Modeling Helicopter Dynamic Loads Using Artificial Neural Networks

by

Michael Nosek

Thesis Submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

MASTER of SCIENCE

in

Aerospace Engineering

COMMITTEE ENDORSMENTS:

[illegible]

Dr. Frederick Lutze, Chairman

[illegible]

Dr. Hugh VanLandingham

[illegible]

Dr. Mark Anderson

June 30, 1993

Blacksburg, Virginia

C.2

LD
5655
V855
1993
N673
C.2

MODELING HELICOPTER DYNAMIC LOADS USING ARTIFICIAL NEURAL NETWORKS

by

Michael Nosek

Dr. Frederick Lutze, Chairman

Aerospace Engineering

(ABSTRACT)

In this thesis, artificial neural networks (ANNs) are used to model helicopter main rotor dynamic loads as a function of flight variables. The motivation to develop an accurate model of such loads is to reduce maintenance and replacement costs by eliminating excessive conservatism currently associated with structural fatigue estimation. Neural networks are used for the modeling procedure because of their capability to model complex, nonlinear relationships for multiple input-multiple output systems.

In support of the dynamic loads modeling discussed above, this thesis also briefly reviews artificial neural network technology, and investigates the modeling of a well-known dynamic system using ANNs.

Acknowledgements

I wish to thank Dr. Frederick Lutze, my advisor and committee chairman, for his guidance and support throughout my Masters program. I would also like to thank the other members of my committee, Dr. Mark Anderson and Dr. Hugh VanLandingham for their patient advice and willing support.

I also thank Mr. Carl Schaefer and our supporters at Naval Air Systems Command, for providing the research data which made this thesis possible.

I thank Michael Tran, for use of his excellent computer algorithm, which greatly facilitated my research. I also thank my roommates, Brad Burnett, Mark Saunders and Sean Weems, for generous use of their computer equipment and advice.

I would like to thank my colleagues and fellow graduate students at Virginia Tech, Ken Bordignon, Tim Cashin, Bob Grogan, and John Keiser, who provided the necessary distractions from graduate studies.

Finally, I thank my parents and family for their continuing support throughout my education.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

In modern aircraft construction and maintenance, the problem of fatigue is primarily an economic issue, rather than a safety issue. Generally, it is possible to meet the desired safety requirements through implementation of fail-safe and safe-life design techniques, along with frequent maintenance and replacement of components [1]. As a result, the issue becomes a question not of the possibility of achieving a given level of safety, but rather, a question of how much cost is involved in attaining that level of safety. For design and construction purposes, the cost of meeting specified safety requirements may take the form of decreased performance of the aircraft, brought about by the addition of extra material and weight, for example. For maintenance scheduling and component replacement, cost is almost exclusively economic. An effective means to economize in this area is to reduce excessive conservatism in fatigue estimation, reducing the demands of maintenance scheduling and preventing disposal of components which may still possess valuable maintenance life.

The primary motivation for developing a model of helicopter dynamic loads is to provide a cost-effective means to accurately estimate fatigue of the main rotor. Currently, maintenance schedules are predetermined by a given number of flight hours for each aircraft. Large safety factors need be incorporated into such schedules, because flight time alone does not determine the deterioration of the helicopter components. Obviously, a helicopter flying straight and level does not suffer the same abuses as a helicopter performing aggressive flight maneuvers for the same amount of time. Safety factors must account for the possibility of extra deterioration by assuming a worst-case environment. By developing a model of how various flight maneuvers affect deterioration of hardware components, maintenance schedules can be intelligently determined in accordance with the aircraft's flight history, eliminating excessive conservatism.

Usually, dynamic system mathematical models are developed from the basic equations governing the motion, Newton's laws, equations describing structural behavior, and for flight vehicles, the equations describing the aerodynamic flow field. As a result, the high fidelity simulation model becomes quite complicated when anything but the basic motion is considered. One example of such a complex system is the dynamic, structural and aerodynamic properties of a helicopter in flight. Another is the modeling of the structural loads encountered at various points in the main rotor mechanism while the helicopter is maneuvering. Development of the model by analytic means is especially difficult for the case of a helicopter, considering the series of interactions affecting the relationship between the flight maneuvers and the structural fatigue: from the external aerodynamic forces acting on the aircraft, to the internal forces, torques, stresses, and strains acting on the helicopter components. Among these relationships are different gear mechanisms, and the interaction of a multitude of parts, not to mention the heavy vibration inherent in a helicopter. The increased level of difficulty inevitably leads to an increased number of assumptions, which can reduce the accuracy of the model. To date, the set of equations which model these loads accurately is still being developed. Consequently, it is appropriate to investigate methods which can predict the behavior of complicated systems for which input and output information is known from experimental measurements, without necessarily modeling the complex causal chain occurring in the interim.

Over the past few years, artificial neural network (ANN) technology has reached maturity and can be applied to such complicated problems. ANNs have been used successfully to model systems where the input and the corresponding output data are known for many data points.

Given several data points, artificial neural networks attempt to develop a relationship between the inputs and outputs. The basic structure of the relationship is determined by the modeler, and is limited mathematically by addition, multiplication, and simple basis functions. A computer algorithm adjusts the multiplications so that the algebraic relationship will closely match the physical relationship between the inputs and outputs. A well-trained neural network model accurately estimates the output data for a given set of inputs, for all data points.

This paper attempts to model dynamic systems using artificial neural networks. Emphasis is placed on achieving models which identify the system, that is, models which "grasp" the underlying dynamics of the system. This objective is in contrast to models which merely predict the output response of the system for a select number of input vectors. The model which grasps the dynamics more closely serves as a better extrapolator, or "predictor". The model which does not grasp the dynamics is said to have merely "memorized" the input/output data for a given data range. Such a model is limited to identifying given input/output data at discrete points of examination, and perhaps can interpolate between those points. In most cases, the neural network model shows evidence of both identification and memorization; the objective here is to structure the neural network (especially with regard to the input vector) to emphasize the former. The neural network which is structured in this manner is easier to train, and can more accurately predict results. By incorporating the dynamics into the ANN model of the helicopter loads, it is hoped to improve the prediction capability of dynamic load modeling, reducing excessive conservatism in fatigue estimation, thus reducing the frequency of component replacements, while still maintaining the safety requirements of the aircraft.

Chapter 2 provides a brief review of neural network technology, followed in Chapter 3 by applications of neural networks to identification of a simple spring mass system. Chapter 4 applies the lessons learned from the spring mass system to aid in modeling of helicopter dynamic loads. The dynamic load models are then evaluated with regard to their potential to estimate fatigue of the helicopter main rotor components.

2 ARTIFICIAL NEURAL NETWORKS

Artificial neural networks are versatile tools with several modern applications in the fields of marketing, medicine, and engineering, to name a few. Such applications include statistical analysis, trend analysis and prediction, pattern recognition, adaptive control systems, and system identification and modeling [2]. Antegnetti [3] presents several specific ANN applications for modern engineering problems. Proceedings of the Second Workshop on Neural Networks[4] is a detailed source of information for ongoing research work that use ANNs. For a detailed look into the internal workings of neural networks, Lippmann [5] and Rumelhart [6] discuss the fundamentals of developing and training neural networks, including development of equations which can be used as a basis for creating computer codes for ANNs. Neural network research journals include: IEEE Transactions on Neural Networks, (monthly since spring 1990); Neural Networks, Journal of the International Neural Networks Society (INNS), (quarterly in 1988, bimonthly since 1989); and IEEE Control Systems Magazine, (special issues on applications of neural networks to control systems, April 1989, April 1990, April 1992.) This chapter provides a brief description of artificial neural networks, their basic operations, and the terminology associated with them.

In its most basic form, an artificial neural network is a series of mathematical operations and simple functions, capable of operating on multiple independent input variables, to produce multiple outputs. In order to successfully use a neural network as a model, the mathematical operations are manipulated so that the vector of network outputs will reasonably match the vector of system outputs, given the same vector of inputs. The feature of the ANNs that make them so useful is their ability to make adjustments in their parameters by "training" so that they closely approximate the system of interest.

