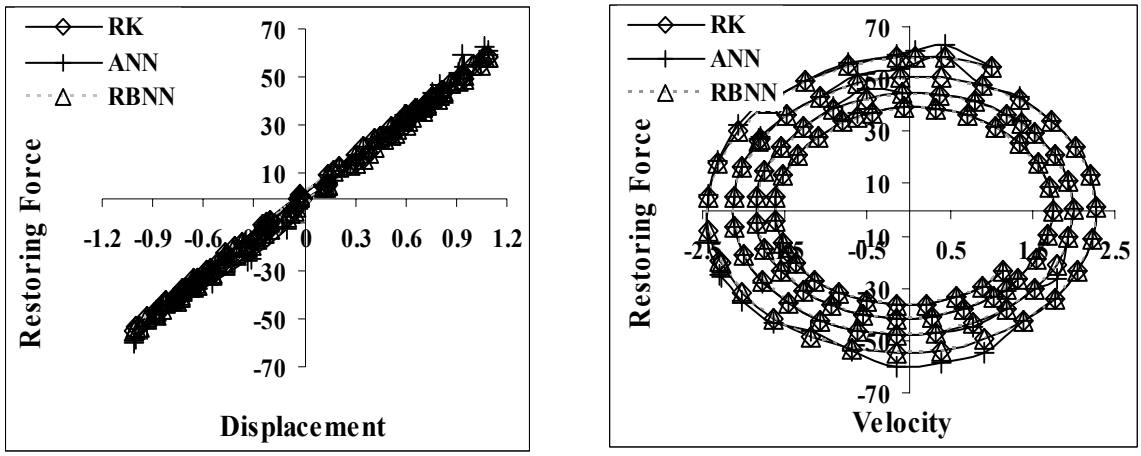


Figure 5.4(c): Validation of *RBNN* and *ANN* model for equation 5.4

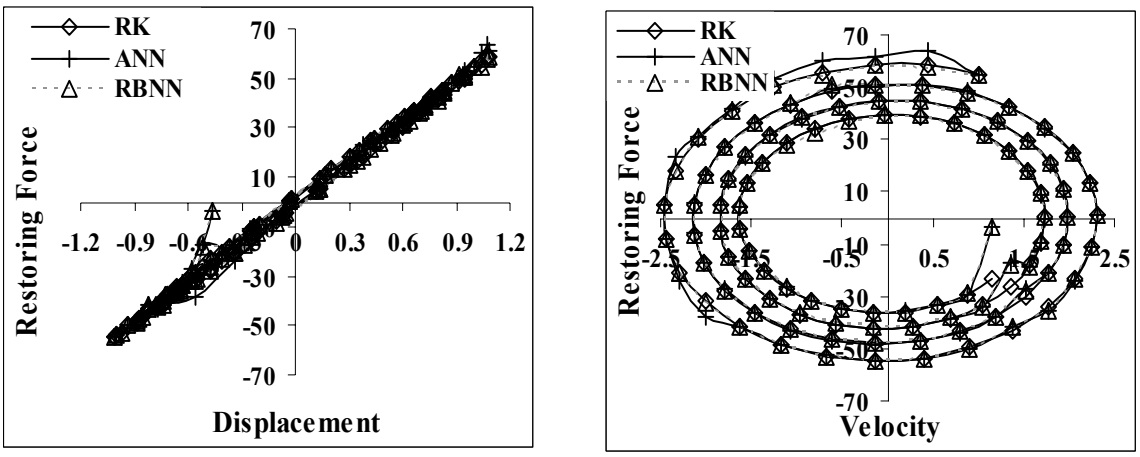
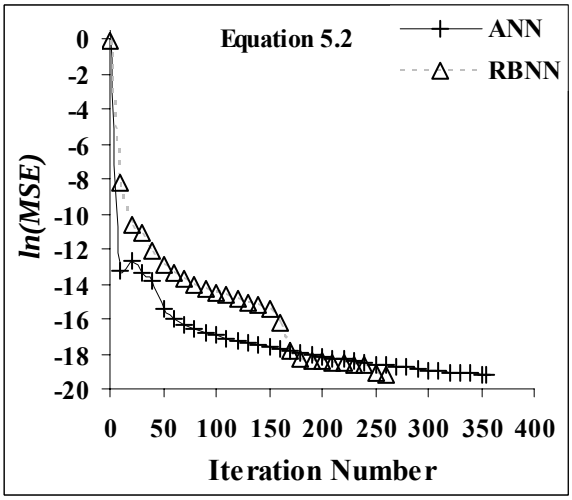


Figure 5.4(d): Validation of *RBNN* and *ANN* model for equation 5.5

5.5 Error Analysis

In order to use *ANNs* as a reliable tool for identification of dynamic system, one has to take into account the factors that influence its prediction capabilities. In this section, the effect of changing the number of neurons in the hidden layer on the error between desired and neuron output has been studied for the traditional *ANN* model. Four cases, namely, numbers of neurons equal to 5, 10, 15 and 20 have been considered in the hidden layer. During the training of the network, it is noted that if the number of nodes in the hidden layer is changed, then different number of iterations are needed to have the desired convergence. This is expected because the change in initialization also changes the time and number of iterations to achieve the convergence, which is usual as in other well-known numerical techniques. In order to have a comparison between the iterations that are carried out to achieve desired accuracy are shown in Figure 5.5 for the case of the equations 5.2 to 5.3. Similarly Figure 5.6 also depicts the comparison between the iterations that are carried out to achieve desired accuracy for case of the equations 5.4 and 5.5. Another popular measurement as explained in Chapter III, for the prediction capability of a network is the mean square error. The graph for *ln*(*MSE*) as a function of iteration number is shown in Figure 5.7 for system equations 5.2 to 5.5. This Figure also gives the comparison of *MSEs* for traditional *ANN* model and *RBNN* model. It has also been observed from these Figures that *RBNN* models takes less time in training and have less *MSE* when compared with the traditional *ANN* models.



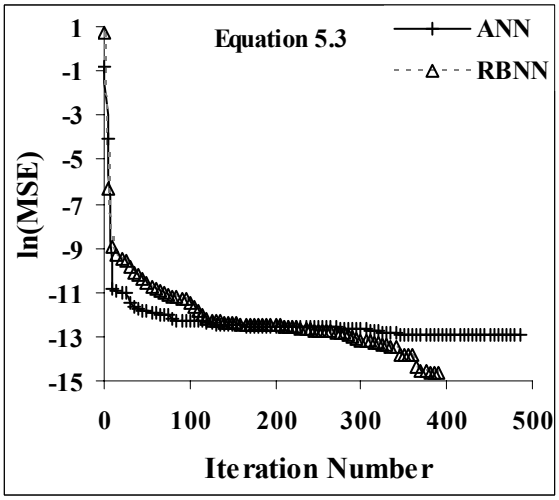
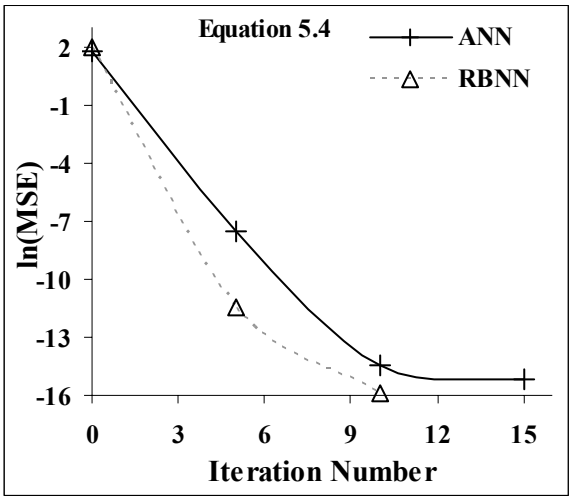


Figure 5.5: Effect of number of nodes in the hidden layer on number of iterations for equation 5.2 to 5.3



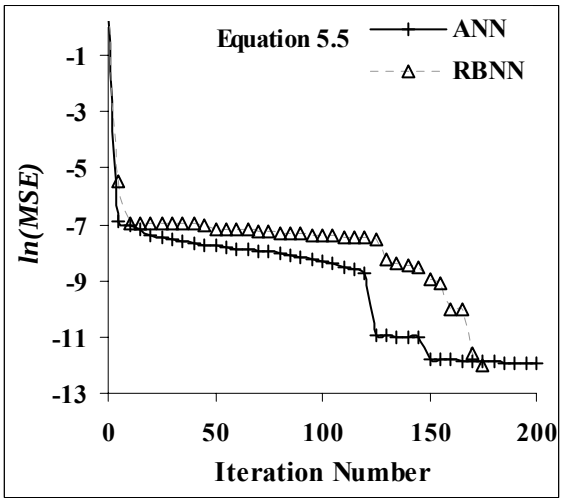


Figure 5.6: Effect of number of nodes in the hidden layer on number of iterations for equation 5.4 to 5.5

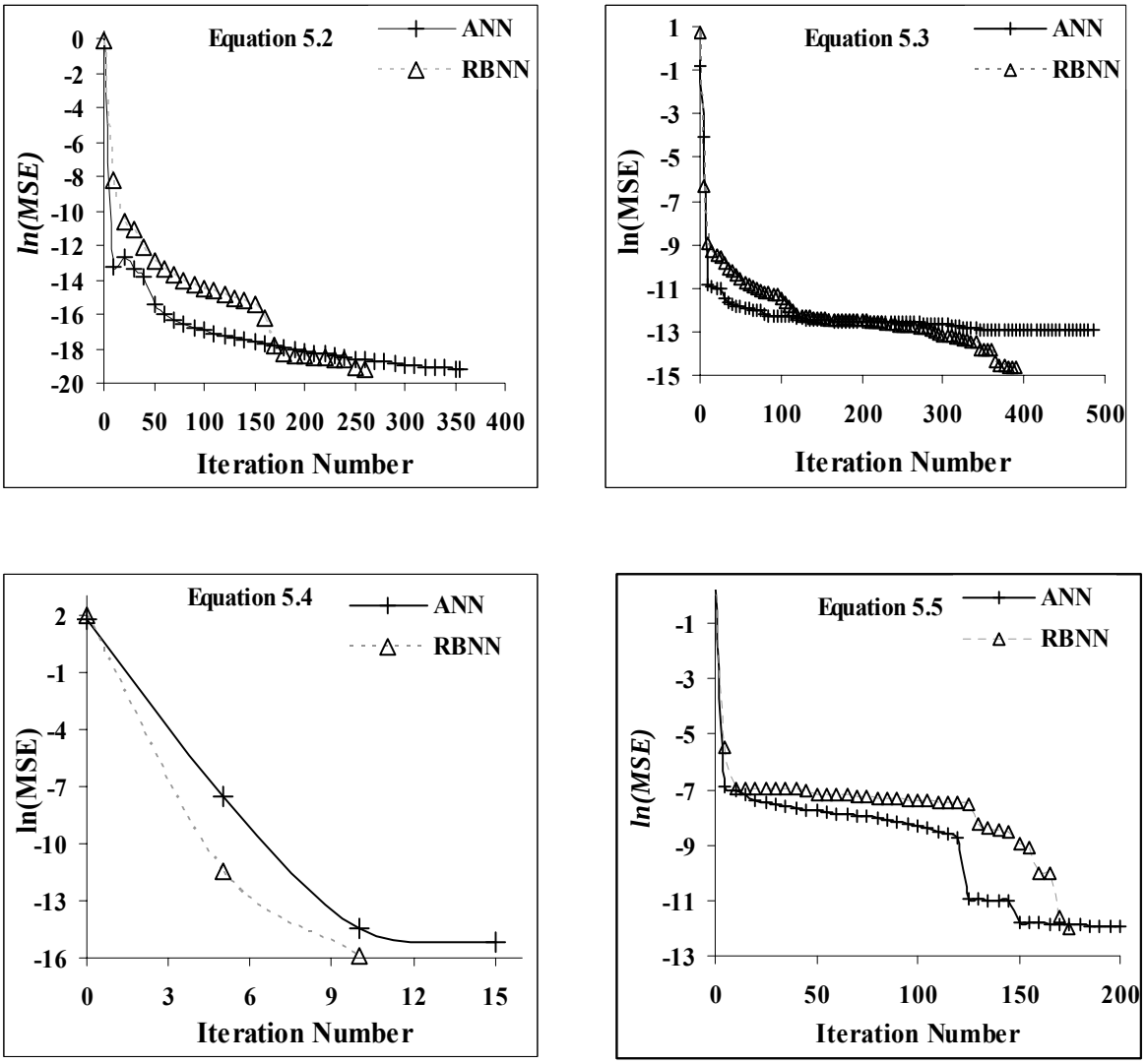


Figure 5.7: Error analysis for the equation 5.2 to 5.5

5.7 Conclusion

The numerical computations demonstrate the powerfulness of the soft-computing technique *viz.* the artificial neural network (*ANN*) in the solution of system identification of structural dynamics problems. Present analysis may be divided into three steps. In the first step, the training patterns for the problem have been generated using fourth order Runge-Kutta (*RK*) method. The second step includes the validation, in which one uses the trained network and get results from new inputs particularly for those inputs for which the network had not been trained. The third and the last step is the error analysis. This was done to have an idea for the number of iterations and the number of nodes in hidden layer to be considered in the analysis. As mentioned, similar time history plots as in Figures

5.2(a) and 5.2(b) can be plotted for each of the four problems defined by equations 5.2 to 5.5 and these may be used for training the network by varying different number of nodes considered in the hidden layer. It is already known that in *ANN*, the optimum number of nodes in the hidden layer required is still an open problem for a particular analysis. From the comparison of the iterations to carry out desired accuracy, it has been observed that the network having 5 nodes in the hidden layer has lowest *MSE* values for traditional *ANN* with respect to the mentioned system identification problem. Accordingly it has been seen that, number of nodes in *RBNN* model can be fixed. The same has been simulated by taking fixed five nodes in the hidden layer. It has also been noted from the Figure 5.6 that the time devoted in training of the proposed *RBNN* model is considerably less in comparison with the traditional *ANN* model. It is also evident from the Figure 5.7 that the prediction accuracy of the proposed *RBNN* model is higher than the traditional *ANN* model. Here, it may be seen that the results are in excellent agreement.

Thus the study investigated the numerical experiments of the four basic and fundamental problems of system identification of structural dynamics *viz.* for (i) Single-Degree of Freedom (*SDF*) system without damping and force, (ii) *SDF* with damping and without force, (iii) *SDF* without damping and with force and (iv) *SDF* with damping and force using the powerful method of traditional *ANN* and also by proposing the new method of *RBNN*.

CHAPTER VI

REGRESSION BASED MULTI-INPUT MULTI-OUTPUT (MIMO) ARTIFICIAL NEURAL NETWORK FOR MULTI-DEGREE OF FREEDOM (MDF) OF DYNAMIC SYSTEMS

(The contents of this Chapter are to be communicated for publication)

CHAPTER VI

REGRESSION BASED MULTI-INPUT MULTI-OUTPUT (MIMO) NEURAL NETWORK FOR MULTIDEGREE OF FREEDOM (MDF) DYNAMIC SYSTEMS

6.1 Introduction

In chapter V, the proposed regression based neural network model has been used in particular for the single degree of freedom dynamic system problems. In various fields of study such as structural dynamics, control Engineering, pattern recognition *etc.*, one has to estimate the complete behaviour of the system with a little knowledge about the dynamics of the system. The systems under these fields of study are described with the help of multiple variables and these are sometimes subjected to disturbances also. These disturbances in the parameters, generally, lead to the complexities in the modeling process of systems. As such, it becomes an important problem to estimate the behaviour of these systems. Accordingly here multidegree of freedom systems and their identification problems are addressed using *ANN* models. In other words the *RBNN* model has been investigated for multi-input multi-output systems.

