

To the Graduate Council:

I am submitting herewith a thesis written by Amy Pearsall Henderson entitled "Neural Network Model of Unsteady, Nonlinear Aerodynamics." I have examined the final electric copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Aerospace Engineering.

Frank G. Collins
Major Professor

We have read this thesis
And recommend its acceptance:

Kenneth K. Kimble

Bruce Whitehead

Accepted for the Council:

Dr. Anne Mayhew
Vice Provost and
Dean of Graduate Studies

(Original signatures are on file in the Graduate Student Services Office.)

**NEURAL NETWORK MODEL OF UNSTEADY, NONLINEAR
AERODYNAMICS**

A Thesis
Presented for the
Master of Science
Degree
The University of Tennessee, Knoxville

Amy Pearsall Henderson
May 2002

DEDICATION

This thesis is dedicated to my husband, Ryan, without his patience and support this degree would not have been possible.

ACKNOWLEDGMENTS

I would like to thank my major professor, Dr. Frank Collins, for his guidance and support toward the completion of this thesis. I would also like to thank the additional members of my committee, Dr. Kenneth Kimble and Dr. Bruce Whitehead. I am also grateful to Dr. Ching Lo for his expert advice and suggestions. Each professor has assisted me with comprehension of different aspects of my thesis and degree.

Many thanks are extended to Dr. Alain Pelletier for sharing the data from his dissertation with me. Development of neural network models would not have been possible without real aerodynamic data to use for training and testing.

ABSTRACT

Many flight control systems are developed from aerodynamic measurements obtained from static wind tunnel testing. These control systems frequently inadequately handle unsteady, nonlinear flight conditions. Dynamic roll angle measurements made in a wind tunnel have been obtained. This aerodynamic data presents a nonlinear, unsteady dynamical system. The roll angle trajectories have been successfully approximated with multilayer feedforward backpropagation neural networks.

TABLE OF CONTENTS

Chapter 1. Introduction	1
Chapter 2. Data Analysis	4
2.1 Aerodynamic Data Description	4
2.2 Phase Space Construction	8
Chapter 3. Overview of Neural Networks	20
3.1 Background	20
3.2 Architecture of the Neural Network	20
3.3 Training of the Neural Network	25
Chapter 4. Neural Network Model Developed for Unsteady Aerodynamics	27
4.1 Selected Architecture	27
4.2 Selected Learning Rule	29
Chapter 5. Results of Neural Network Developed to Model Unsteady Aerodynamics	32
5.1 Performance of Networks Lacking Dependence on Sting Angle	32
5.2 Performance of Networks With Dependence on Sting Angle	49
Chapter 6. Conclusions	52
Works Cited	53
Appendices	56
Vita	88

LIST OF TABLES

Table 2.1	Roll attractors and correlation dimensions of the roll angle history at each sting angle.	9
Table 5.1	Mean Squared Error of networks trained with data over portion of test domain.	33
Table 5.2	Mean Squared Error of networks trained with data over entire test domain.	34
Table 5.3	Comparison of error magnitude means and standard deviations for converged networks. Time delay of fourteen was used for all cases.	35

LIST OF FIGURES

Figure 2.1	Double-delta wing experimental setup.	5
Figure 2.2	Roll motion history of 80°/65° double-delta wing.	6
Figure 2.3	Phase plane trajectories of 80°/65° double-delta wing.	7
Figure 2.4	Phase portrait of roll angle at 35° sting with time delay of one.	10
Figure 2.5	Average mutual information vs. time delay for 35° sting angle.	12
Figure 2.6	Phase portrait of 35° sting angle with time delay of ten.	13
Figure 2.7	Phase portrait of 35° sting angle with time delay of fourteen.	13
Figure 2.8	Phase portrait of 35° sting angle with time delay of twenty	14
Figure 2.9	Phase portrait of 30° sting angle with time delay of ten.	15
Figure 2.10	Phase portrait of 30° sting angle with time delay of fourteen.	15
Figure 2.11	Phase portrait of 30° sting angle with time delay of twenty.	16
Figure 2.12	Phase portrait of 33° sting angle with time delay of ten.	17
Figure 2.13	Phase portrait of 33° sting angle with time delay of fourteen.	17
Figure 2.14	Phase portrait of 33° sting angle with time delay of twenty.	18
Figure 2.15	Three dimensional phase portrait of 35° sting angle with time delay of fourteen.	19
Figure 3.1	Multilayer feed-forward neural network layer diagram.	21
Figure 3.2	Transfer functions.	23
Figure 5.1	Layer diagram for selected network 4-28-1.	36
Figure 5.2	Output and errors for network 4-28-1 with the training set interval of 4.	37
Figure 5.3	Output and errors for network 4-28-1 with the test set interval of 3.	39
Figure 5.4	Output and errors for network 4-28-1 with the test set interval of 5.	41
Figure 5.5	Output and errors for network 4-28-1 with the test set interval of 6.	43
Figure 5.6	Output and errors for overfit network 4-40-1 with the training set interval of 6.	45

Figure 5.7	Output and errors for overfit network 4-40-1 with the test set interval of 5.	47
Figure 5.8	Errors for sting angle dependent network 3(1)-50-1 with the training set interval of 4.	50
Figure 5.9	Errors for sting angle dependent network 3(1)-50-1 with the test set interval of 3.	51
Figure A.1	Output and errors for network 3-37-1 with the training set interval of 4.	58
Figure A.2	Output and errors for network 3-37-1 with the test set interval of 3.	60
Figure A.3	Output and errors for network 3-37-1 with the test set interval of 5.	62
Figure A.4	Output and errors for network 3-37-1 with the test set interval of 6.	64
Figure A.5	Output and errors for network 3-40-1 with the training set interval of 4.	66
Figure A.6	Output and errors for network 3-40-1 with the test set interval of 3.	68
Figure A.7	Output and errors for network 3-40-1 with the test set interval of 5.	70
Figure A.8	Output and errors for network 3-40-1 with the test set interval of 6.	72
Figure A.9	Output and errors for network 4-30-1 with the training set interval of 4.	74
Figure A.10	Output and errors for network 4-30-1 with the test set interval of 3.	76
Figure A.11	Output and errors for network 4-30-1 with the test set interval of 5.	78
Figure A.12	Output and errors for network 4-30-1 with the test set interval of 6.	80

LIST OF NOMENCLATURE

ϕ	Roll angle.
σ	Sting angle.
$\mathbf{f}$	System of nonlinear ordinary differential equations.
T	Time delay of reconstructed phase space.
d	Embedding dimension of reconstructed phase space
$\mathbf{x}_1$	Scalar time series.
$\mathbf{x}_d$	Reconstructed variable of dimension, d .
$I(T)$	Average mutual information at a time delay.
$P(x(n))$	Individual probability density of value at index n .
$P(x_1(n), x_2(k))$	Joint probability density between dimensions.
$\mathbf{P}$	Matrix of network inputs.
$\mathbf{IW}$	Vector of input weights.
$\mathbf{LW}^S$	Vector of layer weights to layer S
$\mathbf{b}^S$	Bias vector to layer S .
$\mathbf{TF}^S$	Transfer function for layer S .
$\mathbf{a}$	Vector of network output values.
$\mathbf{t}$	Vector of target output values.
$F(\mathbf{x})$	Mean square error performance index.
μ	Learning rate for training algorithm.
i	Interval used to extract training or test points.

CHAPTER ONE

Introduction

Traditionally, the control laws used by the autopilot of a new aircraft or aircraft configuration are developed from static testing of the aircraft in a wind tunnel. A picture of the dynamic aerodynamic behavior of the aircraft is obtained by recording several parameters and forces at many static flight conditions. Flight testing is then used to validate and refine the control laws of the autopilot.

Several aircraft may have been lost during expensive flight testing before an autopilot is correctly 'trained'. Additionally, many dynamic and unsteady aerodynamic conditions for a specific aircraft configuration may become apparent only after prolonged service. Unsteady, nonlinear behavior cannot be adequately modeled by static wind tunnel testing combined with traditional linear curve fitting techniques. However, dynamic testing of an aircraft in a wind tunnel may yield useful nonlinear results. This dynamic testing could potentially evolve into virtual wind tunnel training of an autopilot. The necessary flight testing could be rendered much safer and inexpensive.

Dynamic wing rock wind tunnel test data has been obtained from Dr. Alain Pelletier. This data was gathered for his dissertation at Notre Dame University in cooperation with the Institute for Aerospace Research, Ottawa, and Wright Patterson Air Force Laboratory. The measurements provided consist of the roll angle motion time history of a double-delta wing at several sting angles, σ . At each sting angle, the wing was given an initial roll angle, ϕ , of 60° and then

allowed to freely roll. The motion at several of the sting angles appears unsteady (See Figure 2.2). In particular, the roll motion of the wing at the 35° sting angle appears chaotic. It is assumed that the nonlinear motion seen can be described by a system, or systems, of nonlinear ordinary differential equations. The number of state variables necessary to describe the system is not obvious from the one-dimensional data. Therefore, the roll angle motion history will be reconstructed into an appropriate dimension phase space using the method of delayed time.

Multi-dimensional phase space may be reconstructed from nonlinear one-dimensional data since the state variables in a nonlinear system are coupled and exhibit interdependency. For the 35° sting angle, an acceptable delay time, T , is determined from the first minimum of the average mutual information. An acceptable embedding dimension, d , of the system is that which does not exhibit any crossing of the trajectory in the reconstructed phase space of dimension d . For a deterministic chaotic system, the minimum embedding dimension is three. The delayed time phase space reconstruction is of the form:

$$\begin{aligned}x_1(n) &= x_1(n) \\x_2(n) &= x_1(n + T) \\\vdots \\x_d(n) &= x_1(n + (d - 1)T)\end{aligned}$$

Multilayer feedforward neural networks have been developed and trained to recursively model the roll angle motion at a sting angle of 35°. The networks

were trained and tested using the roll motion reconstructed at a time delay of fourteen. Dimensions of both three and four have been successfully trained. The majority of this work was spent modeling the chaotic motion at a sting angle of 35° . However, a useful flight controller would need the capability to generalize the roll motion at intermediate sting angles. For this reason, a network was trained on the roll motion resulting from sting angles of 30° and 35° . This network was then tested with roll data for the 33° sting angle. The results of these networks are analyzed in Chapter 5. All of the networks developed were trained using the Levenberg-Marquardt backpropagation training algorithm. Matlab[®] and the Matlab[®] Neural Network Toolbox were used for all computation and analysis in this work.

The provided data set is characterized and analyzed in Chapter 2. Chapter 3 develops the brief background necessary to assist with an understanding of the neural networks used in this work. Chapter 4 describes the specific architecture and training algorithm utilized for the neural networks used to approximate the roll angle. The results of this approximation are presented in Chapter 5. Chapter 6 concludes with statements concerning the suitability of the neural network model for use in a flight control system. The Appendix contains an explanation of the algorithms developed and the inclusion of additional figures of the neural network outputs.

CHAPTER TWO

Data Analysis

2.1 Aerodynamic Data Description

Roll motion histories for a sharp-edged $80^\circ/65^\circ$ double-delta wing with a centerbody were provided by Dr. Alain Pelletier. This data was recorded as part of his dissertation, *A Study of the Nonlinear Aerodynamic Characteristics of a Slender Double-Delta Wing in Roll* [1], completed at Notre Dame University. The experimental roll angle measurements were obtained at a chord Reynolds number of 300,000. The low speed, low turbulence Notre Dame wind tunnel used had a 2 ft X 2 ft test section. This experiment was performed at sting angles of 25° , 30° , 33° , 35° , 40° , and 45° . For each data set, the wing was positioned at an initial roll angle of 60° and then allowed to roll freely on an air bearing attached to the centerbody. The roll angle was recorded at intervals of five milliseconds for sixty seconds. This allowed 12,000 data points to be recorded at each sting angle. Figure 2.1 illustrates the experimental setup of the double-delta wing. Figure 2.2 displays the roll motion over time for each sting angle. A phase plane view of the trajectories is shown in Figure 2.3. The data provided by Dr. Pelletier represents a dynamic data set of the roll motion for the double-delta wing.

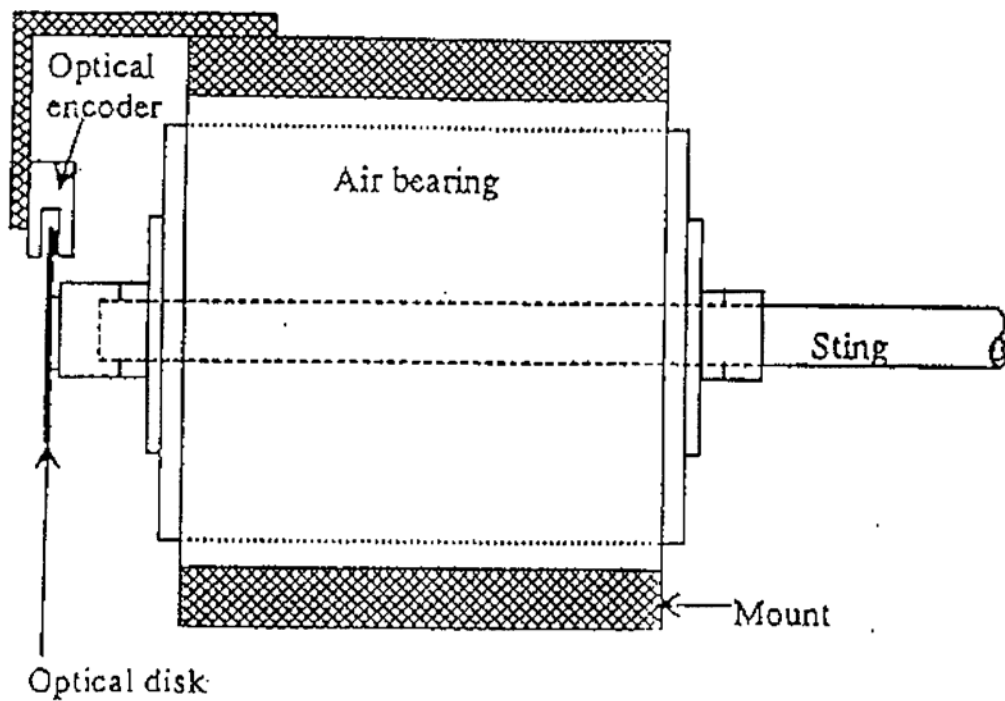
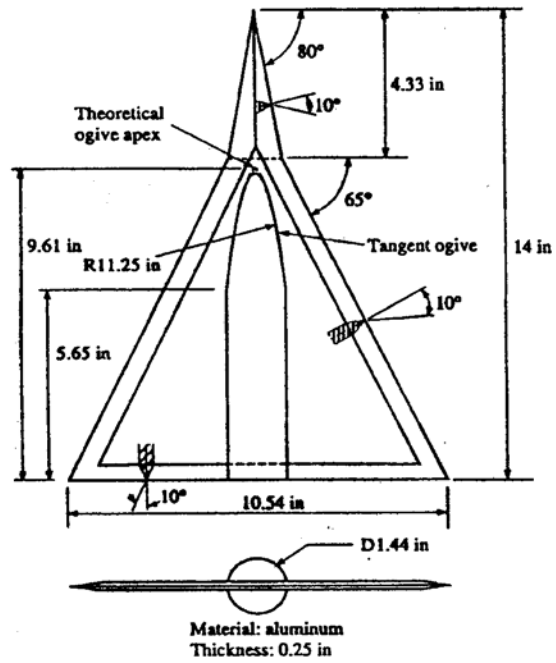


Figure 2.1 Double-delta wing experimental setup [1].

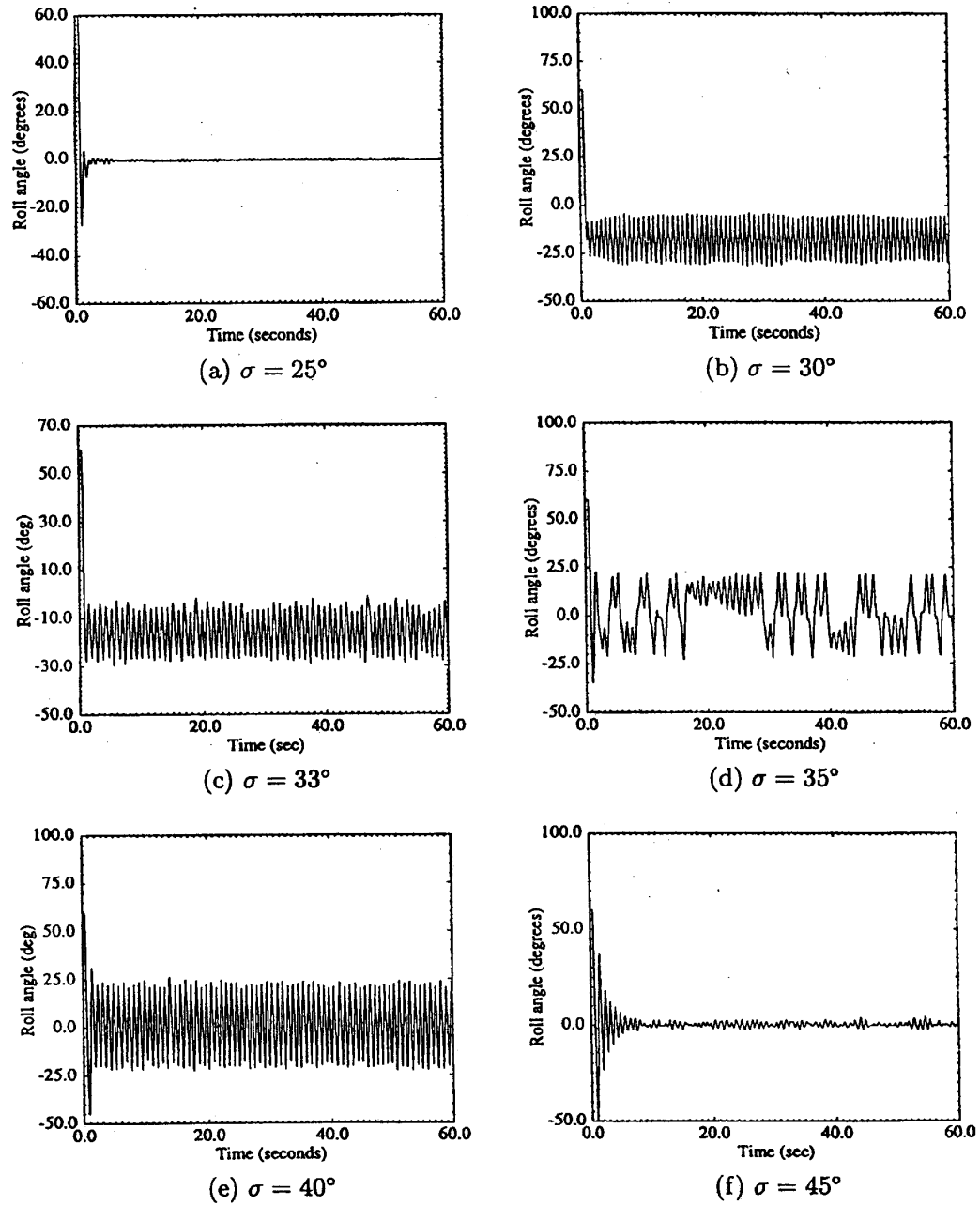


Figure 2.2 Roll motion history of $80^\circ/65^\circ$ double-delta wing [1].

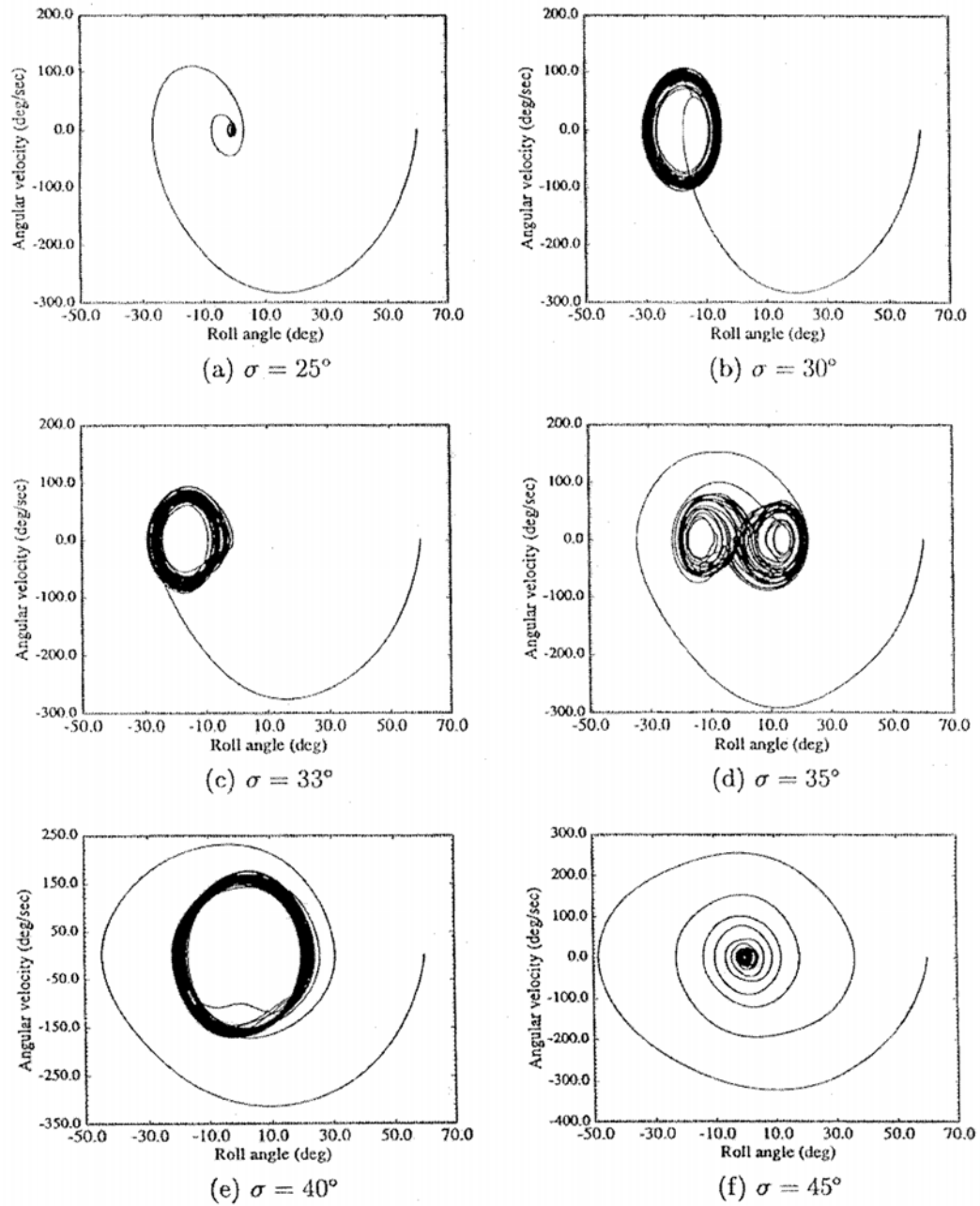


Figure 2.3 Phase plane trajectories of $80^\circ/65^\circ$ double-delta wing [1].

A stable attractor may be seen at a sting angle of 25°. As the sting angle is increased, the roll motion trajectories demonstrate unsteady, nonlinear behavior. Limit cycle behavior is apparent in the trajectory of the roll angle at sting angles of 30°, 33°, and 40°. The phase space portrait for a sting angle of 35° displays two distinct attractors reminiscent of a chaotic system. The roll motion at a sting angle of 45° could represent an intermittently chaotic response. Table 2.1 lists the roll attractors detected in the data. The correlation dimension as computed by Dr. Pelletier is also included in Table 2.1. A correlation dimension greater than one suggests a chaotic system [2].

2.2. Phase Space Construction

The one-dimensional data for the 35° sting angle consists of the roll angle motion time history. The roll dynamics depend upon a set of infinite dimensional coupled nonlinear partial differential equations. The type of nonlinear motion seen may be described by a system of finite coupled nonlinear ordinary differential equations:

$$\frac{d\mathbf{x}(t)}{dt} = f(\mathbf{x}(t), \sigma)$$

An example of this is the Lorenz model for atmospheric convection. The necessary number of state variables, also known as the embedding dimension, will be determined by phase space reconstruction as described by Abarbanel.

Table 2.1 Roll attractors and correlation dimensions of the roll angle history at each sting angle.

Sting Angle, $\sigma(\text{deg})$	Attractor(s), $\phi(\text{deg})$	Correlation Dimension
25	$\cong 0$	
30	-18	1.08 ± 0.06
33	-14 to -15	
35	± 10.92	1.5 ± 0.06
40	$\cong 1.0$	1.10 ± 0.06
45	0.1	
$\Delta t = 0.005 \text{ sec}, \phi_0 = 60^\circ, \text{Re}_c = 300,000$		

Mutli-dimensional phase space may be reconstructed from nonlinear one-dimensional data since the state variables in a nonlinear system are coupled and exhibit interdependency. An appropriate delay time, T , will be determined from the first minimum of the average mutual information.

An acceptable embedding dimension, d , of the system will be that which does not exhibit any crossing of the trajectory in the reconstructed phase space of dimension d . The time delayed phase space is a portrait of the original state space. The delay time phase space reconstruction is of the following form.

$$\begin{aligned}
 x_1(n) &= x_1(n) \\
 x_2(n) &= x_1(n+T) \\
 &\vdots \\
 x_d(n) &= x_1(n+(d-1)T)
 \end{aligned}$$

The time delay is an integer increase to the index of the one-dimensional data.

This allows the values of the additional state variables to be predefined.

Choosing a time delay that is too small does not allow sufficient independency between the state variables. A very large time delay permits an excessive independence within the system. Any correlation between the variables would be random. An acceptable time delay may be determined by examining the idea

of information generation in a chaotic system. “Stable linear systems generate zero information, ... “ [2].

Chaotic motion may occur in deterministic dynamical systems that are described by three or more coupled nonlinear ordinary differential equations. Chaotic systems exhibit a sensitivity to initial conditions. This suggests that trajectories, which start out near one another, may differ exponentially within a short time. However, the possible trajectories will be closely spaced within a constrained region of phase space. Every trajectory of a chaotic, dynamical system is unique. The dimension of phase space must be sufficient to prevent crossing of the reconstructed trajectory. The roll motion at 35° may be chaotic, but it is not random. Figure 2.4 depicts the phase plane of the roll angle data with a time delay of one. A definite structure is clear in the phase plane [3].

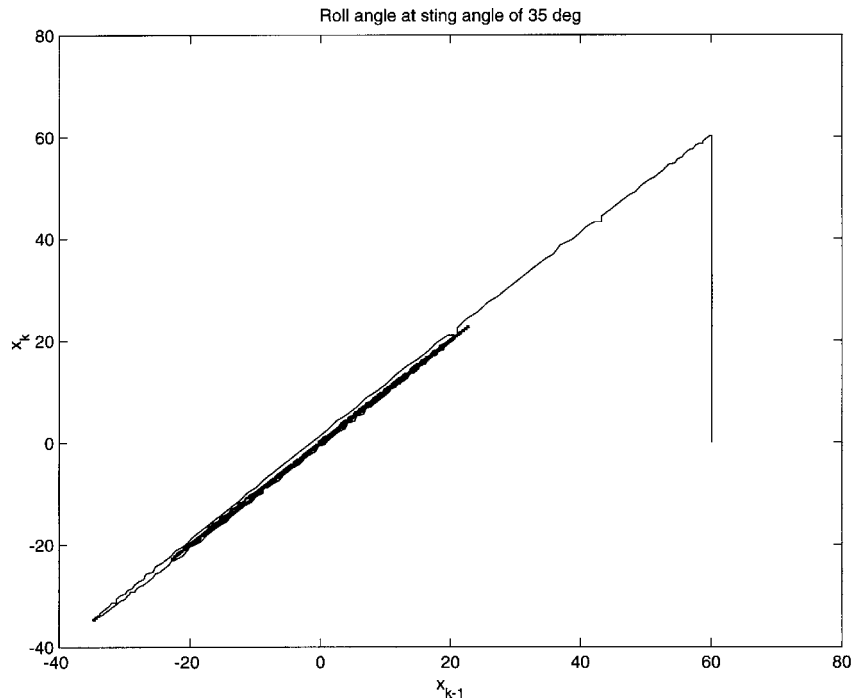


Figure 2.4 Phase portrait of roll angle at 35° sting with time delay of one [3].

Information generation is a property unique to nonlinear dynamics [2]. The average mutual information (in bits) between sets of reconstructed data may be calculated from the following theoretical expression.

$$I(T) = \sum_{x_1(n), x_2(k)} P(x_1(n), x_2(k)) \log_2 \left[\frac{P(x_1(n), x_2(k))}{P(x_1(n))P(x_2(k))} \right]$$

$P(x_1(n))$ and $P(x_2(k))$ represent the individual probability densities of $x_1(n)$ and $x_2(k)$, respectively. The joint probability density between the original and reconstructed dimension is expressed as $P(x_1(n), x_2(k))$. The above expression assumes that the statistical density distribution between the joint measurements is given by the joint probability density. The optimal time delay is chosen as that which yields the first minimum of the average mutual information. Please see Figure 2.5 for the evolution of the average mutual information as the time delay is increased. The nonlinear motion at a sting angle of 35° appears to slowly approach an asymptote. The rate of descent becomes more gradual with each additional time delay. The average mutual information has been calculated through a time delay of twenty. A minimum has not been found via estimation of the probability densities.

The array x_2 is created by the program `datacreate.m`. The average mutual information relating to a particular time delay is computed in algorithm `mutinfo.m`. The code for these programs may be found in Appendix B. Please see comments within the code for a detailed explanation of the procedures used.