2.1 A GENERIC NEURAL NETWORK

A graphical representation of a typical artificial neural network is shown in Figure 2-1. The circles represent "nodes." A column of nodes is referred to as a "layer". The left most nodes are the input nodes, the right most nodes are the output nodes, and nodes in between are "hidden nodes". Similarly, the left most layer is the input layer, followed by the hidden layer(s), followed by the output layer. (Although the neural network represented in Figure 2-1 contains 1 hidden layer, a typical neural network may contain any number of hidden layers, or none at all.) The lines connecting the nodes are "interconnections". Interconnections are generally made in all possible combinations between adjacent layers.

Interconnections which are made directly from the input nodes to the output nodes are referred to as "feed-through" connections. Each interconnection has associated with it a multiplier, or a "weight". The nodes act as summers for the incoming signals from the interconnections. The blocks represent a squashing function, designed to "squash" the incoming signal to a numerical value within a given range, usually between 0 and 1, or between -1 and +1. Commonly used squashing functions include the arctangent, hyperbolic tangent, sine, cosine, and sigmoid functions. (Some examples of squashing functions are graphed in Figure 2-2.) The output layer may also contain squashing functions, but most often will contain a linear function or none at all. Overall, the neural network "structure" or "geometry" is defined by the number of layers, the number of nodes in each layer, the interconnections, the feed-through connections, and the squashing functions used. In addition, "bias nodes" can be included in the neural network structure. A bias node behaves like any other node, except that its input signal is a constant value of unity. (See Figure 2-1)

2.2 TRAINING A NEURAL NETWORK

The training of a neural network involves three phases: initialization, operation, and improvement. After completing the first phase, initialization, the next two phases are alternated until the network is considered to have less than some acceptable defined error. If the network fails to obtain an acceptable error level, it may be appropriate to start again by re-initializing the network, and then continue to alternate the last two phases until acceptable results are obtained. Each of the three phases are discussed in detail in the following sections.

2.2.1 NEURAL NETWORK INITIALIZATION

"Initialization" of the network involves specifying the geometry (explained in Section 2.1 above), and specifying the initial values of the weights on each interconnection. Also specified is the learning sensitivity, η, which will be explained in Section 2.2.3. As a rule of thumb, the weights are generally assigned randomly in the range from -0.3 to +0.3 [7]. Furthermore, the data is initialized through scaling, as will be explained more fully in Section 2.4. After initializing the network and the data, the network is ready to operate in the feedforward mode.

2.2.2 FEEDFORWARD OPERATION

For the feedforward operation, an input vector is selected from the scaled data and entered into the network. The network operates on the input vector to produce an output vector, which can be compared to the true output vector.

When operating the neural network in the feedforward mode, the sequence of operations can be thought of as traveling from left to right across the diagram. Each scaled input value is transmitted along the appropriate interconnection to each of the nodes of the first hidden layer, after being multiplied by the weight associated with that interconnection. The receiving node acts as a summer, adding the incoming values before the squashing function is applied to the sum. This process continues until the output layer is reached, and the output signals obtained, thus concluding the feedforward operation for a single input vector.

In addition to the graphical representation of the feedforward process (Figure 2-1), the output vector can be represented mathematically as a function of the inputs. For an artificial neural network with one hidden layer and no feed-through connections, the output vector is expressed in the following manner:

$$\begin{aligned} O_p &= \sum_{n=0}^{N} W2_{np} f_n\left(\sum_{m=1}^{M} W1_{mn} I_m\right) \\ &= \sum_{n=0}^{N} W2_{np} f_n(F_n^-) \\ &= \sum_{n=0}^{N} W2_{np} F_n^+ \end{aligned} \tag{2-1}$$

where,

M = number of nodes in input layer(not including bias)
N = number of nodes in hidden layer
m = index of input node
n = index of hidden node
p = index of output node

Wj_{mn} = value of weight for interconnection
from node m of layer j
to node n of layer $j+1$

O_p = value of output node p
I_m = value of input node m

F_n^- represents the stimulus to the squashing function,
i.e. the weighted sum of the incoming signals

f_n represents the squashing function
associated with node n

$F_n^+ = f_n(F_n^-)$ = outgoing signal from the squashing function

The output vector obtained from the feedforward operation can be compared to the true output vector of the system, and some measure of error calculated for training the network or evaluating its performance. The algorithm for training the network is based on the mean square error. Performance evaluation may be based on a cost function which is some measure of error for all output vectors of interest.

2.2.3 BACK-PROPAGATION TRAINING

The purpose of training the network is to adjust the weights on the interconnections in order to minimize the cost function. The algorithm for adjusting the weights is based on "back-propagation" of the error, and is represented schematically in Figure 2-3. For this procedure, a "training set" is first constructed, consisting of any number of data sets, where a single "data set" is an input vector and its associated output vector. An input vector is selected and then entered into the neural network, which operates in the feedforward mode to generate an output vector. For back-propagation training, the mean square error of the output vector is defined as follows:

$$E_{mean\ square} = \frac{1}{2}\sum_{p=1}^{P} (O_p - D_p)^2 \tag{2-2}$$

where

O_p = *the network value of output p*
D_p = *the desired, or "true" value of output p*

For each data set, the gradient of the mean square error with respect to the internal weights is used to change the weights so as to reduce the error. Weight adjustments ΔW are made "opposite in direction and in proportion to the magnitude of the derivative of the (mean square) error with respect to the weights,"[7] or,

$$\Delta W \propto -\frac{\partial E}{\partial W}$$

(2-3)

These adjustments are completed in a "reverse order", starting with those closest to the output layer and working back to the input layer.

Weight adjustments for the second layer are developed as follows:

Starting with equation (2-3),

$$\Delta W2_{np} \propto -\frac{\partial E}{\partial W2_{np}}$$

(2-4)

expanding with the chain rule,

$$\Delta W2_{np} \propto -\frac{\partial E}{\partial O_p}\frac{\partial O_p}{\partial W2_{np}}$$

(2-5)

evaluating the partial derivatives,

$$\Delta W2_{np} \propto -(O_p-D_p)\,F_n^{+}$$

(2-6)

and implementing a proportionality constant, η, the appropriate adjustments for the second layer become:

$$\Delta W2_{np} = \eta\,(D_p-O_p)\,F_n^{+}$$

where, $(0 \leq \eta \leq 1)$

(2-7)

The step size of each weight adjustment is controlled by its proportionality constant, η, also referred to as a learning sensitivity. This constant is assigned a value between zero and unity, and is the same value for all weights in a given layer.

Weight adjustments for the first layer are derived in a similar manner, starting with equation (2-3):

$$\Delta W1_{mn} \propto -\frac{\partial E}{\partial W1_{mn}} \tag{2-8}$$

expanding with the chain rule,

$$\Delta W1_{mn} \propto -\frac{\partial E}{\partial F_n^{+}} \frac{\partial F_n^{+}}{\partial F_n^{-}} \frac{\partial F_n^{-}}{\partial W1_{mn}} \tag{2-9}$$

the partial derivatives are then evaluated:

$$\begin{aligned} -\frac{\partial E}{\partial F_n^{+}} &= -\frac{\partial E}{\partial O_p} \frac{\partial O_p}{\partial F_n^{+}} \\ &= (D_p - O_p)\, W2_{np} \end{aligned} \tag{2-10}$$

$$\frac{\partial F_n^{+}}{\partial F_n^{-}} = f_n{}'(F_n^{-}) \tag{2-11}$$

where $f_n{}'(F_n^{-})$ is the slope of the squashing function of node n, evaluated at its level of stimulus, F_n^{-}

and,

$$\frac{\partial F_n^-}{\partial W1_{mn}} = I_m \tag{2-12}$$

Substituting equations (2-10), (2-11), and (2-12) into (2-9), and including the proportionality constant, the adjustments for the weights of the first layer are defined by the following equation:

$$\Delta W1_{mn} = \eta \, (D_p - O_p) \, W2_{np} \, f_n'(F_n^-) \; I_m \tag{2-13}$$

After the appropriate adjustments have been made for all layers, one iteration of training is completed. When all data sets in the training set have been sampled, then one "flop" or "epoch" has been completed. The procedure is repeated for several epochs (sometimes thousands) until an acceptable error (cost function) is obtained for the training set as a whole. When an acceptable error has been achieved, the weights are then frozen and the network can be further evaluated, as will be explained in Section 2.3.