6.2 Identification of the Model

In order to describe the procedure, a multi degree of freedom (*MDF*) system with/without damping and with/without force has been considered (*Chopra 2004, Warbuton 1964 and Biggs 1964*). Let us describe the structural dynamics model first for an example of a two degree of freedom system. Accordingly let m_1 and m_2 are two masses, k_1 and k_2 are the stiffness, c1 and c2 are the damping factor and $f_1(t)$ and $f_2(t)$ are the external exciting force applied to the system of two degrees of freedom. The corresponding equation of motion of the system may easily be written as

$$m_1 \ddot{x}_1 + (c_1 + c_2)\dot{x}_1 - c_2 \dot{x}_2 + (k_1 + k_2) x_1 - k_2 x_2 = f_1(t) \quad \ldots(6.1)$$

$$m_2 \ddot{x}_2 - c_2 \dot{x}_1 + c_2 \dot{x}_2 - k_2 x_1 + k_2 x_2 = f_2(t) \quad \ldots(6.2)$$

The matrix form of this system may be obtained as:

$$\begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix}\begin{bmatrix} \ddot{x}_1 \\ \ddot{x}_2 \end{bmatrix}+\begin{bmatrix} c_1+c_2 & -c_2 \\ -c_2 & c_2 \end{bmatrix}\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix}+\begin{bmatrix} k_1+k_2 & -k_2 \\ -k_2 & k_2 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}=\begin{bmatrix} f_1 \\ f_2 \end{bmatrix} \quad \ldots (6.3)$$

and in general a multi-degree of freedom (*MDF*) system may be written by the following equation of motion in matrix form:

$$M\{\ddot{x}\}+C\{\dot{x}\}+K\{x\}=F(t) \quad \ldots (6.4)$$

Here, for this n degree of freedom system, $[M]_{n\times n}, [C]_{n\times n}$ *and* $[K]_{n\times n}$ are known as mass, damping and stiffness matrices respectively and those may be given by

$$M=\begin{bmatrix} m_1 & 0 & \ldots & \ldots & 0 \\ 0 & m_2 & 0 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & 0 & m_{n-1} & 0 \\ 0 & \ldots & \ldots & 0 & m_n \end{bmatrix}_{n\times n}$$

$$C=\begin{bmatrix} c_1+c_2 & -c_2 & 0 & 0 & 0 \\ -c_2 & c_2+c_3 & -c_3 & 0 & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & \ldots & -c_{n-1} & c_{n-1}+c_n & -c_n \\ 0 & \ldots & \ldots & -c_n & c_n \end{bmatrix}_{n\times n}$$

$$K=\begin{bmatrix} k_1+k_2 & -k_2 & 0 & 0 & 0 \\ -k_2 & k_2+k_3 & -k_3 & 0 & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & \ldots & -k_{n-1} & k_{n-1}+k_n & -k_n \\ 0 & \ldots & \ldots & -k_n & k_n \end{bmatrix}_{n\times n}$$

Equation 6.4 can now be written as

$$C\{\dot{x}\}+K\{x\}=F(t)-M\{\ddot{x}\} \quad \ldots (6.5)$$

and we write

$$g(x,\dot{x})=\{g\}=F(t)-M\{\ddot{x}\} \quad \ldots (6.6)$$

where x is the system displacement vector and $g(x,\dot{x})$ as defined above is the unknown vector of restoring forces. As mentioned above, M is the mass matrix and $F(t)$ is the system excitation. We will incorporate the idea of application of developed *RBNN* in the system identification problem for multi degree of freedom system. In particular for clarity of the application we will consider the two degree of freedom system. The same explanations may be given for n degree of freedom system. Accordingly the restoring forces for the two degree of freedom system are given by

$$\begin{Bmatrix} g_1 \\ g_2 \end{Bmatrix} = \begin{Bmatrix} f_1 \\ f_2 \end{Bmatrix} - \begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix} \begin{bmatrix} \ddot{x}_1 \\ \ddot{x}_2 \end{bmatrix}$$

$$\begin{Bmatrix} g_1 \\ g_2 \end{Bmatrix} = \begin{Bmatrix} f_1 \\ f_2 \end{Bmatrix} - \begin{bmatrix} m_1 \ddot{x}_1 \\ m_2 \ddot{x}_2 \end{bmatrix} \quad \ldots (6.7)$$

$$Restoring\ force = \begin{Bmatrix} g_1 \\ g_2 \end{Bmatrix} = \begin{bmatrix} f_1 - m_1 \ddot{x}_1 \\ f_2 - m_2 \ddot{x}_2 \end{bmatrix}$$

The equation 6.4 can also be written in a different form as

$$\ddot{x} = M^{-1}\{-C\{\dot{x}\} - K\{x\} + F(t)\} \quad \ldots (6.8)$$

$$\ddot{\mathrm{x}} = \mathrm{M}^{-1}\{-\mathrm{g}(\mathrm{x},\dot{\mathrm{x}}) + \mathrm{F}(\mathrm{t})\} \quad \ldots (6.9)$$

In this Chapter, based on the above mentioned equations two modeling and identification problems have been undertaken:

i) Partially known system: the vector of restoring forces $g(x,\dot{x})$ is identified from the input/output data, assuming that the mass vector M is known.

ii) Completely unknown system: even the masses are not known; the system behaviour is modeled on the basis of input/output data only.

Partially known system

In this case the masses m_i, $i = 1, ..., N$ are known or can be estimated, but the restoring force vector function $g(x,\dot{x})$ is unknown.

The neural network identification technique has been used to identify the restoring

force $g(x,\dot{x})$ for the system governed by equation 6.6. Figure 6.1 gives the block diagram for identification process.

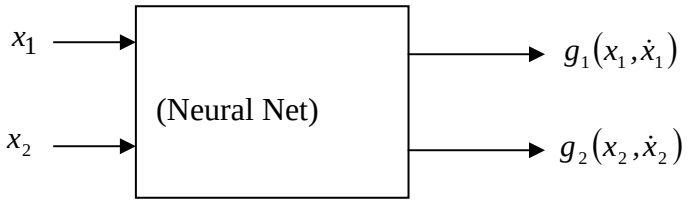


Figure 6.1: Block diagram used to identify partially known system

In this case the mass vector $m=[m_1,\ \ m_2,\ \ .....\ \ m_N]^T$ as well as the vector of restoring forces $g(x,\dot{x})$ is unknown. A neural network has been used to model the system behaviour on the basis of input/output data only.

The neural network identification technique has been used to identify the acceleration $\ddot{x}$ for the system governed by equation 6.9. Figure 6.2 gives the block diagram for identification process.

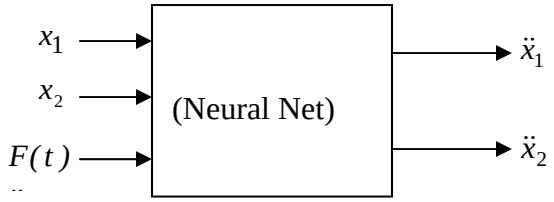


Figure 6.2: Block diagram used to identify completely unknown system

6.3 Training Patterns

In this section, the details of the generation of the training patterns are given. As regards the fourth order Runge-Kutta method has been used to compute these patterns by solving the coupled equation 6.3. The values of $x_1(t)$, $\dot{x}_1(t)$, $\ddot{x}_1(t)$, $x_2(t)$, $\dot{x}_2(t)$, $\ddot{x}_2(t)$, $g_1(x_1,\dot{x}_1)$ and $g_2(x_2,\dot{x}_2)$ with specified initial conditions may easily be computed numerically by utilizing equations 6.3 and 6.7. The patterns that are actually used for training

are corresponding to the discrete values of t, starting at $t = 0$, with a step-size of 0.01 and up to $t = 100$. As such, 1000 patterns in all have been used to train the network. These patterns can be considered as the approximations to the parameters of the system due to the inherent approximation structure of Runge-Kutta method. These patterns have been generated for the following four problems (with respect to the system without/with damping and without/with force) given by the corresponding differential equations from equation 6.4 as

$$M\{\ddot{x}\} + K\{x\} = 0 \quad \text{... (6.10)}$$

$$M\{\ddot{x}\} + C\{\dot{x}\} + K\{x\} = 0 \quad \text{… (6.11)}$$

$$M\{\ddot{x}\} + K\{x\} = F(t) \quad \text{… (6.12)}$$

$$M\{\ddot{x}\} + C\{\dot{x}\} + K\{x\} = F(t) \quad \text{… (6.13)}$$

.For the empirical experimentation, values of the parameters associated with these equations are taken as: $m_1 = 12; m_2 = 12$, $k_1 = 7214; k_2 = 14900$, $c_1 = 21; c_2 = 60$ and $f(t) = \sin[\omega_0 + (\omega_F - \omega_0)(t / t_F)]t$, where $\omega_0 = 10 rad/s, \omega_F = 15 rad/s$ and $t_F = 10.0 s$. All the units are in their consistent units. Using these values of the parameters, training patterns $x_1(t)$, $\dot{x}_1(t)$, $\ddot{x}_1(t)$, $x_2(t)$, $\dot{x}_2(t)$, $\ddot{x}_2(t)$, $F(t)$, $g_1(x_1, \dot{x}_1)$ and $g_2(x_2, \dot{x}_2)$ have been generated for each of the cases given in equations (6.10) to (6.13).

For the sake of completeness, Figures 6.3(a) and 6.3(d) give the time history plots for the two degree of freedom systems of the equations 6.10 to 6.13. The network is then trained with the help of these patterns and those are used in the identification for partially known and completely unknown systems. (Figure 6.3 shows results up to 10 secs.)