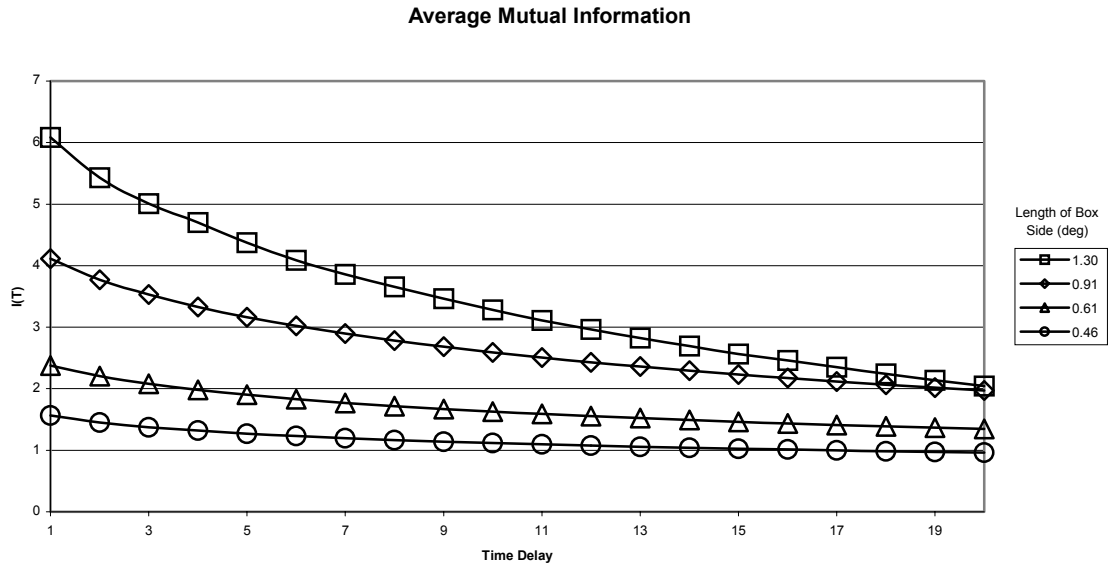


Figure 2.5 Average mutual information vs. time delay for 35° sting angle.

Figures 2.6-2.8 illustrate the reconstructed phase space for the 35° sting angle in two dimensions at three different time delays. The phase plane trajectory begins to unfold from a time delay of ten to fourteen as the average mutual information of the dynamical system decreases. A time delay of fourteen correlates to the first minimum of the average mutual information. The trajectory has begun to unravel by the time delay of twenty as the average mutual information of the system increases. This cycle will repeat as the time delay is varied. Any of the minimums may be used to reconstruct the phase space of the dynamical system. The first minimum will be chosen as advised by Abarbanel. This is the time delay for which the general dependence between the dimensions of the phase space is least.

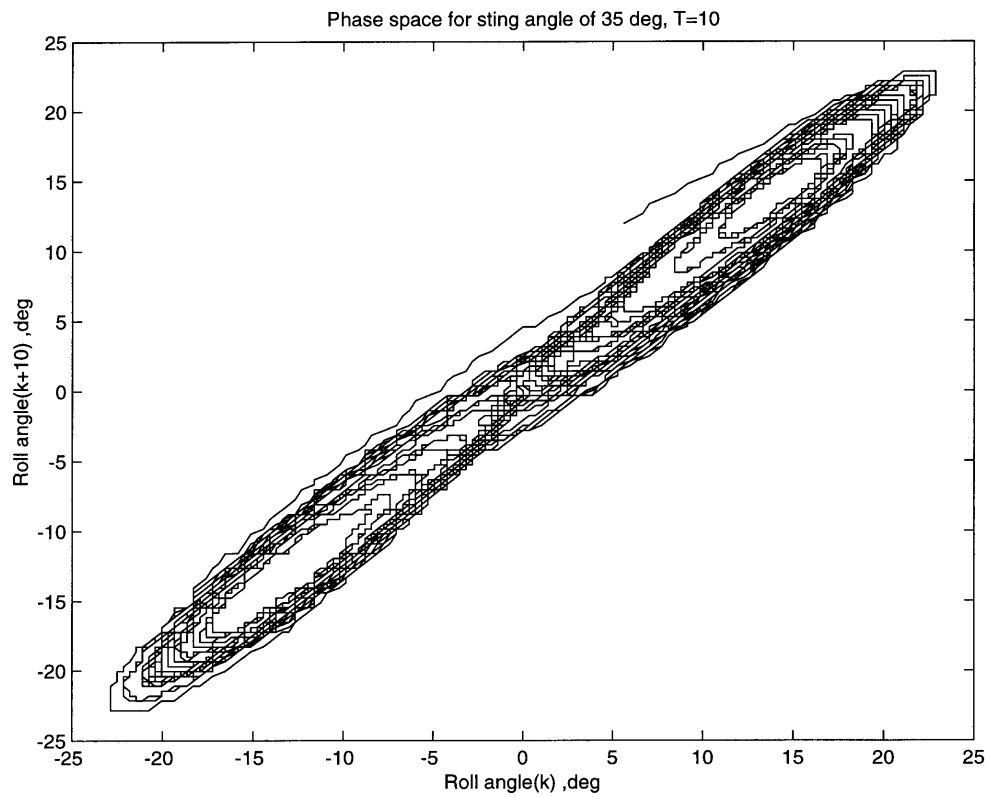


Figure 2.6 Phase portrait of 35° sting angle with time delay of ten.

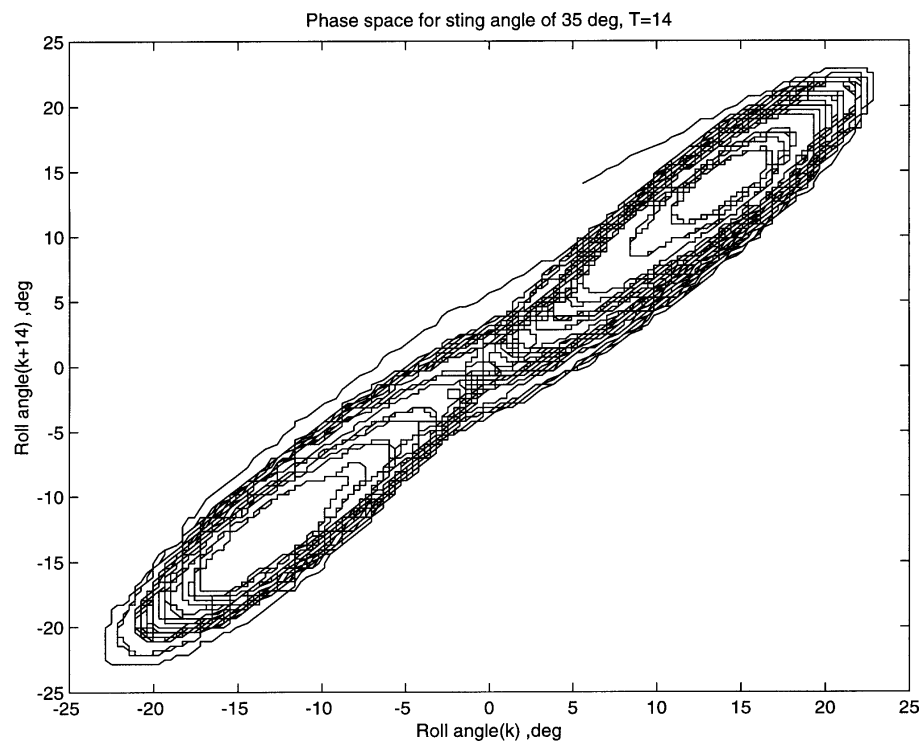


Figure 2.7 Phase portrait of 35° sting angle with time delay of fourteen.

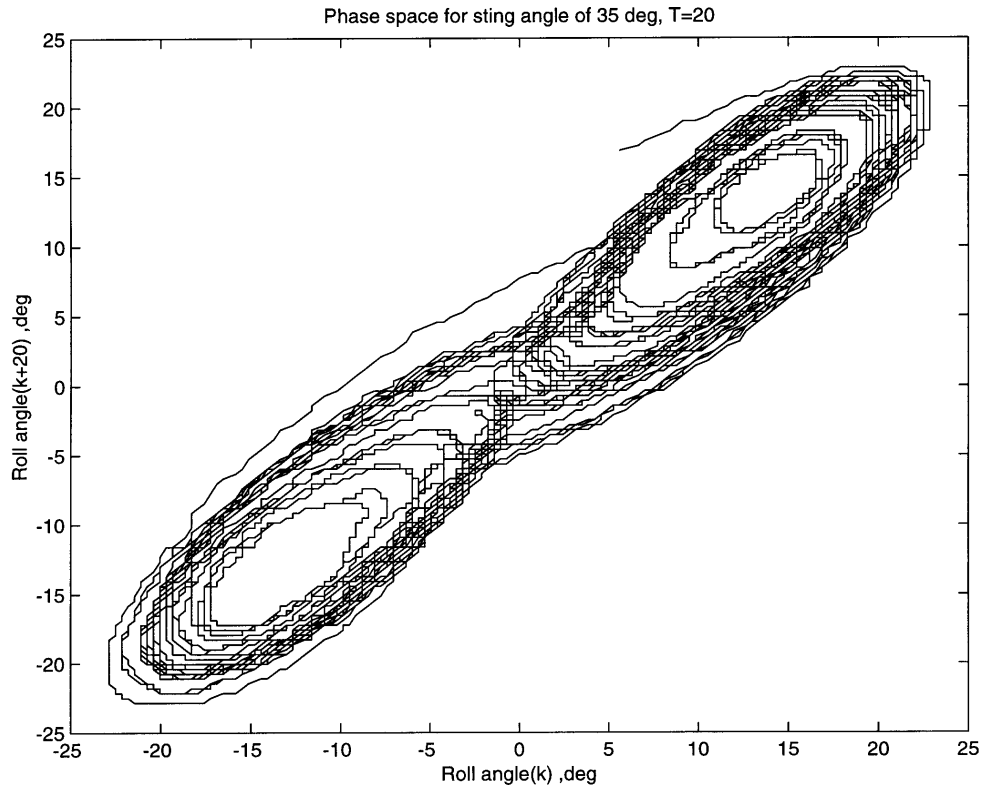


Figure 2.8 Phase portrait of 35° sting angle with time delay of twenty.

The attractors of the roll angle at the 35° sting angle are well defined at a time delay of fourteen. A time delay of fourteen will also be used to reconstruct the limit cycle behavior at sting angles of 30° and 33° to facilitate the training of a neural network that incorporates dependence on the sting angle. Figures 2.9-2.14 illustrate the phase space trajectories at these sting angles. The first 300 recorded points at each sting angle have not been used. This allows the neural network to model only the attractor portions of the trajectories.

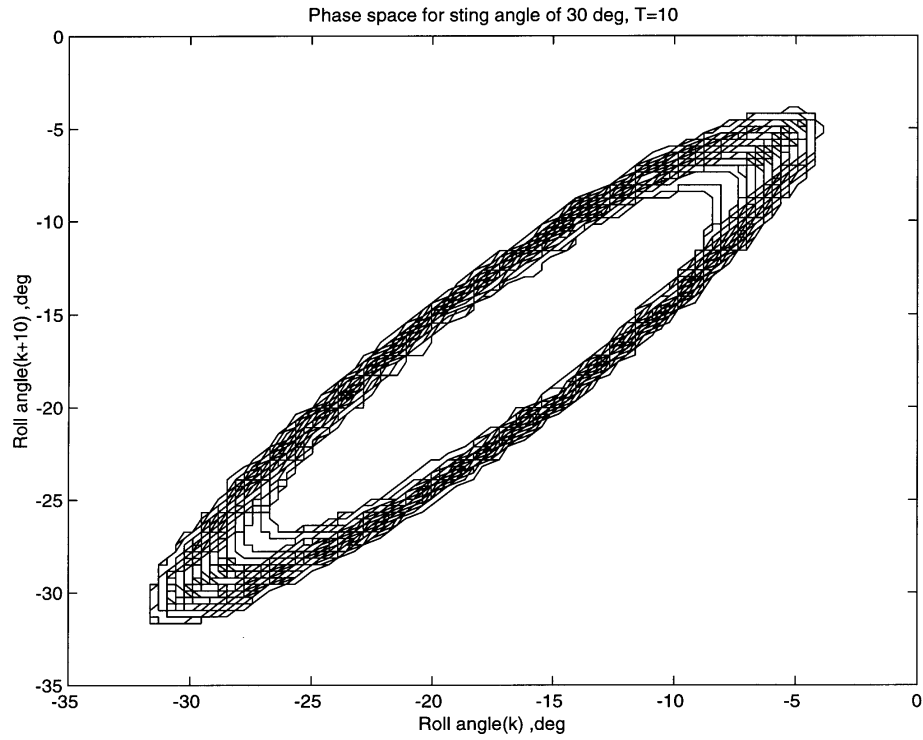


Figure 2.9 Phase portrait of 30° sting angle with time delay of ten.

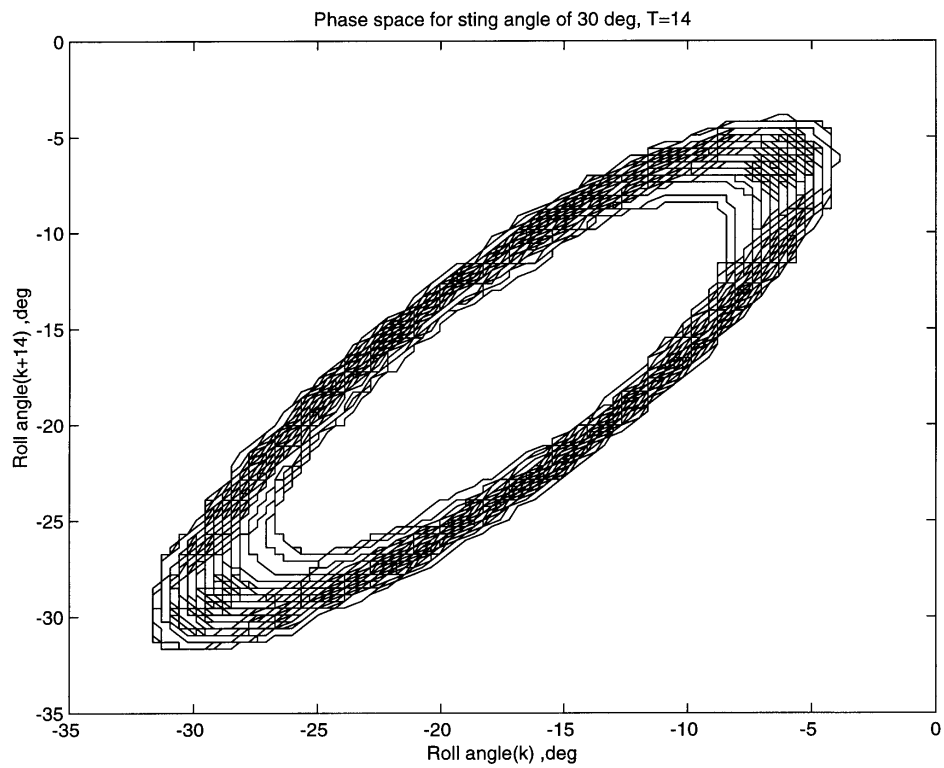


Figure 2.10 Phase portrait of 30° sting angle with time delay of fourteen.

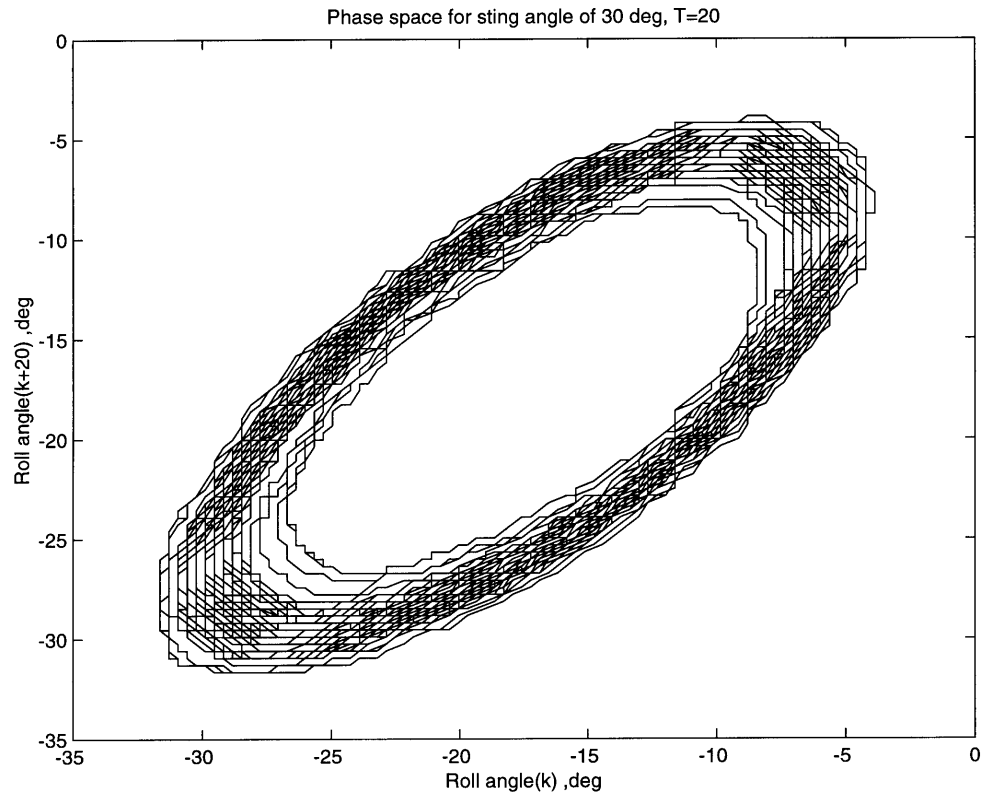


Figure 2.11 Phase portrait of 30° sting angle with time delay of twenty.

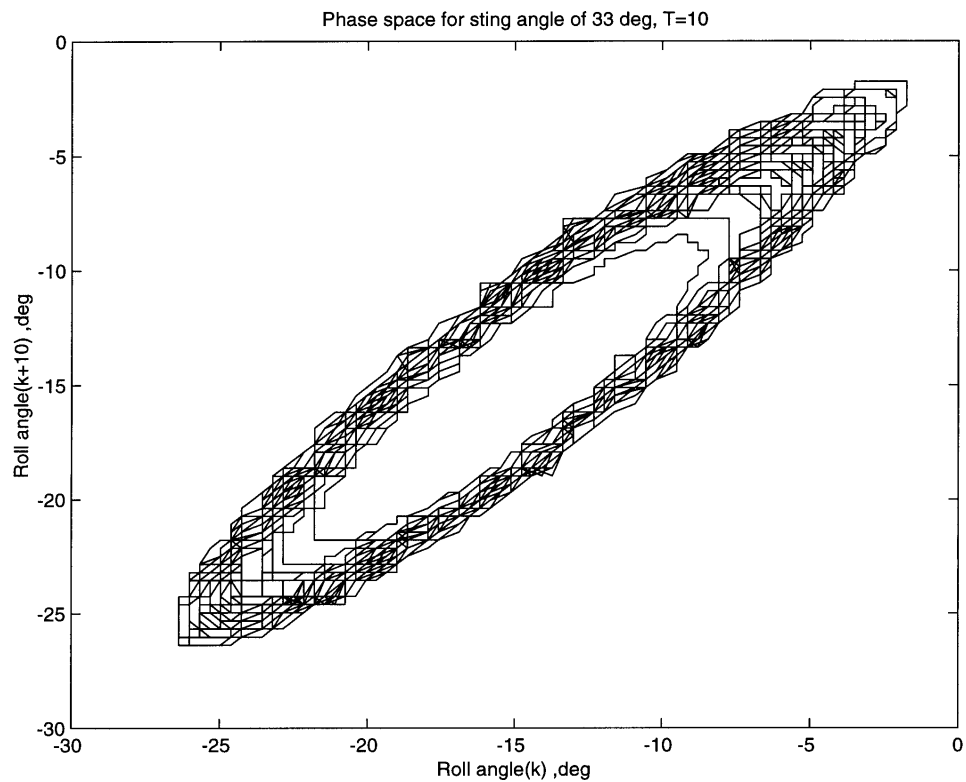


Figure 2.12 Phase portrait of 33° sting angle with time delay of ten.

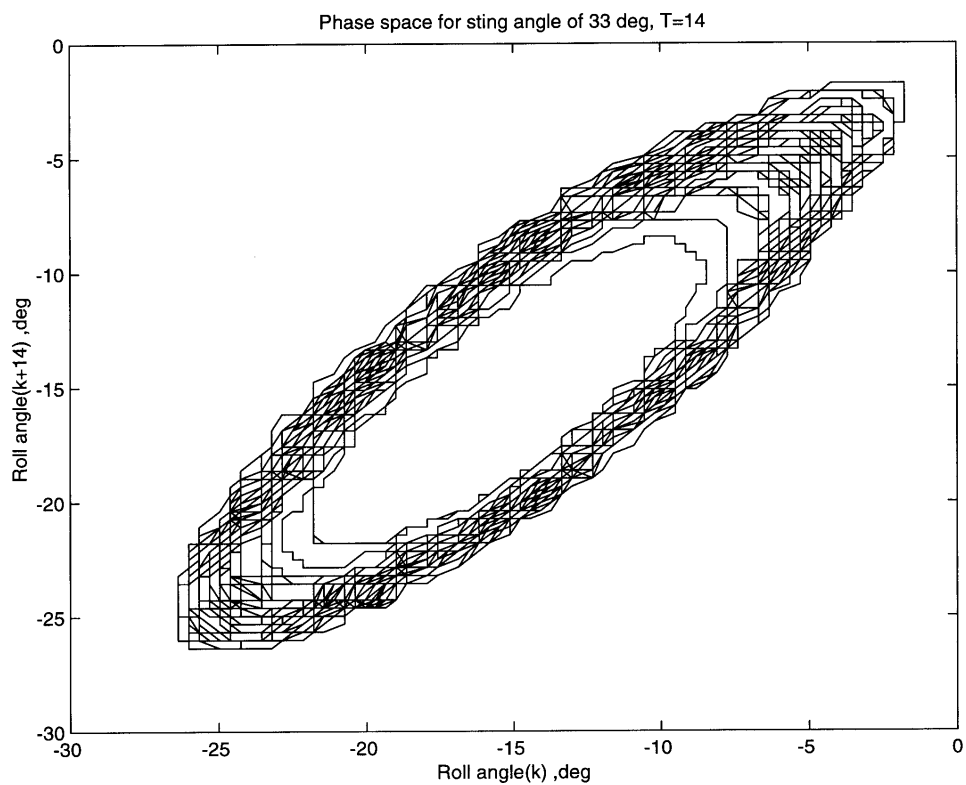


Figure 2.13 Phase portrait of 33° sting angle with time delay of fourteen.

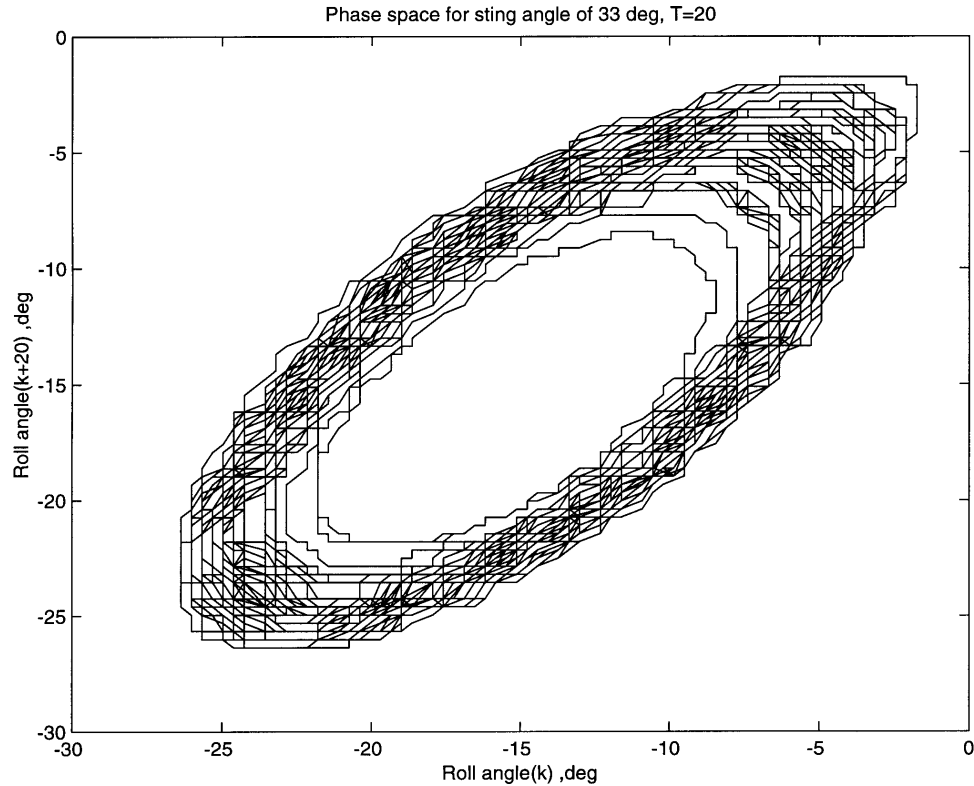


Figure 2.14 Phase portrait of 33° sting angle with time delay of twenty.

The trajectory must be unfolded so that it does not cross itself since each trajectory of a deterministic chaotic system is unique. Also, a chaotic system must have at least three dimensions. Figure 2.15 shows the symmetry of the winged attractor in three dimensions for the sting angle of 35°. It appears that three dimensions are sufficient to unfold the trajectory. Additional dimensions will not affect the accuracy of the neural network model. They may, however, decrease the complexity of the resulting neural network. Chapter 5 relates the results of neural networks approximating the roll angle, in three, four, and five dimensions, for the 35° sting angle.

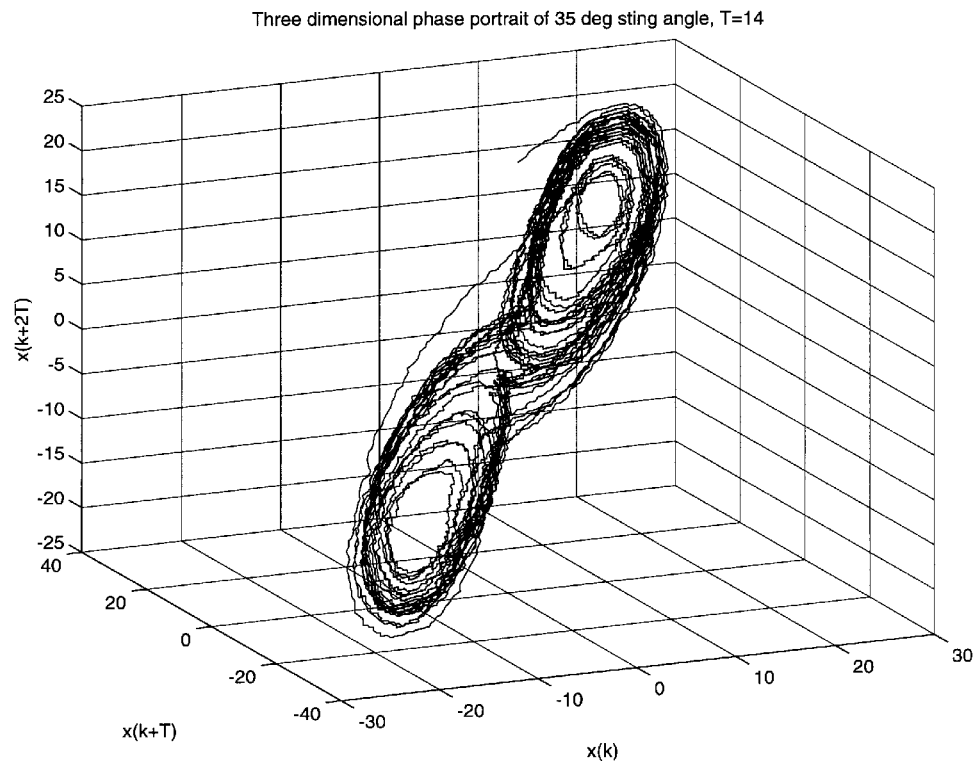
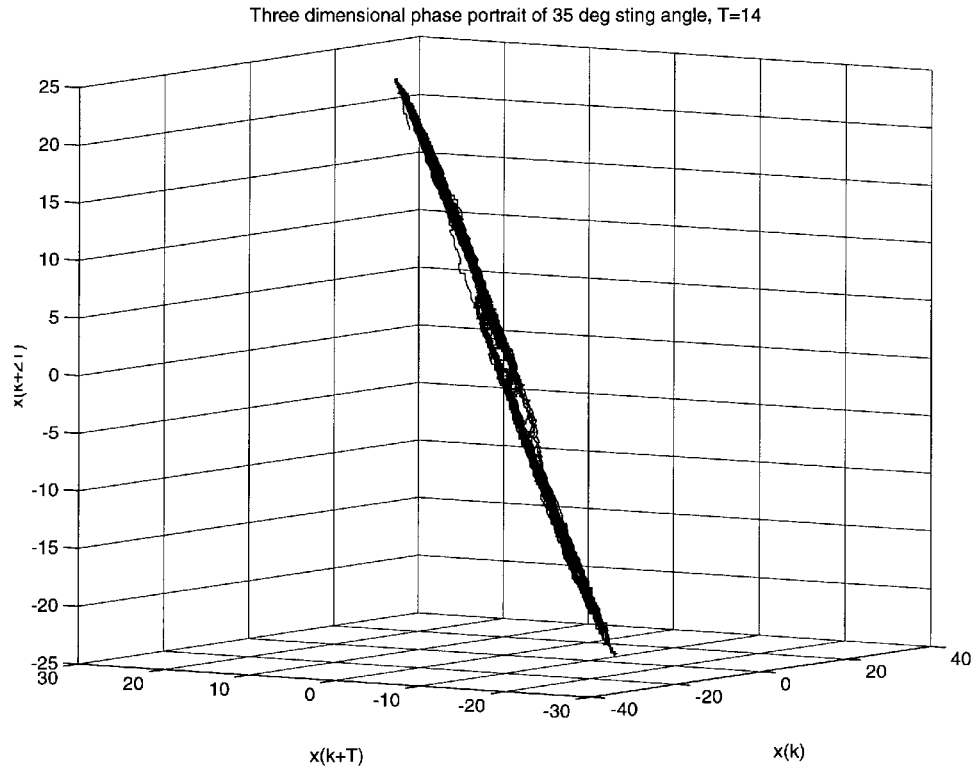


Figure 2.15 Three dimensional phase portrait of 35° sting angle with time delay of fourteen.

CHAPTER THREE

Overview of Neural Networks

3.1 Background

Neural networks first came into use with the perceptron developed by Frank Rosenblatt in the late 1950's [4]. The perceptron was capable of classifying linearly separable data sets. The addition of more nodes, or layers of nodes, could assist the network in performing increasingly difficult classification problems. Today, neural networks have evolved into several applications of pattern recognition, classification, and function approximation. Neural networks have been used in trajectory control of robots on assembly lines, they have been incorporated into speech recognition algorithms, and they have been successfully applied in flight control systems [4].