It is important to note that only a single adjustment is made for each weight for each training iteration. Further adjustments are not made until a new input vector is entered into the network, thus beginning a new iteration. By not adjusting the weights to completion, the error is not minimized for any particular data set. Rather, the algorithm samples all data sets, one at a time, making "small" adjustments for each, thus reducing the error for the training set as a whole. For the same reason, each data set is usually selected only once per epoch. Exceptions to this rule are sometimes made in order to focus the algorithm's attention on "problem" outputs, which maintain a significantly higher error than that of the training set as a whole. The purpose of this modification is to "even out" the error distribution of the training set.

The back propagation algorithm often requires user intervention, as the training of artificial neural networks is not a precise science. During training, the user controls the algorithm primarily by adjusting the learning sensitivities. Larger values of η cause the weight adjustments to occur in larger step sizes, and may accelerate the learning process. Yet, smaller values may be necessary for the algorithm to properly converge or to find the minimum error solution. Because the algorithm does not always converge toward a definite solution, it is up to the user to determine if a minimum-error solution has been obtained. Oftentimes the error will appear to slowly "creep" toward an asymptotic value, and then suddenly drop to lower value. Another ambiguity of ANN training frequently arises when the error of the training set continues to decrease, but may cause the error of the testing set to rise.

2.3 EVALUATING THE NEURAL NETWORK

After the network is trained to an acceptable error, the weights are frozen for the purpose of testing the network. A "testing file" or "testing set" is created by selecting certain data sets, each consisting of an input vector, and its corresponding output vector. Testing involves operating the network with the test data sets in the feedforward mode for the test input vectors, and comparing the resulting output vectors with the desired test output vectors. The performance can then be evaluated by some means of an error analysis for all data sets using some type a cost function, or a graphical comparison. Depending on the desired performance evaluation, the testing set may be comprised of the data from the training set, from separate test data, or from a combination of the two. Evaluation of the network based on the training set is a good indication of how well the network has "memorized" the data under which it was trained. Evaluation of the network based on separate test data is a good indication of how well the network has extrapolated, or predicted the system.

2.4 DATA PREPARATION

When using the back-propagation algorithm discussed above, it is often required to pre-condition the data in two ways:

1. Randomizing the data may be required in order to help the algorithm converge to a global minimum in its search routine. This step is often necessary because if the data is fed to the ANN sequentially, weights are likely to adjust only slightly to "keep in step with" the continually advancing data, reaching only a local minimum at each step of training. Additional information on randomizing data is presented in Chapter 4.

2. Scaling the data is often necessary so that the algorithm does not become "saturated". Saturation of the algorithm occurs when the weight adjustments are essentially frozen, in effect paralyzing the training algorithm. Certain weight adjustments in back-propagation are proportional to the slope of the squashing function evaluated at the value of its stimulus signal. (See equation 2-13.) If the slope of the squashing function is near zero at the value of the signal, as in the case of signals with magnitudes significantly greater than unity (see Figure 2-2), then the weight adjustments essentially will not occur. Properly scaling the data helps to ensure that the magnitudes of the signals will not be too large, thus preventing saturation. Additional details of scaling are presented in Chapter 4.

In summary, the development of a successful artificial neural network requires structuring the network, scaling and randomizing the data, training the network, and then evaluating its performance.