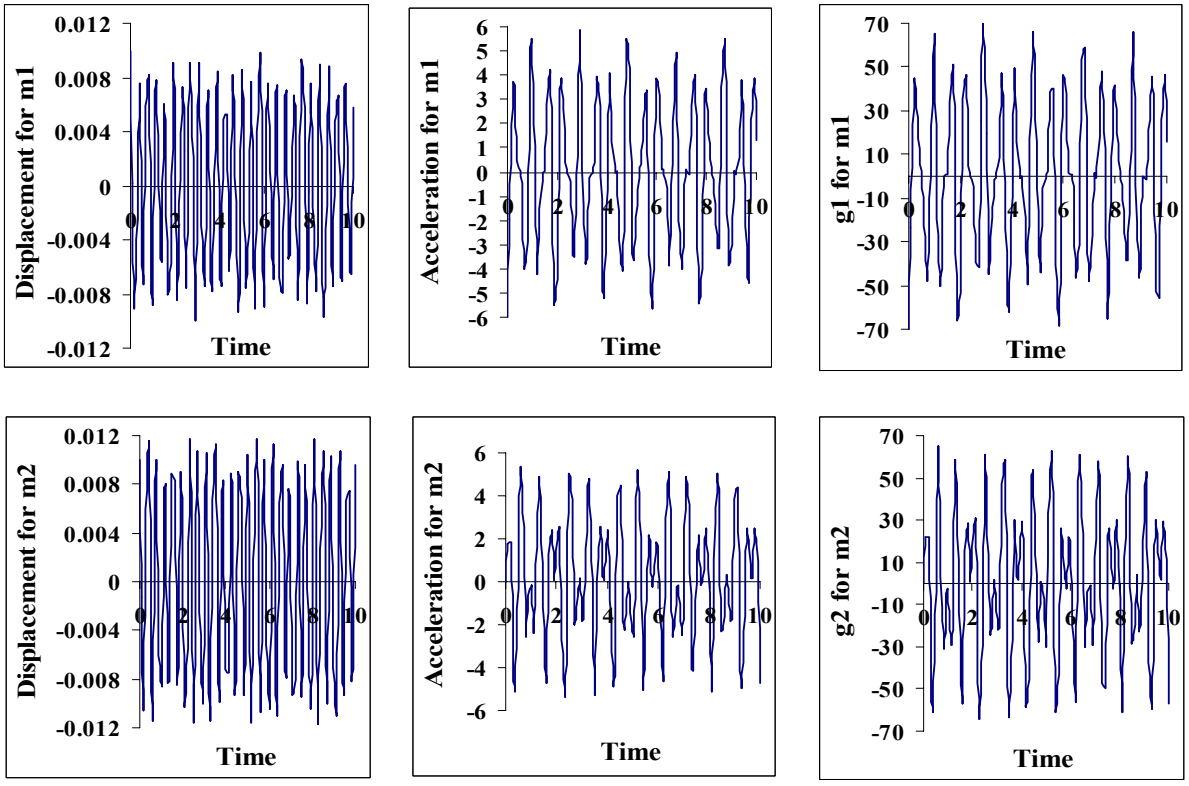


Figure 6.3(a): Time history plots for equation 6.10

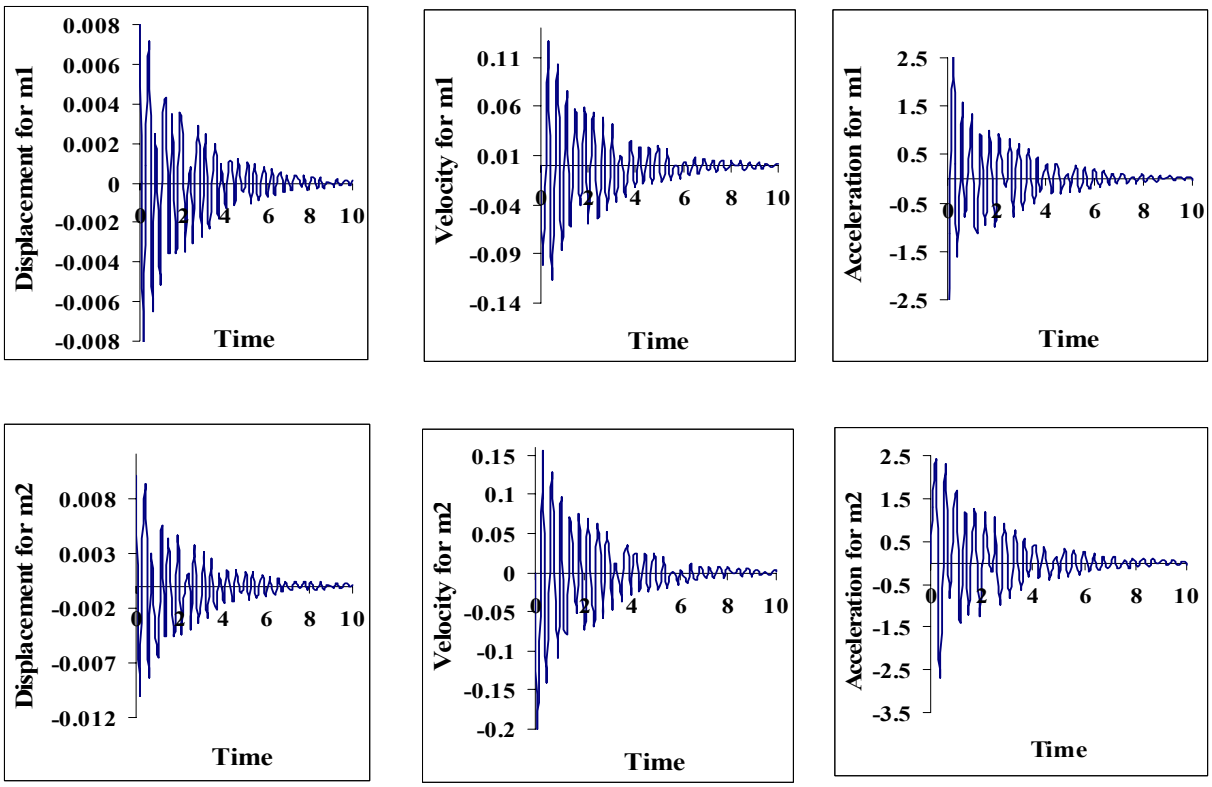


Figure 6.3(b): Time history plots for equation 6.11

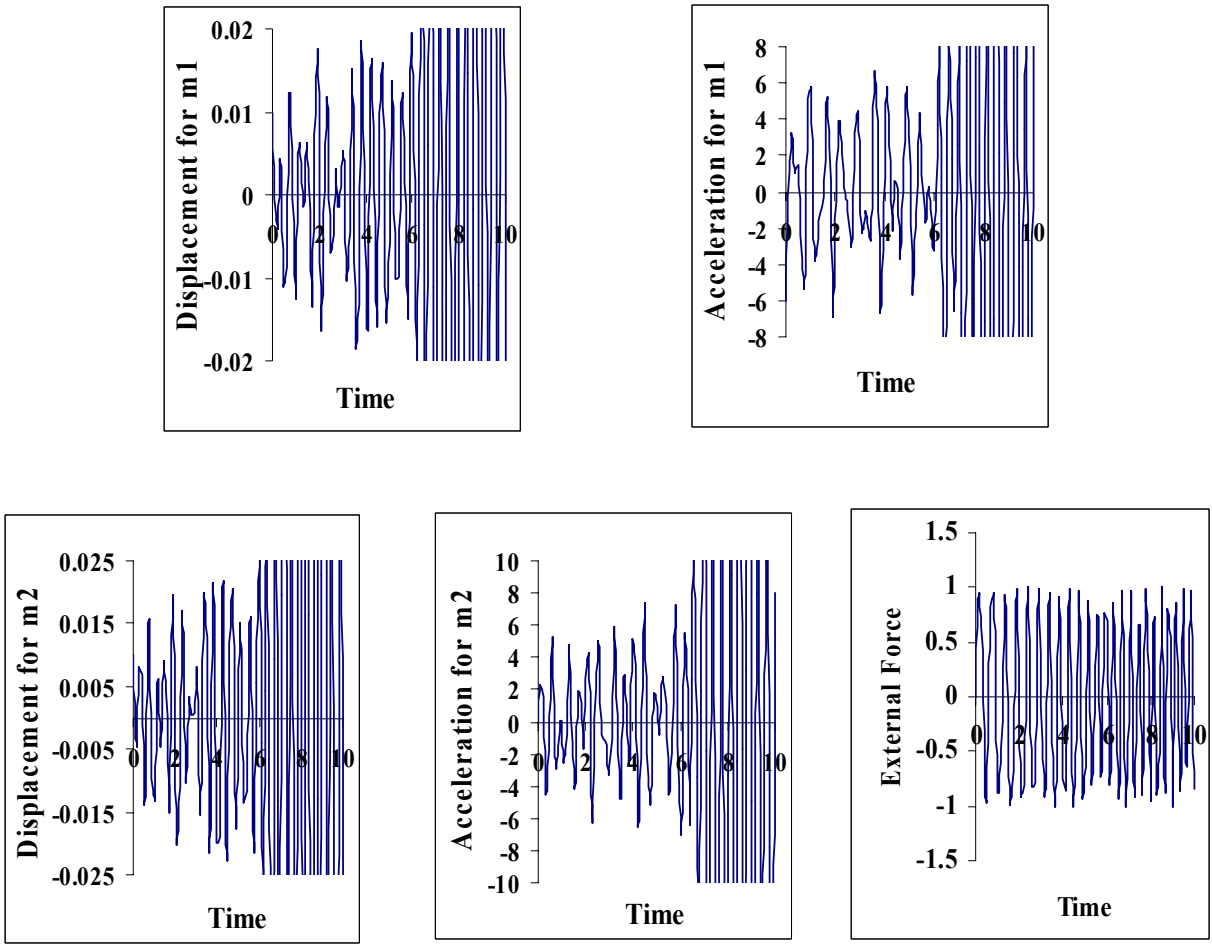


Figure 6.3(c): Time history plots for equation 6.12

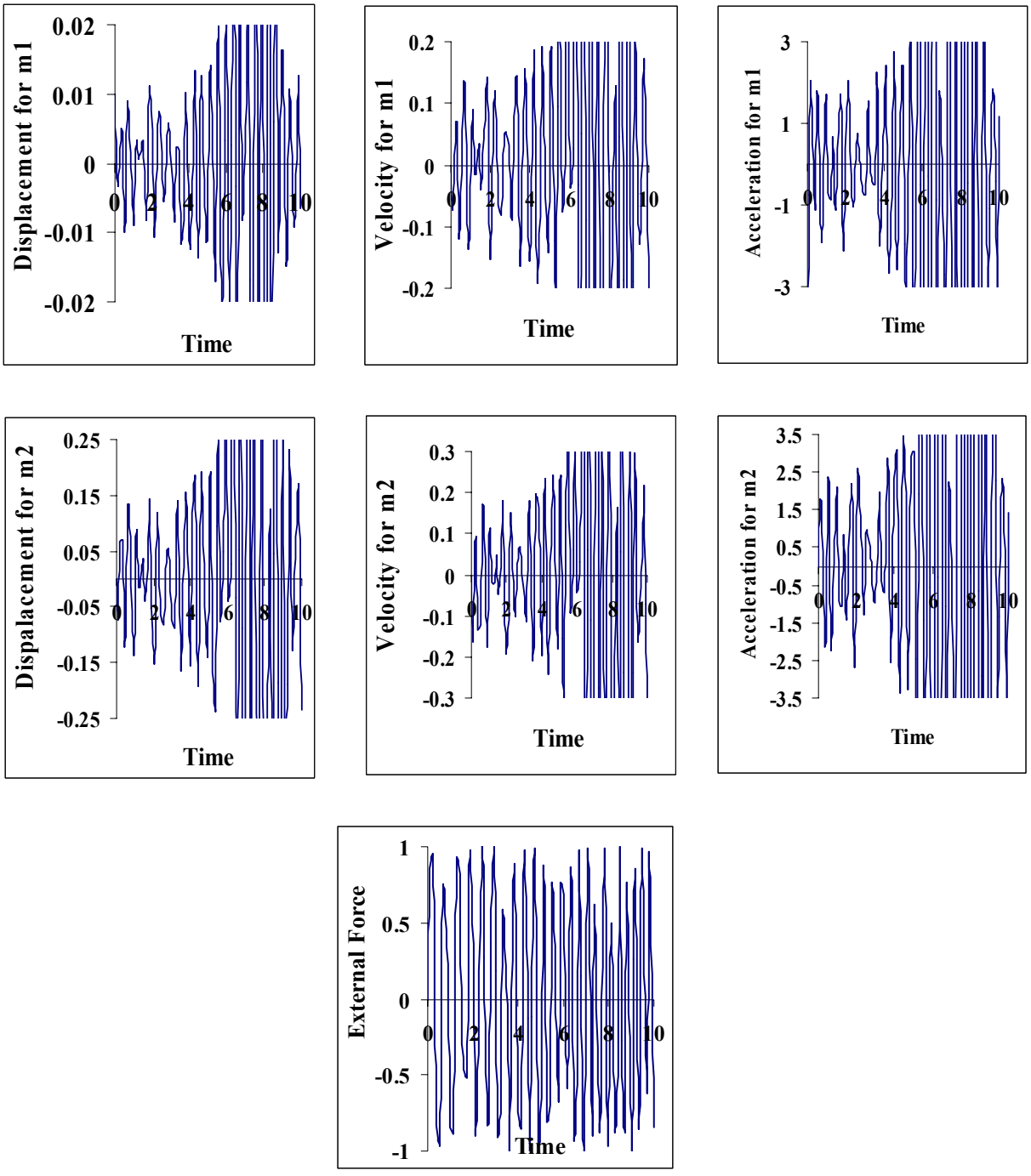


Figure 6.3(d): Time history plots for equation 6.13

6.4 Training Algorithm

In this section, we will first illustrate the training process for traditional *ANN*. Then the training algorithm for *RBNN* model that has been proposed here, will be introduced.

6.4.1 The Traditional *ANN* Model

Three-layer architecture for Neural Network is considered to model the partially known system. The input layer consists of two inputs, namely displacement x_1, x_2 measurements and the output layer consists of two outputs in the form of restoring forces $g(x_1, \dot{x}_1)$ and $g(x_2, \dot{x}_2)$, which can be considered as an approximation to the actual restoring force of the system. The different number of nodes in the hidden layer has been considered to facilitate a comparative study on the architecture of the network. Again, three-layer architecture for Neural Network is considered to model the completely unknown system. The input layer consists of three inputs, namely displacement x_1, x_2 and excitation of the system $F(t)$ measurements and the output layer consists of two outputs in the form of acceleration $\ddot{x}_1$ and $\ddot{x}_2$ of the system. The training algorithm is based on the feedforward back propagation algorithm. The connection weights interconnecting the neurons between different layers are taken through a random number generator built-in function in the *NNT*.

6.4.2 The *RBNN* Training Algorithm

Here the *RBNN* training algorithm has been described for two input and two outputs and the same idea may easily be extended for multi input multi output systems. Accordingly, let us take *N* training patterns as

$$\{(x_1, y_1, R_1, S_1); (x_2, y_2, R_2, S_2); ...; (x_N, y_N, R_N, S_N)\}.$$

These patterns $\{(x_i, y_i, R_i, S_i), i = 1,2,...,N\}$ may, in general, consist of the vector quantities. However, as described above in the present Chapter, we have considered a two input two output problem. We denote the input and output of the given patterns by ***x, y*** and ***R***, ***S*** respectively. The neural network as given in Figure 6.4 is considered here for the development of the training algorithm. Ten neurons in the hidden layer has been taken to explain the training methodology.

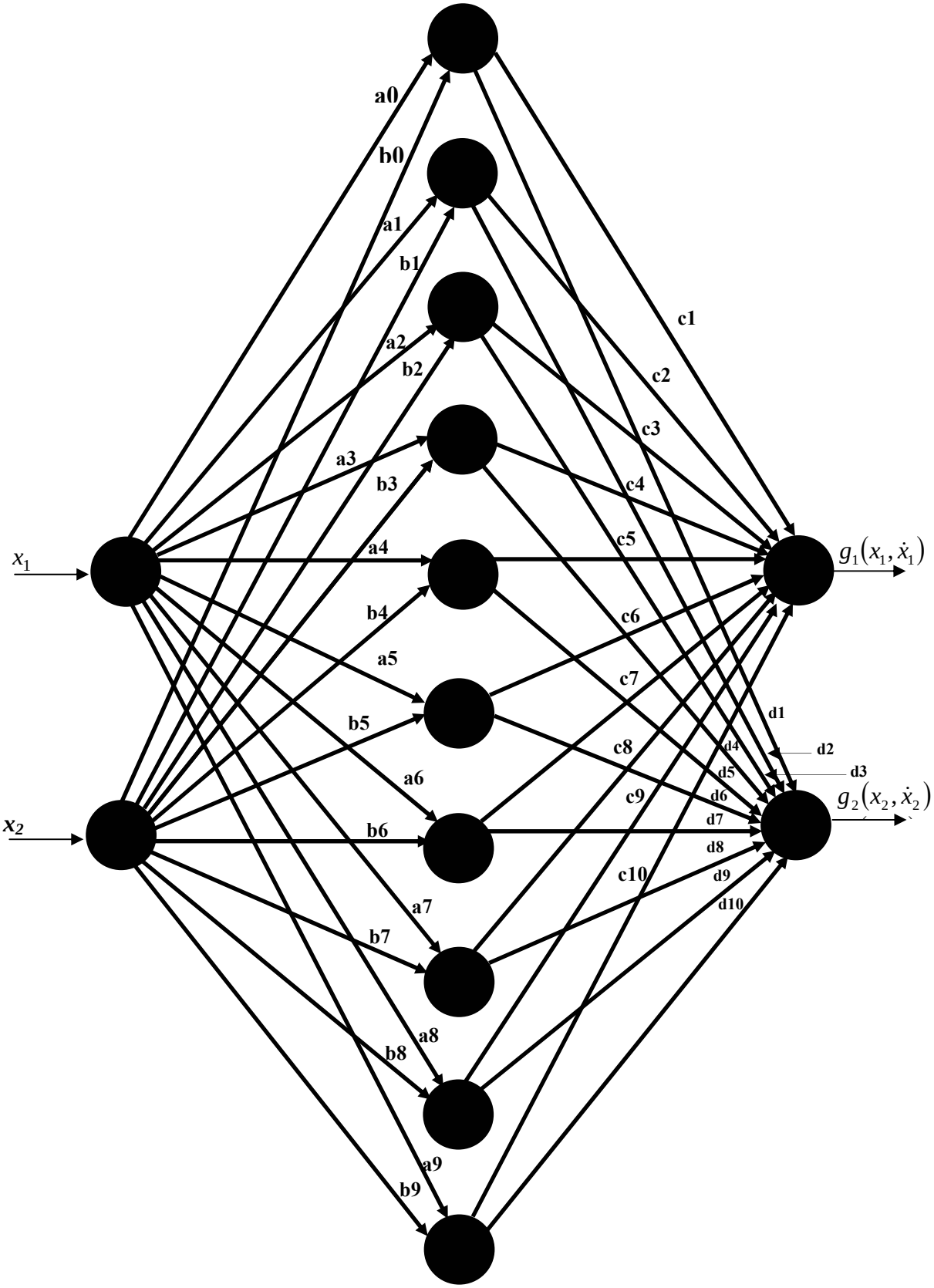


Figure 6.4: *ANN* architecture used to identify multi degree of freedom system

Let a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8 and a_9 are the coefficients of the multivariate regression polynomial fitted between input variables and the first out put by the following equation

$$p(x,y) = a_0 + a_1x + a_2y + a_3xy + a_4x^2 + a_5y^2 + a_6x^3 + a_7y^3 + a_8x^2y + a_9xy^2$$

... (6.14)

and b_0, b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8 and b_9 are the coefficients of the multivariate regression polynomial fitted between input variables and the second out put by the equation

$$q(x,y) = b_0 + b_1x + b_2y + b_3xy + b_4x^2 + b_5y^2 + b_6x^3 + b_7y^3 + b_8x^2y + b_9xy^2$$

... (6.15)

utilizing the given *N* training patterns.

Now we define the weight matrix $\boldsymbol{W}$ from the input to hidden layer as,

$$\boldsymbol{W} = \begin{bmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \\ w_{3,1} & w_{3,2} \\ w_{4,1} & w_{4,2} \\ w_{5,1} & w_{5,2} \\ w_{6,1} & w_{6,2} \\ w_{7,1} & w_{7,2} \\ w_{8,1} & w_{8,2} \\ w_{9,1} & w_{9,2} \\ w_{10,1} & w_{10,2} \end{bmatrix} = \begin{bmatrix} a_0 & b_0 \\ a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \\ a_4 & b_4 \\ a_5 & b_5 \\ a_6 & b_6 \\ a_7 & b_7 \\ a_8 & b_8 \\ a_9 & b_9 \end{bmatrix}$$

described, the system identification problems for the partially known and completely unknown systems we are having either the input variables two or three as shown in Figure 6.4. We discuss here the methodology for muti input and multi output systems *viz.* with two inputs and two outputs only. Inputs are the displacements x_1 and x_2 for two degree of freedom system and corresponding two outputs are the restoring forces g_1 and g_2. However in the following discussion the methodology will be addressed in general by denoting the input variable x_1 and x_2 as x and y and output variable g_1 and g_2 as R and S.

Next, we compute the outputs of the neurons of hidden layer using the following activation functions,

$$p_1^i = \frac{1}{1+e^{-\alpha\,(a_o + b_o)}}, \qquad i = 1, 2, 3, ..., N \qquad \ldots (6.16)$$

$$p_2^i = \frac{1}{1+e^{-\alpha\,(a_1 x_i + b_1 x_i)}}, \qquad i = 1, 2, 3, ..., N \qquad \ldots (6.17)$$

$$p_3^i = \frac{1}{1+e^{-\alpha\,(a_2 y_i + b_2 y_i)}}, \qquad i = 1, 2, 3, ..., N \qquad \ldots (6.18)$$

$$p_4^i = \frac{1}{1+e^{-\alpha\,(a_3 x_i y_i + b_3 x_i y_i)}}, \qquad i = 1, 2, 3, ..., N \qquad ... (6.19)$$

$$p_5^i = \frac{1}{1+e^{-\alpha\,(a_4 x_i^2 + b_4 x_i^2)}}, \quad i = 1, 2, 3, ..., N \qquad ... (6.20)$$

$$p_6^i = \frac{1}{1+e^{-\alpha\,(a_5 y_i^2 + b_5 y_i^2)}}, \quad i = 1, 2, 3, ..., N \qquad ... (6.21)$$

$$p_7^i = \frac{1}{1+e^{-\alpha\,(a_6 x_i^3 + b_6 x_i^3)}}, \quad i = 1, 2, 3, ..., N \qquad ... (6.22)$$

$$p_8^i = \frac{1}{1+e^{-\alpha\,(a_7 y_i^3 + b_7 y_i^3)}}, \quad i = 1, 2, 3, ..., N \qquad ... (6.23)$$

$$p_9^i = \frac{1}{1+e^{-\alpha\,(a_8 x_i^2 y_i + b_8 x_i^2 y_i)}}, \quad i = 1, 2, 3, ..., N \qquad ... (6.24)$$

$$p_{10}^i = \frac{1}{1+e^{-\alpha\,(a_9 x_i y_i^2 + b_9 x_i y_i^2)}}, \quad i = 1, 2, 3, ..., N \qquad ... (6.25)$$

where α is again the training parameter to be chosen suitably. Now, the Weight matrix $\boldsymbol{V}$ from the hidden to output layer is defined as

$$\boldsymbol{V} = \begin{bmatrix} v_{1,1} & v_{2,1} \\ v_{1,2} & v_{2,2} \\ v_{1,3} & v_{2,3} \\ v_{1,4} & v_{2,4} \\ v_{1,5} & v_{2,5} \\ v_{1,6} & v_{2,6} \\ v_{1,7} & v_{2,7} \\ v_{1,8} & v_{2,8} \\ v_{1,9} & v_{2,9} \\ v_{1,10} & v_{2,10} \end{bmatrix} = \begin{bmatrix} c_1 & d_1 \\ c_2 & d_2 \\ c_3 & d_3 \\ c_4 & d_4 \\ c_5 & d_5 \\ c_6 & d_6 \\ c_7 & d_7 \\ c_8 & d_8 \\ c_9 & d_9 \\ c_{10} & d_{10} \end{bmatrix}$$

where c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8, c_9 and c_{10} are the coefficients of the multivariate linear regression polynomial equation 6.26 and $d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8, d_9$ and d_{10} are the coefficients of the multivariate linear regression polynomial equation 6.27 as given below.

$$c_1 p_1^i + c_2 p_2^i + c_3 p_3^i + c_4 p_4^i + c_5 p_5^i + c_6 p_6^i + c_7 p_7^i + c_8 p_8^i + c_9 p_9^i + c_{10} p_{10}^i \quad \ldots (6.26)$$

$$d_1 p_1^i + d_2 p_2^i + d_3 p_3^i + d_4 p_4^i + d_5 p_5^i + d_6 p_6^i + d_7 p_7^i + d_8 p_8^i + d_9 p_9^i + d_{10} p_{10}^i \quad \ldots (6.27)$$