3.2 Architecture of the Neural Network

The human brain is capable of incredible generalization and pattern recognition. The parallel computing power of the brain allows it to accomplish facial recognition and interpret a spoken language of many accents. Neural network architecture is based upon an operational concept of the biologic neuron and nervous system. The processing unit of a neural network is the neuron, or node. The network is constructed of one or more layers that each consist of at least one node. Some authors refer to the set of inputs as a layer. This work assumes that a layer performs a processing function. For this reason, the inputs

will not be referred to as a layer. The first layer is labeled the output layer if it is the only layer. In the case of two or more layers, the last layer is called the output layer and all others are referred to as hidden layers. Please refer to Figure 3.1 for an illustration of a feedforward neural network.

The determination of the number of layers and the number of nodes in each layer is problem specific and not well defined. Too few neurons may result in underfitting the problem. A network suffers a loss of problem generalization if too many neurons are used. The network overfits the problem by exactly learning the cases of the training points. A general rule is that the number of training pairs should exceed the number of parameters (total number of weights and biases).

Each node in the first layer is connected to every input via a weighted connection. Each node of the second and following layers receives weighted connections from every nodal output, or activation value, of the preceding layer. Each connection has a unique weight, which is multiplied with the output over

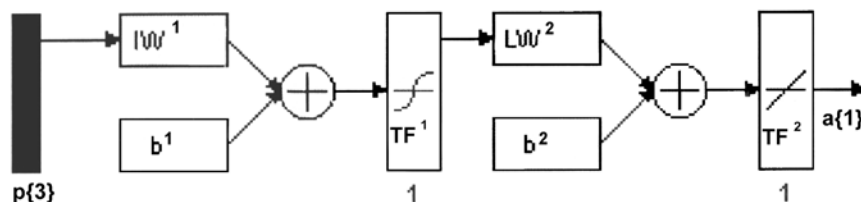
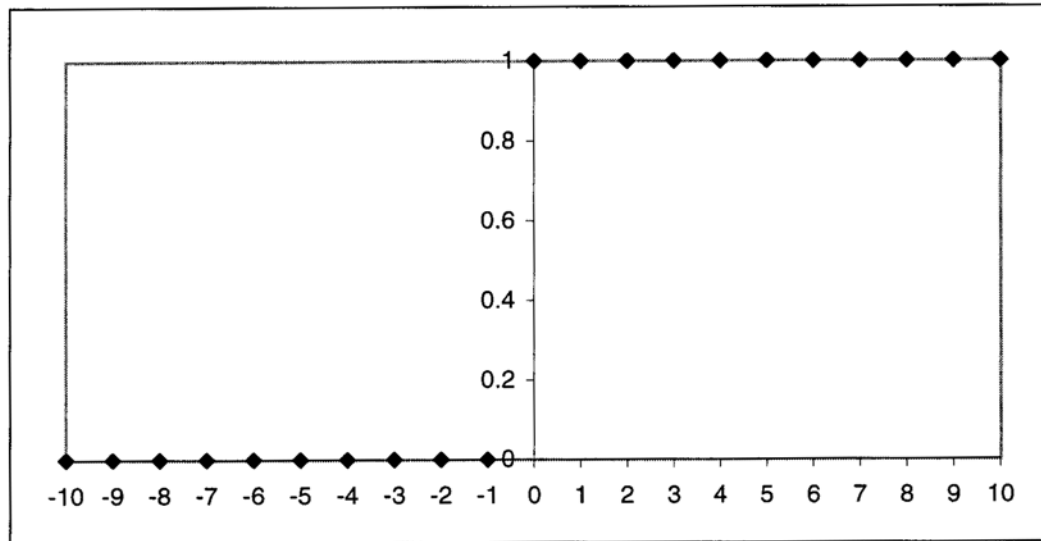


Figure 3.1 Multilayer feed-forward neural network layer diagram.

that connection. A node sums together its respective collection of weighted inputs and biases. The biases may be viewed as weights with a constant input of one. Biases may be used to shift a node's output away from the origin. Biases and weights are adjustable parameters of the network and are set during the training phase of the network. A node's net input is the argument of a transfer function, TF , in each node. The mapping of the transfer function then becomes the activation value of a node. The output for the q^{th} input set of the network pictured in Figure 3.1 is displayed below. The superscripts indicate the layer that the weight matrix, $\mathbf{LW}$, bias vector, $\mathbf{b}$, transfer function, TF , or net input, $\mathbf{m}$, belong to.

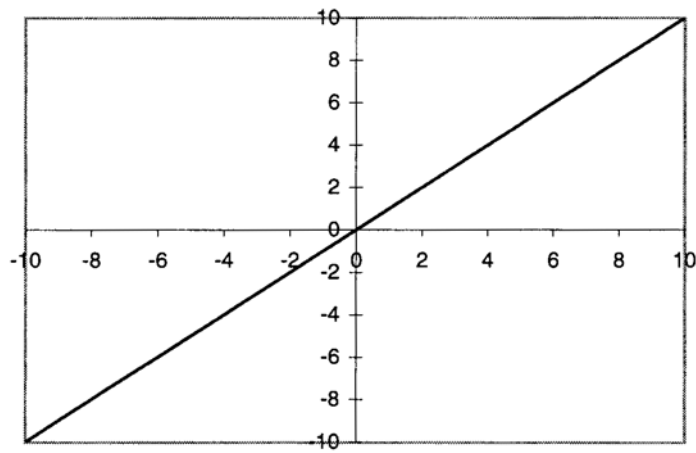
$$a(q) = TF^2(\mathbf{LW}^2[TF^1(\mathbf{IW}^1\mathbf{p}(q) + \mathbf{b}^1)] + \mathbf{b}^2) = TF^2(\mathbf{LW}^2[TF^1(\mathbf{m}^1)] + \mathbf{b}^2)$$

Every node of a layer uses the same transfer function. The selection of a transfer function for a layer is problem specific. For example, the hard limit transfer function returns either a one or a zero and may be useful for binary classifications. The hard limit, linear, log-sigmoid, and hyperbolic tangent sigmoid transfer functions are illustrated below in Figure 3.2. The sigmoid functions are useful for function approximation of nonlinear problems. A network which uses tangent sigmoid transfer functions in the hidden layer and linear transfer functions in the output layer should be able to generalize almost any continuous function to some degree, as long as a sufficient number of hidden neurons is used [4].



$$a = \begin{cases} 0, & m < 0 \\ 1 & m \geq 0 \end{cases}$$

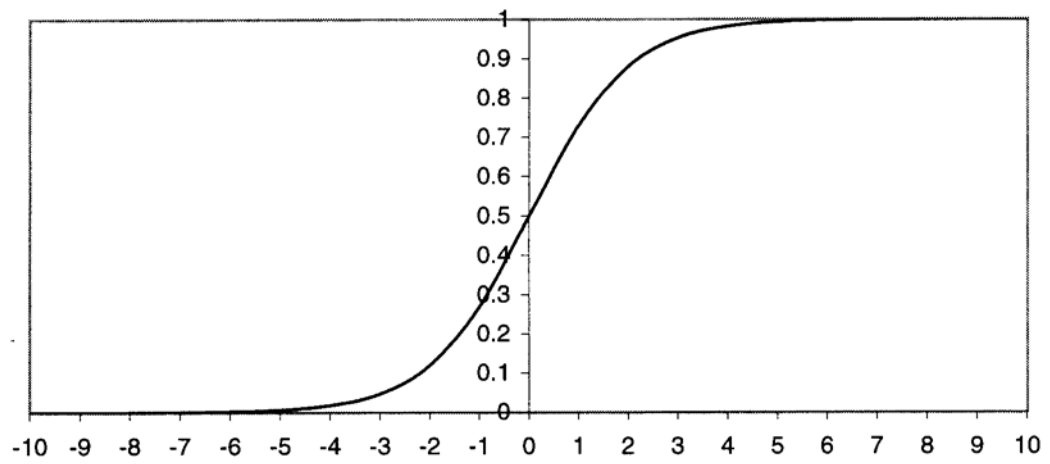
a) Hard Limit



$$a = m$$

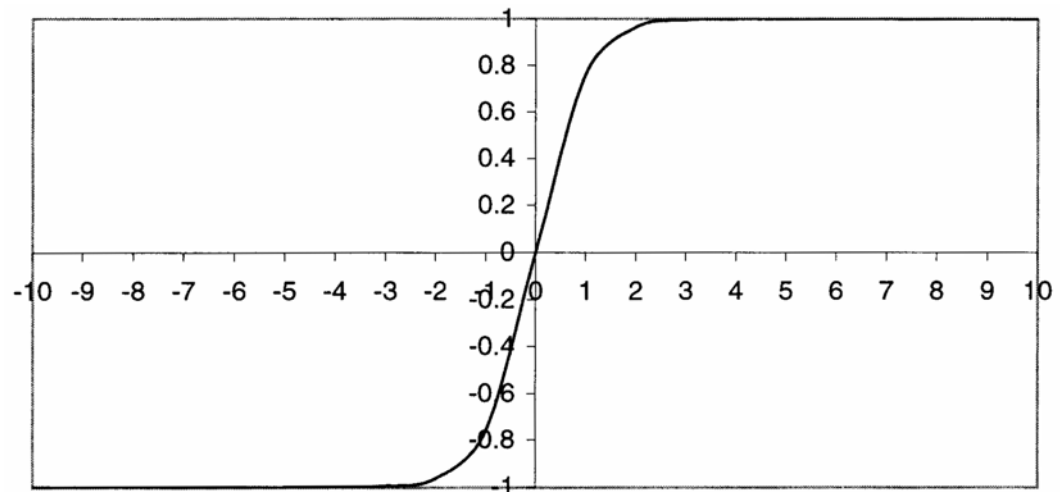
b) Linear

Figure 3.2 Transfer functions.



$$a = \frac{1}{1 + e^{-m}}$$

c) Logarithmic Sigmoid



$$a = \frac{e^m - e^{-m}}{e^m + e^{-m}}$$

d) Hyperbolic Tangent Sigmoid

Figure 3.2 Continued

3.3 Training of the Neural Network

A neural network is trained on a set of data representative of the entire problem domain. Training of the network may be either supervised or unsupervised. With supervised learning, target output values are provided for each set of input points. This allows the training algorithm to compute the error of each output and determine the appropriate change to the weights and biases. Unsupervised learning does not provide target values for the outputs. The weights and biases are adjusted in response to the inputs. This type of learning is useful for some types of clustering problems.

Incremental, or adaptive training, is accomplished by calculating the change to the weights after each input set is presented. In batch training the weight updates are computed following each pass, or epoch, through the training set. A network is tailored to a specific problem by setting the values of the weights and biases to yield a desired output. Several training algorithms, or learning rules, have been developed to adjust the values of the weights and biases toward an optimal setting. Many of these algorithms attempt to minimize the mean square error of the network outputs. This is done by following the mean square error performance index gradient in the direction of steepest descent. Several variations of the steepest descent method exist to optimize accuracy and speed. The selection of a learning rule is often problem specific.

Training continues until the mean square error has reached a given value, a certain number of epochs through the data have occurred, or a given amount of time has passed.

CHAPTER FOUR

Neural Network Model Developed for Unsteady Aerodynamics

Primary emphasis in this work has been placed on approximating the roll angle motion recorded at a sting angle of 35° . Unless otherwise noted, all statements and figures pertinent to the neural network model refer to this case.

4.1 Selected Architecture

Multilayer feed-forward backpropagation was selected as the basic architecture for a neural network model of the unsteady aerodynamics provided by Dr. Pelletier. This architecture has been documented as successful at function approximation of nonlinear continuous and chaotic trajectories.

Each network consisted of one hidden layer utilizing the hyperbolic tangent sigmoid transfer function. Also, each network possessed one output layer with one node using the linear transfer function. As noted in Chapter Two, the dimension of the roll motion phase space may be approximated by reconstructed coordinates in three or four dimensions with a time delay of 14. The number of dimensions correlates to the number of inputs to the neural network. Therefore, networks with three, four, and five inputs were constructed and trained.

The data set was constructed by removing every fourteenth point from the roll angle motion history. For three inputs to the network, the target output training vector was created by extracting every fourth point from the reconstructed data set. Every fifth point of the data set was extracted for a network with four inputs.

This vector, $\mathbf{t}$, contains the desired outputs of the network to be used for calculation of the mean squared error during training. The inputs, $\mathbf{P}$, for each target output value consist of the preceding three or four points in the reconstructed data set. In this way, recursive input and target output pairs were constructed to train and test the network.

A quarter of the input and target pairs were used as the training set for every network architecture. These were constructed by selecting target values from the reconstructed data set with an interval of four. These are then grouped with their corresponding inputs to compromise the training set. To improve generalization, it is important to use a training set that is representative of the entire roll trajectory. Converged networks were tested for generalization with intervals of three, five, and six. The interval, i , used to generate trajectories and error graphs is indicated on the title line of the respective figures.

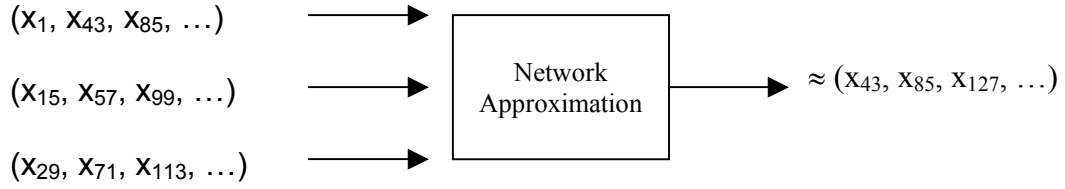
The below expressions are representative of a three dimensional training set with an interval of four.

$$\begin{aligned}\mathbf{RawData} &= [x_1, x_2, \dots, x_n] \\ \mathbf{DataSet} &= [x_1, x_{15}, x_{29}, x_{43}, \dots] \\ \mathbf{t} &= [x_{43}, x_{99}, x_{155}, x_{211}, x_{267}, \dots] \\ \mathbf{P} &= \begin{bmatrix} x_1 & x_{57} & x_{113} & x_{169} & x_{225} & \cdots \\ x_{15} & x_{71} & x_{127} & x_{183} & x_{239} & \cdots \\ x_{29} & x_{85} & x_{141} & x_{197} & x_{253} & \cdots \end{bmatrix}\end{aligned}$$

The below expressions are representative of a three dimensional test set with an interval of three.

$$\mathbf{t} = [x_{43}, x_{85}, x_{127}, \dots]$$

$$\mathbf{P} = \begin{bmatrix} x_1 & x_{43} & x_{85} & \cdots \\ x_{15} & x_{57} & x_{99} & \cdots \\ x_{29} & x_{71} & x_{113} & \cdots \end{bmatrix}$$



The uncertainty in the recorded roll angle for a sting angle of 35° is 0.54° [1].

Therefore, each measurement was rounded to four significant figures to avoid introducing additional errors into the neural network approximation. The algorithm *create_input_target.m* was used to generate the input and target pairs and is included in Appendix B.

4.2 Selected Learning Rule

Backpropagation training algorithms propagate errors from the output layer back through the network. These errors are then used to compute weight and bias updates. The Levenberg-Marquardt (LMBP) learning rule was selected for use in the function approximation of the roll motion. This training algorithm is fast and accurate when compared to other backpropagation training algorithms. Also, it has better performance with function approximation when compared to other

rapid algorithms such as resilient backpropagation. Resilient backpropagation appears to perform better with pattern recognition schemes [5]. LMBP calculates the Jacobian matrix, or the derivatives of the errors, with each epoch through the training set. A sufficiently large network could stress the memory resources of a computing environment. The network sizes developed in this work did not pose a significant memory burden.

Levenberg-Marquardt backpropagation is a supervised training scheme applied to the training set in batch mode. It is a variation of Gauss-Newton's method and is a numerical optimization of the steepest descent algorithm. Like all backpropagation learning rules, LMBP attempts to modify the network parameters so as to minimize the mean square error performance index. An expression for the mean square error with one network output is shown below.

$$F(x) = \frac{1}{Q} \sum_{q=1}^Q (t_q - a_q)^2 = \frac{1}{Q} \sum_{q=1}^Q e_q^2$$

In the above expression, e_q is the error for the q th input/target pair [4].

Multilayer nonlinear networks can produce a complex error surface with many local minima. Applying LMBP with the use of momentum can help ensure that convergence is reached at the global minimum of the performance index [5]. It is prudent to attempt several training runs with different initial weights. It is important to initialize the weights with small random numbers to avoid difficulties in reaching the global minimum. The performance function for multilayer

nonlinear networks tends to have a saddle point near the origin. Also, flat regions may be found far from the origin [4]. The training algorithm could become mired in these regions if large initial weights are used.

Newton's method optimizes a performance index by applying an update to the network parameters.

$$\begin{aligned}\mathbf{x}_{k+1} &= \mathbf{x}_k - \mathbf{A}_k^{-1} \mathbf{g}_k \\ \mathbf{A}_k &\equiv \nabla^2 F(\mathbf{x}_k), \nabla^2 F(\mathbf{x}) \cong 2\mathbf{J}^T(\mathbf{x})\mathbf{J}(\mathbf{x}) \\ \mathbf{g}_k &\equiv \nabla F(\mathbf{x}_k), \nabla F(\mathbf{x}) = 2\mathbf{J}^T(\mathbf{x})\mathbf{e}(\mathbf{x})\end{aligned}$$

The approximation to the Hessian is valid if the sensitivity of the performance function to the inputs is small. In some cases, the Hessian may not be invertible. Levenberg-Marquardt backpropagation differs from the Gauss-Newton learning rule in that the Hessian is rendered invertible by the following substitution to the parameter update.

$$\mathbf{x}_{k+1} = \mathbf{x}_k - [\mathbf{J}^T(\mathbf{x}_k)\mathbf{J}(\mathbf{x}_k) + \mu_k \mathbf{I}]^{-1} \mathbf{J}^T(\mathbf{x}_k)\mathbf{e}(\mathbf{x}_k)$$

As μ_k is increased, LMBP approaches the steepest descent algorithm with a small learning rate. As μ_k is decreased to zero, LMBP becomes the Gauss-Newton method. An initial value of 0.001 was used for μ_k for all training periods of converged networks in this paper.

The reader is referred to Chapters 11 and 12 of *Neural Network Design* by Hagan, Demuth, and Beale [4] for further details of the LMBP derivation.

CHAPTER FIVE

Results of Neural Network Developed to Model Unsteady Aerodynamics

Primary emphasis in this work has been placed on approximating the roll angle motion recorded at a sting angle of 35° . Unless otherwise noted, all statements and figures pertinent to the neural network model refer to this case.

5.1 Performance of Networks Lacking Dependence on Sting Angle

Chapter Four related the method of preparation for the data sets used in the training and testing of the neural networks. An interval of four was used for all training. Test sets were composed of intervals of three, five, and six. Figures depicting the output or error of a network will have an entry in the title line indicating the interval, i , used to generate the results depicted in that figure. This chapter and the above mentioned figures refer to network architectures in the following manner:

S^1 - S^2 - S^3 ,

S^1 = number of inputs = number of dimensions

S^2 = number of nodes in hidden layer

S^3 = number of outputs (One value is output by the network with the recursive model used.)

Recall from Chapter Two that the trajectory became sufficiently independent at a time delay of fourteen. Attempts to train a neural network with a time delay of one in three, four, and five dimensions did not meet with convergence. For the purposes of this paper, convergence is defined as a final mean squared error

less than or equal to 0.01. The results of training attempts with a time delay of one may be seen in Table 5.1.

A time delay of fourteen was used for all subsequent training attempts. Thirty-seven and forty nodes in the hidden layer proved sufficient for convergence in three dimensions. Networks utilizing four dimensions turned out to be less complex than those with three dimensions. Twenty-eight and thirty nodes reached convergence with four inputs. Networks with five inputs did not show greater accuracy than those with three or four inputs. Tables 5.1 and 5.2 display the resulting mean squared error following training attempts in three, four, and five dimensions with a time delay of fourteen. These tables report the best convergence achieved after several training attempts using different initial values of the weights and biases.

Table 5.1 Mean Squared Error of networks trained with data over portion of test domain.

Time Delay	Dimension	S ²	Mean Squared Error
1	3	5	0.040
	3	10	0.038
	4	5	0.029
	4	10	0.027
	5	5	0.028
	5	10	0.027
14	3	5	0.151
	3	15	0.048
	3	20	0.027

Table 5.2 Mean Squared Error of networks trained with data over entire test domain.

Time Delay	Dimension	S ²	Mean Squared Error
14	3	20	0.061
	3	30	0.026
	3	37	0.0098
	3	40	0.0078
	4	15	0.056
	4	20	0.038
	4	28	0.0101
	4	30	0.0071
	5	15	0.026
	5	25	0.025
	5	35	0.024
	5	45	0.023

The converged networks 3-37-1, 3-40-1, 4-28-1, and 4-30-1 were tested and compared regarding their accuracy and ability to generalize. The absolute error and relative error are a means of comparing the performance between architectures. The absolute and relative errors of each output value, as depicted graphically, are defined below.

$$abs = t - a, \text{deg}$$

$$rel = \frac{abs}{|t|}, \text{deg}$$

t = target network output value, a = actual network output value

Table 5.3 compares the mean values and standard deviations of the error magnitudes. Test points with an interval of six appear to generalize better than intervals of three and five. However, fifty percent of the test data used on an

Table 5.3 Comparison of error magnitude means and standard deviations for converged networks. Time delay of fourteen was used for all cases.

Dimension	S ²	Interval	Absolute Error		Relative Error	
			Mean	Standard Deviation	Mean	Standard Deviation
3	37	4	0.6806	0.9599	0.1832	0.5007
		3	1.4833	2.8034	0.7743	4.0236
		5	1.0272	1.8540	0.6411	3.5399
		6	1.1071	1.5470	0.3706	1.0252
3	40	4	0.6950	1.1000	0.1963	0.8803
		3	1.0243	1.0632	0.3940	1.3072
		5	1.1214	1.3098	0.4640	1.7230
		6	0.9152	0.9136	0.3121	0.9879
4	28	4	0.6000	0.5992	0.1762	0.5018
		3	1.0100	1.0210	0.3058	0.7715
		5	0.9090	1.2600	0.3683	1.1094
		6	0.8290	0.8862	0.2168	0.4113
4	30	4	0.8791	1.2814	0.3110	0.9342
		3	1.2210	1.7940	0.4248	1.2032
		5	1.2946	2.1190	0.5536	2.1643
		6	1.1176	2.0035	0.3847	1.0026
4	40	4	0.0213	0.0880	0.0032	0.0151
		5	1.4848	2.5460	0.8196	3.7137
3(1)	50	4	0.1694	0.1405	0.0279	0.0949
		3	3.1105	1.9363	0.3392	0.4475

interval of six were also a part of the training set. Intervals of three and five each share twenty-five percent of their test points with the training set. The intervals of three seem to perform slightly better than intervals of five.

Network 3-40-1 performs better with generalization than network 3-37-1. The mean squared error of network 4-30-1 is lower than that of network 4-28-1.

However, network 4-28-1 performs better with generalization. A large portion of possible convergence and generalization success is dependent upon the initial values of the network parameters. This is why it is important to train an architecture with several different initial values.

Network 4-28-1 is best adapted to the problem at hand. It is the least complex of the converged networks, with 169 network parameters, and generalizes well with all cases of the test data. A diagram of the final network architecture is displayed in Figure 5.1. The output and errors of network 4-28-1 at all intervals are shown graphically in Figures 5.2-5.5. Graphical results for the other converged networks may be found in Appendix A.

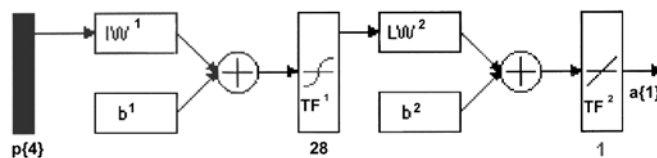


Figure 5.1 Layer diagram for selected network 4-28-1.

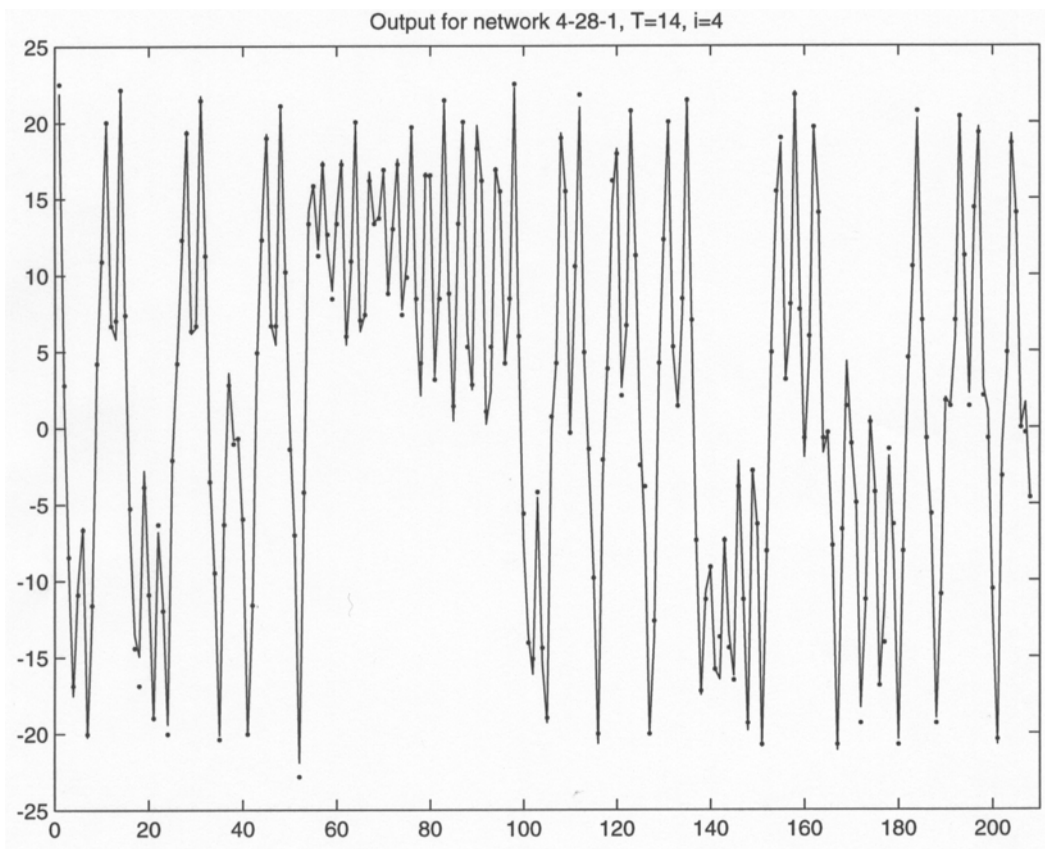


Figure 5.2 Output and errors for network 4-28-1 with the training set interval of 4.