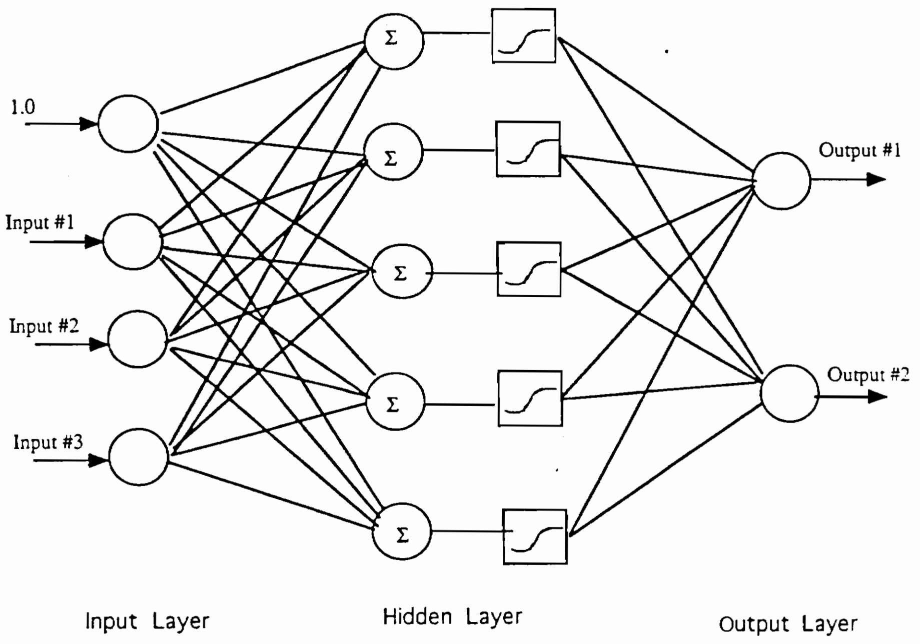


Figure 2-1: Artificial Neural Network

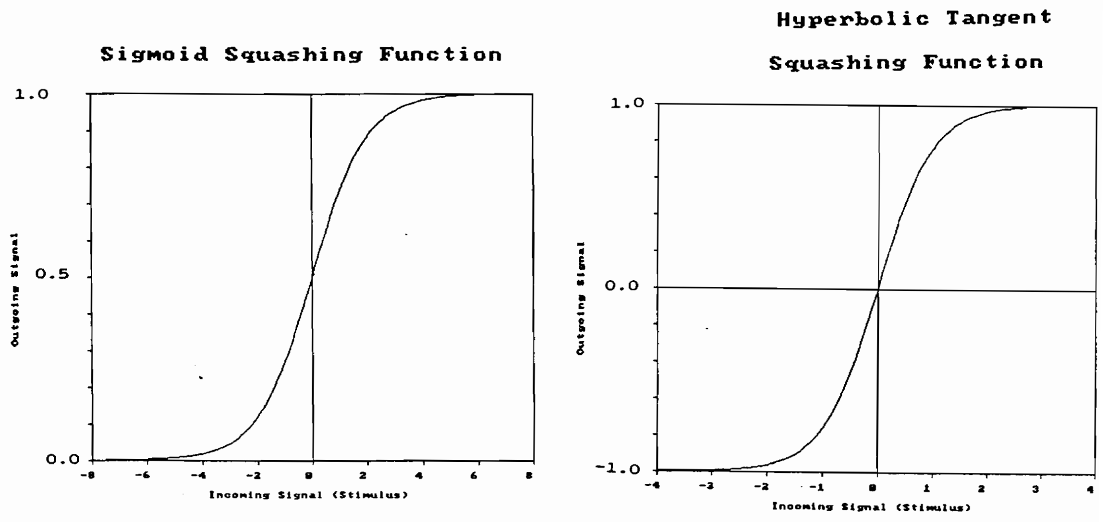


Figure 2-2: Example Squashing Functions

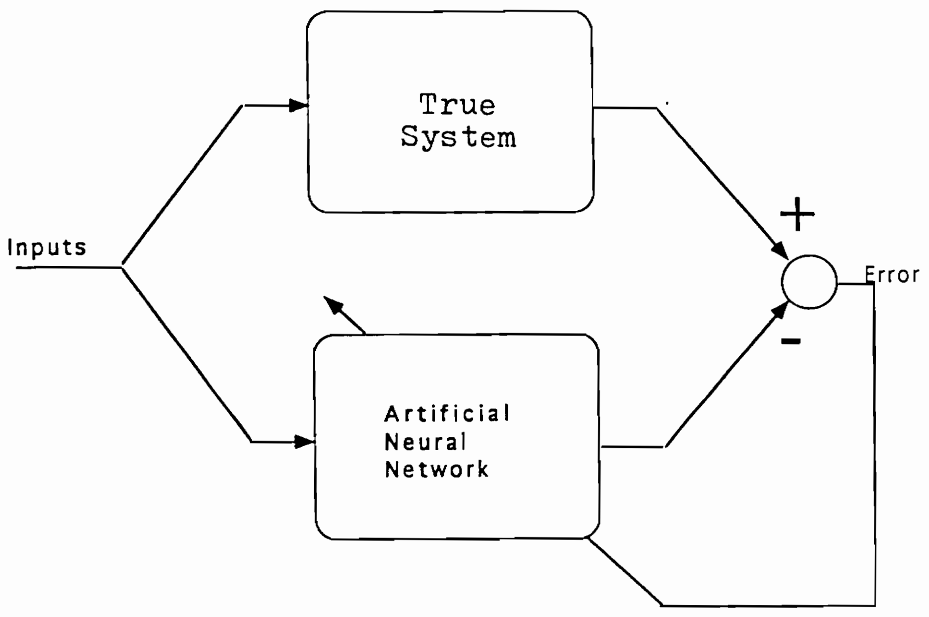


Figure 2-3: Schematic of Back Propagation Training

3 NEURAL NETWORKS APPLIED TO A FAMILIAR DYNAMIC SYSTEM

Before using ANNs to model the unknown helicopter dynamics, it was decided to apply this approach to model a well-known physical system. The purpose of this exercise is to gain familiarity with the development of neural network models, including their capabilities, their assets and shortcomings, particularly with regard to dynamic systems. Techniques applied to known mathematical models can be tested readily and their usefulness ascertained. On the basis of such experience, it is hoped the model of the helicopter dynamics can be developed to a greater accuracy. For the purpose of this investigation into the behavior of neural networks, a FORTRAN computer code was written based on the back propagation algorithm and equations developed in Chapter 2.

The known system for the purpose of this study is the motion of a spring-mass oscillator. The equation of motion of the mass is represented by the following differential equation:

$$\ddot{x}=-\frac{b}{m}\dot{x}-\frac{k}{m}x+u(t) \tag{3-1}$$

where,

m=mass
x=position of mass
b=viscous damping constant
k=spring stiffness constant
u(t)=forcing function

The first experiment attempts to train an artificial neural network to learn the response of the mass as a function of time. The results of the experiment will show how neural networks are capable of "memorizing" input/output data without necessarily learning the dynamic characteristics of the system. The second experiment focuses on identifying the system by structuring the neural network according to the governing differential equation. After successfully identifying the dynamic system, the experiment continues with several attempts to identify the dynamic characteristics when input information is limited. The experiment concludes with an observation of how the neural network behaves when provided more information than necessary.

A detailed discussion of each experiment is presented below. The results of the experiments are then summarized in Table 3-1. The performance of each network is evaluated according to four different criteria, based on the comparison of the ANN output with the true output of the system. The comparisons are mainly subjective, relying on visual inspection of the graphs rather than a numerical error analysis. The purpose is to obtain insight into the behavior of neural networks applied to dynamic systems. Lessons learned can then be applied to the problem of estimating helicopter dynamic loads in Chapter 4.

The four criteria for evaluating each neural network are:

1. Level of difficulty in training the network. The difficulty of training is best measured by the number of iterations required to achieve the desired result. Here, one iteration refers to one set of internal weight adjustments for a given data set. (See Section 2.2.3)

2. Capability of the network to learn the characteristics of the system within the training range.

3. Capability of the network to predict the characteristics of the system outside of the training range.

4. Determination of whether or not the ANN has "grasped", or identified the system dynamics. Performance evaluation in this area does not exclusively focus on how closely the outputs match. Rather, it may disregard the fine details and look at the general behavior of the network output, comparing it to the general behavior of the true system. "Does it attempt to 'follow' the true output?" A good test of this figure of merit is to alter the forcing function and see if the network output is able to correctly adapt to such a change.

3.1 LEARNING THE SYSTEM RESPONSE

The first attempt to train an artificial network model considers the motion of the spring-mass oscillator with no damping and no forcing function. The spring constant and mass are assigned values of k=1 N/m and m=.048 Kg. The initial position and velocity are x_o=0.3 m and v_o=0.8 m/s. The analytic solution to the system is: [3]

$$x = x_o \cos(\omega_n t) + \frac{v_o}{\omega_n} \sin(\omega_n t) \qquad (3\text{-}2)$$

where,

$$\omega_n = \sqrt{\frac{k}{m}} = 4.56 \ rad/s$$

The desired output of the model is the value of the position, x. The network is created with one hidden layer consisting of 16 nodes, plus a bias. The squashing function is the sigmoid function:

$$f(x) = \frac{1}{1+e^{-x}} \qquad (3\text{-}3)$$

The input layer consists of 4 nodes, plus a bias. To train the neural network, data sets are constructed consisting of an input vector and the value of the output (position) obtained from the true solution. The input vector for this exercise consists of values of time raised to the first, second, third, and fourth powers (t^1,t^2,t^3,t^4). 1,401 data sets over the interval of time from -0.4 to 1.0 seconds are created. A single iteration of the back-propagation training consists of selecting one of the input/output sets at random, calculating the network output, calculating the error, and adjusting the weights accordingly. The results of training the network for 34,000 iterations are graphed in Figure 3-1a. As seen from the graph, the neural network has learned the output response for the training range only. The training range is then expanded by including data sets in the interval of $-1.8 \leq t \leq 1.8$ seconds. (1 data set for every 1/1000 of a second) Starting with the weights fixed from the first 20,000 iterations of the previous case, training is resumed using the expanded training file. After 16,000 more iterations, the results of Figure 3-1b are obtained. Again, the network has performed well within the training interval, but has performed poorly outside of this interval, as evidenced by the flat lines in the figures. The network has matched the input/output data, but has not identified the dynamic characteristics of the system, the causal relationship which determines the position of the mass. This result is not surprising, for artificial neural networks develop algebraic relationships between the inputs and outputs, and the inputs for this case (t^1,t^2,t^3,t^4) contain no information about the dynamic characteristics of the system.

3.2 LEARNING THE SYSTEM DYNAMICS

In order to incorporate the dynamics into the model, the network is structured according to the parameters which define the motion of the system, as in the governing differential equation, equation 3-1. As such, the new inputs are selected as velocity, $x'(t)$, acceleration $x''(t)$, and the external stimulus, $u(t)$.

The spring-mass system for this experiment is defined by the following parameters: m=.005 Kg, k= 1.0 N/m, b=0.0, x_o=0.0, v_o=0.0, and $u(t)=.5\sin(0.7\omega_n t)$. The neural network geometry is defined by 1 hidden layer consisting of 16 nodes plus a bias. Inputs nodes include $u(t)$, $x'(t)/20$, and $x''(t)/400$, plus a bias. (The velocity and acceleration are scaled such that the inputs are of similar magnitudes, and are not greater than unity.) The training data consists of 1,001 data sets in the interval $0 \leq t \leq 1.0$ seconds.

As seen in Figure 3-2a, the network trains relatively quickly, requiring less than one epoch (=1,001 iterations) to learn the characteristics of the system both inside and outside of the training interval. An additional 4,000 iterations improve the fine details of the output, so that the desired output and the training output are almost identical. The weights are then fixed and the network is further tested for the case when the control is changed to $u(t) = 0.5\sin(.35\omega_n t)$. The results in Figure 3-2b show that the network adapts to the change in the control. Unlike the first experiment, the network has identified the dynamic characteristics of the system, rather than merely matching input/output data.

Having successfully identified the system, the experiment continues for the case when acceleration information is not available. Figure 3-3 demonstrates that when only the control and velocity inputs are included, the ANN performance deteriorates.

To compensate for the lacking acceleration data, information can be included in the form of past, or delayed inputs. Because the spring mass oscillator is a dynamic system, involving cause-effect mathematical relationships, past control inputs and past states may also aid in identifying the system. Potentially, adding delays to the network can be interpreted as increasing the order of the state derivative inputs. For example, by feeding the neural network present and past values of velocity x′ at fixed intervals of Δt, information on acceleration x" is indirectly provided:

$$\begin{aligned}\ddot{x} &\approx \frac{\dot{x}(t)-\dot{x}(t-\Delta t)}{\Delta t} \qquad (3\text{-}4)\\ &\approx C_1\dot{x}(t)+C_2\dot{x}(t-\Delta t)\end{aligned}$$

Thus, the algorithm has the potential to adjust the weights,

C_1 and C_2, in such a manner as to "estimate" the acceleration. That is, acceleration is approximated as the weighted linear combination of present and past values of velocity.