The above are fitted to the data set

$$\{(p_1^i, p_2^i, p_3^i, p_4^i, p_5^i, p_6^i, p_7^i, p_8^i, p_9^i, p_{10}^i, \quad R_i): \quad i = 1, 2, \ldots, N\}$$

$$\{(p_1^i, p_2^i, p_3^i, p_4^i, p_5^i, p_6^i, p_7^i, p_8^i, p_9^i, p_{10}^i, \quad S_i): \quad i = 1, 2, \ldots, N\}$$

Output of the network is now calculated as,

$$o_1 = c_1 p_1^i + c_2 p_2^i + c_3 p_3^i + c_4 p_4^i + c_5 p_5^i + c_6 p_6^i + c_7 p_7^i + c_8 p_8^i + c_9 p_9^i + c_{10} p_{10}^i \quad \ldots (6.28)$$

$$o_2 = d_1 p_1^i + d_2 p_2^i + d_3 p_3^i + d_4 p_4^i + d_5 p_5^i + d_6 p_6^i + d_7 p_7^i + d_8 p_8^i + d_9 p_9^i + d_{10} p_{10}^i \quad \ldots (6.29)$$

The errors associated with this *RBNN* model is next defined as,

$$E_1 = \frac{1}{2} \sum_{i=1}^{N} (o_{1i} - R_i)^2 \quad \ldots (6.30)$$

$$E_2 = \frac{1}{2}\sum_{i=1}^{N}(o_{2i} - S_i)^2 \qquad \text{... (6.31)}$$

This error E_1 and E_2 is again used to train the network as described below by redefining the elements of $\boldsymbol{W}$ and $\boldsymbol{V}$ according to the following process.

$$E_1 = \frac{1}{2}\sum_{i=1}^{N}(c_1 p_1^i + c_2 p_2^i + c_3 p_3^i + c_4 p_4^i + c_5 p_5^i + c_6 p_6^i + c_7 p_7^i + c_8 p_8^i + c_9 p_9^i + c_{10} p_{10}^i - R_i)^2$$

Substituting the expressions of p_j^i , j=1,...10 from equations 6.16 to 6.25 in the above we have

$$E_1 = \frac{1}{2}\sum_{i=1}^{N}\left[\frac{c_1}{1+e^{-\alpha(a_0+b_0)}} + \frac{c_2}{1+e^{-\alpha(a_1x_i+b_1x_i)}} + \frac{c_3}{1+e^{-\alpha(a_2y_i+b_2y_i)}} + \frac{c_4}{1+e^{-\alpha(a_3xy+b_3xy)}} + \frac{c_5}{1+e^{-\alpha(a_4x_i^2+b_4x_i^2)}} + \frac{c_6}{1+e^{-\alpha(a_5y_i^2+b_5y_i^2)}} + \frac{c_7}{1+e^{-\alpha(a_6x_i^3+b_6x_i^3)}} + \frac{c_8}{1+e^{-\alpha(a_7y_i^3+b_7y_i^3)}} + \frac{c_9}{1+e^{-\alpha(a_8x_i^2y_i+b_8x_i^2y_i)}} + \frac{c_{10}}{1+e^{-\alpha(a_9x_iy_i^2+b_9x_iy_i^2)}} - R_i\right]^2$$

Similarly the error E_2 is obtained as

$$E_2 = \frac{1}{2}\sum_{i=1}^{N}(d_1 p_1^i + d_2 p_2^i + d_3 p_3^i + d_4 p_4^i + d_5 p_5^i + d_6 p_6^i + d_7 p_7^i + d_8 p_8^i + d_9 p_9^i + d_{10} p_{10}^i - S_i)^2$$

$$E_2 = \frac{1}{2}\sum_{i=1}^{N}\left[\frac{d_1}{1+e^{-\alpha(a_0+b_0)}} + \frac{d_2}{1+e^{-\alpha(a_1x_i+b_1x_i)}} + \frac{d_3}{1+e^{-\alpha(a_2y_i+b_2y_i)}} + \frac{d_4}{1+e^{-\alpha(a_3xy+b_3xy)}} + \frac{d_5}{1+e^{-\alpha(a_4x_i^2+b_4x_i^2)}} + \frac{d_6}{1+e^{-\alpha(a_5y_i^2+b_5y_i^2)}} + \frac{d_7}{1+e^{-\alpha(a_6x_i^3+b_6x_i^3)}} + \frac{d_8}{1+e^{-\alpha(a_7y_i^3+b_7y_i^3)}} + \frac{d_9}{1+e^{-\alpha(a_8x_i^2y_i+b_8x_i^2y_i)}} + \frac{d_{10}}{1+e^{-\alpha(a_9x_iy_i^2+b_9x_iy_i^2)}} - S_i\right]^2$$

The partial derivatives of E_1 and E_2 w. r. t. a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8 and a_9 and b_0, b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8 and b_9 may easily be calculated as explained in Chapter III and IV. Using these partial derivates, the weights matrix $\boldsymbol{W}$ is now redefined as,

$$W = \begin{bmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \\ w_{3,1} & w_{3,2} \\ w_{4,1} & w_{4,2} \\ w_{5,1} & w_{5,2} \\ w_{6,1} & w_{6,2} \\ w_{7,1} & w_{7,2} \\ w_{8,1} & w_{8,2} \\ w_{9,1} & w_{9,2} \\ w_{10,1} & w_{10,1} \end{bmatrix} = \begin{bmatrix} a_0 - \frac{\partial E_1}{\partial a_0} & b_0 - \frac{\partial E_1}{\partial b_0} \\ a_1 - \frac{\partial E_1}{\partial a_1} & b_1 - \frac{\partial E_1}{\partial b_1} \\ a_2 - \frac{\partial E_1}{\partial a_2} & b_2 - \frac{\partial E_1}{\partial b_2} \\ a_3 - \frac{\partial E_1}{\partial a_3} & b_3 - \frac{\partial E_1}{\partial b_3} \\ a_4 - \frac{\partial E_1}{\partial a_4} & b_4 - \frac{\partial E_1}{\partial b_4} \\ a_5 - \frac{\partial E_1}{\partial a_5} & b_5 - \frac{\partial E_1}{\partial b_5} \\ a_6 - \frac{\partial E_1}{\partial a_6} & b_6 - \frac{\partial E_1}{\partial b_6} \\ a_7 - \frac{\partial E_1}{\partial a_7} & b_7 - \frac{\partial E_1}{\partial b_7} \\ a_8 - \frac{\partial E_1}{\partial a_8} & b_8 - \frac{\partial E_1}{\partial b_8} \\ a_9 - \frac{\partial E_1}{\partial a_9} & b_9 - \frac{\partial E_1}{\partial b_9} \end{bmatrix} \quad \dots (6.32)$$

Using the matrix ***W*** as defined above, we can calculate the new outputs of the hidden neurons using equations 6.16 to 6.25.

Now, the weight matrix ***V*** is computed again by fitting the multivariate linear regression polynomials as given in equation 6.26 and 6.27 with the new outputs of the hidden layer. Utilizing the present new ***V***, the output of the network can again be calculated with the help of equations 6.28 and 6.29. This training procedure is repeated until we get the desired tolerance level of error E_1 and E_2 and thus the *RBNN* model is said to be learnt from the given training patterns.

Similar to above, models for three inputs, two outputs or in general muti input and multi output systems may easily be written for particular problems.

6.4.3 The Experiment

The procedure described in subsection 6.4.1 and 6.4.2 for training the traditional *ANN* model and proposed *RBNN* model has been implemented in the MatLab environment for partially known system and completely unknown system. In the traditional model, the output of the network is computed by built-in transfer functions, namely, tan-sigmoid (tansig) and linear (purelin) of the *NNT* performed at two stages *i.e.*, the stage of hidden layer and the stage of output layer. Four cases of the number of nodes in the hidden layer have been considered to facilitate a comparative study on the architecture of the network. The number of nodes in the hidden layer has been considered as 5, 10, 15 and 20 in these cases. The neural network based on this feedforward back propagation algorithm has been trained with Levenberg-Marquardt training function of the *NNT*.

In proposed *RBNN* model, regression polynomials of degree three are fitted to the training patterns. The coefficients of this polynomial are taken as the connecting weights for the hidden layer as described earlier. The output of the neurons in the hidden layer is calculated using activation function defined in equations 6.16 to 6.25. These values are used to fit multivariate regression polynomials of the form equation 6.26 and 6.27 and we can calculate the output of the network using equation 6.28 and 6.29 for ten neurons in the hidden layer. At this stage, the error of the *RBNN* model is calculated and a decision is taken whether the network has been trained or not. If the tolerance level of the error is not achieved, we redefine the connection weights $\boldsymbol{W}$ according to equation 6.32 and repeat the procedure otherwise we say that the network has got trained.

In this case the network has been converged with the desired accuracy as shown in the Figures 6.5(a) to 6.5(d) for the problems defined by equations 6.10 to 6.13. The output of the network, $g_1(x,\dot{x})$, $g_2(x,\dot{x})$ and the *MSE* between neural and desired output is calculated. In these Figures, *RK* represents the values of the displacement, velocity, acceleration and restoring force computed by numerical methods, *ANN* represents these values obtained by the traditional *ANN* models, *RBNN* represents the values of these parameters obtained from the proposed model with ten nodes in the hidden layer. The performance of the proposed model is given in the Figures 6.5(a) to 6.5(d) for the partially know system and completely unknown system.

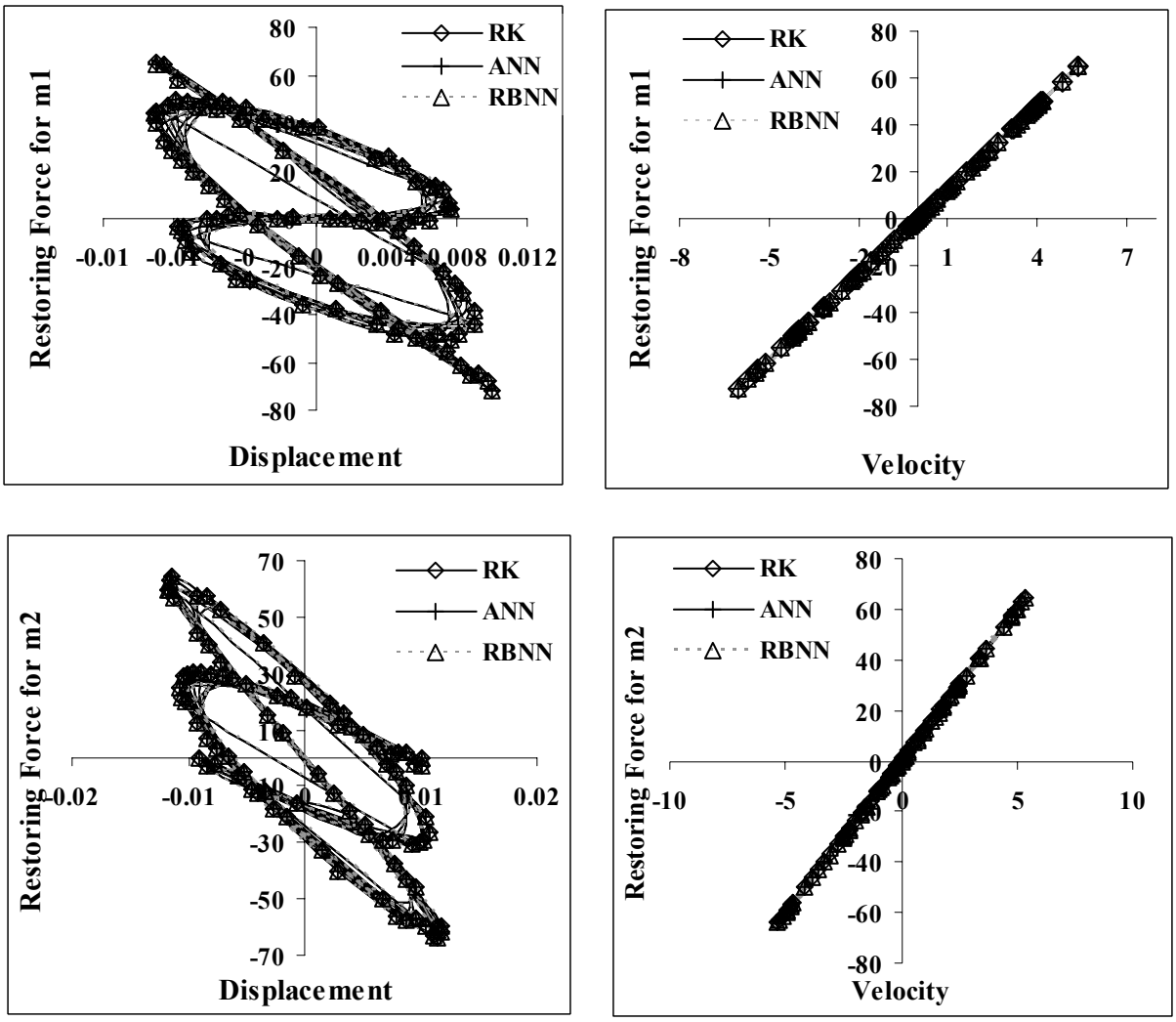


Figure 6.5(a): Identification Results for Training of *RBNN* and *ANN* model for Equation 6.10

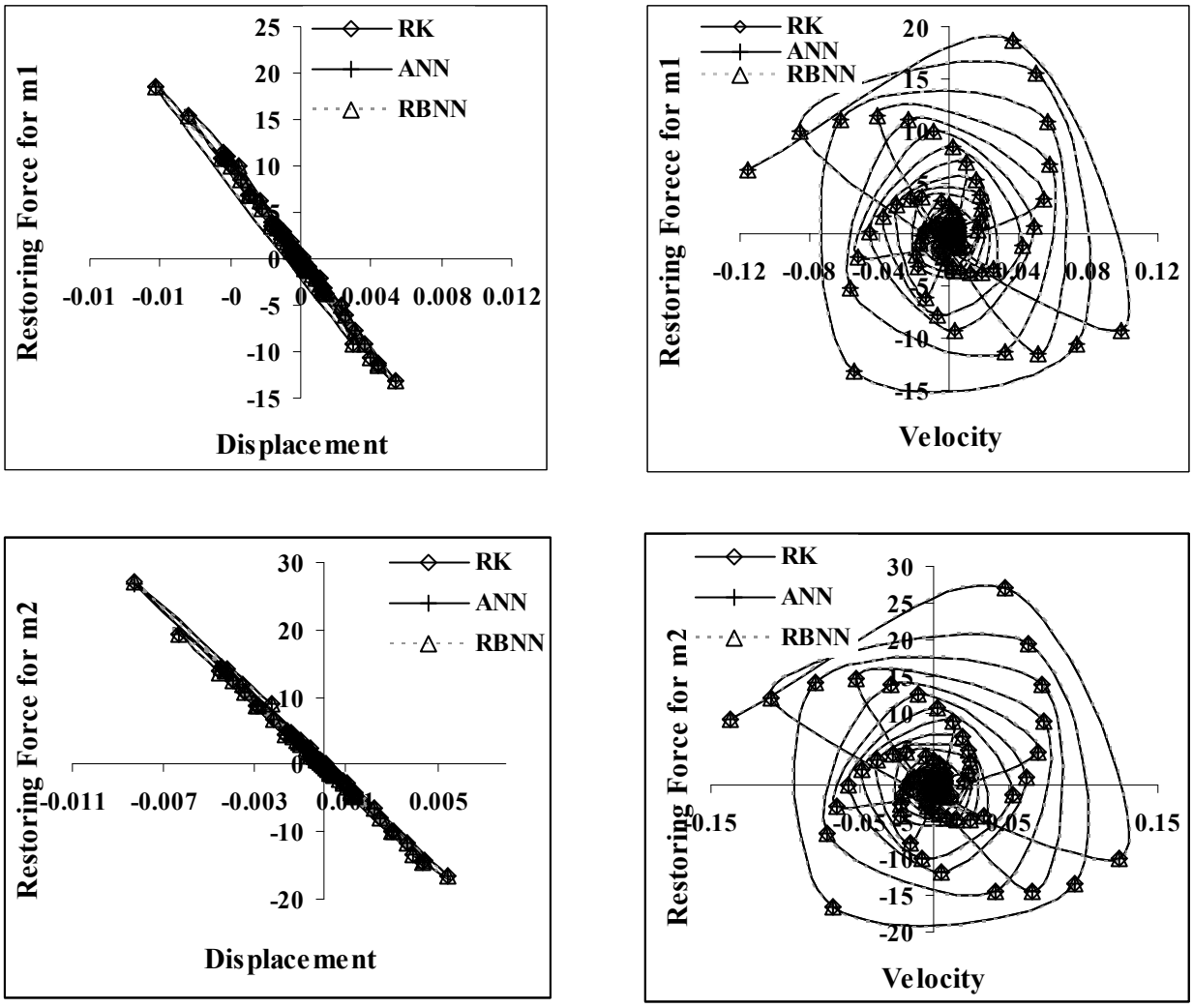


Figure 6.5(b): Identification Results for Training of *RBNN* and *ANN* model for Equation 6.11