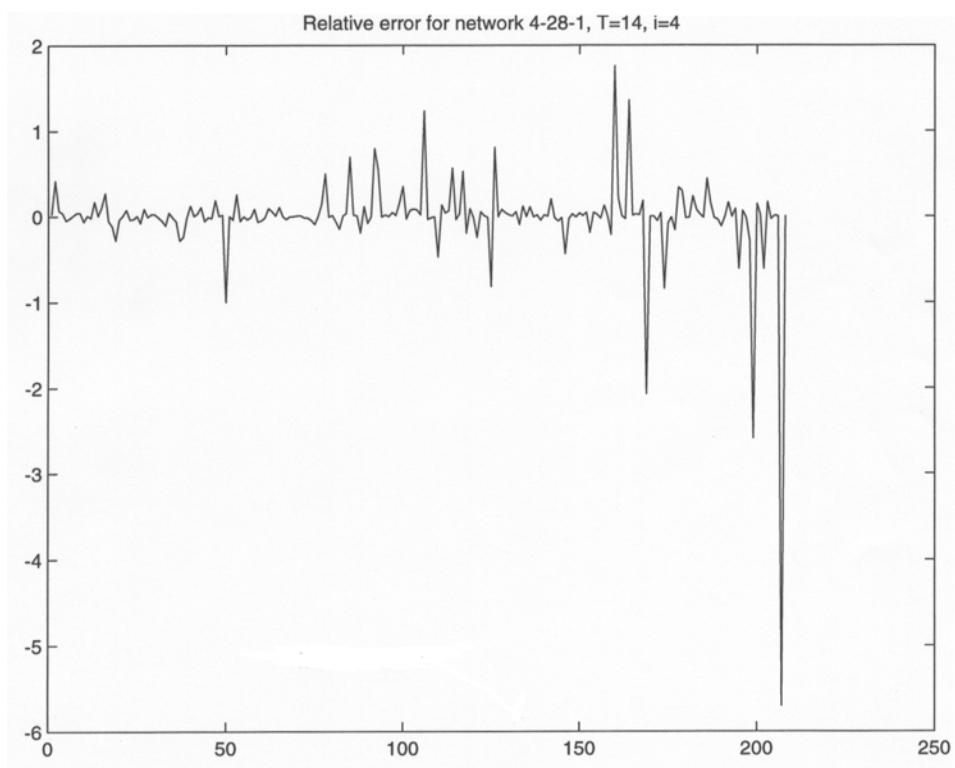
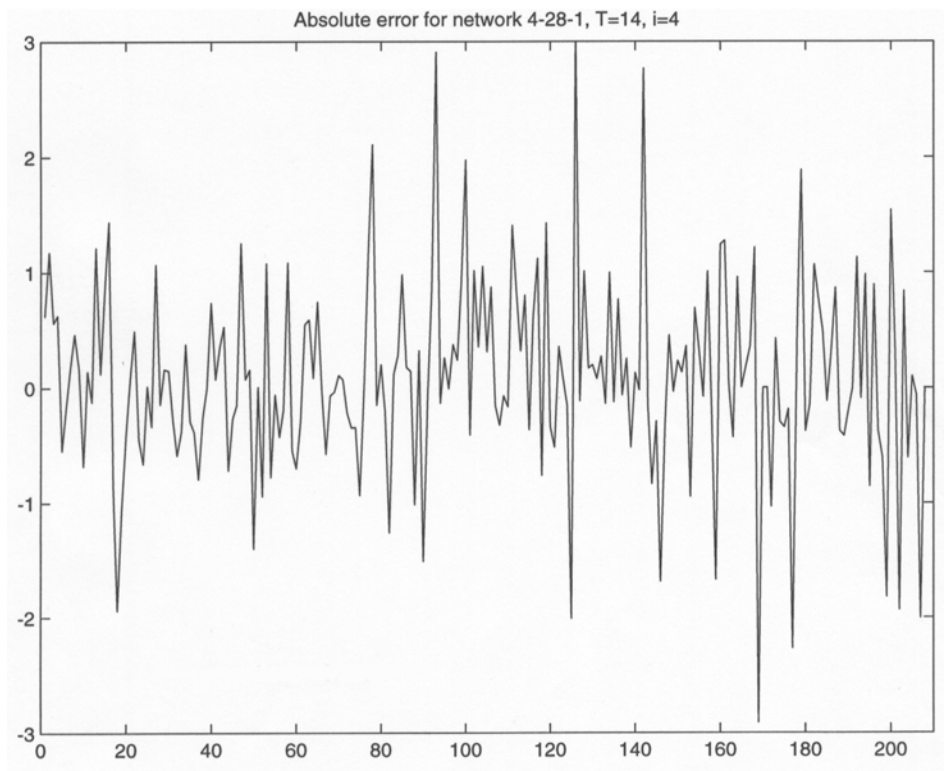


Figure 5.2 Continued

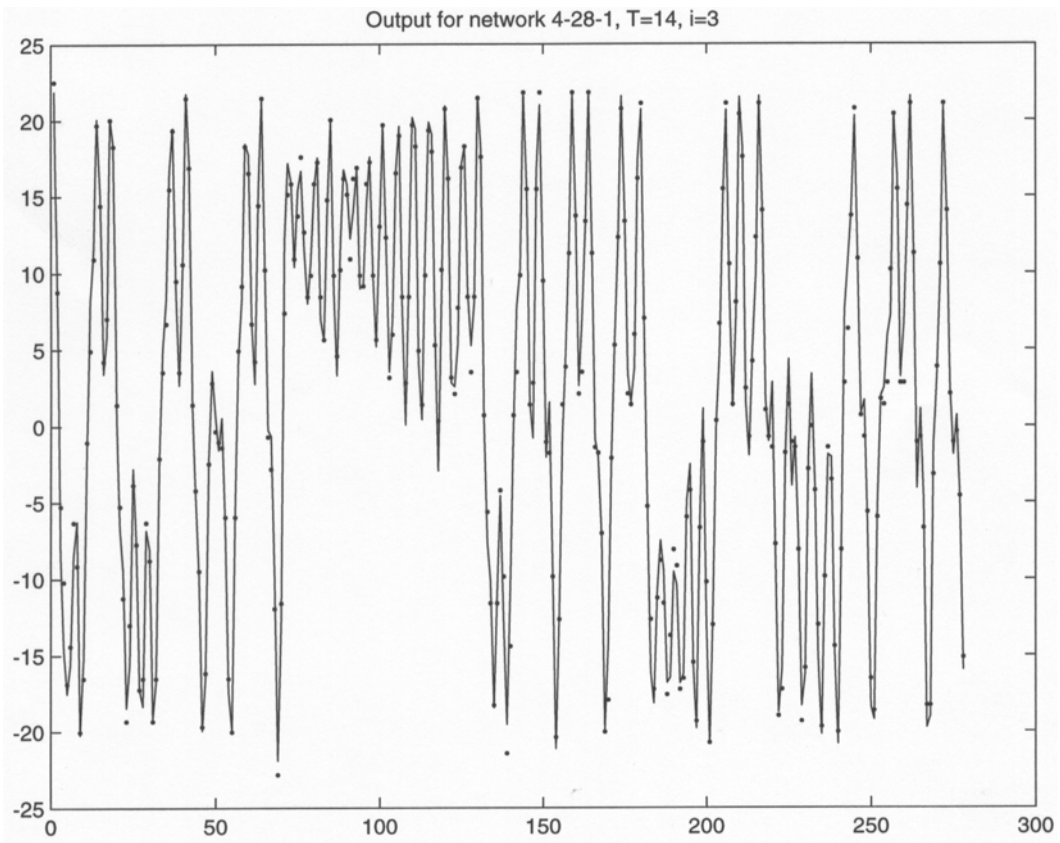


Figure 5.3 Output and errors for network 4-28-1 with the test set interval of 3.

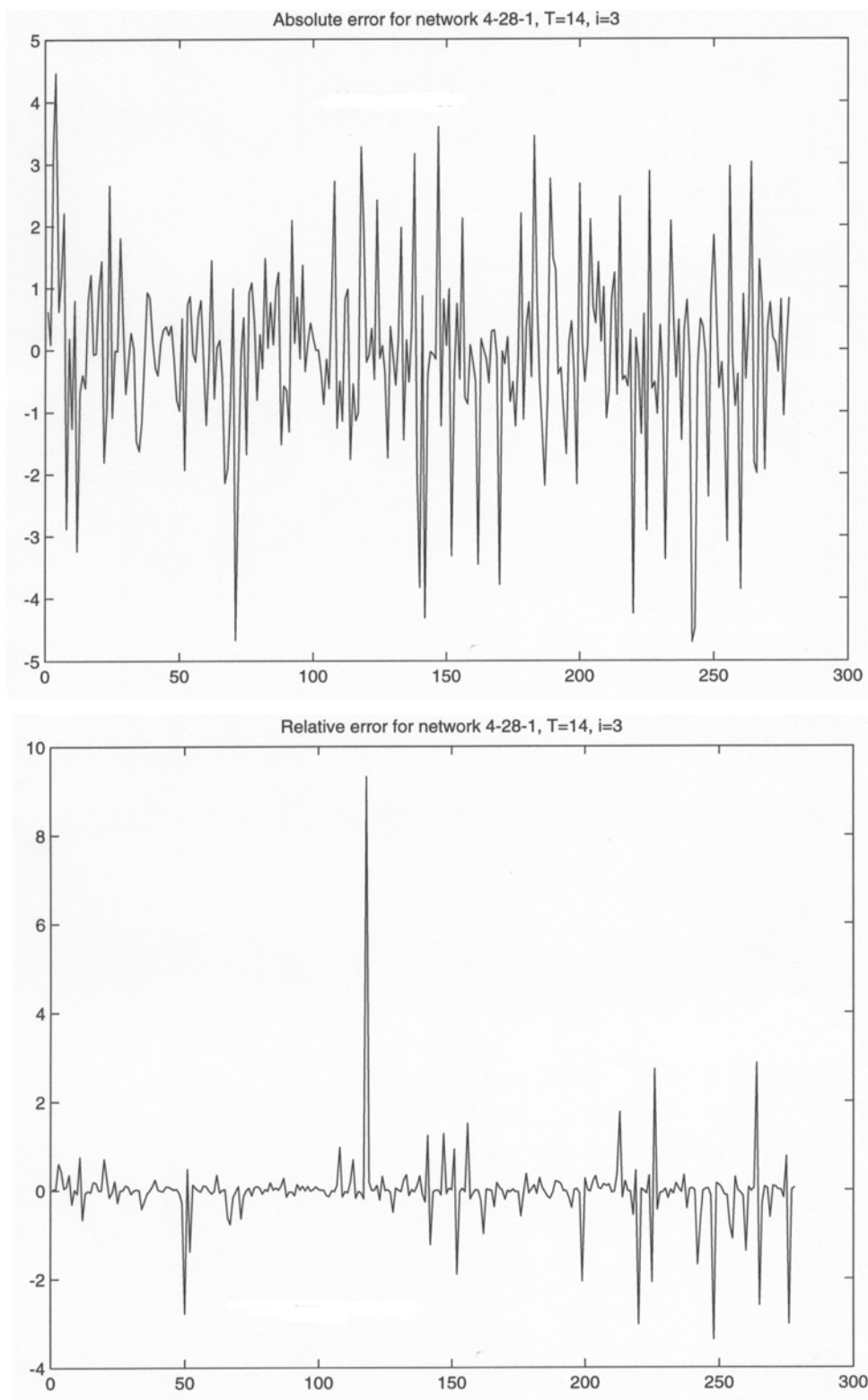


Figure 5.3 Continued

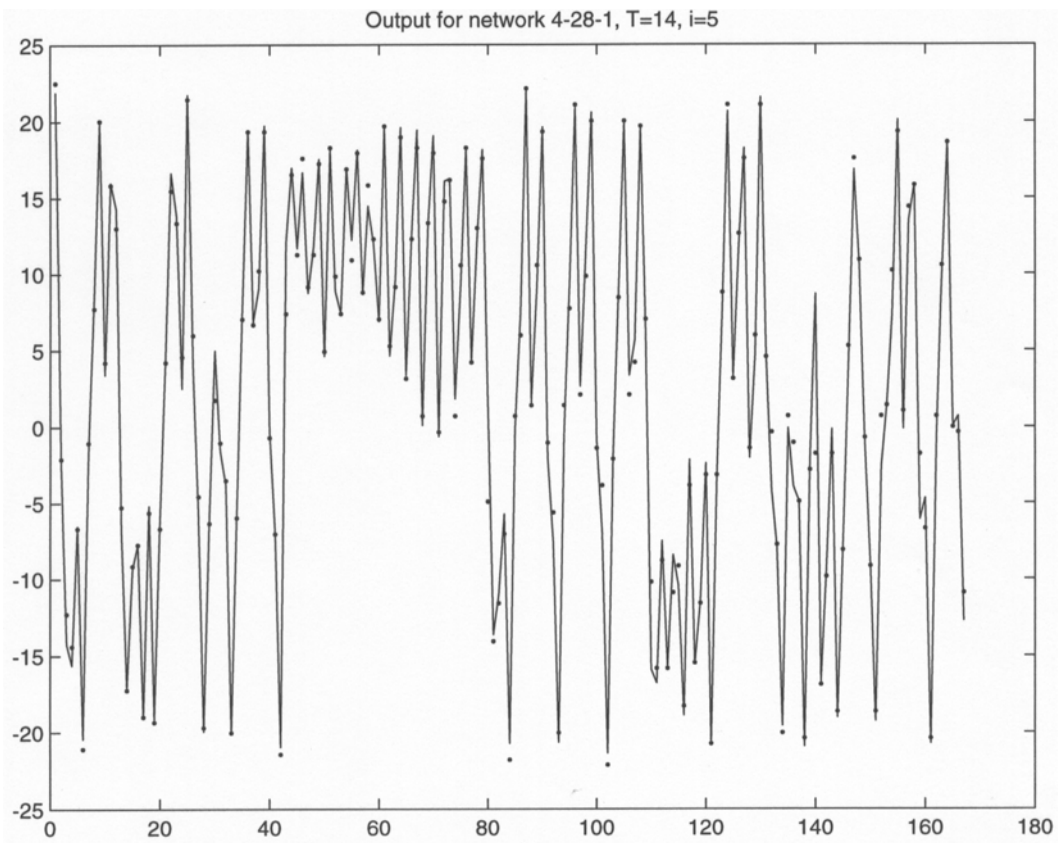


Figure 5.4 Output and errors for network 4-28-1 with the test set interval of 5.

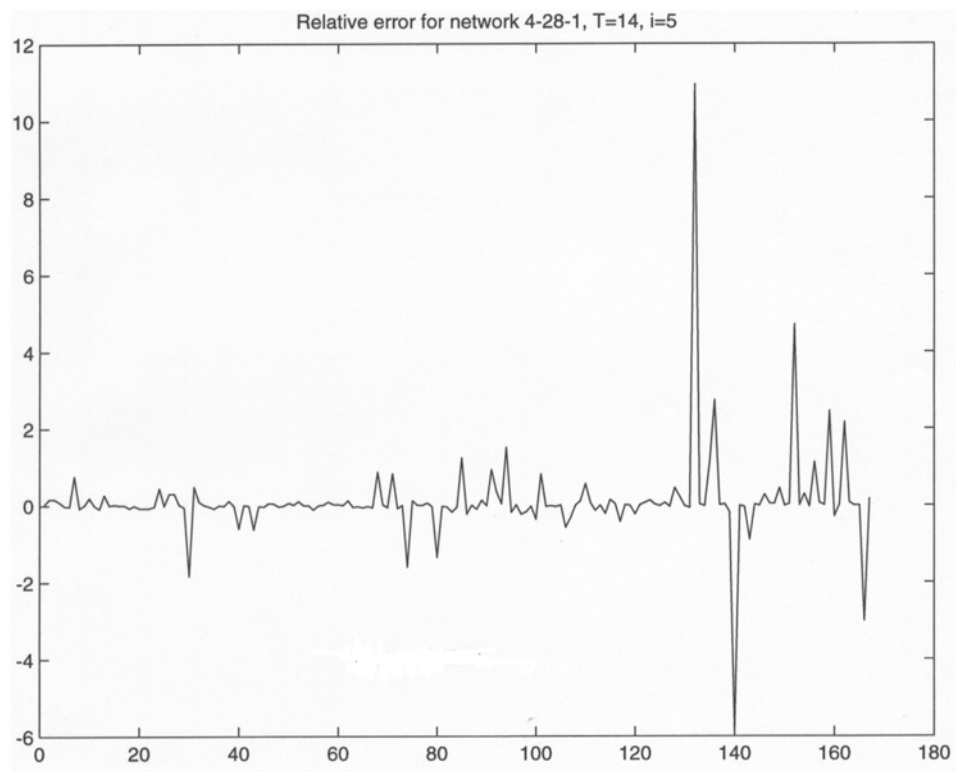
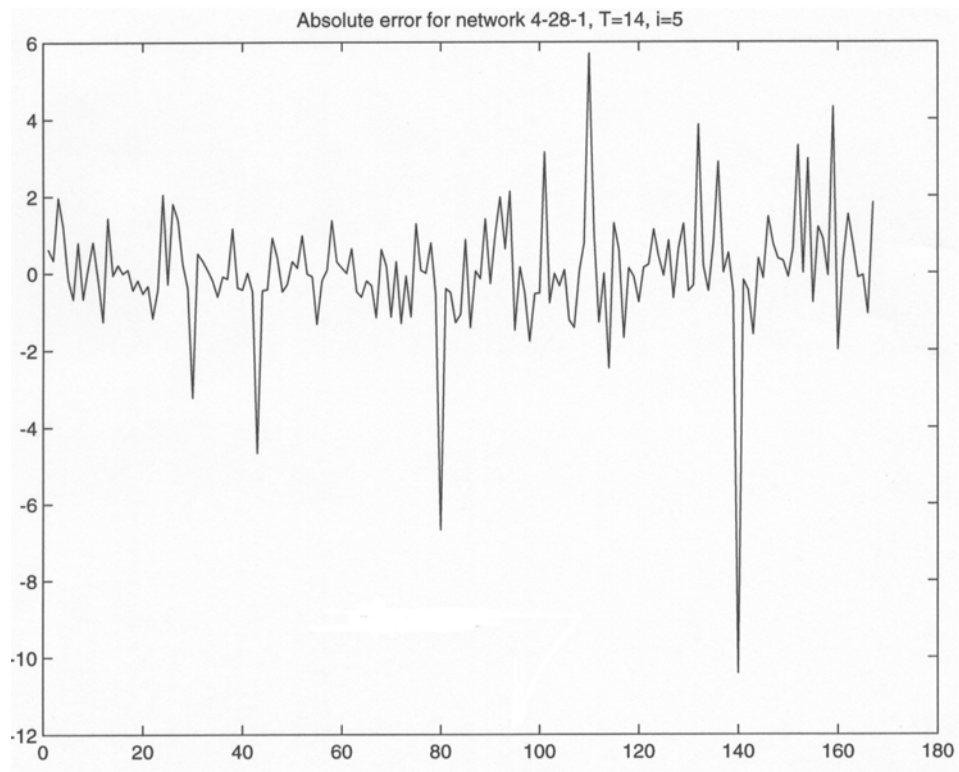


Figure 5.4 Continued

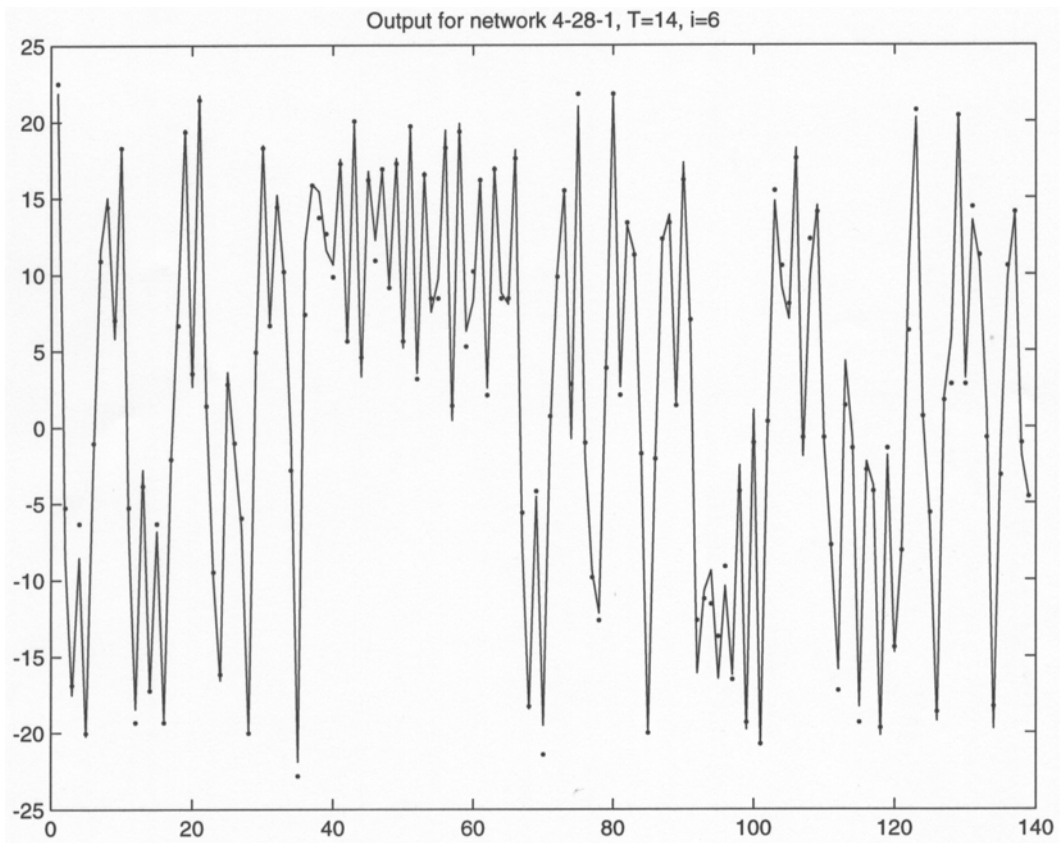


Figure 5.5 Output and errors for network 4-28-1 with the test set interval of 6.

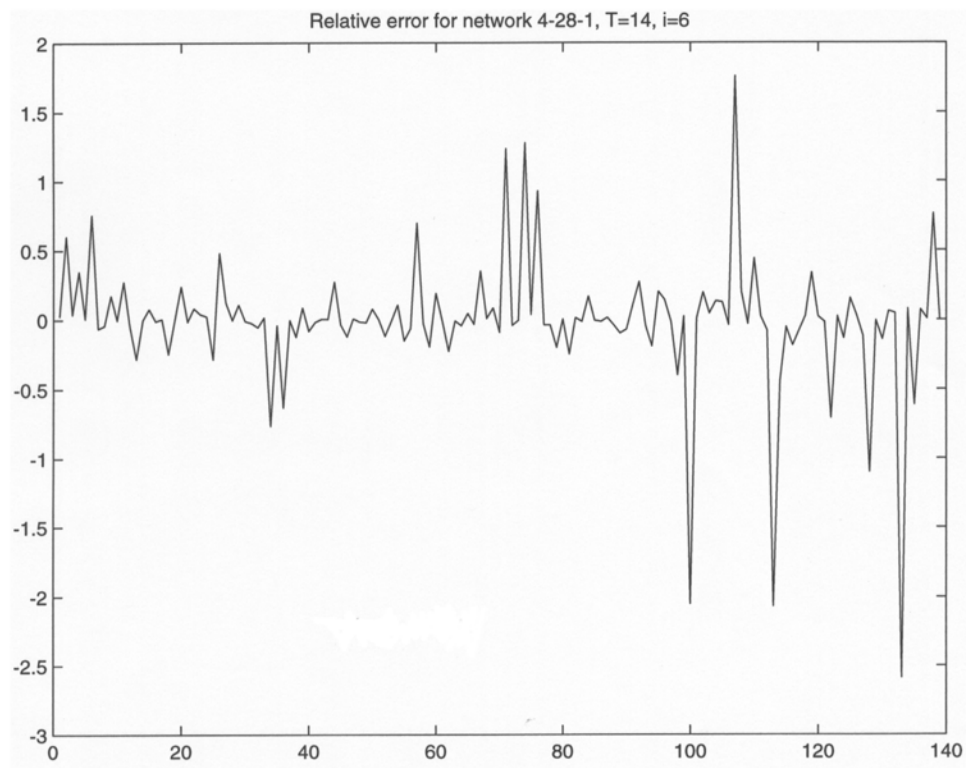
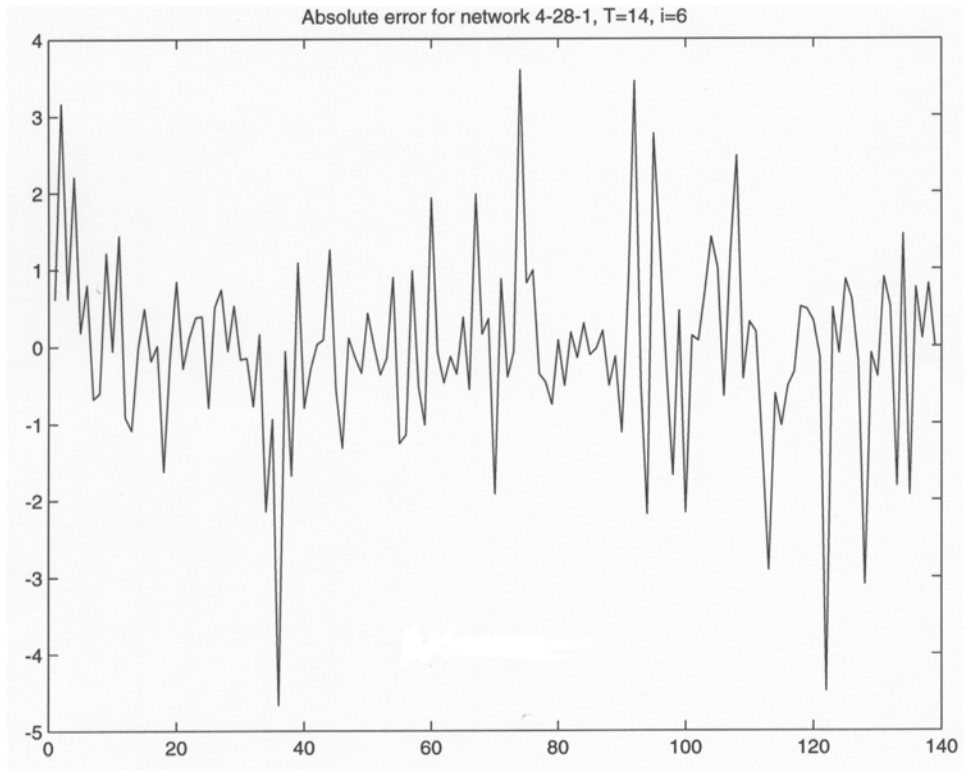


Figure 5.5 Continued

Recall from Chapter Three that it is possible to construct a network that overfits problem data. Network 4-40-1 is an example of such a network. This architecture possesses 241 parameters. The training set of interval four presents 208 data points. The network follows the training set remarkably well. However, the generalization to an interval of five is extremely poor. The network has exactly learned the function represented by the training set. Figures 5.6-5.7 present the output and errors of network 4-40-1.

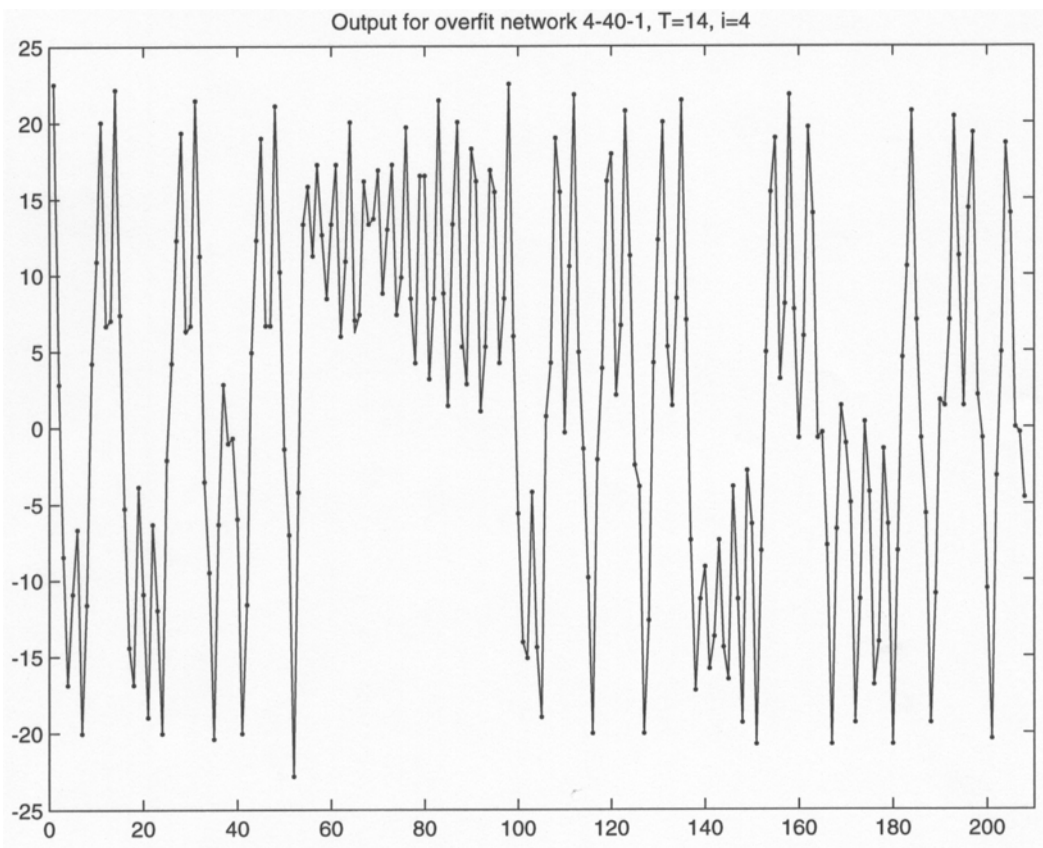


Figure 5.6 Output and errors for overfit network 4-40-1 with the training set interval of 4.