Several networks are examined for the purpose of evaluating the performance of the ANN when time delays are included. Through the use of trial-and-error, a search is conducted for the combination of delays that will enable the ANN to most successfully identify the system. The results, summarized in Table 3-1 and Figures 3-1 through 3-8, show that although not as successful as including higher order derivatives, the delays can be somewhat helpful in training the network to learn the dynamics of the system. The most successful case which did not include acceleration information is seen in Figures 3-9a and 3-9b. Included in this case are delays of position, velocity, and control.

The final experiment associated with the spring-mass system addresses the situation when the neural network is trained with more information than necessary. One might expect that the algorithm will simply adjust the appropriate weights to a value of zero as a means of ignoring unnecessary inputs. For this experiment, a network is constructed using the same geometry as for the case where the inputs included $u(t)$, $x'(t)/20$, and $x''(t)/400$, except that the three more input nodes are added: $x'(t-.3)/20$, $u(t-.3)$, $x(t-.3)$. The network is trained over the same interval of $0 \leq t \leq 1.0$, and is tested by fixing the weights and changing the control to $u(t)=.5\sin(.35\omega_n t)$. The results of Figures 3-10a and 3-10b show that the network has performed reasonably well in identifying the system, but not as well as 3-input network (Figures 3-2a and 3-2b). Thus, including more information than necessary has hindered the learning process.

3.3 SUMMARY OF PERFORMANCE

The performance of the various ANNs discussed above are summarized in Table 3-1. The first three columns define the geometry of the input layer for each network by specifying the inputs, input delays, and output delays. Columns 4 through 7 describe the performance of the network in terms of training, testing, and system identification. The last column indicates the graphs upon which the performance evaluation is based.

Training difficulty is best quantified by the number of iterations required. The higher the difficulty, the more iterations are required to achieve acceptable results. Each number represents the number of iterations in thousands that the network undergoes before satisfactory results are obtained, or until the performance no longer improves significantly with further training. Training, testing, and system identification performance are evaluated according to the following descriptions of excellent, good, fair, and poor.

Table 3-1 : ANN Performance Summary for Spring Mass System

Inputs	Input Delays	Output Delays	Training Rate (# of iterations) (Thousands)	Training Performance	Testing Performance	System Identif-ication	Graphical Comparison
t,t^2, t^3,t^4			35	good	poor	poor	3-1a,b
x'/20 x"/400 u(t)			1 5	good excellent	good excellent	excel. excel.	3-2a,b
x'/20 u(t)			10	fair	fair	fair	3-3
x'/20	u(t-.1)		7	poor	poor	poor	3-4
x'/20	u(t-.3)		7	poor	poor	fair	3-5
x'/20 u(t)	u(t-.3)		7	fair	fair	fair	3-6
x'/20 u(t)	x'(t-.3)		7	fair	fair +	good	3-7
x'/20 u(t)	u(t-.3)	x(t-.3)	10 50	fair excellent	poor poor	fair fair	3-8
x'/20 u(t)	x'(t-.3)/20 u(t-.3)	x(t-.3)	10	excellent	good	good	3-9a,b
x'/20 u(t) x"/400	x'(t-.3)/20 u(t-.3)	x(t-.3)	15	fair	fair	good	3-10a,b

3.4 CONCLUSIONS

Having investigated the behavior of neural networks applied to a known physical system, the following conclusions are made. It is hoped that these conclusions can be used to improve the accuracy of the helicopter dynamic loads modeling. It is understood that these conclusions are made on the basis of this study only, and may or may not apply to dynamic systems in general.

1. Because artificial neural networks are limited to developing algebraic relationships, the dynamics of the system must be provided through the input data. The choice of input data may determine whether the ANN identifies the system (Figures 3-2a,b); merely "memorizes" data for a given range (Figures 3-1a,b); accomplishes a little of both (Figures 3-7,3-8); or fails altogether (Figure 3-4).

2. In identifying the dynamics of the system, best results are obtained using the higher order derivatives. For the spring-mass system, the second order derivatives were used to develop the model to its full potential. This is demonstrated by the matching curves in Figures 3-2a and 3-2b, which could not be duplicated for networks lacking acceleration information.

3. In cases where higher-order derivatives are lacking, delays can be included in an attempt to improve training. Figures 3-9a and 3-9b demonstrate how adding delays of position, velocity, and the control function improved results slightly from when these delays were absent.

4. Including more information to the ANN is not necessarily better. Comparing Figures 3-10a and 3-10b with Figures 3-2a and 3-2b, it is apparent that better results are obtained when only the necessary inputs are supplied. Otherwise, the performance of the neural network model can be compromised.