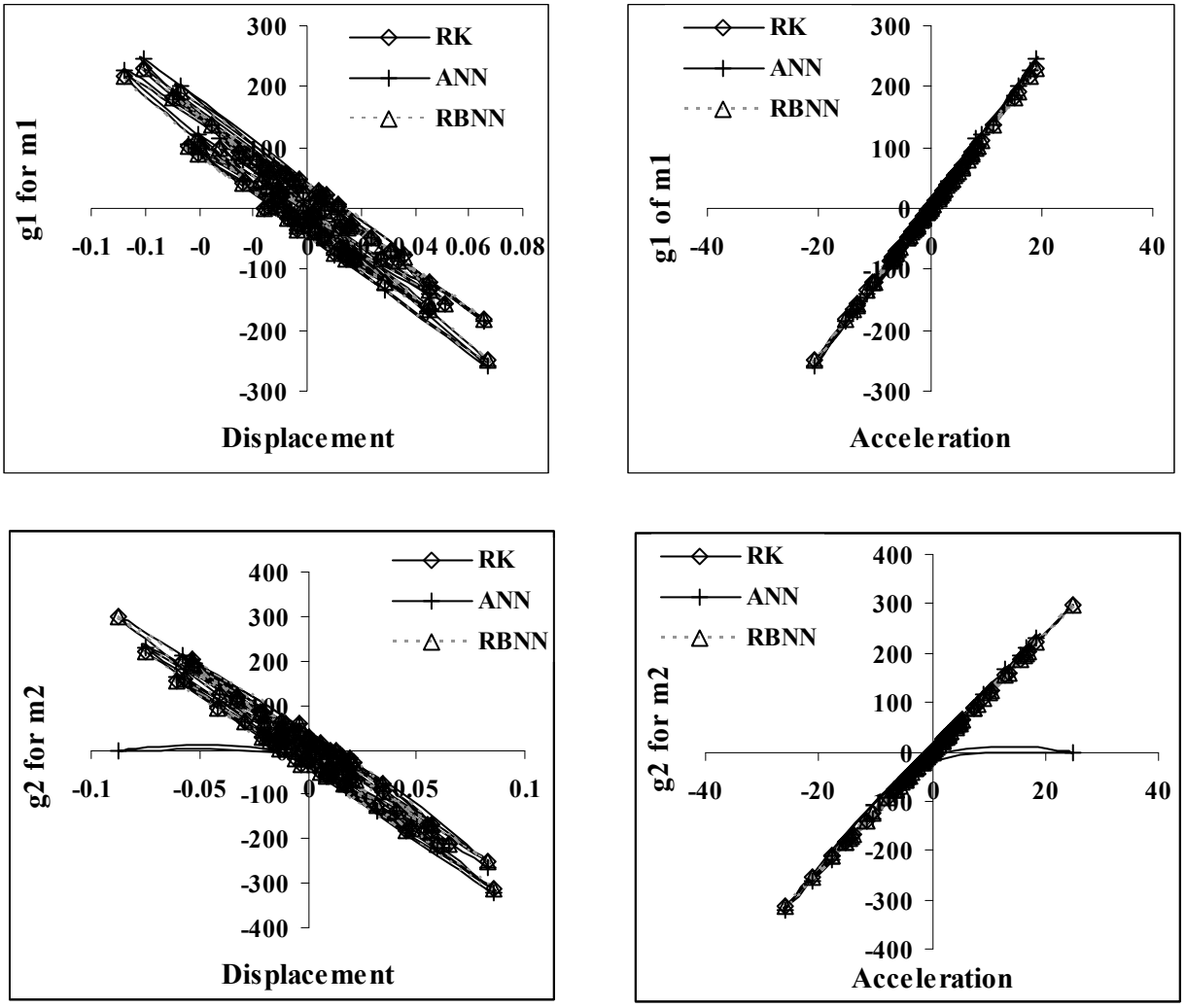


Figure 6.5(c): Identification Results for Training of *RBNN* and *ANN* model for Equation 6.12

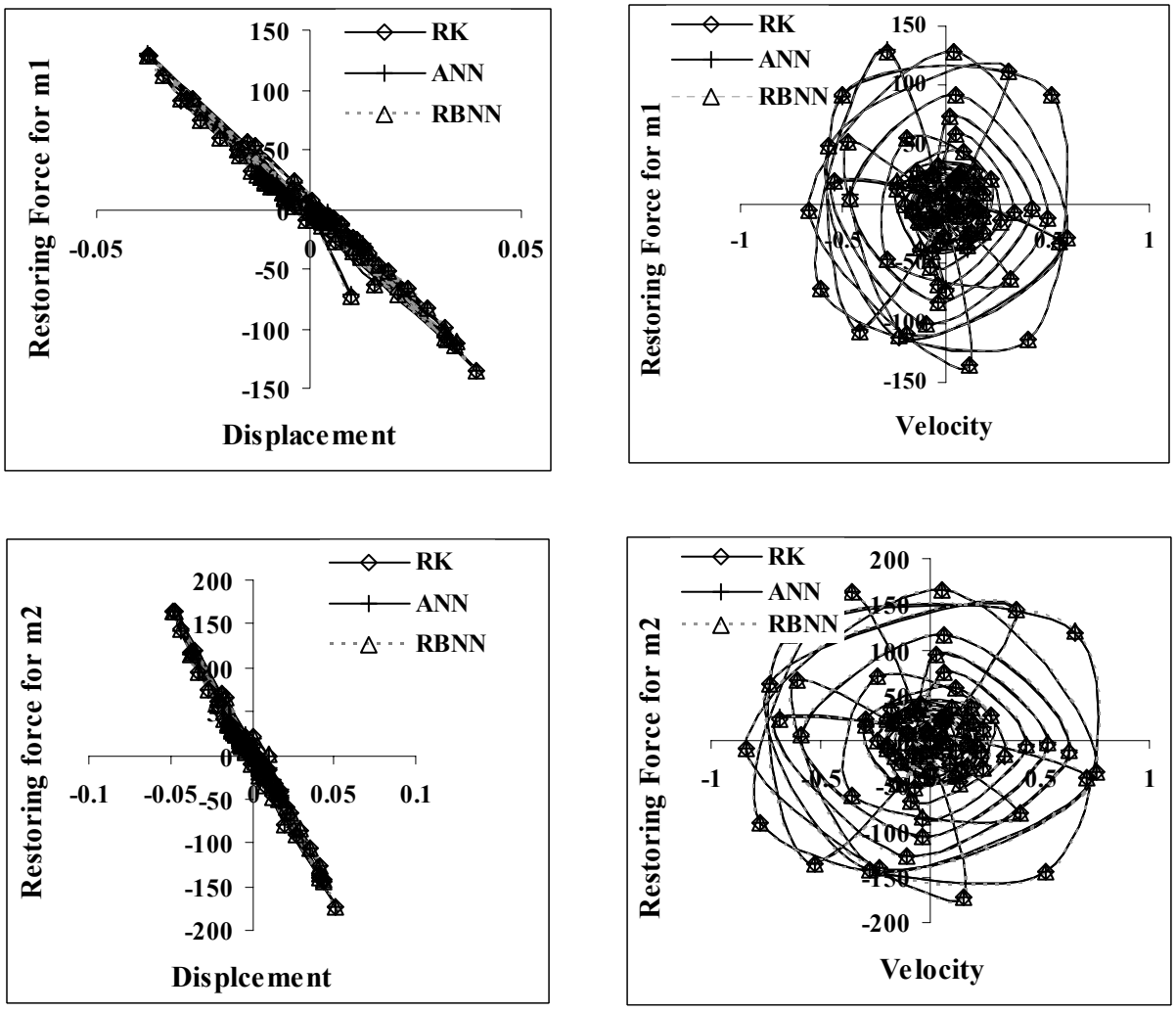


Figure 6.5(d): Identification Results for Training of *RBNN* and *ANN* model for Equation 6.13

6.5 Validation Phase

In the validation phase, the identifying ability of the network has been analysed. The patterns for testing, i.e. $x_1(t)$, $x_2(t), g_1(x_1,\dot{x}_1)$ and $g_2(x_2,\dot{x}_2)$ for partially known system and $x_1(t)$, $x_2(t), F(t)\ \ddot{x}_1(t)$ and $\ddot{x}_2(t)$ for completely unknown system are again generated numerically with same initial conditions but with different step-size h. The state variable plots of these records obtained by the traditional numerical method and by the proposed *RBNN* method are presented in Figures 6.6 (a) to 6.6(d) for the equations 6.10 to 6.13. It can be noted from these Figures that the *RBNN* model gives the better results when compared with traditional *ANN* model. The *MSE* values for all the four defined problems

have been computed. It has been that the *MSE* values for the traditional *ANN* models are higher as compared to the *RBNN* model.

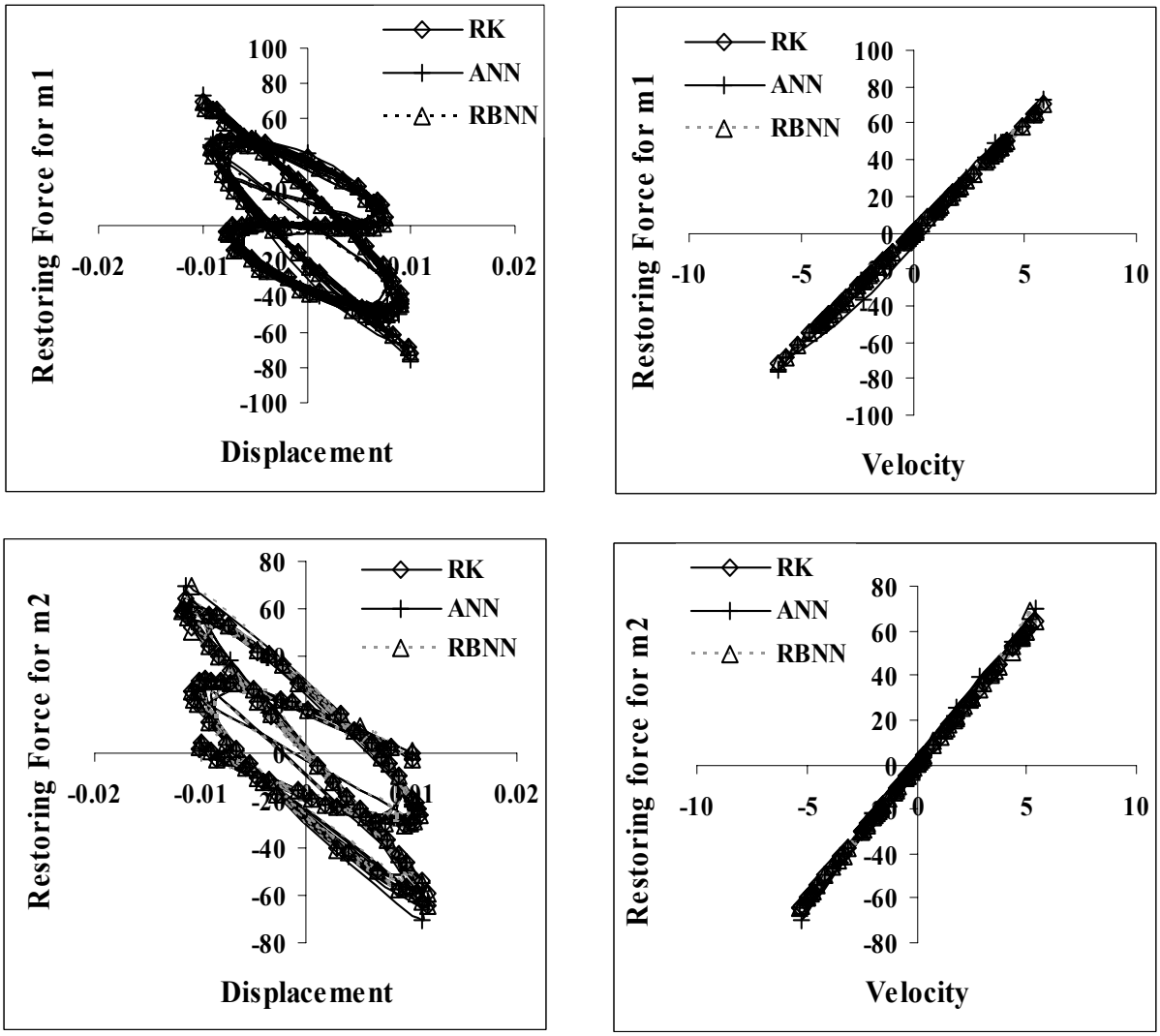


Figure 6.6(a): Identification Results for Validation of *RBNN* and *ANN* model for Equation 6.10

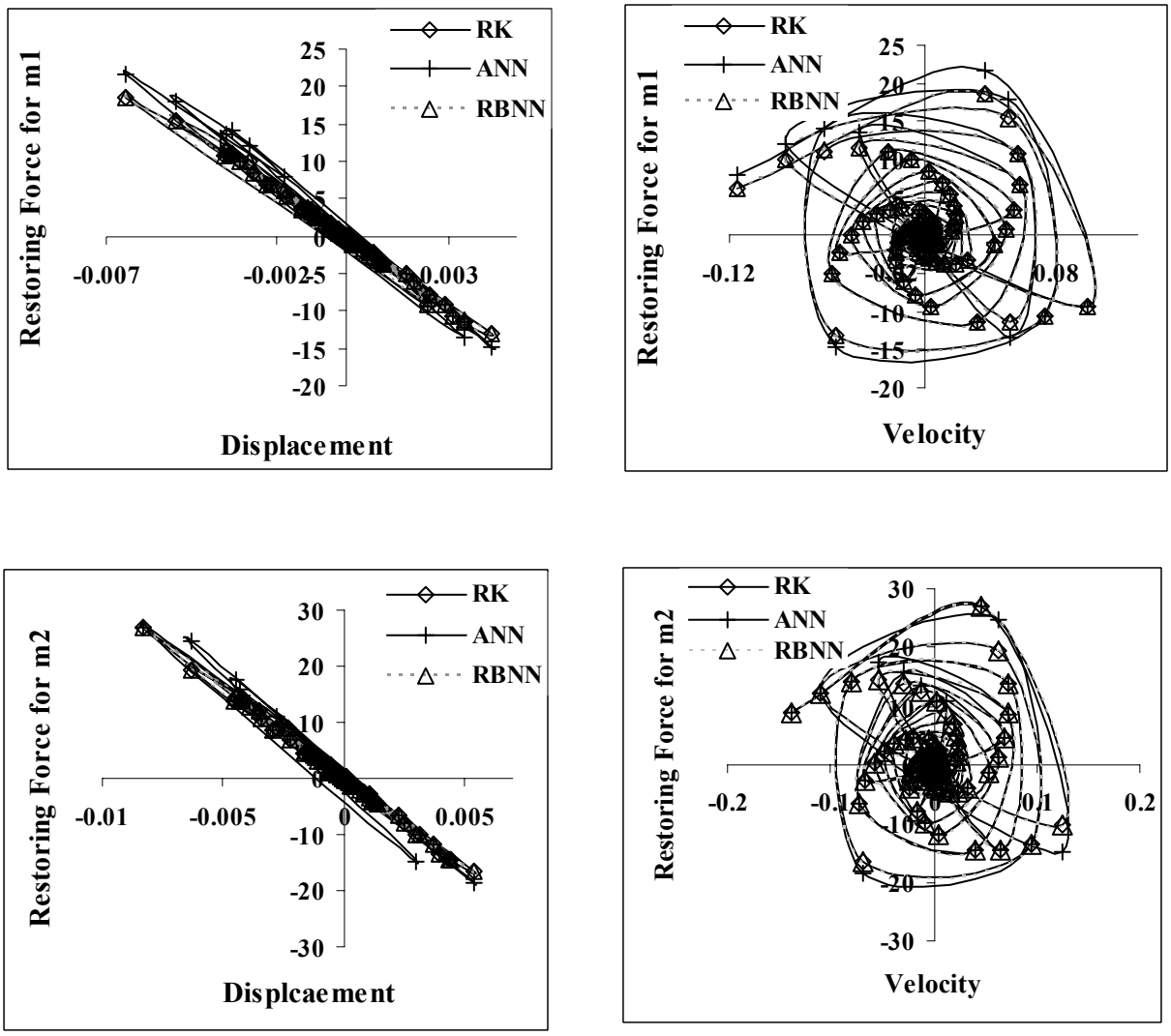


Figure 6.6(b): Identification Results for Validation of *RBNN* and *ANN* model for Equation 6.11