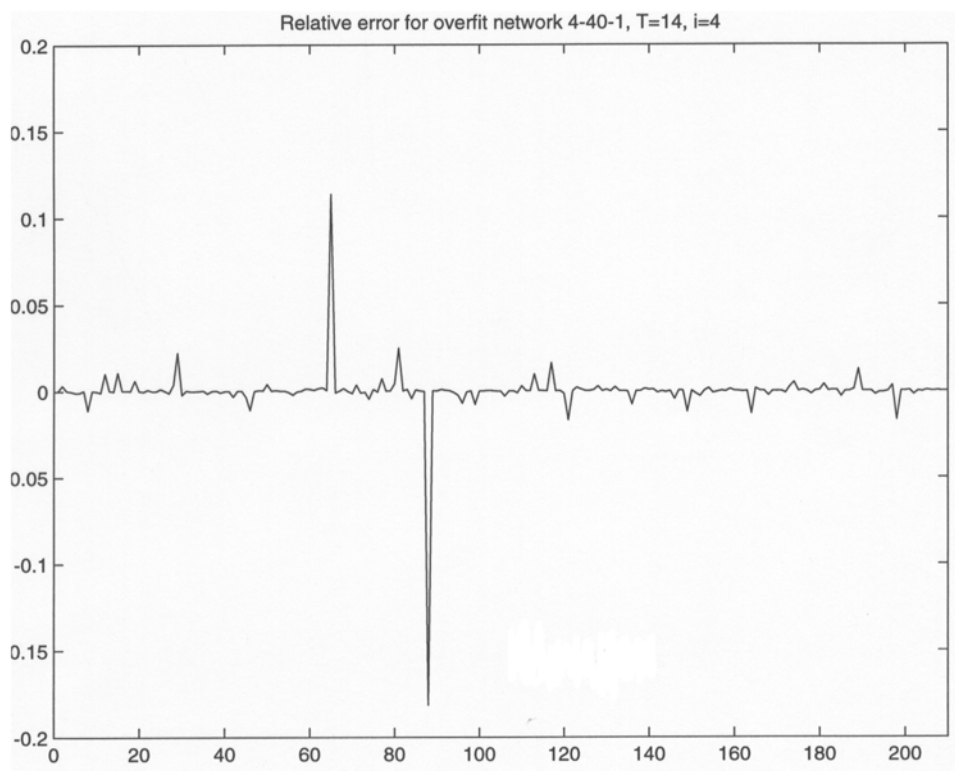
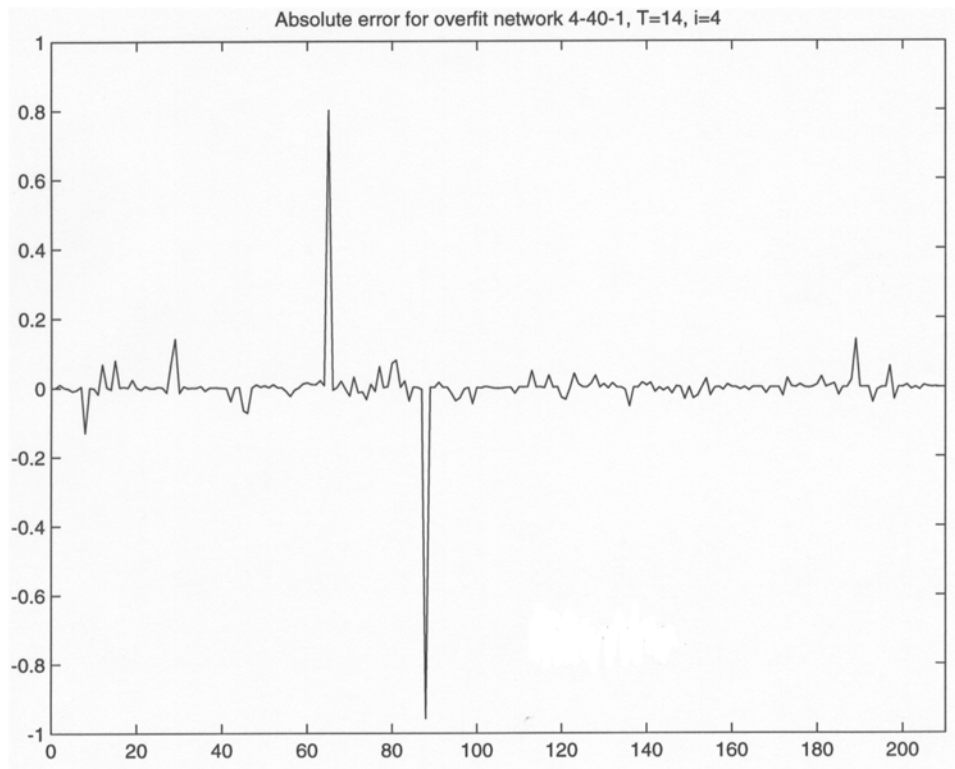


Figure 5.6 Continued

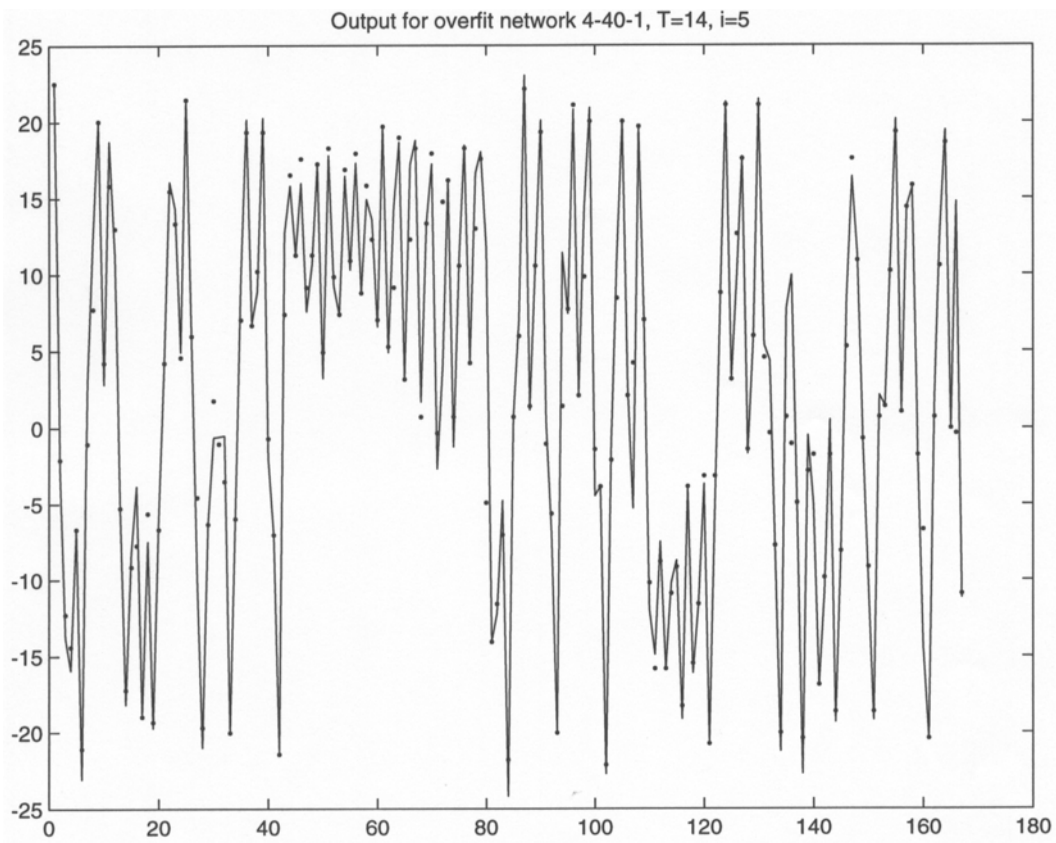


Figure 5.7 Output and errors for overfit network 4-40-1 with the test set interval of 5.

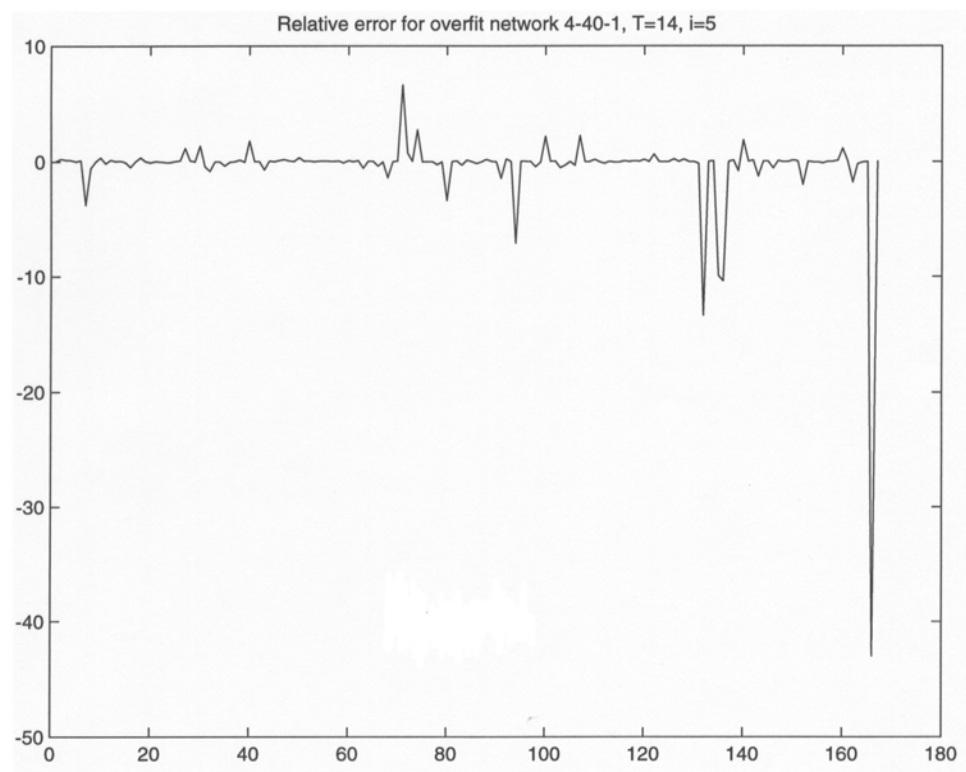
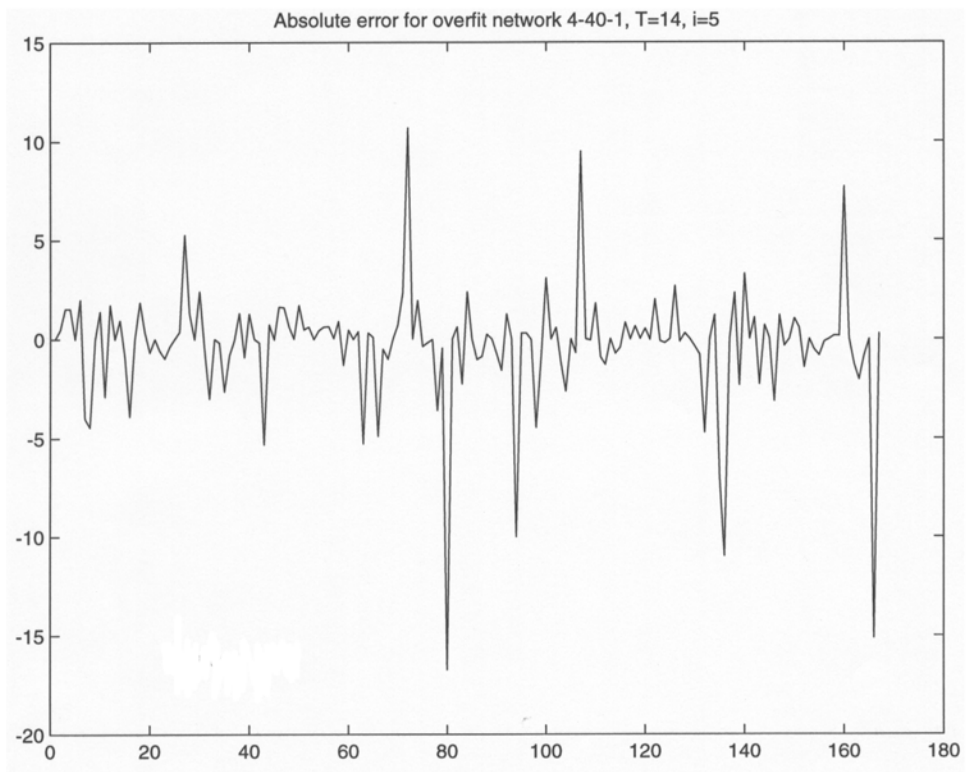


Figure 5.7 Continued

5.2 Performance of Networks With Dependence on Sting Angle

The utility of the described neural network models were finally tested by incorporating a dependence upon the sting angle. A fourth input was appended to the three dimensional architecture. The input matrix for the training set was of the following form.

$$\begin{aligned}
 \mathbf{RawData} &= \begin{bmatrix} \sigma_1, x_1, x_2, \dots, x_n \\ \sigma_2, y_1, y_2, \dots, y_n \end{bmatrix} \\
 \mathbf{DataSet} &= \begin{bmatrix} \sigma_1 x_1, x_{15}, x_{29}, x_{43}, \dots \\ \sigma_2, y_1, y_{15}, y_{29}, y_{43}, \dots \end{bmatrix} \\
 \mathbf{t} &= [x_{43}, x_{99}, y_{43}, y_{99}, \dots] \\
 \mathbf{P} &= \begin{bmatrix} x_1 & x_{57} & y_1 & y_{57} \cdots \\ x_{15} & x_{71} & y_{15} & y_{71} \cdots \\ x_{29} & x_{85} & y_{29} & y_{85} \cdots \\ \sigma_1 & \sigma_1 & \sigma_2 & \sigma_2 \cdots \end{bmatrix}
 \end{aligned}$$

Sting angles of 30° and 35° were used for the training data. The converged network was then tested with roll angle measurements recorded at a sting angle of 33° . A network with fifty nodes in the hidden layer showed remarkable success with the test data at an interval of three. The resulting errors of network 3(1)-50-1 are shown in Table 5.3. Figures 5.8-5.9 illustrate this information graphically.

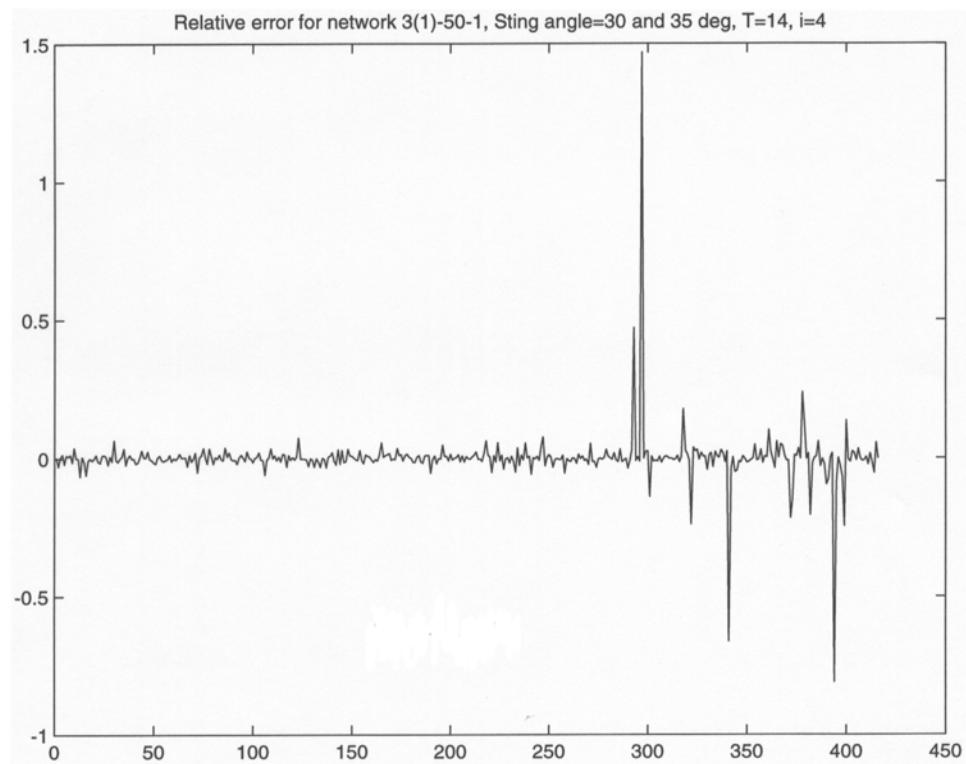
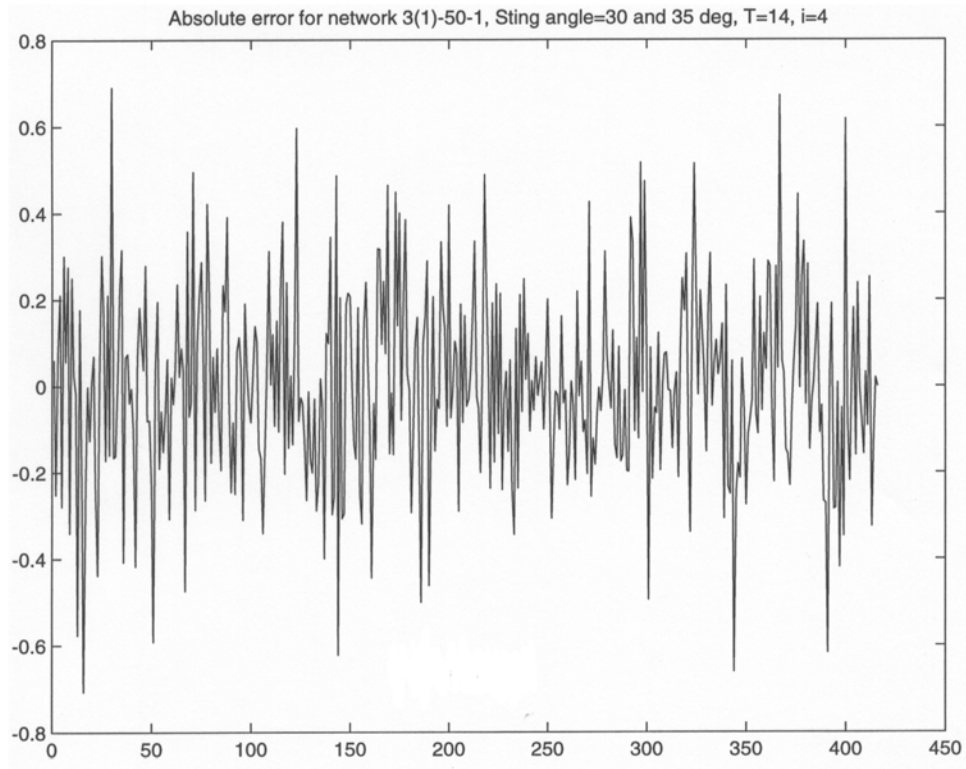


Figure 5.8 Errors for sting angle dependent network 3(1)-50-1 with the training set interval of 4. The training set includes data from sting angles of 30° and 35° only.

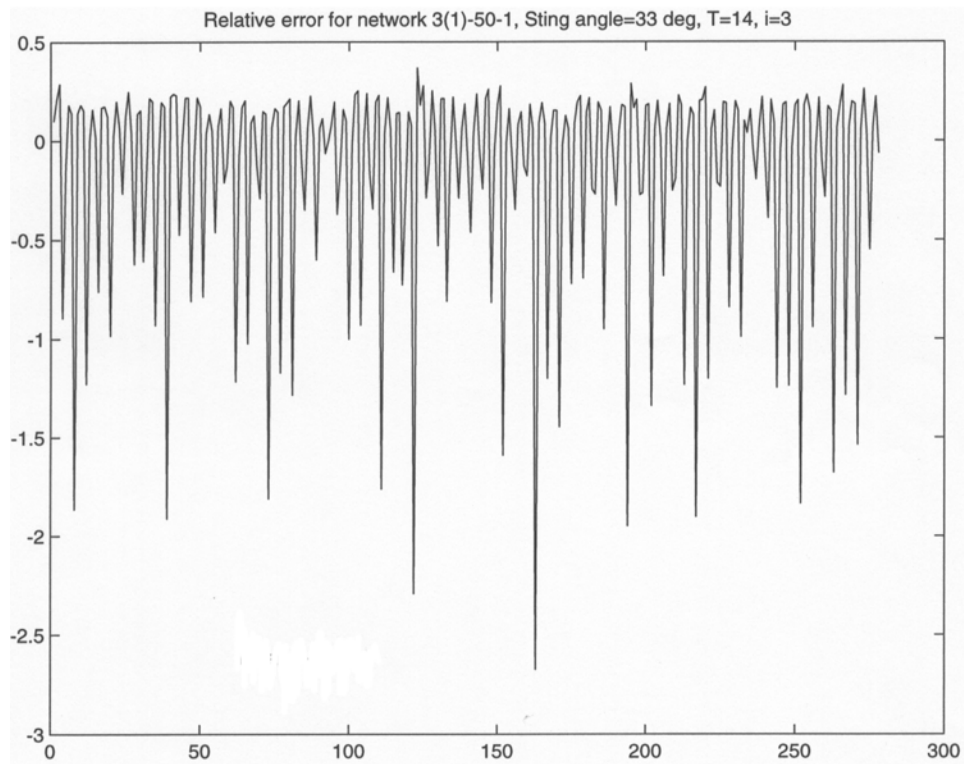
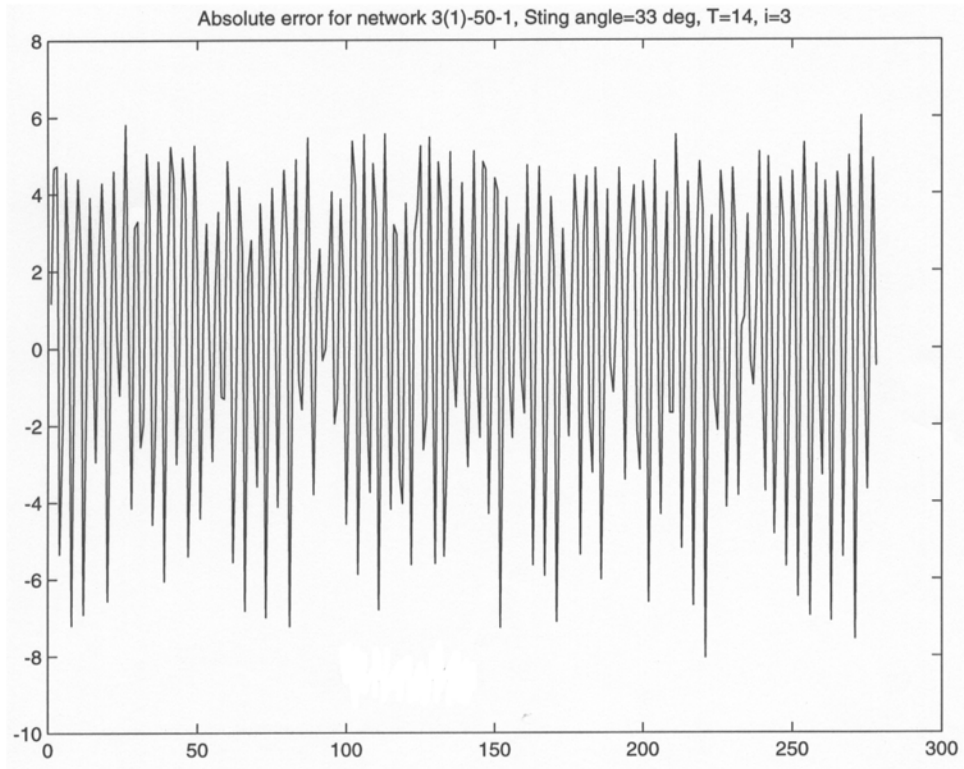


Figure 5.9 Errors for sting angle dependent network 3(1)-50-1 with the test set interval of 3. The test set includes data from a sting angle of 33° .

CHAPTER SIX

Conclusions

Multilayer feedforward backpropagation networks are well suited to approximating unsteady, nonlinear aerodynamic data of the type recorded by Dr. Pelletier. The network models are capable of simulating short-term roll motion behavior. Future work might include incorporating the output of the neural network as a part of the inputs to continue the roll angle approximation. Dynamic data is essential for creation of an accurate control system. The development of a controller that incorporates dependence upon the sting angle would require much training data at intermediate sting angles. Any serious effort toward development of a control system requires high fidelity, real-world data.

WORKS CITED

WORKS REFERENCED

- ¹Pelletier, A. *A Study of the Nonlinear Aerodynamic Characteristics of a Slender Double-Delta Wing in Roll*. Ph.D. Dissertation, University of Notre Dame, Indiana, 1998.
- ²Abarbanel, H. *Analysis of Observed Chaotic Data*. Institute for Nonlinear Science, Springer-Verlag, New York, 1996.
- ³Peitgen, H., Jurgens, H., Saupe, D. *Chaos and Fractals New Frontiers of Science*. Springer-Verlag, New York, 1992.
- ⁴Hagan, M., Demuth, H., Beale, M. *Neural Network Design*. PWS Publishing Company, Massachusetts, 1996.
- ⁵Demuth, H., Beale, M. *Neural Network Toolbox User's Guide, For Use with MATLAB[®]*, Version 4. The Mathworks, Inc., Massachusetts, 2001.

WORKS CONSULTED

Atkins, D. *Using Neural Networks For Interpolating Aerodynamic Data*. M. S.

Thesis, University of Tennessee, Knoxville, 1999.

Collins, F., Kimble, K., Henderson, A., Lo, C. "Aerodynamic Model for Chaotic Wing Rock for a Double-Delta Wing Model." Presented at the 40th AIAA

Aerospace Sciences Meeting and Exhibit, Reno, NV, January, 2002.

APPENDICES

APPENDIX A

Additional Figures

Graphical results of network outputs and errors for networks 3-37-1, 3-40-1, and 4-30-1 are included in Appendix A.

LIST OF FIGURES IN APPENDIX A

Figure A.1	Output and errors for network 3-37-1 with the training set interval of 4.	58
Figure A.2	Output and errors for network 3-37-1 with the test set interval of 3.	60
Figure A.3	Output and errors for network 3-37-1 with the test set interval of 5.	62
Figure A.4	Output and errors for network 3-37-1 with the test set interval of 6.	64
Figure A.5	Output and errors for network 3-40-1 with the training set interval of 4.	66
Figure A.6	Output and errors for network 3-40-1 with the test set interval of 3.	68
Figure A.7	Output and errors for network 3-40-1 with the test set interval of 5.	70
Figure A.8	Output and errors for network 3-40-1 with the test set interval of 6.	72
Figure A.9	Output and errors for network 4-30-1 with the training set interval of 4.	74
Figure A.10	Output and errors for network 4-30-1 with the test set interval of 3.	76
Figure A.11	Output and errors for network 4-30-1 with the test set interval of 5.	78
Figure A.12	Output and errors for network 4-30-1 with the test set interval of 6.	80

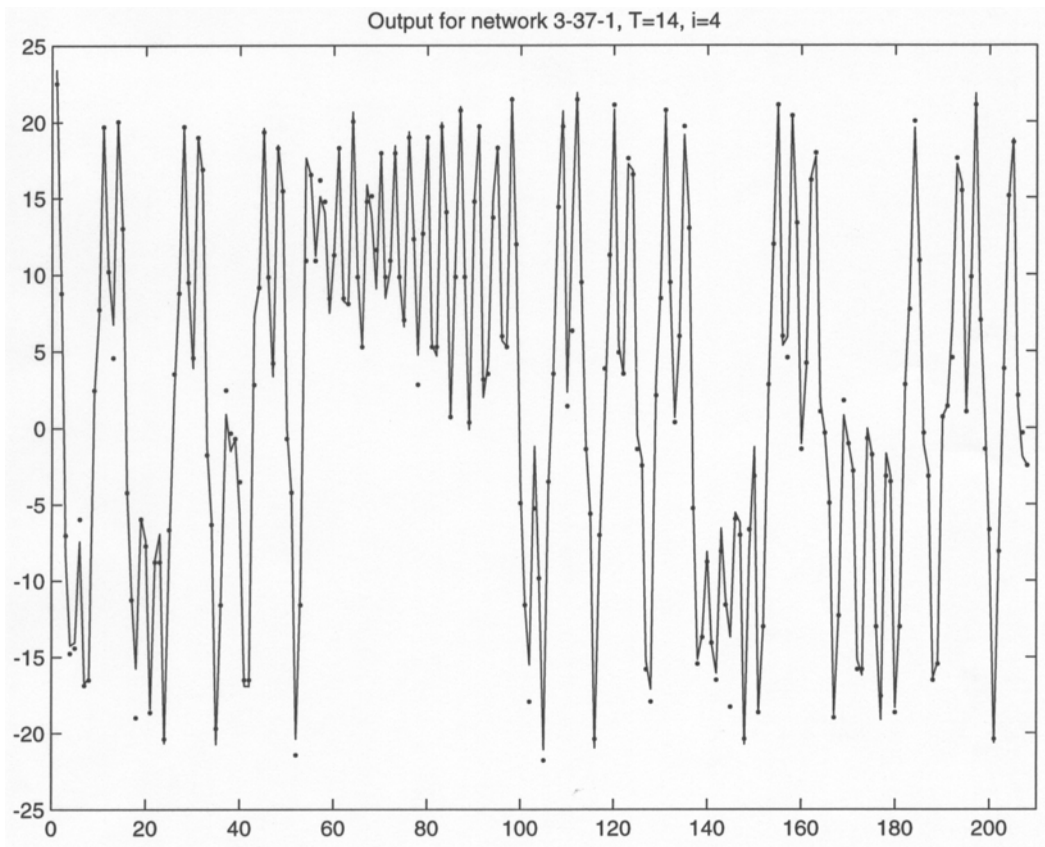


Figure A.1 Output and errors for network 3-37-1 with the training set interval of 4.

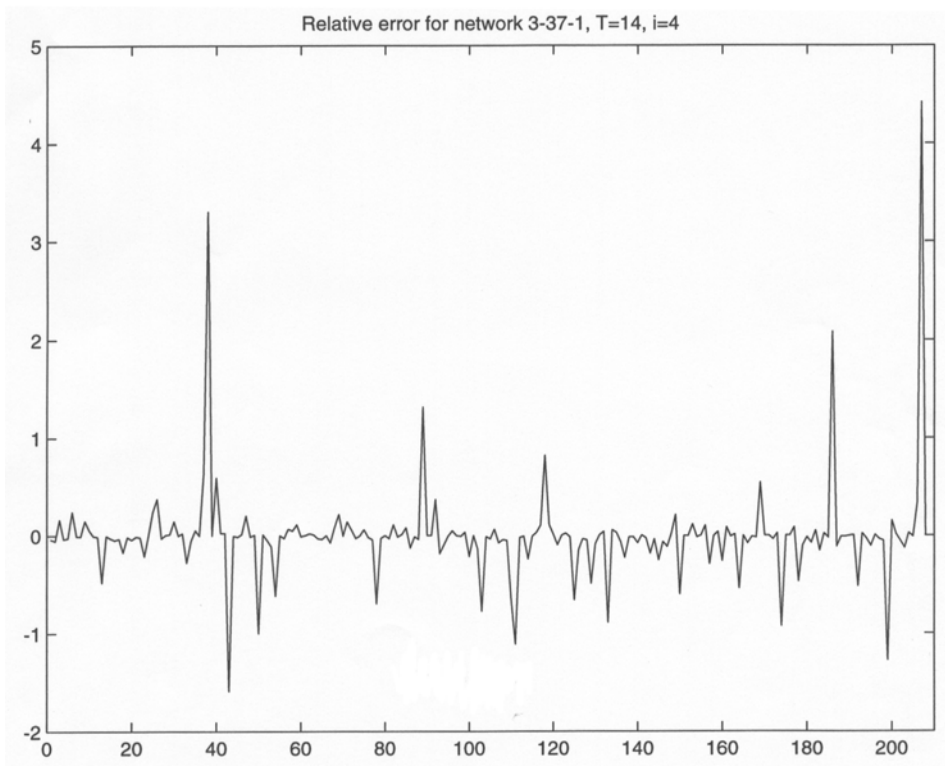
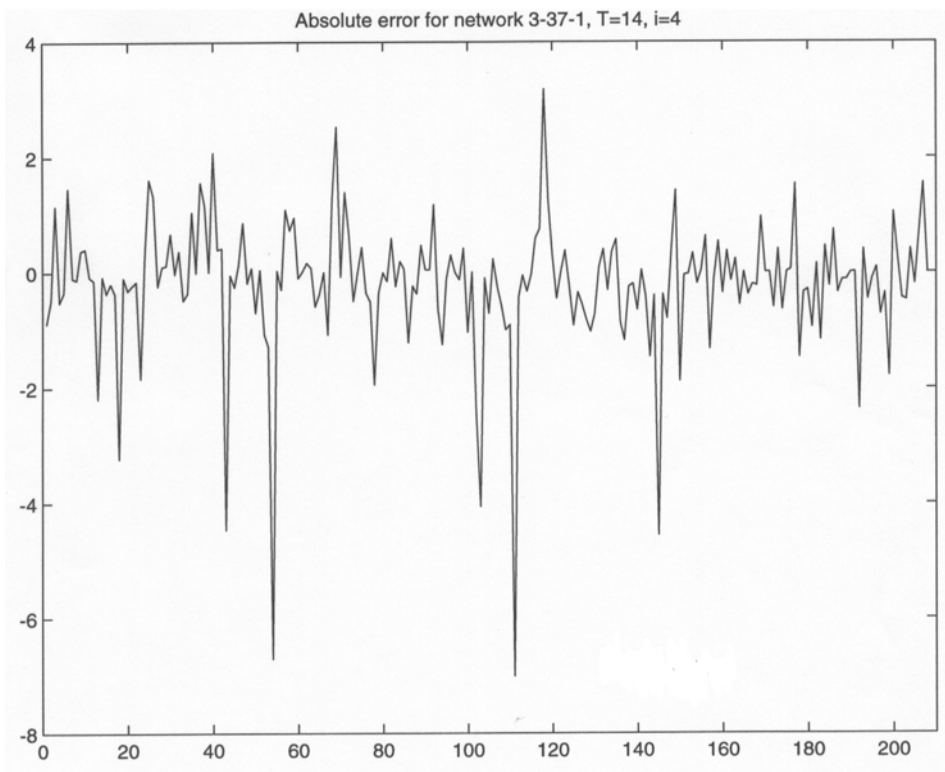


Figure A.1 Continued

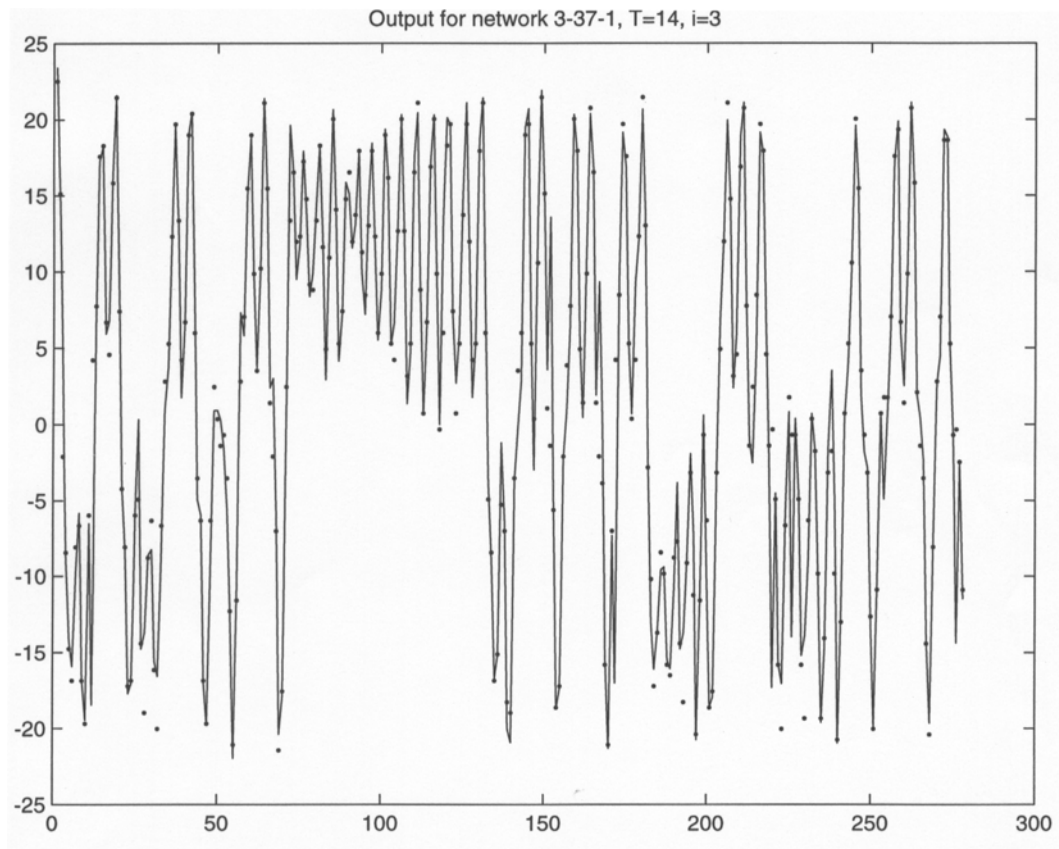


Figure A.2 Output and errors for network 3-37-1 with the test set interval of 3.