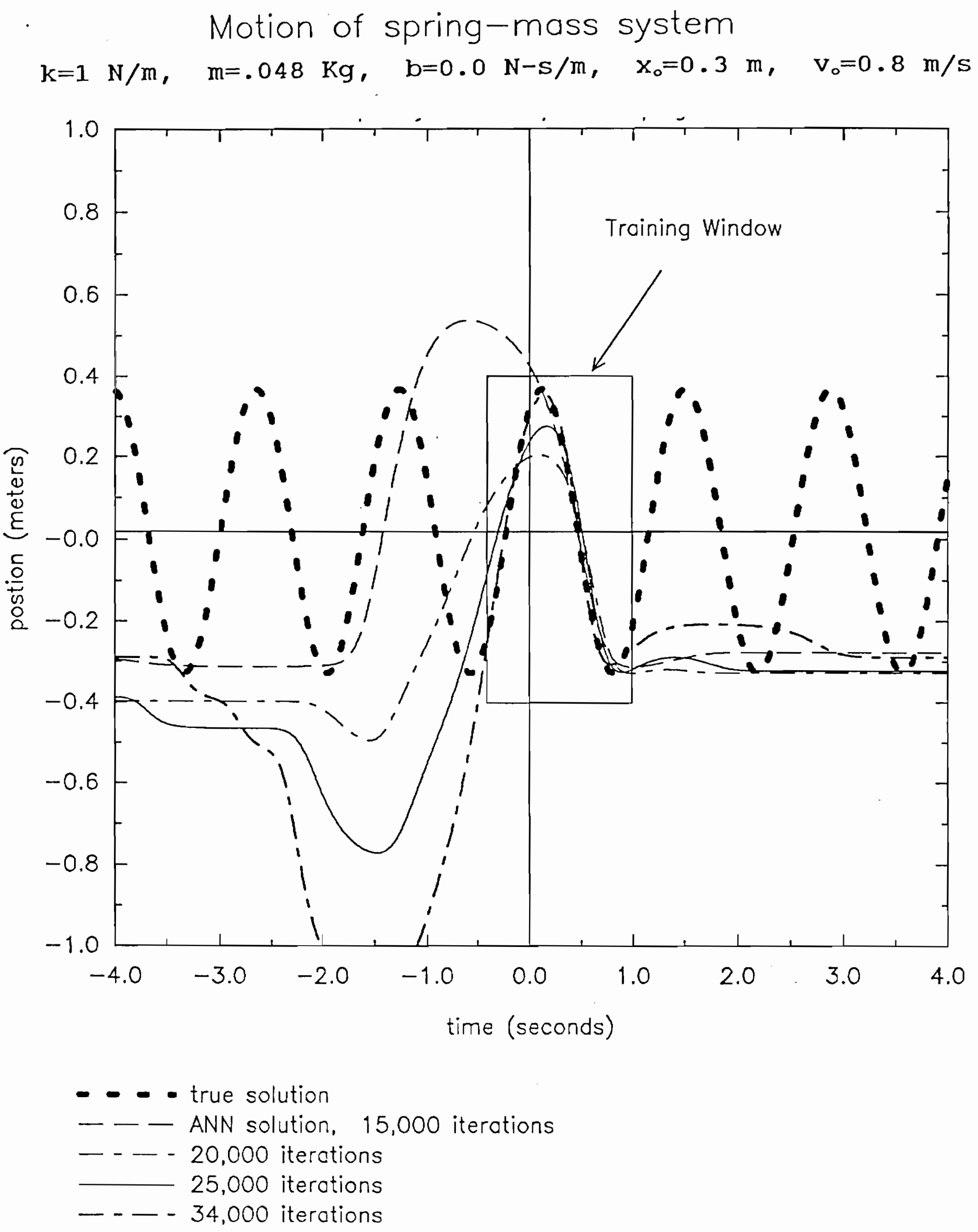


Figure 3-1a: Training Spring Mass System
Inputs = t, t^2, t^3, t^4

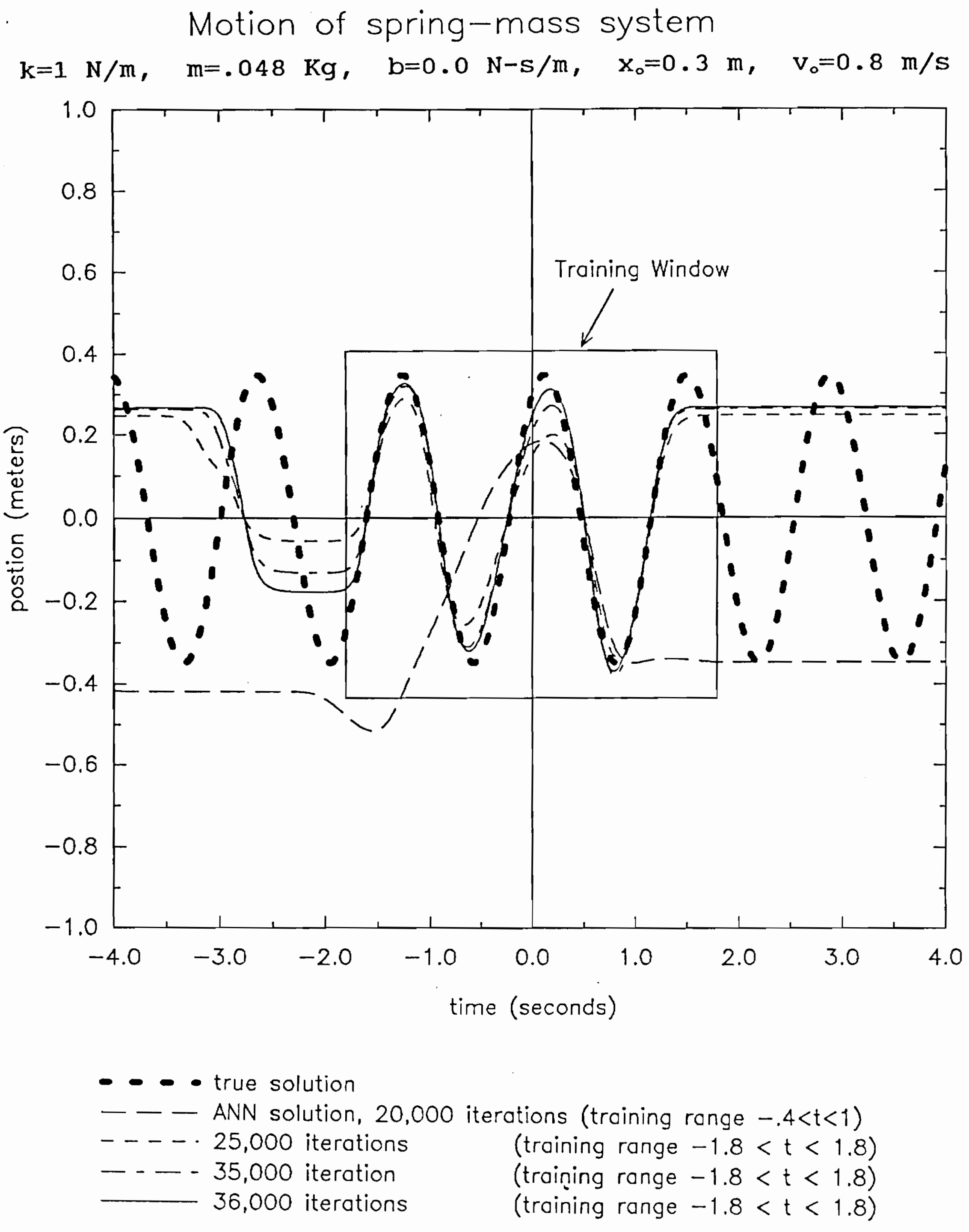


Figure 3-1b: Training Spring Mass System
Inputs = t, t^2, t^3, t^4

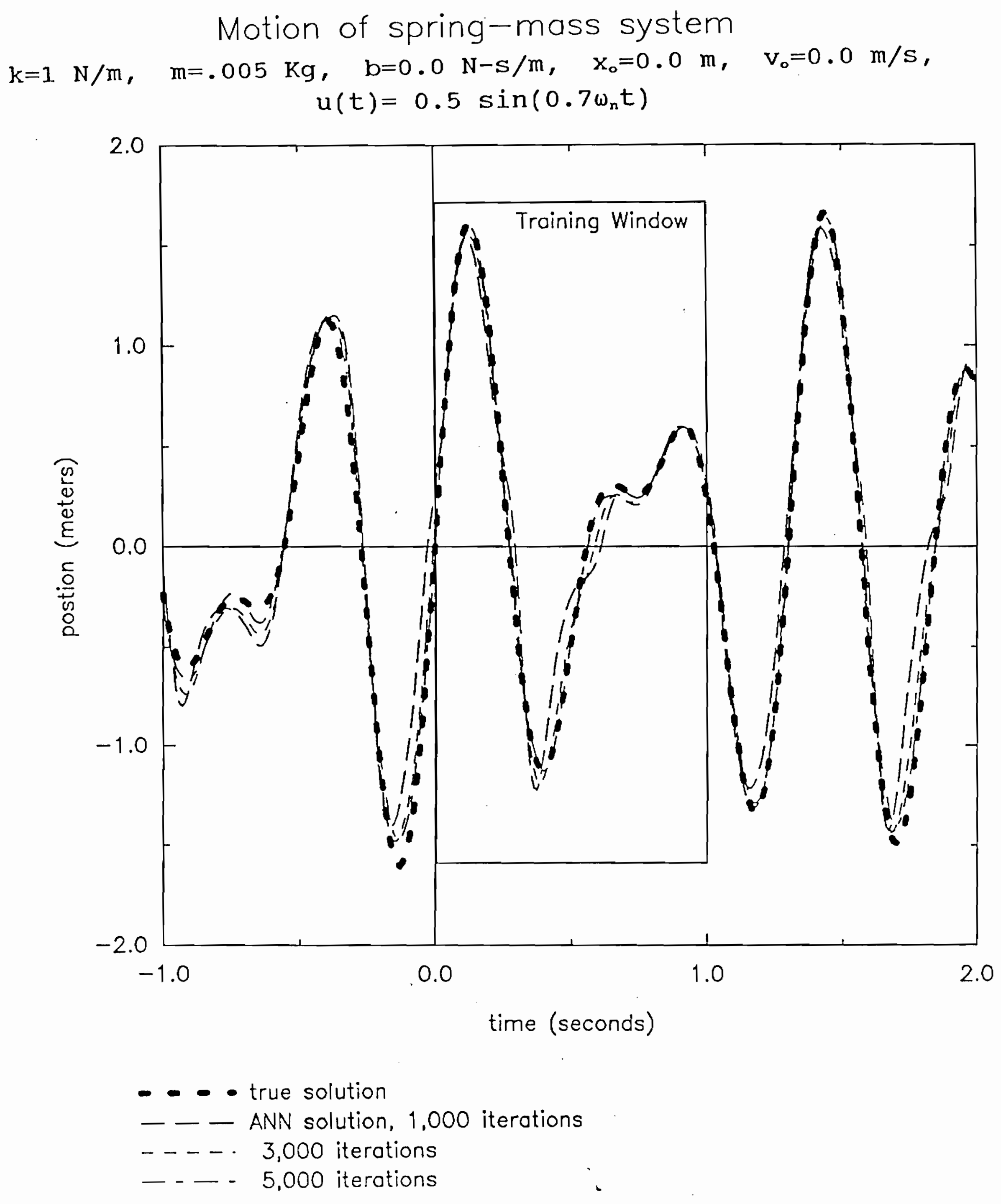
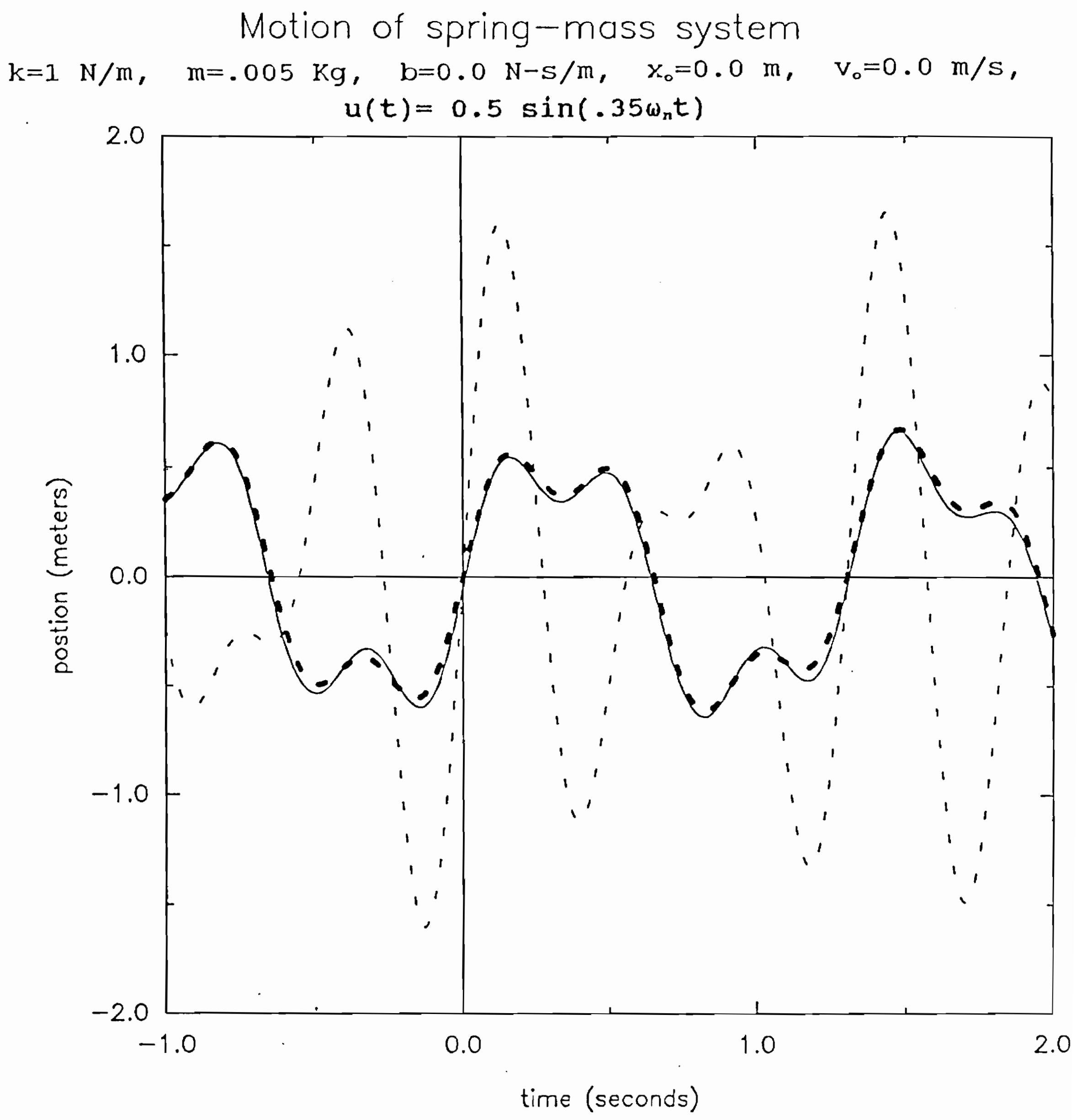


Figure 3-2a: Training Spring Mass System