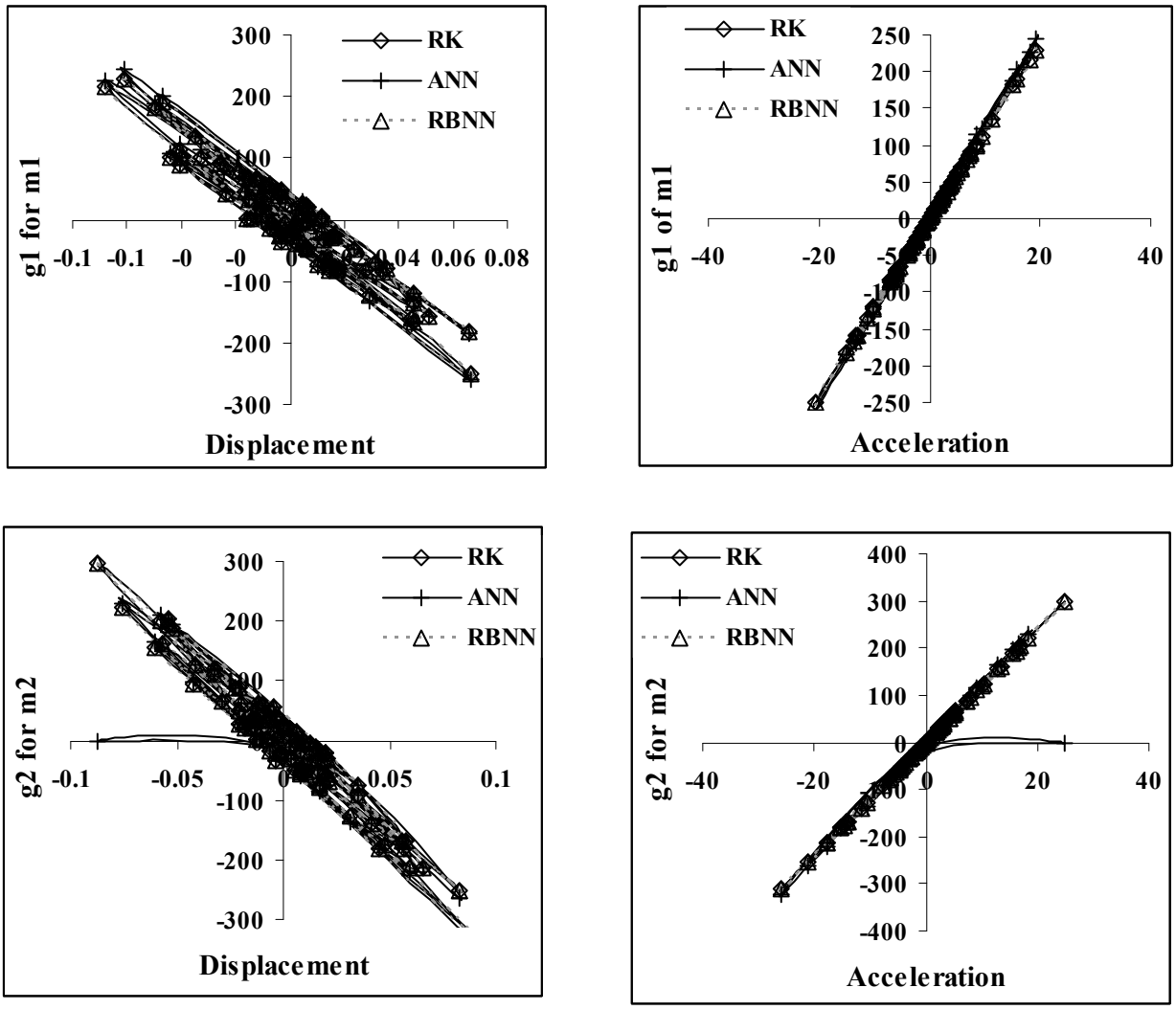


Figure 6.6(c): Identification Results for Validation of *RBNN* and *ANN* model for Equation 6.12

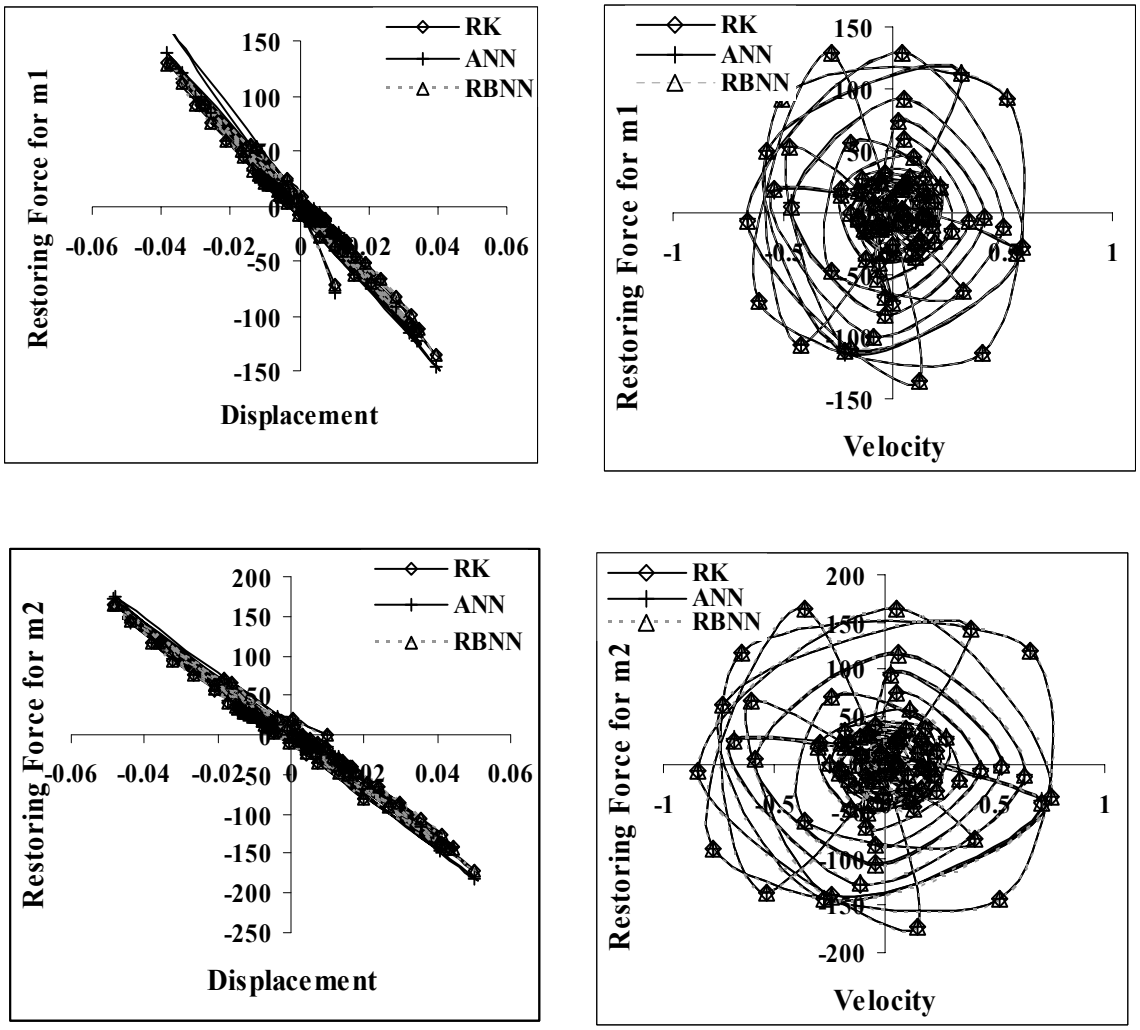


Figure 6.6(c): Identification Results for Validation of *RBNN* and *ANN* model for Equation 6.13

6.6 Error Analysis

In this section, the effect of changing the number of neurons in the hidden layer on the error between desired and neuron output has been studied for the traditional *ANN* model. Four cases, namely, numbers of neurons equal to 5, 10, 15 and 20 have been considered in the hidden layer. During the training of the network, it is noted that if the number of nodes in the hidden layer is changed, then different number of iterations are needed to have the desired convergence. This is expected because the change in initialization also changes the time and number of iterations to achieve the convergence, which is usual as in other well-known numerical techniques. In order to have a comparison between the iterations that are

carried out to achieve desired accuracy are shown in Figure 6.7 for four cases of the equations 6.10 to 6.13.

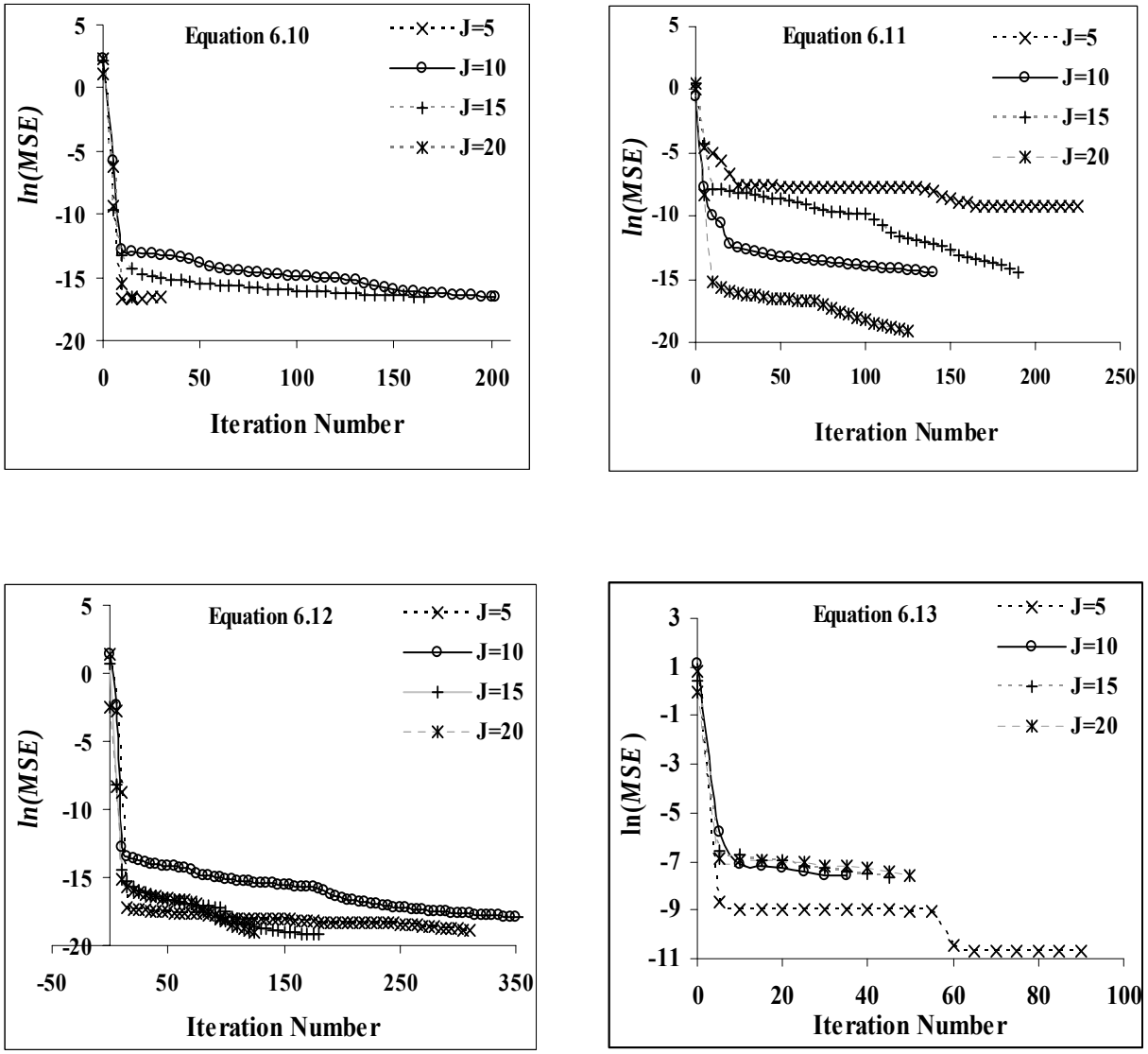


Figure 6.7: Effect of number of nodes in the hidden layer on number of iterations for equations 6.10 to 6.13

It has been noted from Figure 6.7 that number iterations that are carried out to achieve desired accuracy are different for different number of nodes in the hidden. The traditional *ANN* model with ten nodes in the hidden layer has lowest *MSE* values as compared with the other cases of number of nodes in the hidden layer. Another popular measurement as explained in Chapter III, for the prediction capability of a network is the mean square error. The graph for ln(*MSE*) as a function of iteration number is shown in Figure 6.8 for system

equations 6.10 to 6.13. These Figures also give the comparison of *MSEs* for traditional *ANN* model and *RBNN* model. It has also been observed from these Figures that *RBNN* models takes less time in training and have less *MSE* when compared with the traditional *ANN* models.

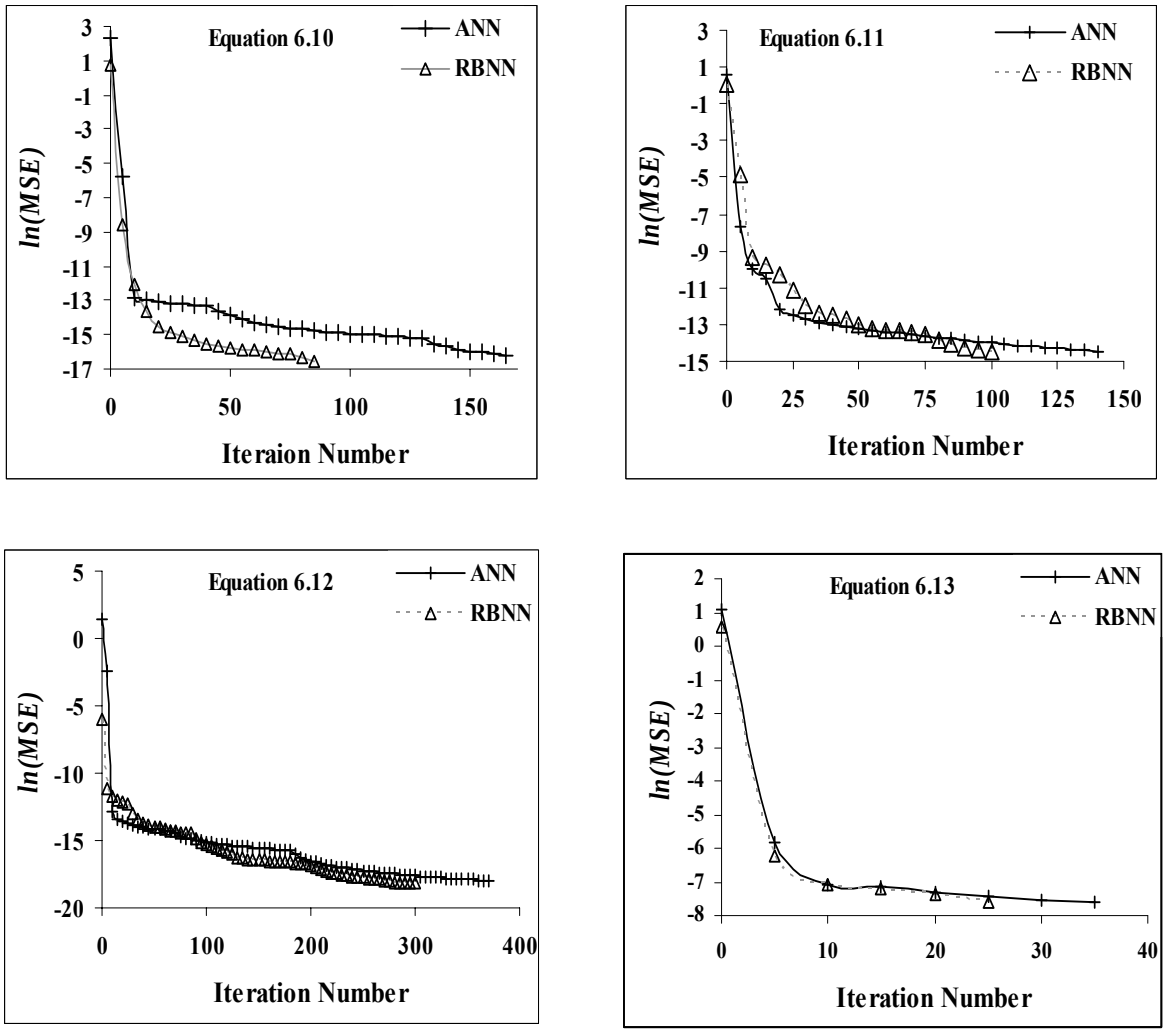


Figure 6.8: Error analysis for the equation 6.10 to 6.13

6.7 Conclusion

Herein, a regression based neural network model has been proposed and simulated for multi-input multi-output artificial neural network and the same is used in multi-degree of freedom identification problems. The testing of this model shows the powerfulness of the soft-computing technique *viz.* the artificial neural network (*ANN*) in the solution of system identification of structural dynamics problems. The Multi Degree of Freedom (*MDF*) system having equation of motion with/without damping and force has been considered for modeling

and identification. Here two problems of identification *viz.* partially known system and completely unknown system have been investigated. In partially known system, the vector of restoring forces $g(x,\dot{x})$ is identified from the input/output data, assuming that the mass vector M is known. On the other hand in completely unknown system, the masses are not known and the system behaviour is modeled on the basis of input/output data only. A three layer architecture for Neural Network is considered to model the partially known system. The input layer consists of two inputs, namely displacement x_1 and x_2 measurements and the output layer consists of two outputs in the form of restoring forces $g(x_1,\dot{x}_1)$ and $g(x_2,\dot{x}_2)$. Similarly three layer architecture for Neural Network is considered to model the completely unknown system too. The input layer consists of three inputs, namely displacements x_1, x_2 and excitation of the system $F(t)$ measurements and the output layer consists of two outputs in the form of accelerations $\ddot{x}_1$ and $\ddot{x}_2$ of the system. Here two different multivariable polynomials are regressed first with each of the input data with the output data for determining the initial weights between input and hidden layer. Then output from the hidden layer is computed as discussed in previous Chapters. Next the initial weight matrix from hidden to output layer was generated by regressing the data of hidden output and actual output of the data considered. This Chapter also includes comparison of identification results between the traditional *ANN* and *RBNN* model. The study investigated the numerical experiments of the four basic and fundamental problems of system identification of structural dynamics using the *ANN* and proposed *RBNN* models.