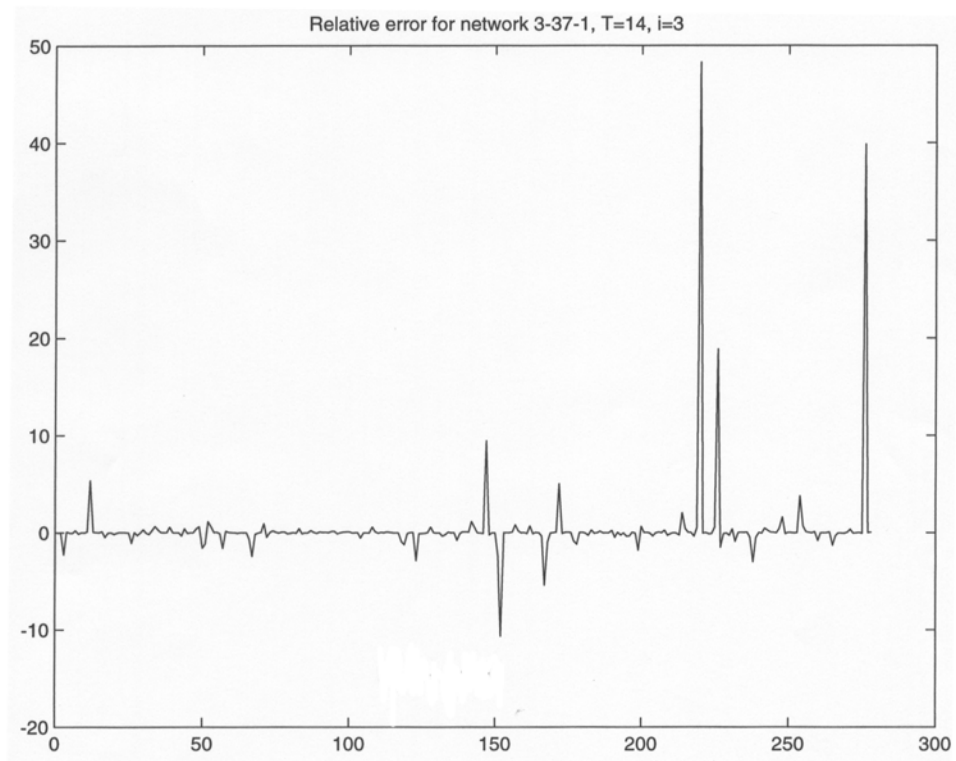
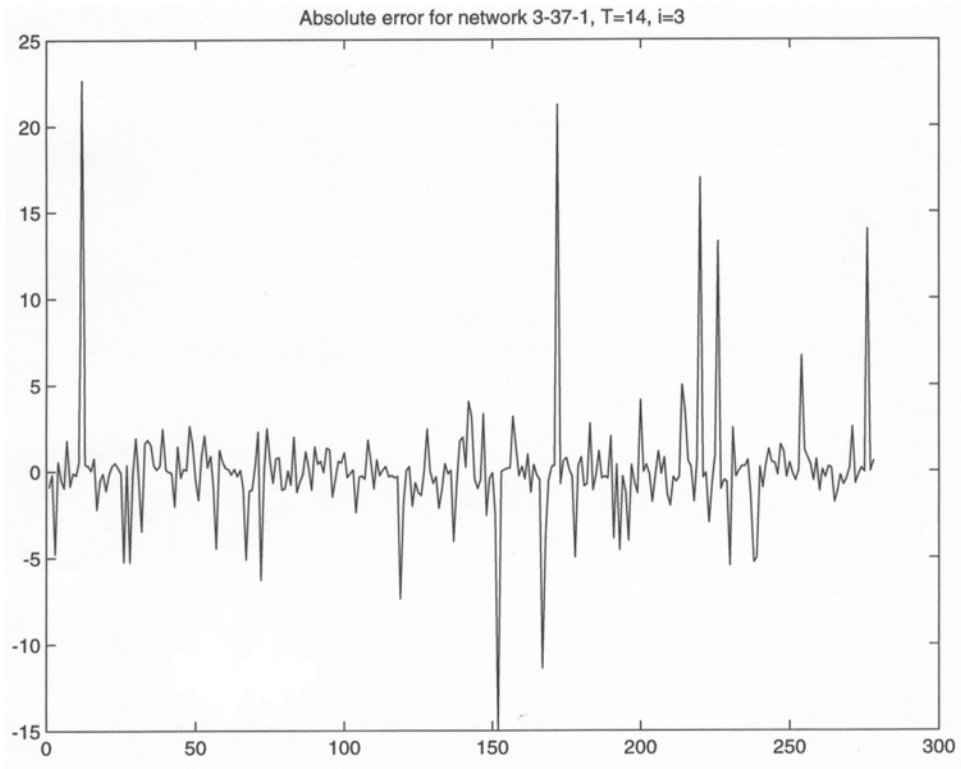


Figure A.2 Continued

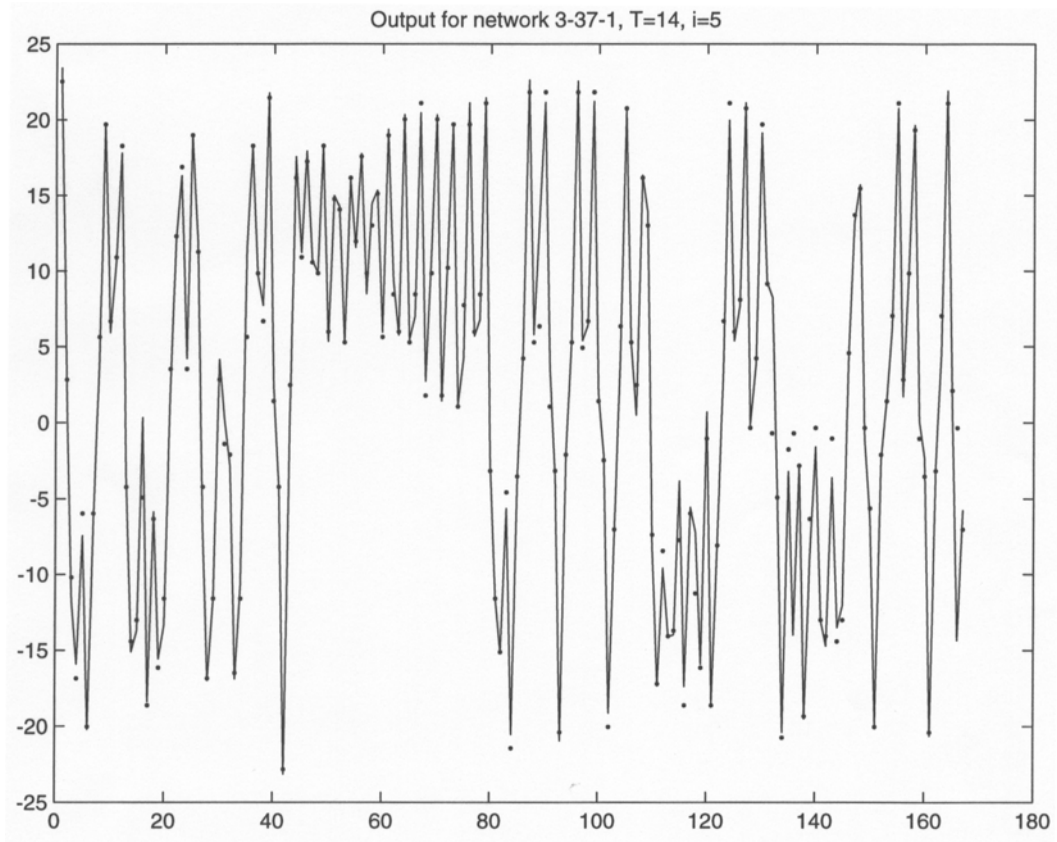


Figure A.3 Output and errors for network 3-37-1 with the test set interval of 5.

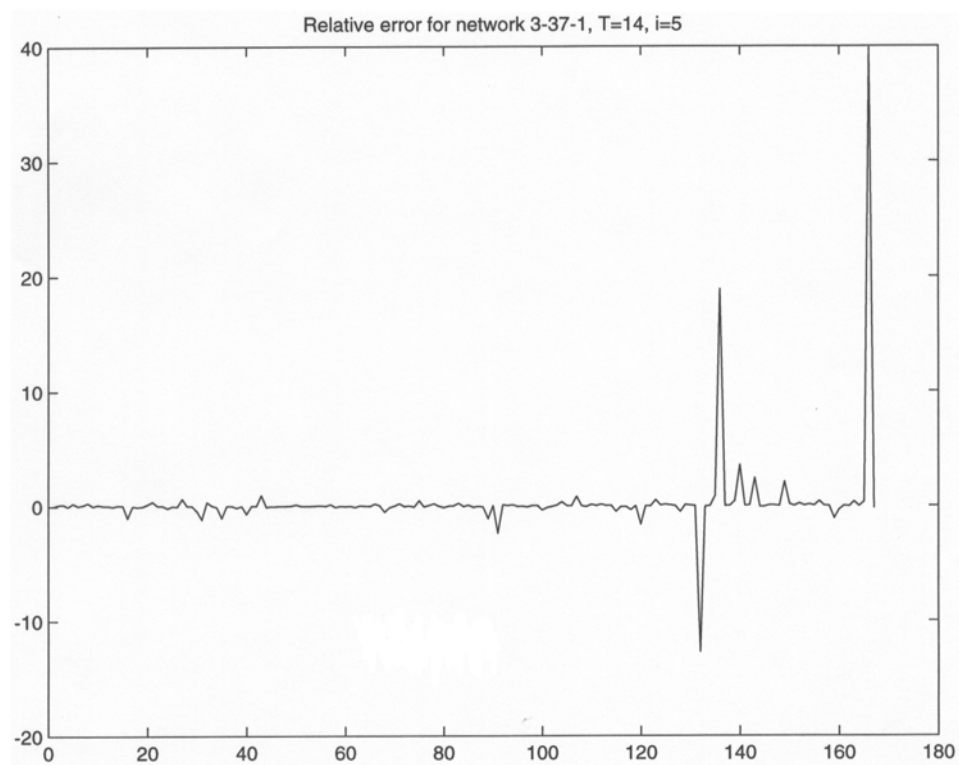
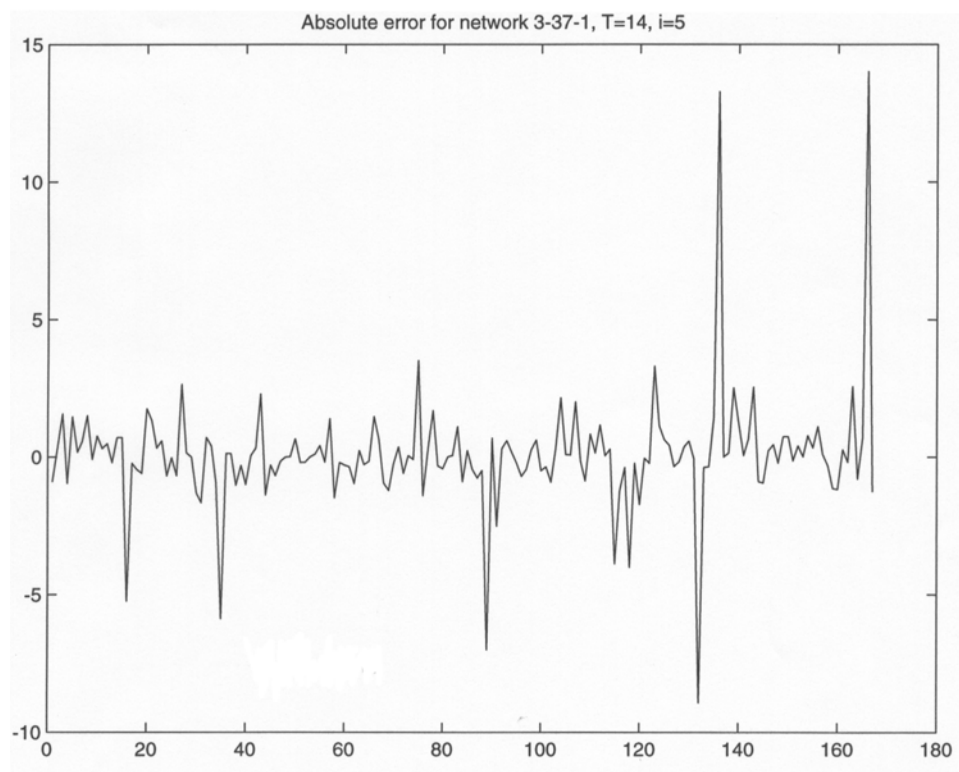


Figure A.3 Continued

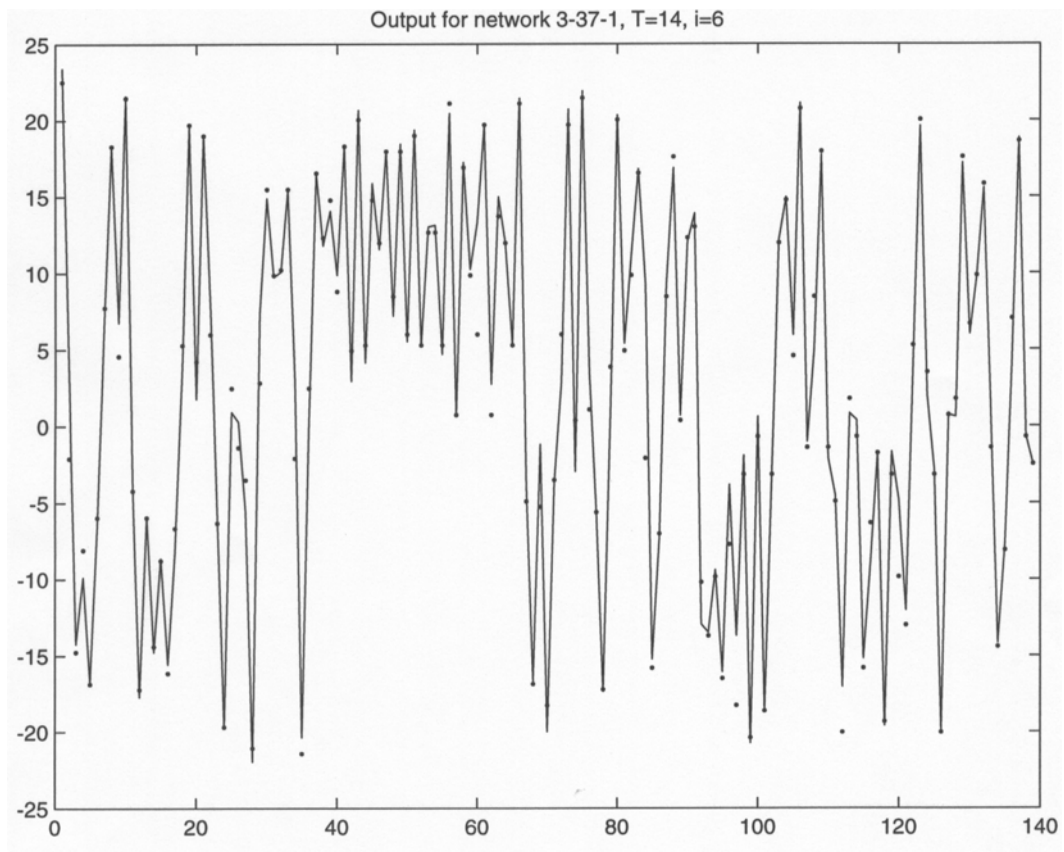


Figure A.4 Output and errors for network 3-37-1 with the test set interval of 6.

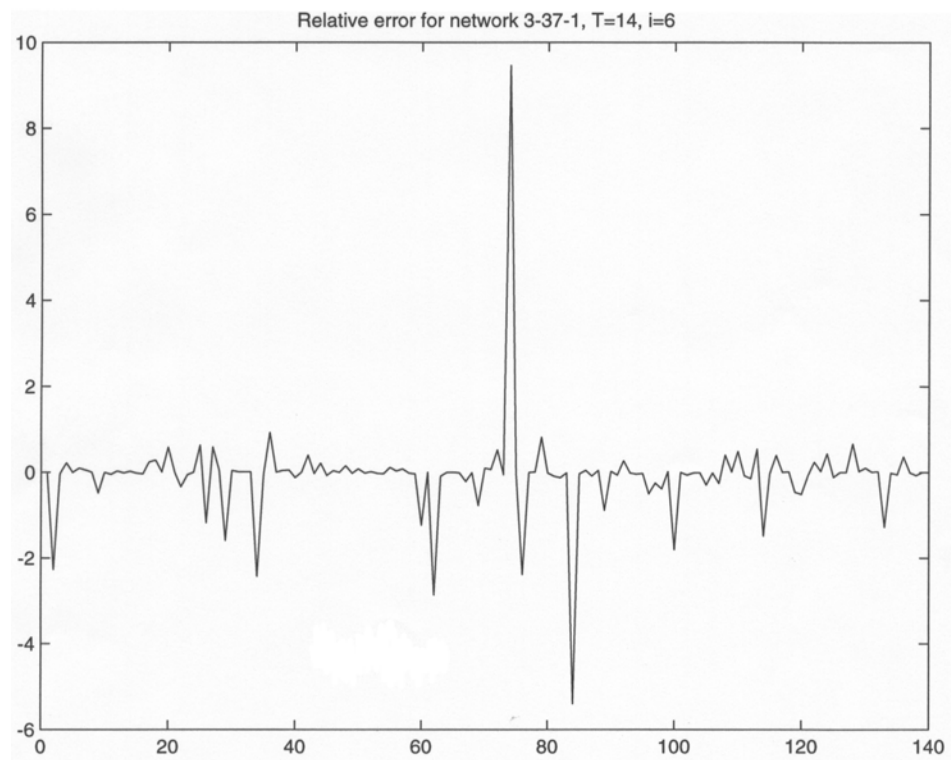
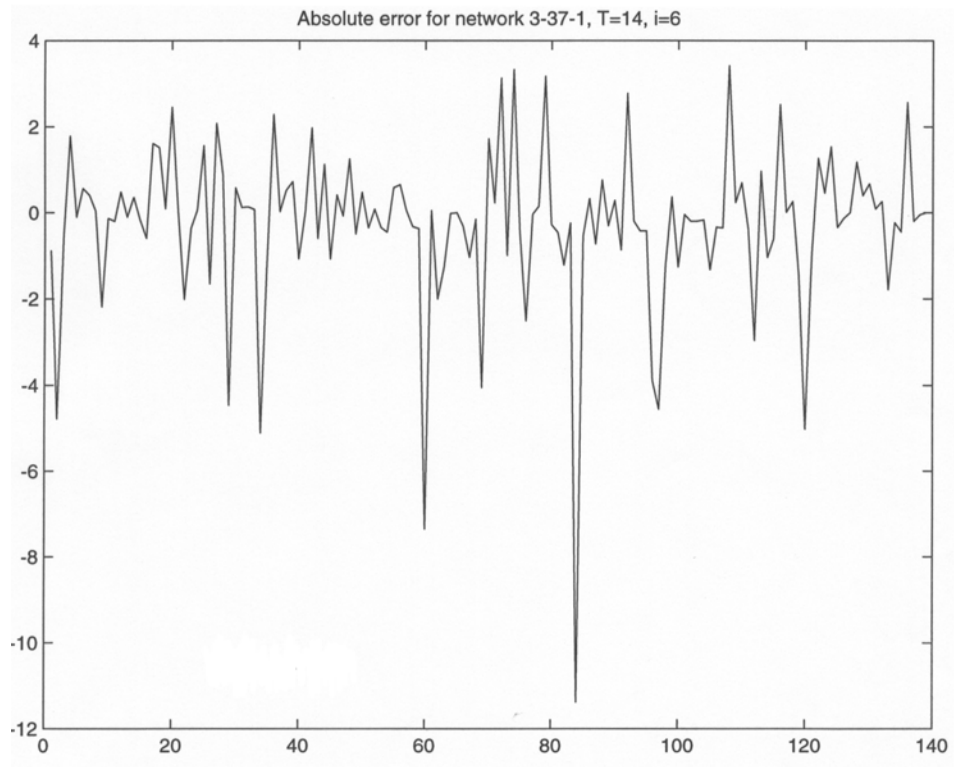


Figure A.4 Continued

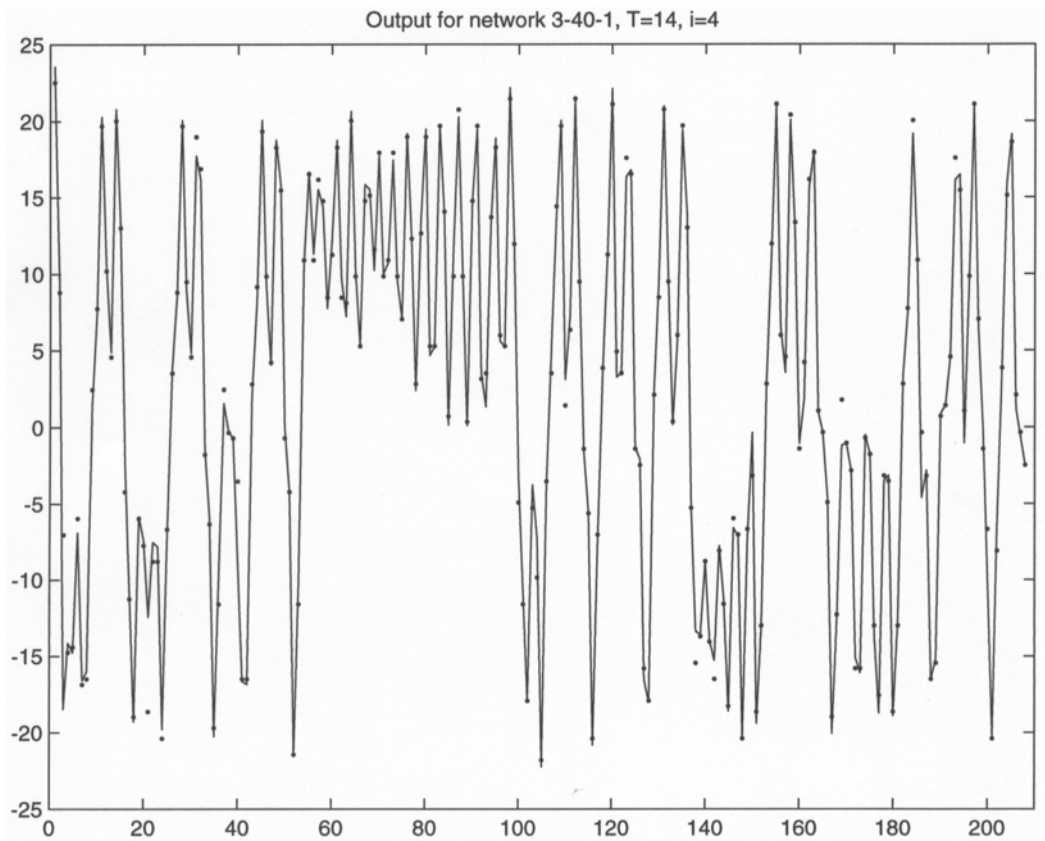


Figure A.5 Output and errors for network 3-40-1 with the training set interval of 4.