Inputs = x′/20,x″/400, u(t)



- - - - true solution, u(t)=.5*sin(.70*wn*t) (old control)
▬ ▬ ▬ ▬ true solution, u(t)=.5*sin(.35*wn*t) (new control)
———— network solution, fixed weights

Figure 3-2b: Testing Spring Mass System
Inputs = x'/20,x"/400, u(t)

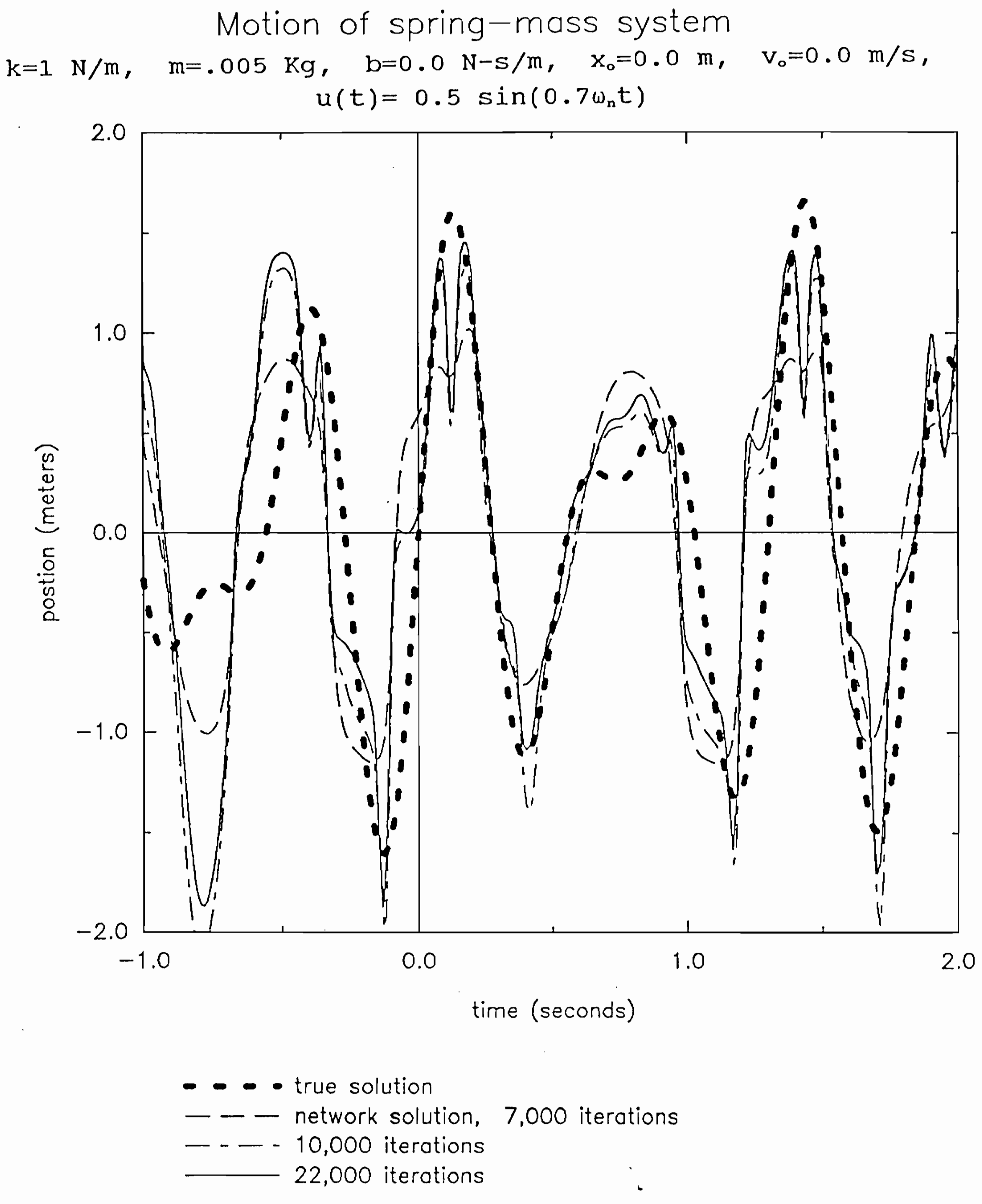


Figure 3-3: Training Spring Mass System
Inputs = x'/20, u(t)