Bibliography

1. Abraham A, (2004), “Meta learning evolutionary artificial neural network”, Journal of Neurocomputing, 56, pp 1-38.
2. Abraham A, Jain R, Thomas J and Han S Y, (2007), “D-SCIDS: Distributed soft computing intrusion detection system”, Journal of Network and Computer Applications, 30, pp 81–98.
3. Abraham A, Philip N S and Saratchandran P, (2003), “Modeling chaotic behaviour of stock indices using intelligent paradigms”, Neural, Parallel & Scientific Computations, 11(1&2), pp 143-160.
4. Abraham A, Philip S and Mahanti P K, (2004), “Soft computing models for weather forecasting”, Journal of Applied Science and Computations, 11(3), pp 106-117.
5. Ahmed R S, (2001), “Identification of nonlinear dynamic systems using rapid neural network”, Proceedings of 27th Annual Conference of the IEEE Industrial Electronics Society, pp 1734-1739.
6. Amini F, Chen H M, Qi G Z and Yang J C S, (1997), “Generalized neural network based model for structural dynamic identification, analytical, experimental studies”, Proceedings of International Conference on Intelligent Information Systems, IEEE computer Society, pp 138-142.
7. Atkins P A, Wright J R and Worden K, (2000), “Extension of force appropriation to the identification of nonlinear multi-degree of freedom systems”, Journal of Sound and Vibration, 237, pp 23-43.
8. Azvine B, (1999), “Soft computing in intelligent multi-modal Systems”, Proceedings of 1st European Symposium on the Application of Soft Computing in Telecommunication.
9. Baldi P F and Hornik K, (1995), “Learning in linear neural networks: A survey”, IEEE Transactions on Neural Networks, 6(4), pp 837-845.
10. Bani H K, Ghaboussi J and Schneider S P, (1999), “Experimental study of identification and control of structures using neural network Part1: Identification”, Journal of Earthquake Engineering and Structural Dynamics, 28(9), pp 995-1018.
11. Bani H K, Ghaboussi J and Schneider S P, (1999), “Experimental study of identification and control of structures using neural network. Part 2: Control”, Journal of Earthquake Engineering and Structural Dynamics, 28(9), pp 1019-1039.

12. Baruch I S, Flores J M, Thomas F and Gortcheva E, (2001), “A multimodel recurrent neural network for systems identification and control”, Proceedings of INNS-IEEE International Joint Conference on Neural Networks, 2, pp 1292-1296.
13. Bernieri A, DApuzzo M, Sansone L and Savastano M, (1994), “Neural network approach for identification and fault diagnosis on dynamic systems”, IEEE Transactions on Instrumentation and Measurement, 43(6), pp 867-873.
14. Bhat R B, (1985), “Natural frequencies of rectangular plates using characteristic orthogonal polynomials in Rayleigh-Ritz method”, Journal of Sound and Vibration, 102(4), pp 493-499.
15. Biedermann J D, (1997), “Representing design knowledge with neural networks”, Microcomputers in Civil Engineering, 12(4), pp 277-285.
16. Biggs J M, (1964), Introduction to Structural Dynamic" McGram Hill.
17. Bonissone P P and Khedkar P S, (2002), “A hybrid soft computing model to predict time-to-break margins in paper machines”, Proceedings of International conference of Society for Optical Engineering, pp 53-64.
18. Bonissone P P, (1997), “Soft Computing: The convergence of emerging reasoning technologies”, Journal of Research in Soft Computing, 1 (1), pp 6-18.
19. Bonissone P P, Goebal K and Khedkar P S, (1999), “Hybrid soft computing systems: Industrial and Commercial Applications”, Proceedings of IEEE special issues on Computational Intelligence, 87(9), pp 1641-1667.
20. Bonissone P, (2003), “Soft Computing and Meta-heuristics” Proceedings of conference on Applications and Science of Neural Networks, Fuzzy Systems and Evolutionary Computation, 5200, pp 133-149.
21. Campos R, Ruiz F, Agell N and Angulo C, (2004), “Financial credit risk measurement prediction by using innovative soft-computing techniques", Journal of Computational Finance, pp 21-23.
22. Cao X, Sugiyama Y and Mitsui Y, (1998), “Application of artificial neural networks to load identification”, Journal of Computers and Structures, 69(1), pp 63-78.
23. Castellano G, Fanelli A M and Pelillo M., (1997), “Iterative pruning algorithm for feedforward neural networks”, IEEE Transactions on Neural Networks, 8(3), pp 519-531.
24. Chakraverty S, (1992), “Numerical solution of vibration of plates”, Ph.D. Thesis, University of Roorkee (Now I I T, Roorkee).

25. Chakraverty S, (1998), “A new computationally efficient numerical method for analysis of vibration of plate structures”, Journal of Scientific Industrial Research, 57, pp 371.
26. Chakraverty S, Bhat R B and Stiharu I, (1999), “Recent research on vibration of structures using boundary characteristic orthogonal polynomials in the Rayleigh-Ritz method”, Journal of Shock Vibration Digest, 31(3), pp 187-194.
27. Chakraverty S, Sharma R K and Singh V P, (2003), “New soft computing based estimation of frequencies for some machine elements to be used in construction machinery”, National workshop on innovative building construction machinery”, (CONSMACH -2003), CBRI, India.
28. Chakraverty S, Sharma R K and Singh V P, (2003), “Soft-Computing approach for dynamic systems”, Journal of New Building Materials and Construction World, pp 50-56.
29. Chakraverty S, Singh V P and Sharma R K, (2006), “Regression based weight generation algorithm in neural network for estimation of frequencies of vibrating plates” Journal of Computer Methods in Applied Mechanics and Engineering, 195, pp 4194-4202.
30. Chassiakos A G and Masri S F, (1991), “Identification of the internal forces of structural Systems using feedfarward multilayer networks”, Journal of Computing Systems in Engineering, 2, pp 125-134.
31. Chassiakos A G and Masri S F, (1992), “Neural network based identification of structural systems”, Proceedings of conference on parallel and distributed computing in engineering systems, Elsevier Science Publishers, Holland, pp 371-375.
32. Chassiakos A G and Masri S F, (1996), “Modelling unknown structural systems through the use of neural networks”, Journal of Earthquake Engineering and Structural Dynamics, 25, pp 117-128.
33. Chen H.M, Tsai K H, Qi G Z, Yang J C S and Amini F, (1995), “Neural network based structural control”, Journal of Computing in Civil Engineering, 9(2), pp 168-176.
34. Chen M and Popplewell N, (1994), “Neural network for earthquake selection in structural time history analysis”, Journal of Earthquake Engineering and Structural Dynamics, 23, pp 303-319.
35. Chihara T S, (1978), “An Introduction to Orthogonal Polynomials”, Gordon &

Breach.

36. Chinnam R B and Ding J, (1998), “Prediction limit estimation for neural network models", IEEE Transactions on Neural Networks, 9(6), pp 1515-1522.
37. Chopra A K, (2001), “Dynamics of Structures: Theory and applications to Earthquake Engineering”, Prentice Hall, Inc. New Jersay USA, ISBN-81-203-2139.
38. Chryssolouris G, Moshin L and Ramsey A, (1996), “Confidence interval prediction for neural network models", IEEE Transactions on Neural Networks, 7(1), pp 229-232.
39. Chu S R, Shoureshi R and Tenorio M, (1990), “Neural networks for system identification”, IEEE Control Systems Magazine, 10(3), pp 31-35.
40. Cong T, Abraham A and Lakhmi J, (2004), “Decision support systems using hybrid neurocomputing”, Journal of Neurocomputing, 61, pp 85-97.
41. Cordon O and Viedma H E, (2003), “Special issue on Soft Computing applications to intelligent information retrieval on the internet”, Journal of Approximate Reasoning, 34, pp 89-95.
42. Crestani F and Pasi G, (2000), “Soft Computing in information retrieval, studies in fuzziness and soft computing series”, Physica-Verlag, 50.
43. Dai L and Singh M C, (1997), “Analytical and numerical method for solving linear and nonlinear vibration problems”, Journal of Sound and Vibration, 34(21), pp 2709-2731.
44. De Nicolao G and Ferrari-Trecate G, (2001), “Regularization networks: Fast weight calculation via Kalman filtering”, IEEE Transactions on Neural Networks, 12(2), pp 228-235.
45. Desu N B, Deb S K and Dutta A, (2006), “Coupled tuned mass dampers for control of coupled vibrations in asymmetric building”, Journal of Structural Control and Health Monitoring, 13(5), pp 897-916.
46. Dutta A., Deb S K and Medhi M, (2006), “Parametric system identification technique for damage detection of multistoreyed shear building”, Proceedings of 1st European Conference on Earthquake Engineering and Seismology, paper no.61.
47. Efe O M and Kaynak O, (2000), “A comparative study of neural network structures in identification of non linear systems”, Journal of Robotics & Autonomous Systems, 30, pp 221-230.

48. Elkordy M F, Chang K C and Lee G C, (1993), “Neural network trained by analytically simulated damaged states”, Journal of Computing in Civil Engineering, 7(2), pp130-145.
49. Faravelli L and Pisano A A, (1997), “A neural network approach to structure damage assessment”, Proceedings of conference on Intelligent Information systems IIS 97, ISBN 0-8186-8218-3, pp 585-588.
50. Faravelli L and Venini P, (1994), “Active structural control by neural networks”, Journal of Structural Control, 1, pp 79-101.
51. Ge M and Lui E M, (2005), “Structural damage identification using dynamic properties”, Journal of Computers and Structures, 83, pp 2185-2196.
52. Goebel K and Bonissone P, (2001), "Soft computing for diagnostics in equipment service", Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AIEDAM) Journal, 15(4), pp 267-279.
53. Gontkevich V S, (1964), “Natural Vibrations of Plates and Shells”, (A. P. Filippov editor), Kiev : Nauk. Dumka. (Transl. By Lockhead missiles & Space Co.(Sunnyvale,Calif.)).
54. Gupta K C and Li J, (2000), “Robust design optimization with mathematical programming neural networks”, Journal of Computers and Structures, 76(4), pp 507-516.
55. Gupta U S, LaL R and Verma C P, (1986), “Buckling and vibrations of polar orthotropic annular plates of variable thickness”, Journal of Sound and Vibration, 104, pp 357-369.
56. Gupta U S, LaL R and Verma C P, (1987), “Vibrations and Buckling parabolically tappered polar orthotropic annular plates on elastic foundation”, Journal of Pure and Applied Mathematics, 18(3), pp 269-281.
57. Hajela P and Berke L (1991), “Neuro biological computational models in structural analysis and design”, Journal of Computers and Structures, 41(4), pp 657-667.
58. Hamming F G, (1975), “Investigation of natural frequencies of circular plates with mixed boundary conditions”, Air Force Inst. Tech., School of Engg. Rept. No. GAE/MC/75-11.
59. He Y and Wu L, (1998), “Control structural seismic response by self-recurrent neural network”, Journal of Earthquake Engineering and Structural Dynamics, 27, pp 641-648.

60. Huang G B, Chen Y Q and Babri H A, (2000), “Classification ability of single hidden layer feed forward neural network”, IEEE Transactions on Neural Networks, 11(3), pp 799-801.
61. Igelnik B, Tabib A M, and LeClair S R, (2001), “A net with complex weights”, IEEE Transactions on Neural Networks, 12(2), pp 236-249.
62. Janos A, Feil B and Abraham A, (2005), “Computational intelligence in data mining informatics”, Journal of Computing and Informatics, 29(1), pp 3-12.
63. Joga Rao C V and Vijay Kumar, (1963), “On admissible functions of flexural vibrations and buckling of annular of plates”, Journal of Aeronautical Society India, 15(1), pp 1-5.
64. Johns D J, (1975), “Comments on an approximate expression for the fundamental frequency of vibration of elastic plates”, Journal of Sound and Vibration, 41(3), pp 385-387.
65. Jones R, (1975), “An approximate expression for the fundamental frequency of vibration of elastic plates”, Journal of Sound and Vibration, 38(4), pp 503-504.
66. Jordan M I and Bishop C M, (1996), “Neural Networks”, ACM Computing Surveys, 28(1), pp 73-75.
67. Kallassy A, (2003), “A new neural network for response estimation”, Journal of Computers and Structures, 81(26, 27), pp 2417-2429.
68. Kao C Y and Hung S L, (2003), “Detection of structural damage via free vibration responses generated by approximating artificial neural networks”, Journal of Computers and Structures, 81(28-29), pp 2631-2644.
69. Karlik B, Ozkaya E, Aydin S and Pakdemirli M, (1998), “Vibrations of a beam-mass systems using artificial neural networks”, Journal of Computers and Structures, 69(3), pp 339-347.
70. Khan M R and Abraham A, (2003), “Short term load forecasting models in Czech Republic using soft computing techniques”, Journal of Knowledge-Based Intelligent Engineering Systems, 7(4), pp 172-179.
71. Kim C M and Dickinson S M, (1989), “On the free, transverse vibration of annular and circular thin, sectorial plates subject to certain complicating effects”, Journal of Sound and Vibration, 134(3), pp 407-421.
72. Kim Y Y and Kapania R K, (2006), “Neural networks for inverse problems using principal component analysis and orthogonal arrays”, Journal of American Institute of Aeronautics and Astronautics, 44(7), pp 1628-1634.

73. Kortesis S and Panagiotopoulos P D, (1993), “Neural network for computing in structural analysis: Method and prospects of applications”, Journal of Numerical Methods in Engineering, 36, pp 2305-2318.
74. Kosmatopoulos E B, Polycarpou M M, Christodoulou M A and Ioannou P A, (1995), “High-order neural network structures for identification of dynamical systems”, IEEE Transactions on Neural Networks, 6(2), pp 422-431.
75. Kramer M A and Leonard J A, (1990), “Diagnosis using backpropagation neural networks-analysis and criticism", Computers and Chemical Engineering, 14(12), pp 1323-1338.
76. Lagaros N D and Papadrakakis M, (2004), “Learning improvement of neural networks used in structural optimization”, Advances in Engineering Software, 35, pp 9-25.
77. Lagaros N D, Charmpis D C and Papadrakakis M, (2005), “An adaptive neural network strategy for improving the computational performance of evolutionary structural optimization”, Journal of Computer Methods Applied Mechanics and Engineering, 194, pp 3374-3393.
78. Lawrence S and Giles C L, (2000), “Overfitting and neural networks: conjugate gradient and backpropagation”, Proceedings of international joint conference on neural networks, IEEE Computer Society, pp 114-119.
79. Leissa A W, (1967), “Vibration of a simply-supported elliptic plate”, Journal of Sound and Vibration, 6(1), pp 145-148.
80. Leissa A W, (1969), “Vibration of Plates” NASA SP160, U.S. Government Washington, DC.
81. Leissa A W, (1973), “The free Vibration of rectangular plate”, Journal of Sound and Vibration, 31(3), pp 257-293.
82. Leissa A W, (1977), “Recent research in plate vibrations, 1973-1976: classical Theory”, Journal of Shock Vibration Digest, 9(10), pp 13 -24.
83. Leissa A W, (1978), “Recent research in plate vibrations, 1973-1976: complicating effects”, Journal of Shock Vibration Digest, 10(12), pp 21-35.
84. Leissa A W, (1981), “Plate vibration research, 1976-1980: classical Theory”, Journal of Shock Vibration Digest, 13(9), pp 11-12.
85. Leissa A W, (1981), “Plate vibration research, 1976-1980: complicating effects”, Journal of Shock Vibration Digest, 13(10), pp 19-36.
86. Leissa A W, (1987), “Recent studies in plate vibrations, 1981-1985: classical

Theory", Journal of Shock Vibration Digest, 19(2), pp 11-18.