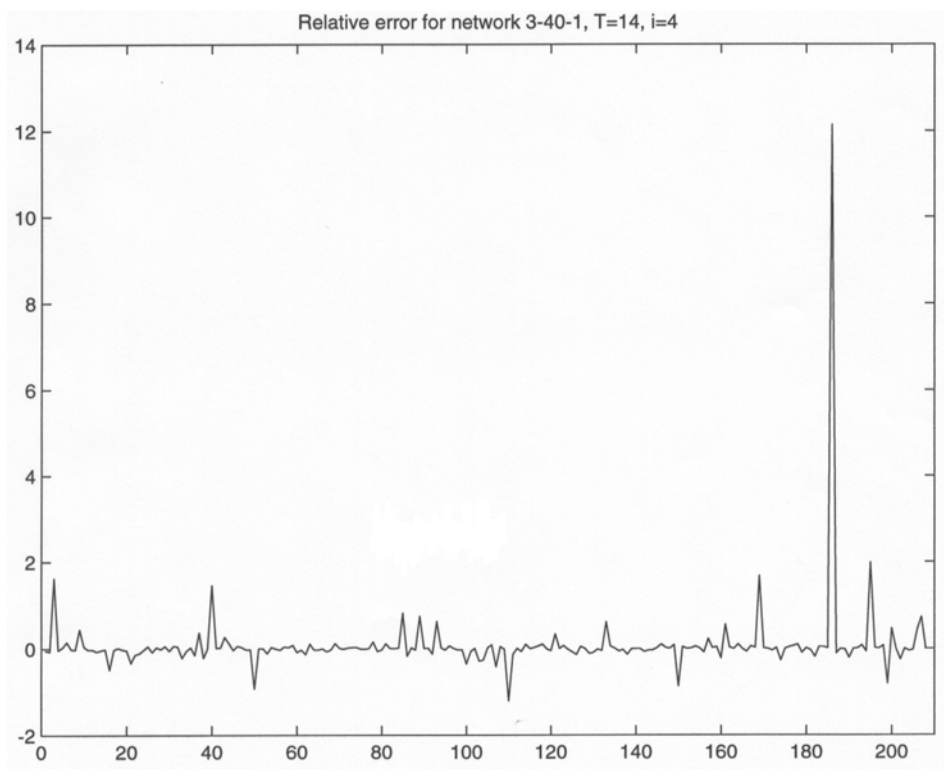
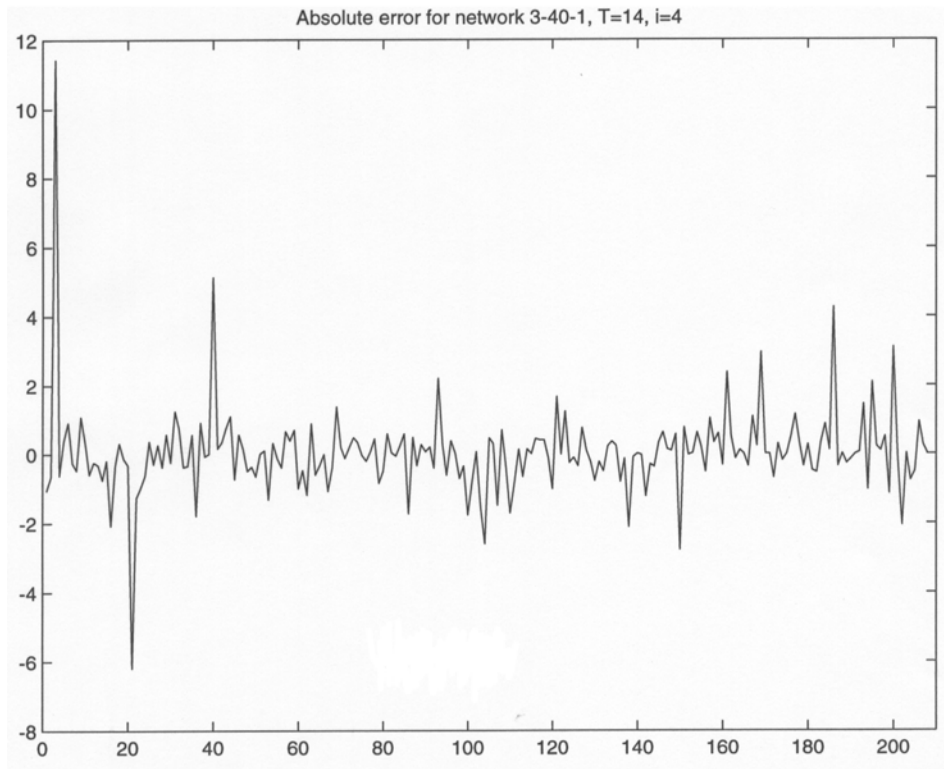


Figure A.5 Continued

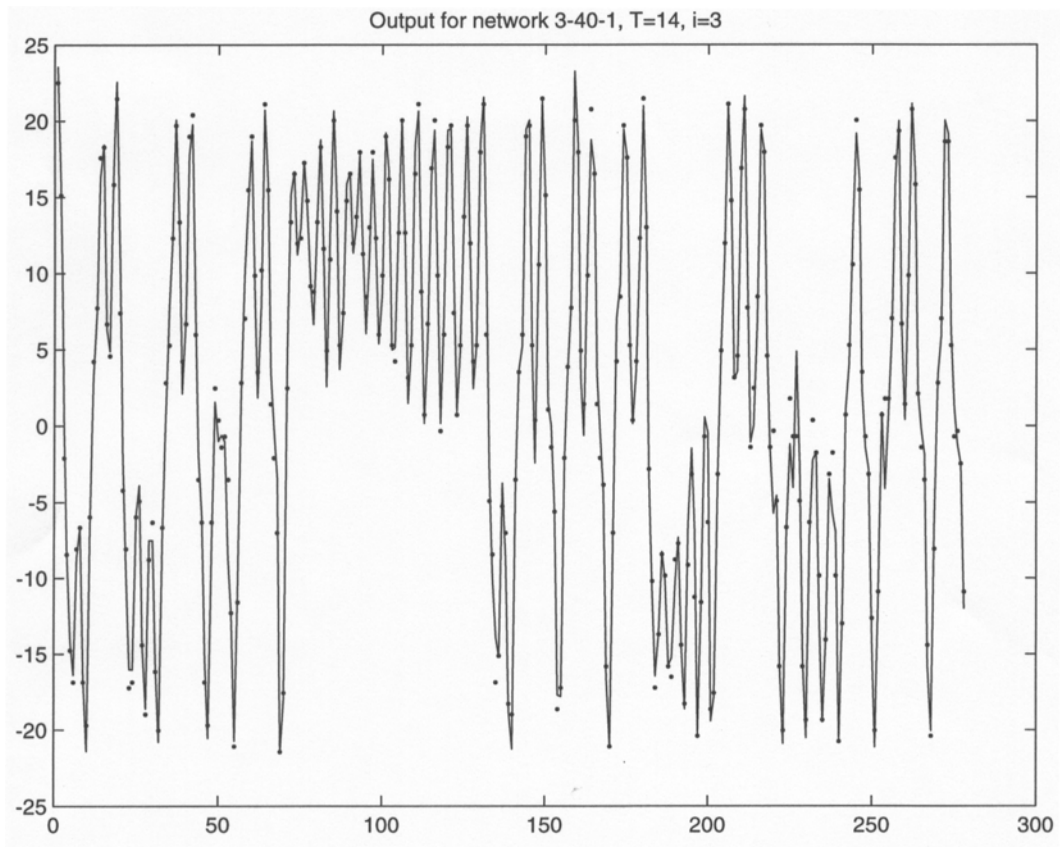


Figure A.6 Output and errors for network 3-40-1 with the test set interval of 3.

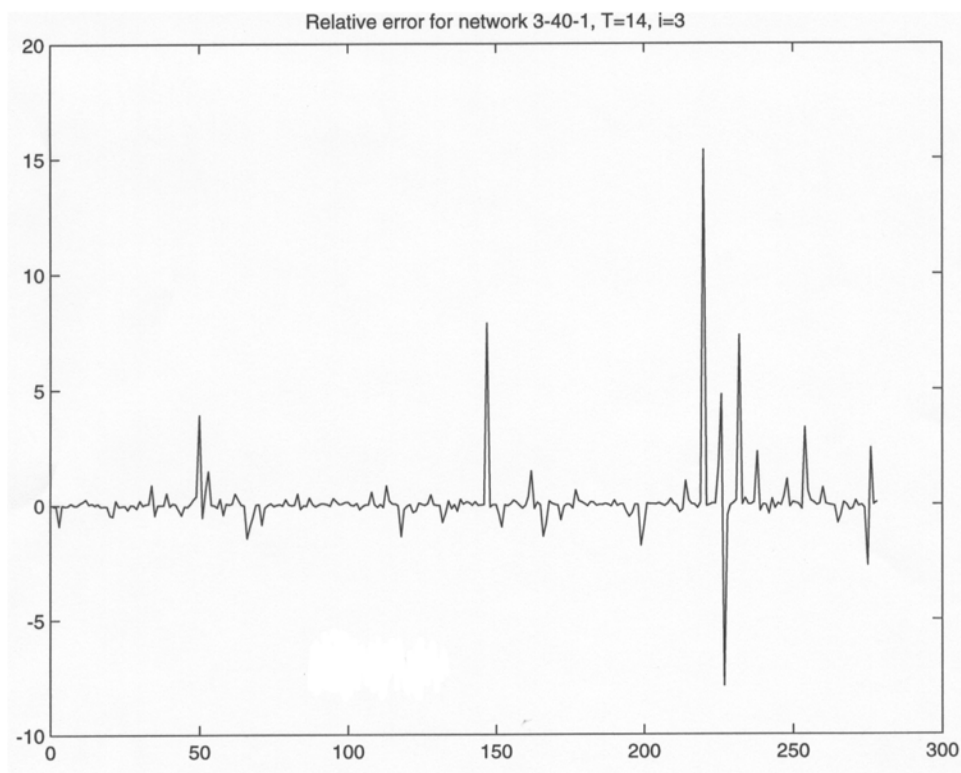
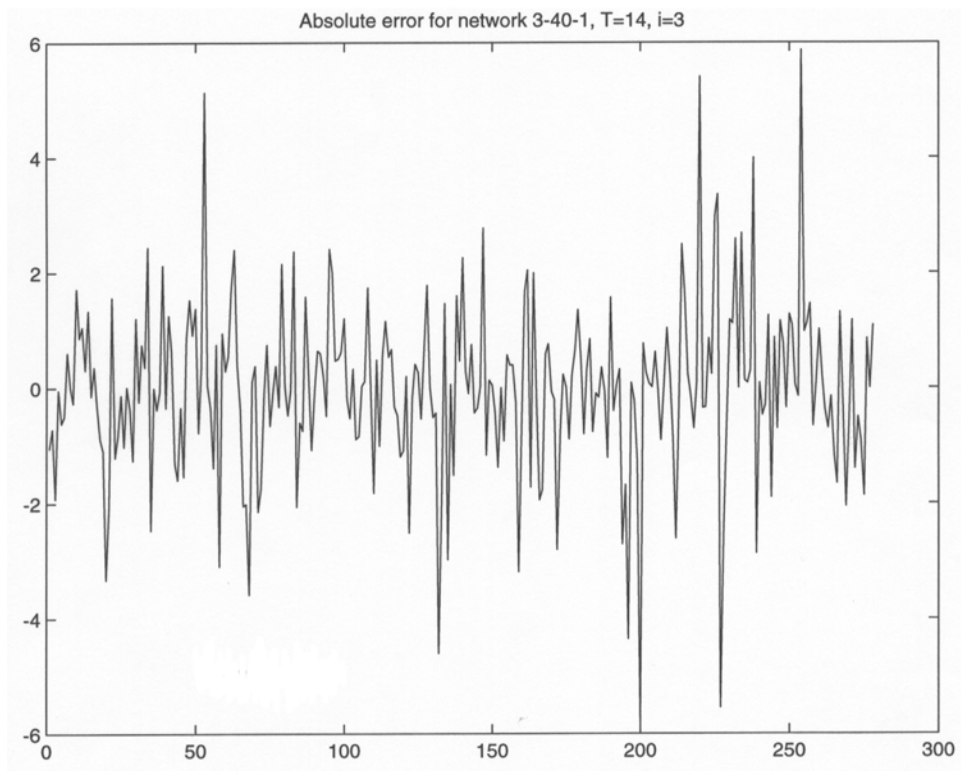


Figure A.6 Continued

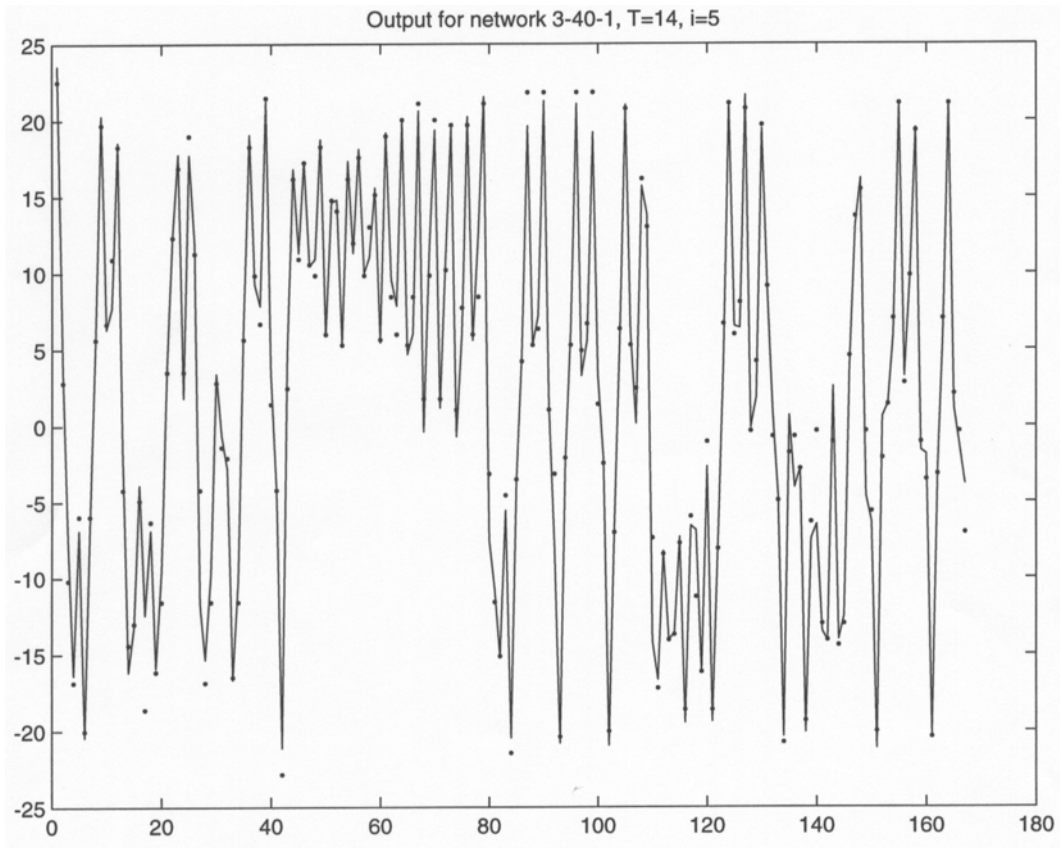


Figure A.7 Output and errors for network 3-40-1 with the test set interval of 5.

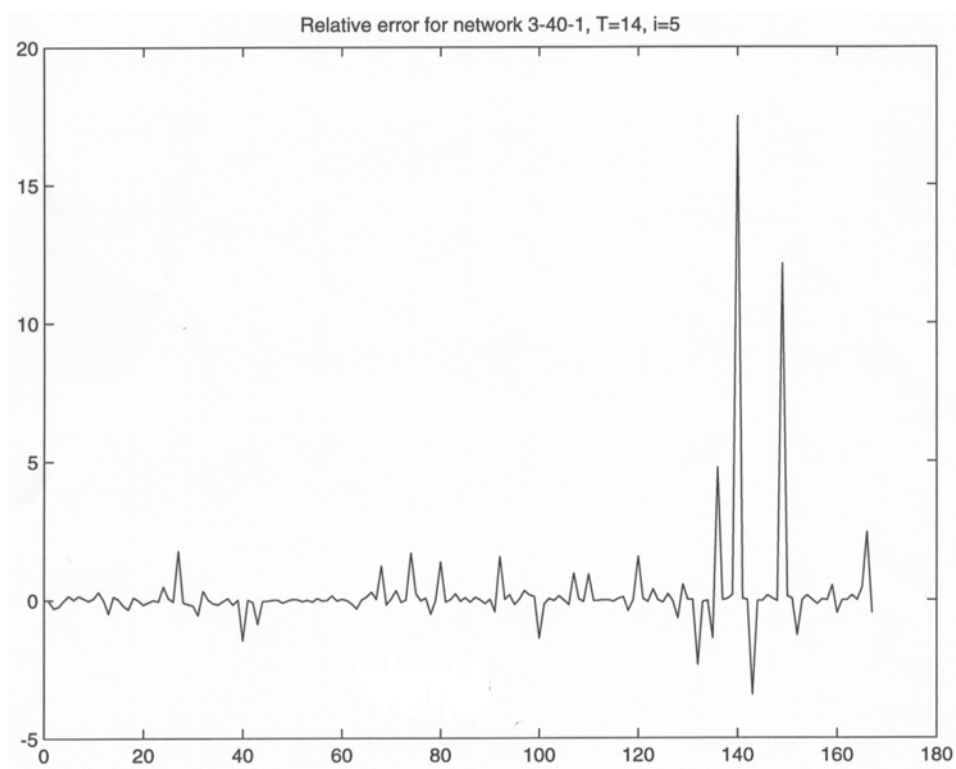
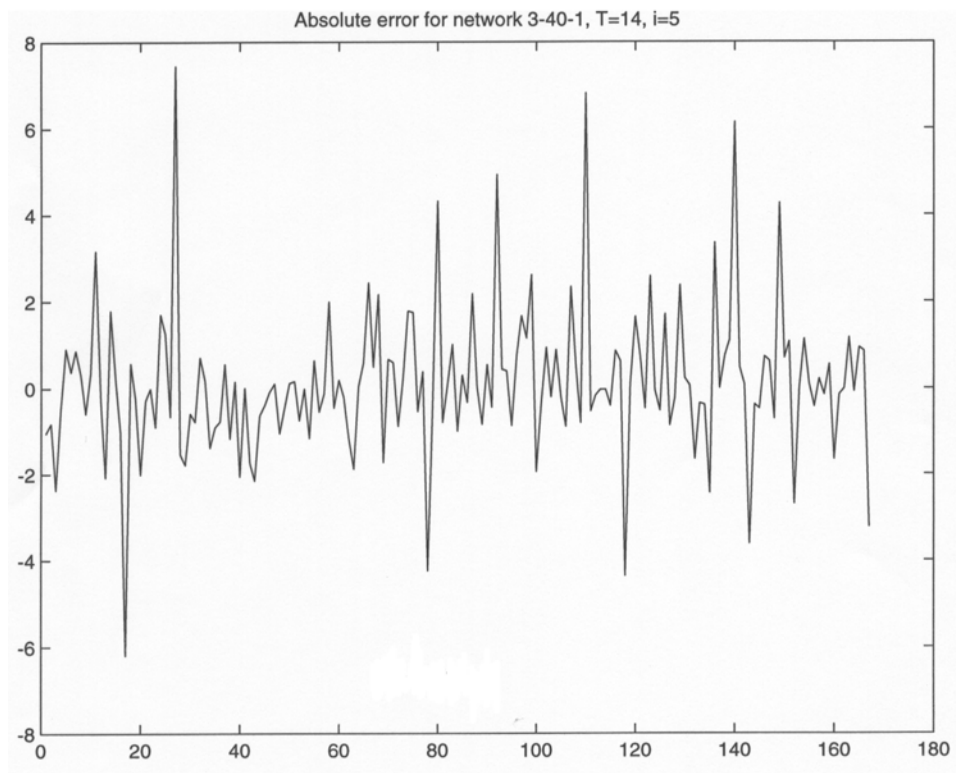


Figure A.7 Continued

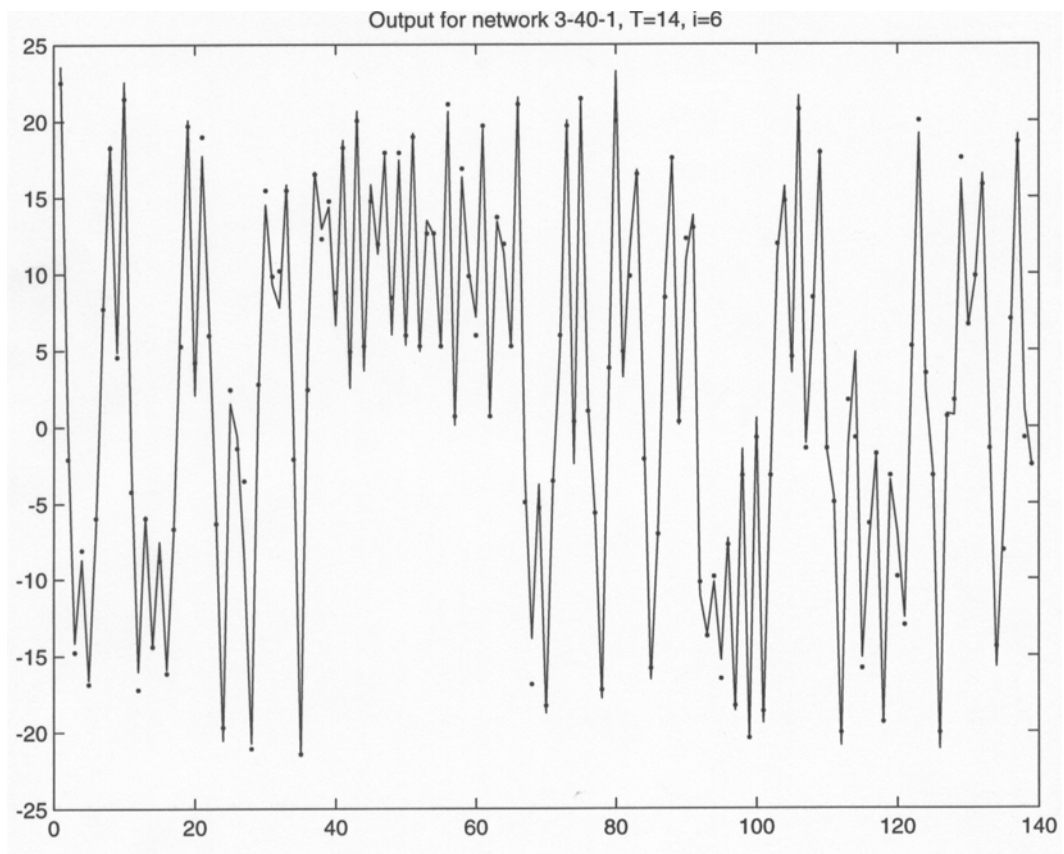


Figure A.8 Output and errors for network 3-40-1 with the test set interval of 6.

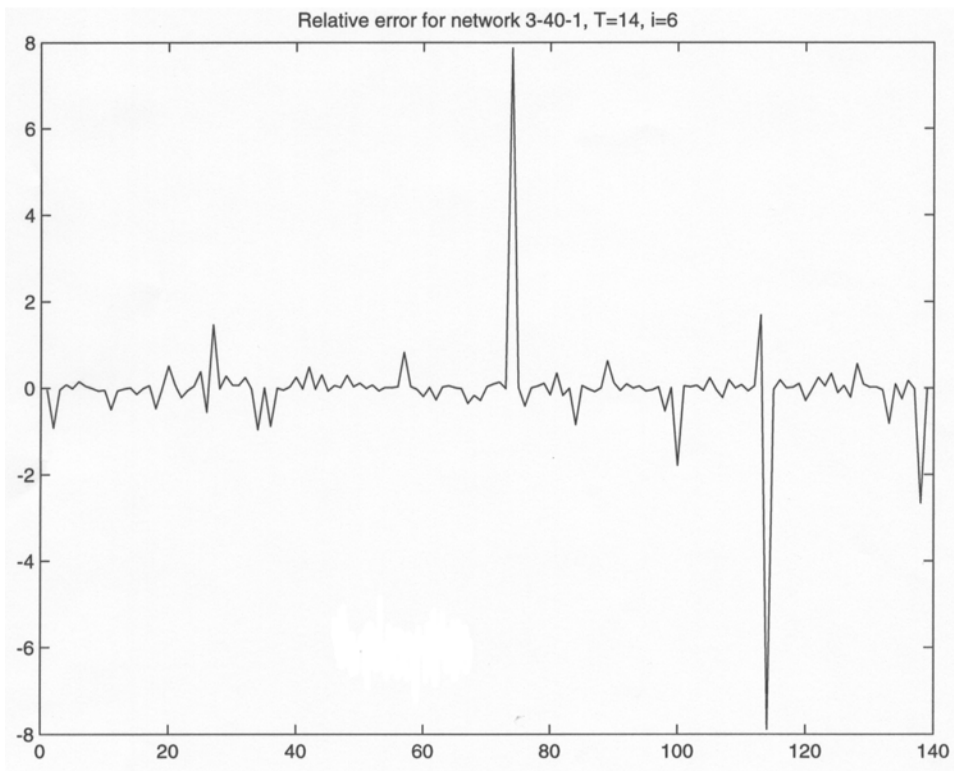
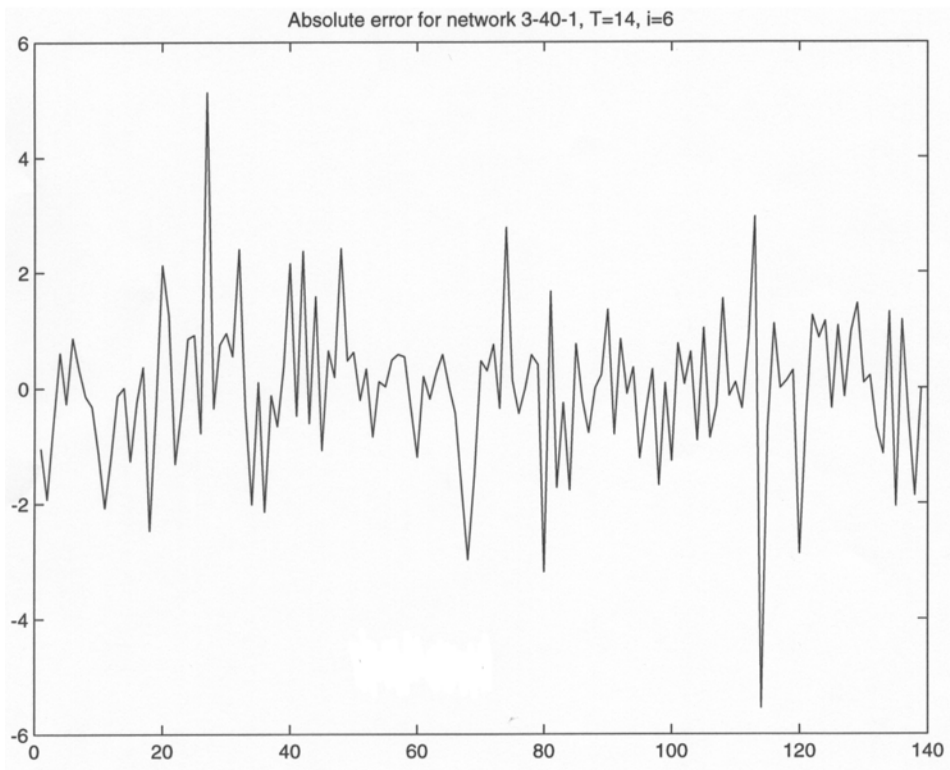


Figure A.8 Continued

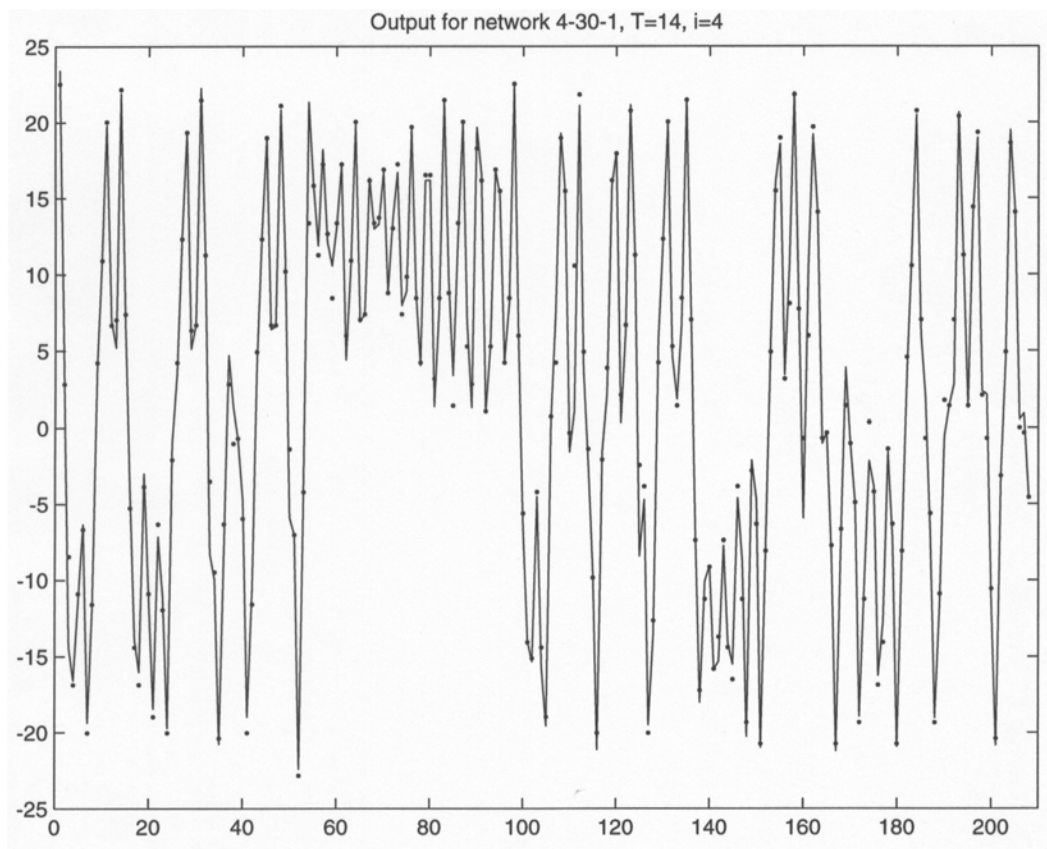


Figure A.9 Output and errors for network 4-30-1 with the training set interval of 4.

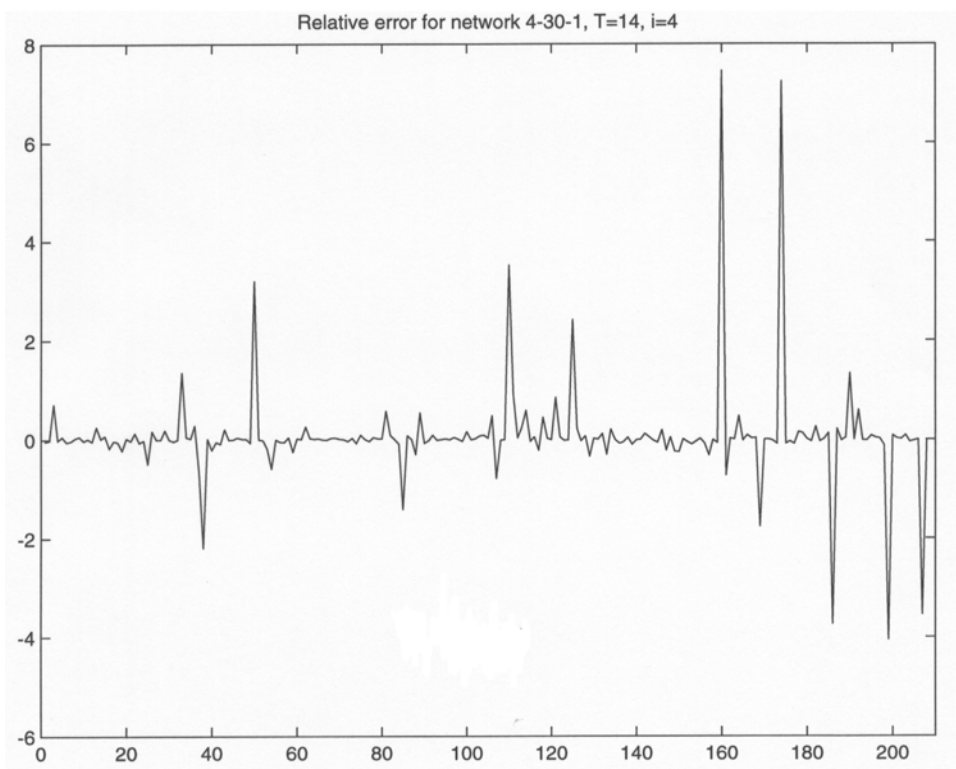
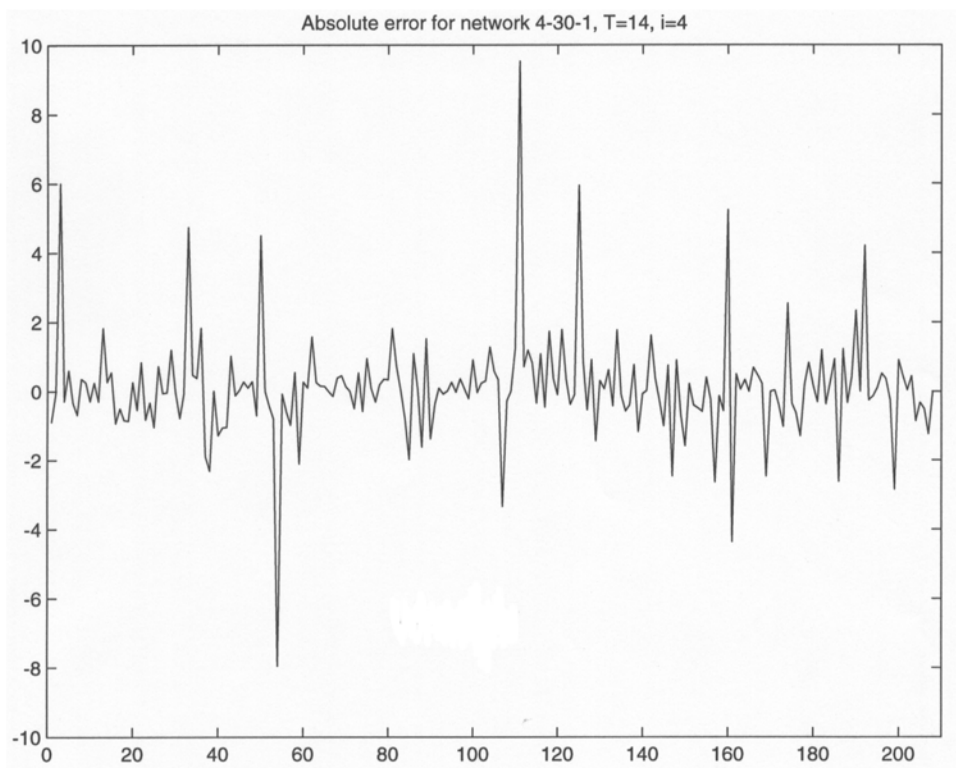


Figure A.9 Continued

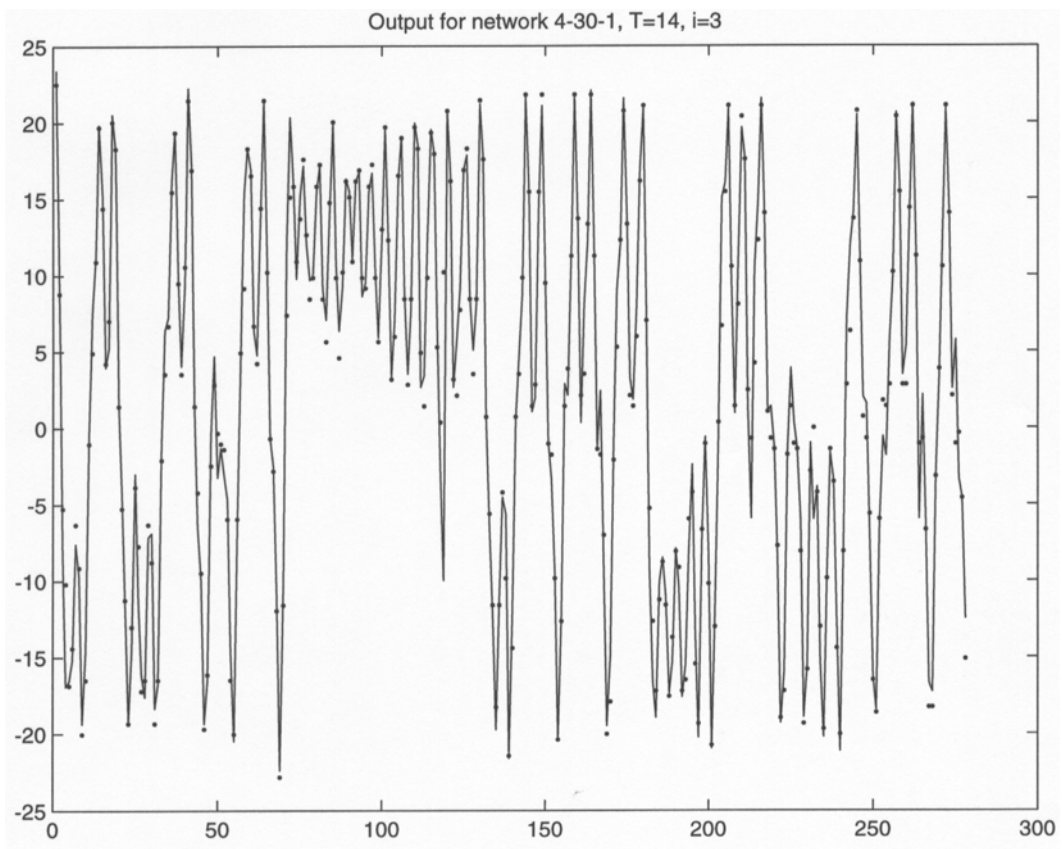


Figure A.10 Output and errors for network 4-30-1 with the test set interval of 3.

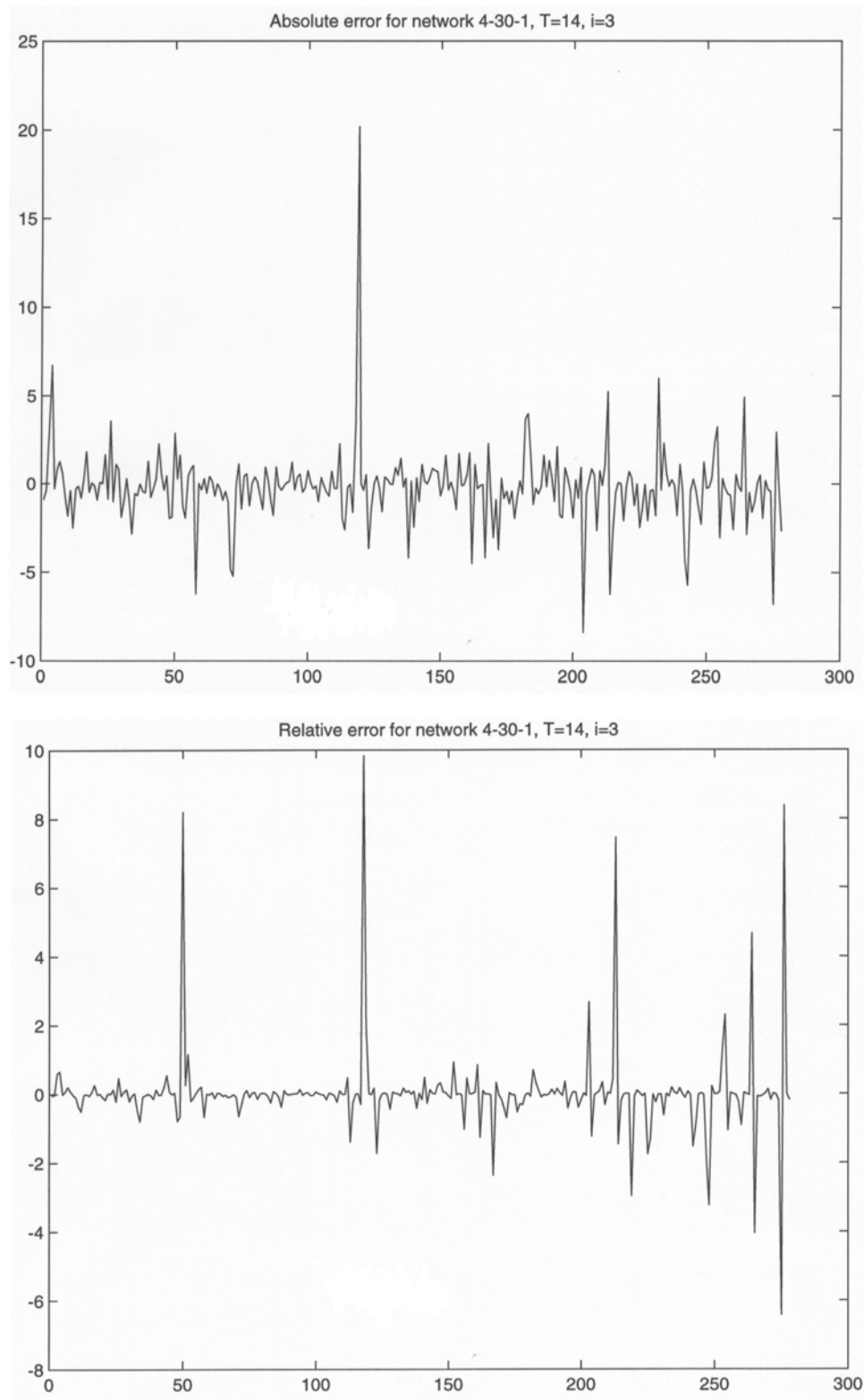


Figure A.10 Continued

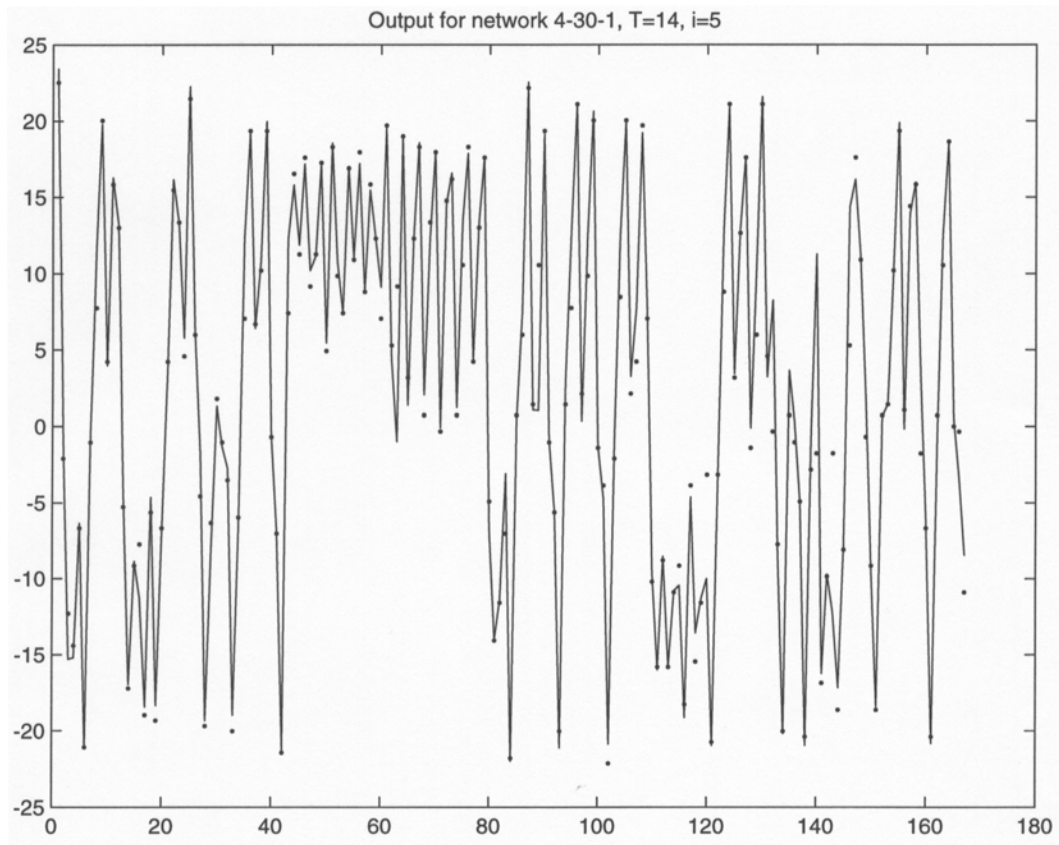


Figure A.11 Output and errors for network 4-30-1 with the test set interval of 5.

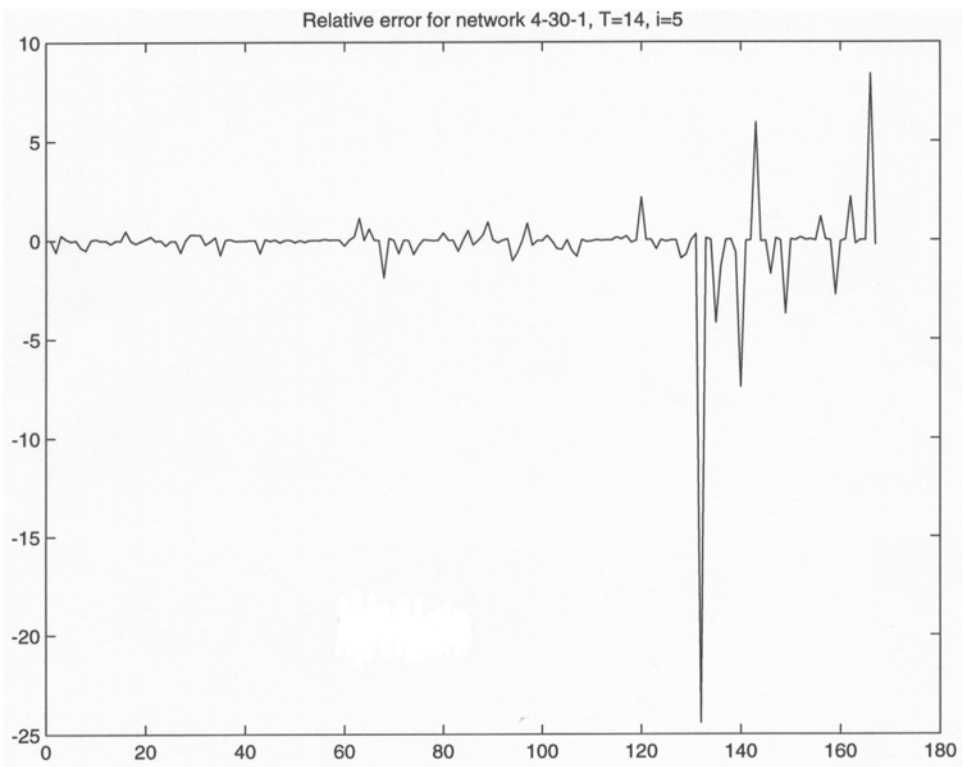
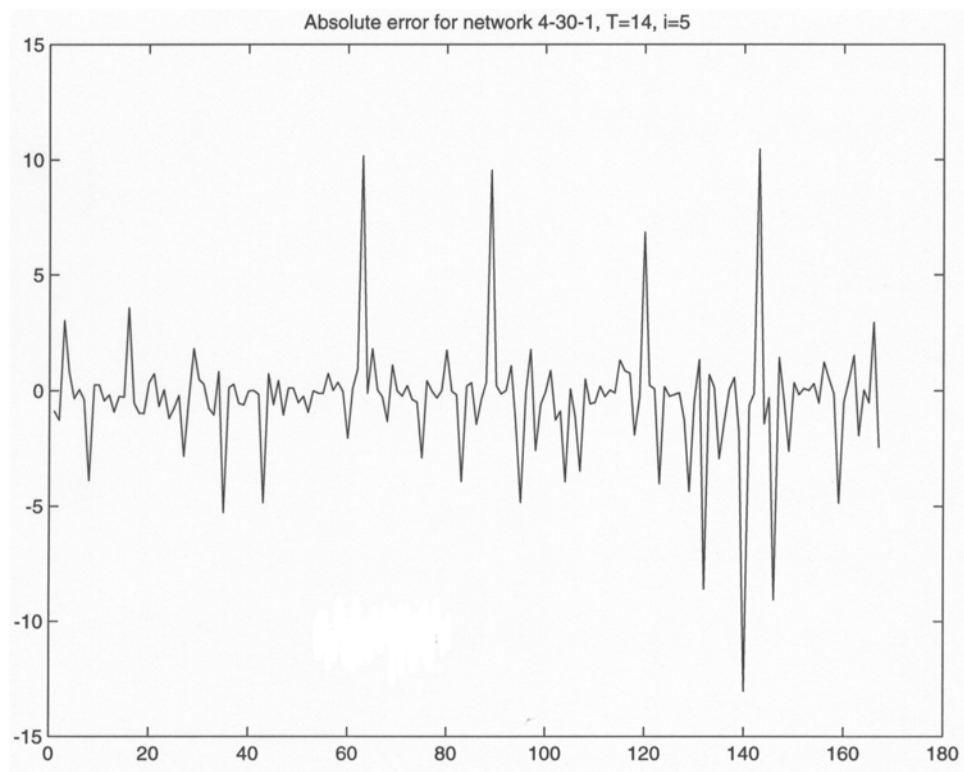


Figure A.11 Continued

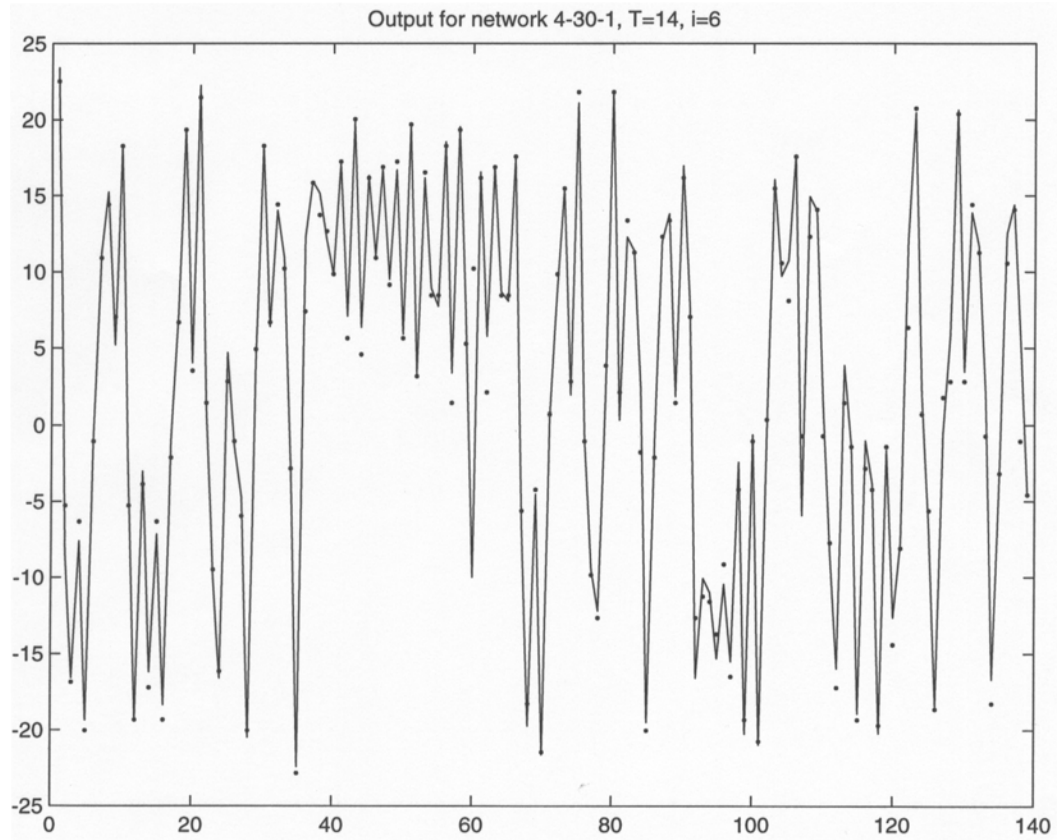


Figure A.12 Output and errors for network 4-30-1 with the test set interval of 6.

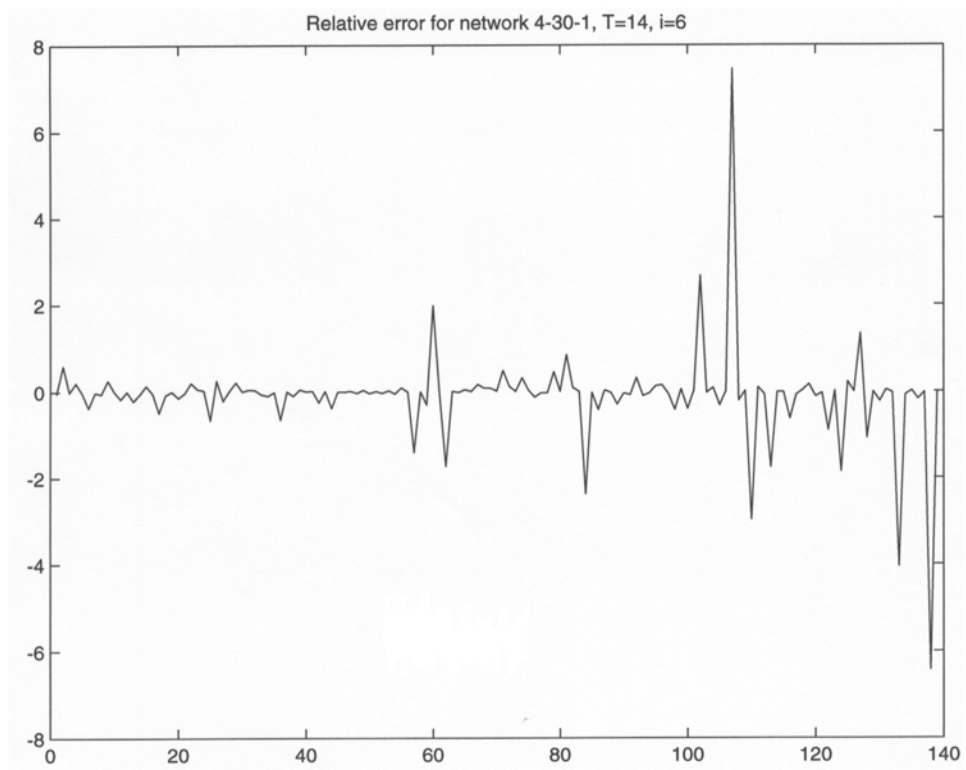
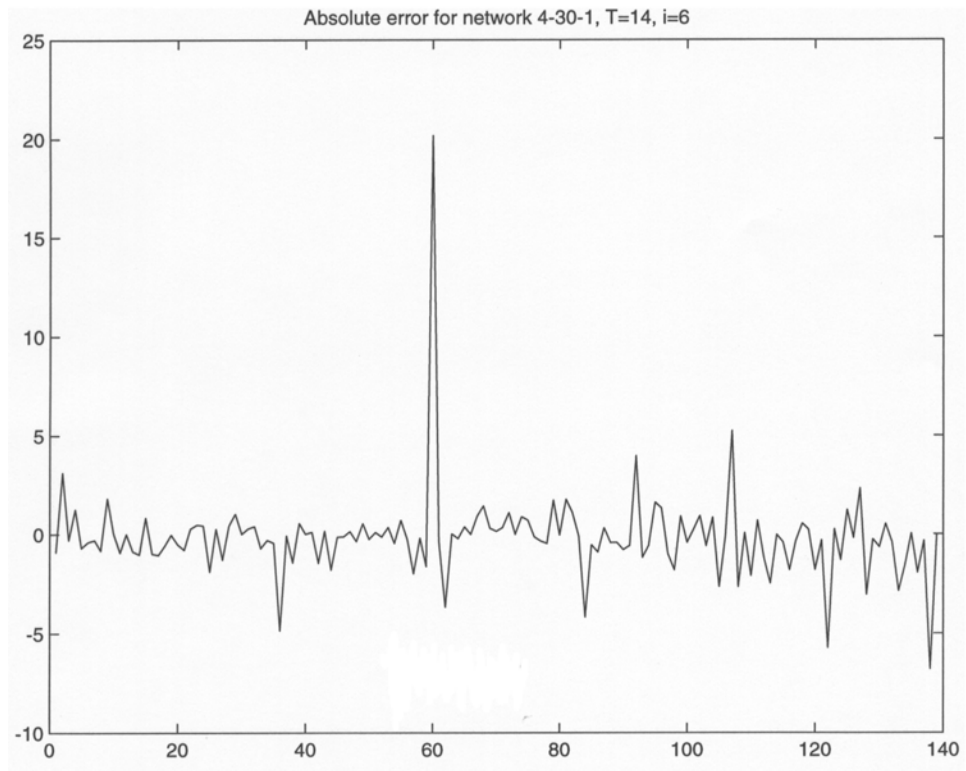


Figure A.12 Continued

APPENDIX B

Software Implementation

Software Description

The MATLAB[®] Neural Network Toolbox proved very useful and efficient for the development and training of the neural networks in this paper. The toolbox possesses several built-in architectures, transfer functions, and learning rules. In addition, a user may incorporate their own transfer functions and learning rules. The package trains large networks quickly and accurately. A graphical user interface may be used to enhance the network development process. Training data and simulation results may be passed to and from the MATLAB[®] workspace easily. This facilitates a rapid analysis of network performance. The accompanying User's Guide by Demuth and Beale provides a comprehensive introduction to the toolbox. In addition, examples from the text *Neural Network Design*, by Hagan, Demuth, and Beale, are included software in the toolbox.

Algorithm Explanation

Average Mutual Information

The program *datacreate.m* is self explanatory and will not be further discussed.

The algorithm *mutinfbox.m* calculates the individual and joint probability densities of the reconstructed data. The phase space is subdivided into the number of boxes specified by the user. The probability densities are then estimated by counting the normalized number of points in each box.

```

function [t ,x]=create_input_target(sting,T,interval,start,stop,d)
%[target,inputs]=create_input_target (sting,T,interval,start,stop,d),
%All arguments are scalars.
%T=time delay, interval=number of indices between target values in
reconstructed data,
%stop=portion of reconstructed data to be used
%d=number of inputs
%Creates input and target vectors to be used in recursive training or
simulation of neural networks.
switch sting
case 35
load c35.mat
case 33
load c33.mat
case 30
load c30.mat
end
%The first 300 points are discarded so that only the attractor portion
of the trajectory is trained.
%The uncertainty in a sting angle of 35 deg is 0.54 deg. The data is
rounded to four significant figures.
x1=chop(x1(300:end),4);
x1=x1(start:T:stop);
m=length(x1);
row=1;
k=1;
col=1;
kstep=interval-d-1;
while k<=m
    if row<(d+1)
        x(row,col)=x1(k);
        row=row+1;
    else
        k=k+kstep;
        row=1;
        col=col+1;
    end
    k=k+1;
end
[a,b]=size(x);
x=x(:,1:b-1);
t=x1(d+1:interval:m);
t=t(1:b-1)';

```

```

%datacreate.m
%Amy Henderson
%Creates time delay data from one-dimensional data

clear
%Load one-dimensional data, x1(n), of length N.
load c35.mat
x1=x1(300:end);
N=length(x1);

%T is an integer increment to the index. Produce next data set n Time
increment is equal to T times the sampling time.
%x1(n+T) = x2(k) for k=1:(N-(T+1)) for T>=2, k=1:N for T=0, k=1:N-T for
T=1.

T=14; %Adjust time delay as appropriate.
d=3;
stop=N-(d-1)*T;

for k=1:stop
    x2(k)=x1(k+T); %Construct delayed time series.
    x3(k)=x1(k+2*T);
end

%Vectors must be the same length, x1 becomes shorter with each increase
of T.
x1=x1(1:stop);

%Output
stop
x2=x2';
x3=x3';
save data.mat x1 x2 x3
save params.mat stop T
%Change to column vector for ease of sorting in Excel. See iprob1.m
for explanation.

%These ascii files are read by Excel and sorted.
save x1.dat x1 -ascii
save x2.dat x2 -ascii
save x3.dat x3 -ascii

```

```

%Amy Henderson
%Estimates average mutual information for reconstructed phase space.
clear
load data.mat
set=[x1 x2];
x1=sort(x1);
x2=sort(x2);
minx1=min(x1);
maxx1=max(x1);
minx2=min(x2);
maxx2=max(x2);

numberbinsx1=input('Enter the number of bins for x1. ');
numberbinsx2=input('Enter the number of bins for x2. ');

x1step=abs(maxx1-minx1)/numberbinsx1
x2step=abs(maxx2-minx2)/numberbinsx2
nx1=zeros(numberbinsx1,1);
nx2=zeros(numberbinsx2,1);
Ntot=length(x1);
binnumberx1=zeros(Ntot,1);
binnumberx2=binnumberx1;
set(:,3:4)=[binnumberx1 binnumberx2];
k=1;
x1val=minx1;
x2val=minx2;
binnumber1=1;
binnumber2=1;
for k=1:Ntot
    set(k,3)=ceil(abs(set(k,1)-minx1)/x1step);
    if set(k,3)==0
        set(k,3)=1;
    end
    set(k,4)=ceil(abs(set(k,2)-minx2)/x2step);
    if set(k,4)==0
        set(k,4)=1;
    end
    if x1(k)<=(x1val+x1step)
        nx1(binnumber1)=nx1(binnumber1)+1;
    else
        while x1(k)>(x1val+x1step)
            if binnumber1==numberbinsx1
                binnumber1=numberbinsx1;
            else
                binnumber1=binnumber1+1;
            end
            x1val=x1val+x1step;
        end
        nx1(binnumber1)=nx1(binnumber1)+1;
    end
    if x2(k)<=(x2val+x2step)
        nx2(binnumber2)=nx2(binnumber2)+1;
    else
        while x2(k)>(x2val+x2step)
            if binnumber2==numberbinsx2

```

```

        binnumber2=numberbinsx2;
    else
        binnumber2=binnumber2+1;
    end
    x2val=x2val+x2step;
end
nx2(binnumber2)=nx2(binnumber2)+1;
end
end
set=sortrows(set,[3 4]);
clear x1 x2 N T d stop
A=x1step*x2step;
Ntotlog2=log2(Ntot);
Ntotmult=(1/Ntot);
Nbox=0;
binnumber1=1;
binnumber2=1;
sum=0;
k=1;
while k<=Ntot
    if set(k,3)<=binnumber1
        if set(k,4)>binnumber2
            if Nbox~=0

mutinf=A*Nbox*(Ntotlog2+log2(Nbox/A*1/(nx1(binnumber1)*nx2(binnumber2))))
);
                sum=sum+mutinf;
                Nbox=0;
            end
            binnumber2=binnumber2+1;
        else
            Nbox=Nbox+1;
            k=k+1;
        end

    else
        if Nbox~=0

mutinf=A*Nbox*(Ntotlog2+log2(Nbox/A*1/(nx1(binnumber1)*nx2(binnumber2))))
);
                sum=sum+mutinf;
                Nbox=0;
            end
            binnumber1=binnumber1+1;
            binnumber2=1;
        end
    end
end
if Nbox~=0

mutinf=A*Nbox*(Ntotlog2+log2(Nbox/A*1/(nx1(binnumber1)*nx2(binnumber2))))
);
    sum=sum+mutinf;
    Nbox=0;
end
mutinf=Ntotmult*sum

```

VITA

Amy Henderson was born in Athens, Tennessee, on September 27, 1978. She graduated from Farragut High School located in Knoxville, Tennessee, in May, 1996. She received a Bachelor of Science degree in Aerospace Engineering from the University of Tennessee in May, 2000. She entered the graduate program at the University of Tennessee in August, 2000, and received the Master of Science degree in Aerospace Engineering in May, 2002. She is currently employed as a Structural Engineer with Lockheed Martin Aeronautics Company in Fort Worth, Texas.