87. Leissa A W, (1987), "Recent studies in plate vibrations, 1981-1985: complicating effects", Journal of Shock Vibration Digest, 19(3), pp 10-24.
88. Leonard Z, (2003), "Hybrid neural network/finite element modeling of wave propagation in infinite domains", Journal of Computers and Structures, 81(8-11), pp 1099-1109.
89. Levin A U and Narendra K S, (1995), "Identification using feed forward networks", Journal of Neural Computation, 7(2), pp 349-357.
90. Li S, Wunsch D C and Geisselmann M G, (2001), "Comparative analysis of regression and artificial neural network models for wind turbine power curve estimation", Journal of Solar Energy Engineering, 123, 327-332.
91. Liang Y C, Zhou C G and Wang Z S, (1997), "Identification of restoring forces in non-linear vibration systems based on neural networks", Journal of Sound and Vibration, 206(1), pp 103-108.
92. Lipmann R A, (1987), "An Introduction to computing with neural nets", IEEE ASSP Magazine, pp 4-21.
93. Liu X, Johnson R, Cheng G, Swift S and Tucker A, (1999), "Soft computing for intelligent data analysis", Proceedings of 18th International Conference of North American Fuzzy Information Processing Society, pp 527-531.
94. Liu Y, Li Y, Liu X and Peng Z, (2003), "Parameters identification and vibration control for modular manipulators", Proceedings of IEEE International conference on Robotics and Automation.
95. Lo S and Basar T, (1998), "Robust nonlinear system identification using neural-network models", IEEE Transactions on Neural Networks, 9(3), pp 407-429.
96. Lopes T A P, Andrade O P and Vianna A L, (1998), "Neural networks on the dynamic models updating", Transactions on Information and communications Technologies, 19, ISSN 1734-3517.
97. Mahmoud M A and Abukiefa M A, (1999), "Neural network solution of the inverse vibration problem", Journal of Non-Destructive Testing and Evaluation International, 32(2) pp 91-99.
98. Marwala T and Hunt H E M, (1999), "Fault identification using finite element models and neural networks", Journal of Mechanical Systems and Signal processing, 13(3), pp 475-490.
99. Marwala T, (2001), "Probabilistic fault identification using a committee of neural

networks and vibration data", American Institute of Aeronautics and Astronautics, Journal of Aircraft, 38(1), pp 138-146.

100. Masri S F and Chassiakos A G, (1979), "A nonparametric identification techniques for nonlinear dynamic problems", Journal of Applied Mechanics, 46, pp 433 -447.
101. Masri S F, Chassiakos A G and Caughey T K, (1992), "Structure-unknown nonlinear dynamics system identification through neural networks," Journal of Smart Materials Structures, 1, pp 45-56.
102. Masri S F, Chassiakos A G and Caughey T K, (1993), "Identification of nonlinear dynamic system using neural networks," Journal of Applied Mechanics, 60, pp 123-133.
103. Meirovitch L, (1980), "Computational Methods in Structural Dynamics" (Sijthoff & Noordhoff, The Netherlands).
104. Mitra S, Pal S K and Mitra P, (2002), "Data Mining in soft computing framework: A Survey", IEEE Transactions on Neural Networks, 13(1), pp 3-14.
105. Molas G L and Yamazaki F, (1995), "Neural networks for quick earthquake damage estimation", Journal of Earthquake Engineering and Structural Dynamics, 24, pp 505-516.
106. Mukherjee A (1997), "Self-organizing neural network for identification of natural modes", Journal of Computing in Civil Engineering, 2(1), pp 74-77.
107. Mukherjee A and Deshpande J M, (1995), "Application of artificial neural networks in structural design expert systems", Journal of Computers and Structures, 54(3), pp 367-375.
108. Mukkamala S, Sung A H and Abraham A, (2005), "Intrusion detection using an ensemble of intelligent paradigms", Journal of Network and Computer Applications" 28, pp 167-182.
109. Muthukumaran P, Bhat R B and Stiharu I, (1998), "Localization of structural vibrations and acoustic radiation through boundary conditioning", J. TCSME, 22, pp 519-532.
110. Nagaraja R S, (1985), "Vibrations of plates of variable cross-section", Ph.D Thesis, IIT Madras.
111. Narendra K S and Parthasarathy K, (1990), "Identification and control of dynamical systems using neural networks", IEEE Transactions on Neural Networks, 1(1), pp 4-27.

112. Narita Y and Leissa A W, (1980), “Natural frequencies of simply-supported circular plates”, Journal of Sound and Vibration, 70(2), pp 221-229.
113. Natke H G, (1982), “Identification of Vibrating Structures”, Spring Publication, berlin.
114. Nayfeh A H, Mook D T, Lobitz D W and Sridhar S, (1978), “Vibrations of nearly annular and circular plates”, Journal of Sound and Vibration, 47(1), pp 75-84.
115. Nikravesh M, Loia V and Azvine B, (2002), “Fuzzy logic and the Internet: World Wide Web and search Engines”, Journal of Soft Computing, 6(5), pp 287-299.
116. Niyogi P and Girosi F, (1999), “Generalization bounds for function approximation from scattered noisy data", Advances in Computational Mathematics, 10(1), pp 51-80.
117. Oh Sang-Hoon, (1997), “Improving the error backpropagation algorithm with a modified error function”, IEEE Transactions on Neural Networks, 8(3), pp 799-803.
118. Olivera J, (1997), " Identification of dynamic system using neural network”, Journal of Architecture and Civil Engineering, 1(4), pp 525-532.
119. Ozkul G A, (1975), “Studies of the vibration and bucking of elliptic membranes and plates”, Ph. D. Thesis, Rutgers University.
120. Packirisamy M, Bhat R and Stiharu I, (1999), “Boundary conditioning technique for structural tuning”, Journal of Sound and Vibration, 220(5), pp 847-859.
121. Papadrakakis M and Lagaros N D, (2003), “Soft computing methodologies for structural optimization”, Journal of Applied Soft Computing, 3(3), pp 283-300.
122. Pardoen G C, (1973), “Static, vibration and bucking analysis of axisymmetric circular plates using finite elements”, Journal of Computers and Structures, 3(2), pp 355-375.
123. Pathak J, Godbole P N and Paul D K, (2001), “ANN based Models: A new solution technology in structural engineering”, Advances in Elastic Vibrations and Smart Structures, ISBN 81-7484-043-5, pp 239-253.
124. Pathak J, Paul D K and Godbole P N, (2004), “Simulation of design earthquake in the Himalayan region using artificial neural network” Proceedings of 13th world conference on earthquake engineering.
125. Prechelt L, (1998), “Automatic early stopping using cross validation: quantifying the criteria", Neural Networks, 11, pp 761-767.
126. Prescott T, (1961), “Applied Elasticity”, Dover Pub. Inc. (originally published by

Longmans, Green & co.)

127. Raju P N, (1962), “Vibrations of annular plates”, Journal of Aeronautical Society India, 14(2), pp 37-52.
128. Reid W P, (1962), “Free vibration of a circular plate”, Journal of Society for Industrial and Applied Mathematics, 10(4), pp 668-674.
129. Ren X M, Rad A B, Chan P T and Lo W L, (2003), “Identification and control of continuous-time nonlinear systems via dynamic networks”, IEEE Transactions Industrial Electronics, 50(3), pp 478-486.
130. Roa M M and Datta T K, (1998), “Use of ANN for linear active control of structures”, Proceedings of Eleventh Symposium on Earthquake Engineering, pp 705-712.
131. Romanelli E and Laura P A A, (1997), “Approximate method for analyzing transverse vibrations of circular, annular plates of non-uniform thickness and a free inner boundary”, Journal of Computers and Structures, 62(4), pp 795-797.
132. Sahu A R, (1998), “Determination of the Change in Bending Frequencies of a Wedge Shape Turbine Blade Due to Small Change in the Radius of Rotating Disc”, Journal of Modeling and Simulation Based Engineering, 2, pp 1176-1182.
133. Sahu A R, (2001), “Effect of small Pre-Twist Change on Frequencies of Torsional Vibrations of a Pre-Twisted Beam of Rectangular Cross-Section” Advances in Elastic Vibrations and Smart Structures, Phoenix Publishing House Pvt. Ltd., New Delhi, ISBN 81-7484-043-5.
134. Sahu A R, Bhargava R R and Gupta A P, (2001), “Advances in Elastic Vibrations and Smart Structures”, Phoenix Publishing House Pvt. Ltd., New Delhi, ISBN 81-7484-043-5.
135. Sarkar D, (1995), “Methods to speed up error back–propagation learning algorithm”, ACM Computing Surveys, 27(4), pp 519-542.
136. Sato K, (1971), “Free Flexural vibrations of an elliptic plate with simply-supported edge”, Journal of Acoustical Society America, 52(3), part 2, pp 919-922.
137. Sato K, (1973), “Free Flexural vibrations of an elliptic plate with free edge”, Journal of Acoustical Society America, 54, pp 547-550.
138. Sato K, (1974), “Free Flexural vibrations of a ring-shaped plate bounded confocal ellipses”, Journal of Acoustical Society America, 56(4), pp 1172-1176.
139. Sharma R K, Singh V P and Chakraverty S, (2003), “Neural Network experiments

on the identification of dynamic systems." Conference on Mathematics & Computer Applications in Engineering & Science, TIET, India.

140. Shi Jonathan J, (2000), "Reducing prediction error by transforming input data for neural networks", Journal of Computing in Civil Engineering, 14(2), pp 109-116.
141. Shibaoka Y, (1965), "On the transverse vibration of an elliptic plate with clamped edge", Journal of Physical Society, 11(7), pp 797-803.
142. Singh B and Chakraverty S, (1991), "Transverse vibration of completely free elliptic and circular plates using orthogonal polynomials in Rayleigh-Ritz method", Journal of Mechanical Sciences, 33(9), pp 741.
143. Singh B and Chakraverty S, (1992 a), "On the use of orthogonal polynomials in Rayleigh-Ritz method for the study of transverse vibration of elliptic plates", Journal of Computers and Structures, 43(3), pp 439.
144. Singh B and Chakraverty S, (1992 b), "Transverse vibration of simply-supported elliptic and circular plates using boundary characteristic orthogonal polynomials in two dimensions", Journal of Sound and Vibration, 152(1), pp 149.
145. Singh B and Chakraverty S, (1993), "Transverse vibration of annular circular and elliptic plates using boundary characteristic orthogonal polynomials in two dimensions", Journal of Sound and Vibration, 162(3), pp 537-546.
146. Singh B and Chakraverty S, (1994), "Boundary characteristic orthogonal polynomials in numerical approximation", Journal of Communications in Numerical Methods in Engineering, 10, pp 1027.
147. Sorwar G and Abraham A, (2004), "DCT based texture classification using soft computing approach", Journal of Computer Science, 17(1), pp 13-23.
148. Srinivas M Andrew H S and Abraham A, (2005), "Intrusion detection using ensemble of (Soft Computing and Hard Computing) intelligent paradigms", Journal of Network and Computer Applications, 28(2), pp 167-182.
149. Stephens J E and VanLuchene R D, (1994), "Integrated assessment of seismic damage in structures", Microcomputers in Civil Engineering, 9, pp119-128.
150. Topping B H V and Bahreinine A, (1997), "Neural Computing for Structural Mechanics", Saxe-cobourg publication, ISBN -1-874672-02-4.
151. Tsukimoto H, (2000), "Extracting rules from trained neural networks", IEEE Transactions on Neural Networks, 11(2), pp 377-389.
152. Tsypkin Y Z, Mason J D, Avedyan E D, Warwick K and Levin I K, (1999), "Neural networks for identification of nonlinear systems under random piecewise

polynomial disturbances", IEEE Transactions on Neural Networks, 10(2), pp 303-312.

153. Vaughn J L and Bershad N J, (1995), "Stochastic convergence analysis of a two-layer perceptron for a system identification model", Proceedings of ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing, 5, pp 3619-3622.
154. Victor M B, Freddy R, Slawomir J N and William H, (2005), "An efficient parameterization of dynamic neural networks for nonlinear system identification", IEEE Transactions on Neural Networks, 16(4}, pp 982-987.
155. Vivoli J and Fillippi P, (1974), "Eigenfrequencies of thin plates and layer potentials", Journal of Acoustical Society America, 55(3), pp 562-567.
156. Waller M D, (1953), "Concerning combined and degenerate vibrations of plates", Acustica, 3, pp 370.
157. Wang Y J and Lin C T, (1998), "Runge-Kutta neural network for identification of dynamical systems in high accuracy", IEEE Transactions on Neural Networks, 9(2), pp 294-307.
158. Wang Z O and Lin C (1997),"Fast learning algorithm of feedfarward neural network and its application to system identification", Acta Automatica Sinica, 23(6), pp 728-735.
159. Warburton G B, (1964), "The Dynamic Behaviour and Structure" Pergamon Press.
160. Wong W O, Yam L H, Li Y Y, Law L Y and Chan K T, (2000), "Vibration analysis of annular plates using mode subtraction method", Journal of Sound and Vibration, 232(4), pp 807-822.
161. Worden K, Ball A and Tomlison G, (1994), "Neural network for fault locations", Proceedings of the Eleventh International Modal Analysis Conference, pp 47-54.
162. Wu X, Ghaboussi J and Garret J H, (1992), "Use of neural network in detection of structural damage", Journal of Computers and Structures, 42, pp 649-659.
163. Yamamoto Y and Nikiforuk P N, (2000), "New supervised learning algorithm for multilayered and interconnected neural networks", IEEE Transactions on Neural Networks, 11(1), pp 36-46.
164. Yen G G, (1994), "Identification and control of large structures using neural networks" Journal of Computers and Structures, 52(5), pp 859-870.
165. Younger A S, Conwell P R and Cotter N E (1999), "Fixed-weight on-line learning", IEEE Transactions on Neural Networks, 10(2), pp 272-283.

166. Yu W and Li X (2001), “Some new results on system identification with dynamic neural networks”, IEEE Transactions on Neural Networks, 12(2), pp 412-417.
167. Yu X H, Chen G A and Cheng S X, (1995), “Dynamic learning rate optimization of the backpropagation algorithm”, IEEE Transactions on Neural Networks, 6(3), pp 669-677.
168. Yuan J and Dickinson S M, (1996), “On the vibration of annular, circular and sectorial plates with cut-outs or on partial supports”, Journal of Computers and Structures, 58(6), pp 1261-1264.
169. Yun C B and Bahng E Y, (2000), “Substructural identification using neural networks”, Journal of Computers and Structures, 77(1), pp 41-52.
170. Zadeh L A, (1994), Fuzzy logic, neural networks, and soft computing”, Communications of the ACM, 37, pp 77-84.
171. Zang C and Imregun M, (2001), “Structural damage detection using artificial neural networks and measured FRF data reduced via principal component projection”, Journal of Sound and Vibration, 242(5), pp 813-827.
172. Zenon W and Leonard Z, (2001), “Neural networks in mechanics of structures and materials: new results and prospects of applications”, Journal of Computers and Structures, 79(22 - 25), pp 2261-2276.
173. Zhu Q and Ma D, (1997), “Modification of neural network structure of dynamic system and its application”, Journal of Chemical Industry and Engineering, 48(6), pp 680-685.
174. Zurada J M, (1994), "Introduction to Artificial Neural Network" West Publ. Co.

LIST OF PUBLICATIONS IN REVIEWED JOURNALS/ CONFERENCES

1. S. Chakraverty, R.K. Sharma and **V.P. Singh,** (2003), "**Soft-Computing Approach for Identification of Dynamic Systems**" Journal of New Building Materials & Construction, 8, 50-56.

2. S. Chakraverty, **V. P. Singh** and R. K. Sharma, (2006), **"Regression Based Weight Generation Algorithm in Neural Networks for Estimation of Frequencies of Vibrating Plate"**, Journal of Computer Methods in Applied Mechanics and Engineering 195, (33-36), 4194-4202.

3. R.K. Sharma, **V.P. Singh** and S. Chakraverty, (2003), "**Neural Network experiments on the identification of Dynamic Systems**." Conf. on Mathematics & Computer Applications in Engineering & Science, Jan 2003, TIET, Patiala.

4. S. Chakraverty, R.K. Sharma and **V.P. Singh,** (2003), **"New soft computing based estimation of frequencies for some machine elements to be used in construction machinery**" **Communicated** to National workshop on innovative building construction machinery" August 2003 at CBRI, Roorkee.

5. **V. P. Singh**, S. Chakraverty, R. K. Sharma and G. K. Sharma, (2006) "**Regression Based Multi Input Single Output ANN for Transverse Vibration of annular Circular and Elliptic Plates" (Communicated).**

6. **V. P. Singh**, S. Chakraverty, R. K. Sharma and G. K. Sharma, "**Regression Based Multi Input Multi Output Artificial Neural Network for system identification of multidegree of freedom Dynamic Systems" (To be communicated)**