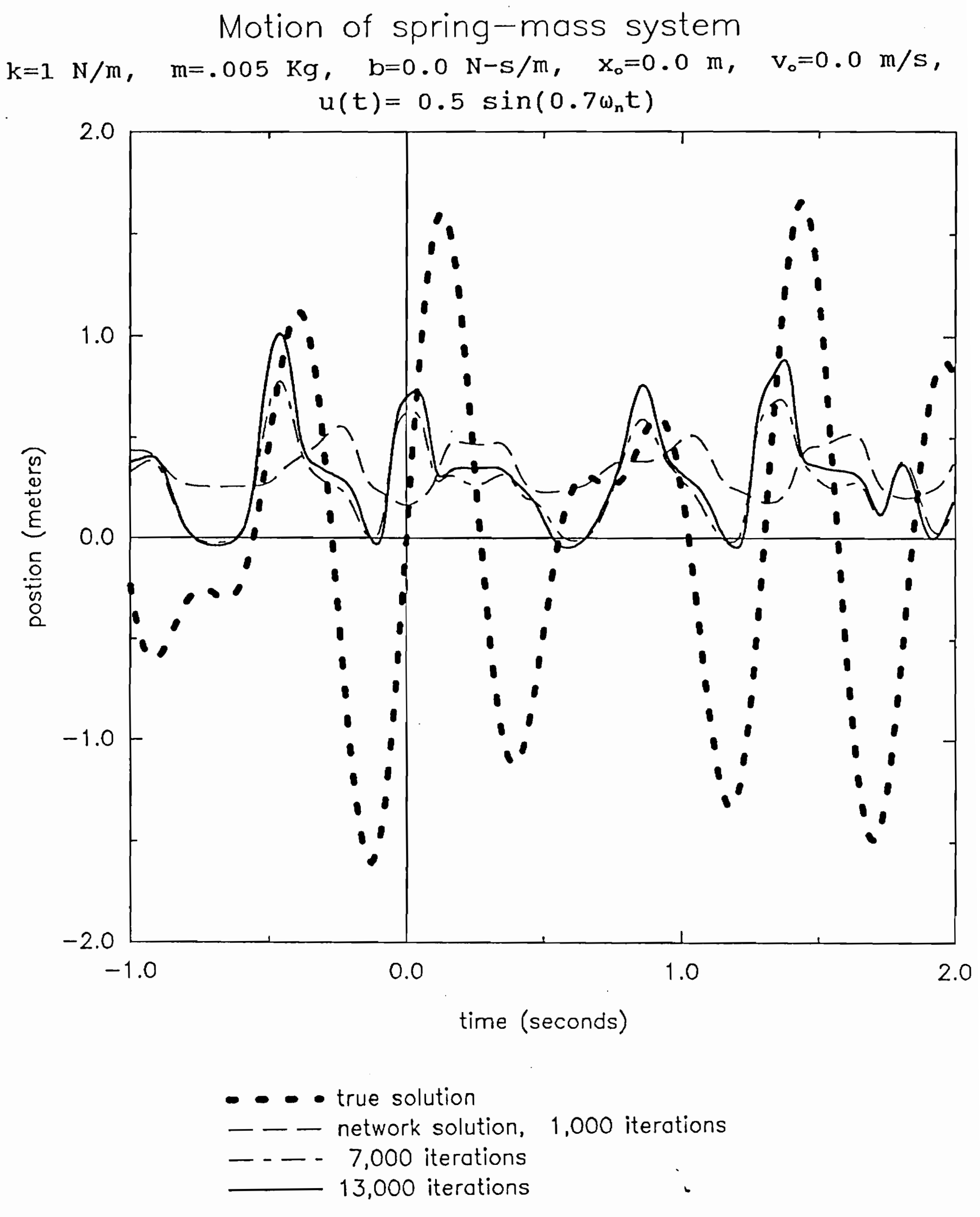


Figure 3-4: Training Spring Mass System
Inputs = x'/20, u(t-.1)

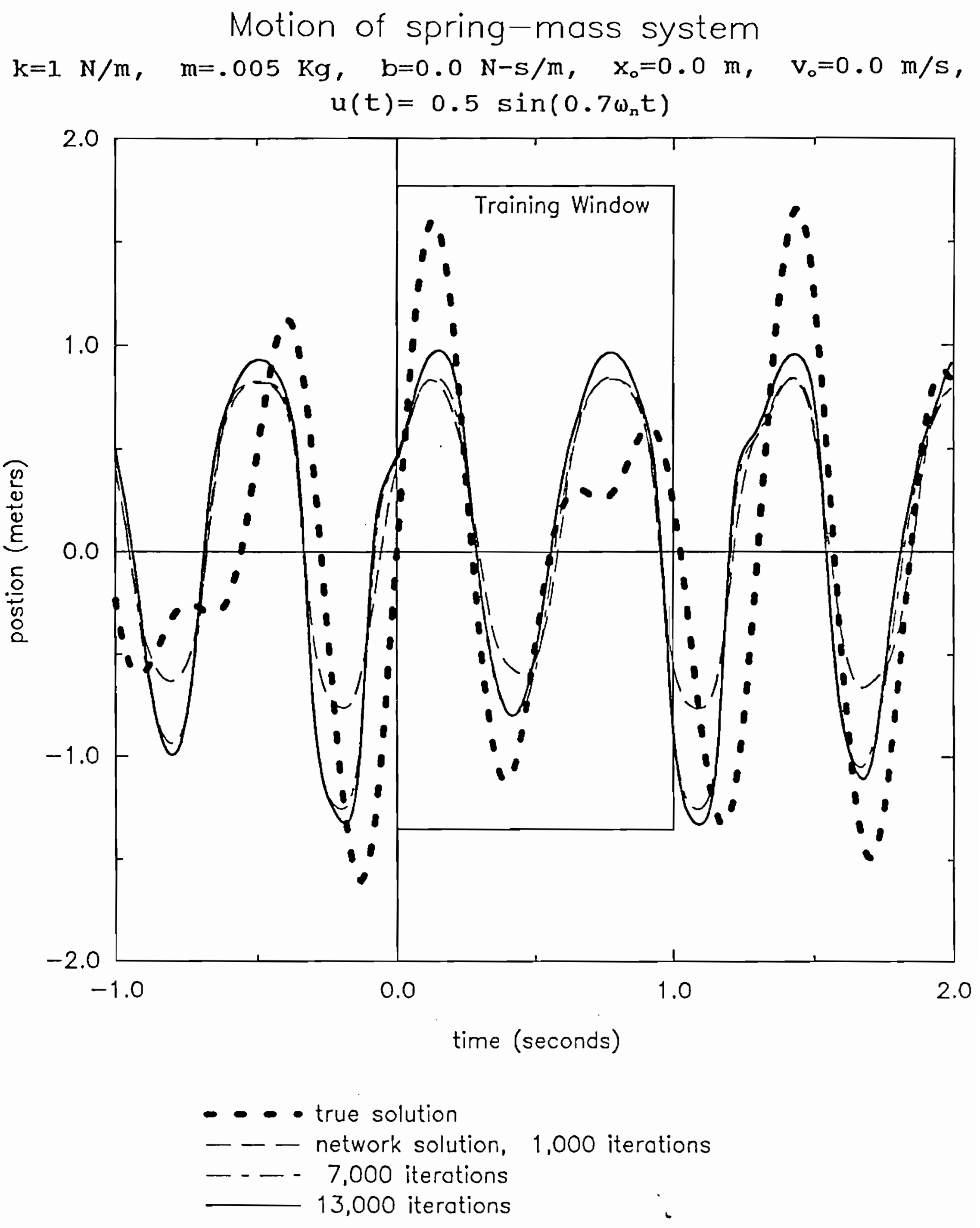


Figure 3-5: Training Spring Mass System
Inputs = x′/20, u(t-.3)

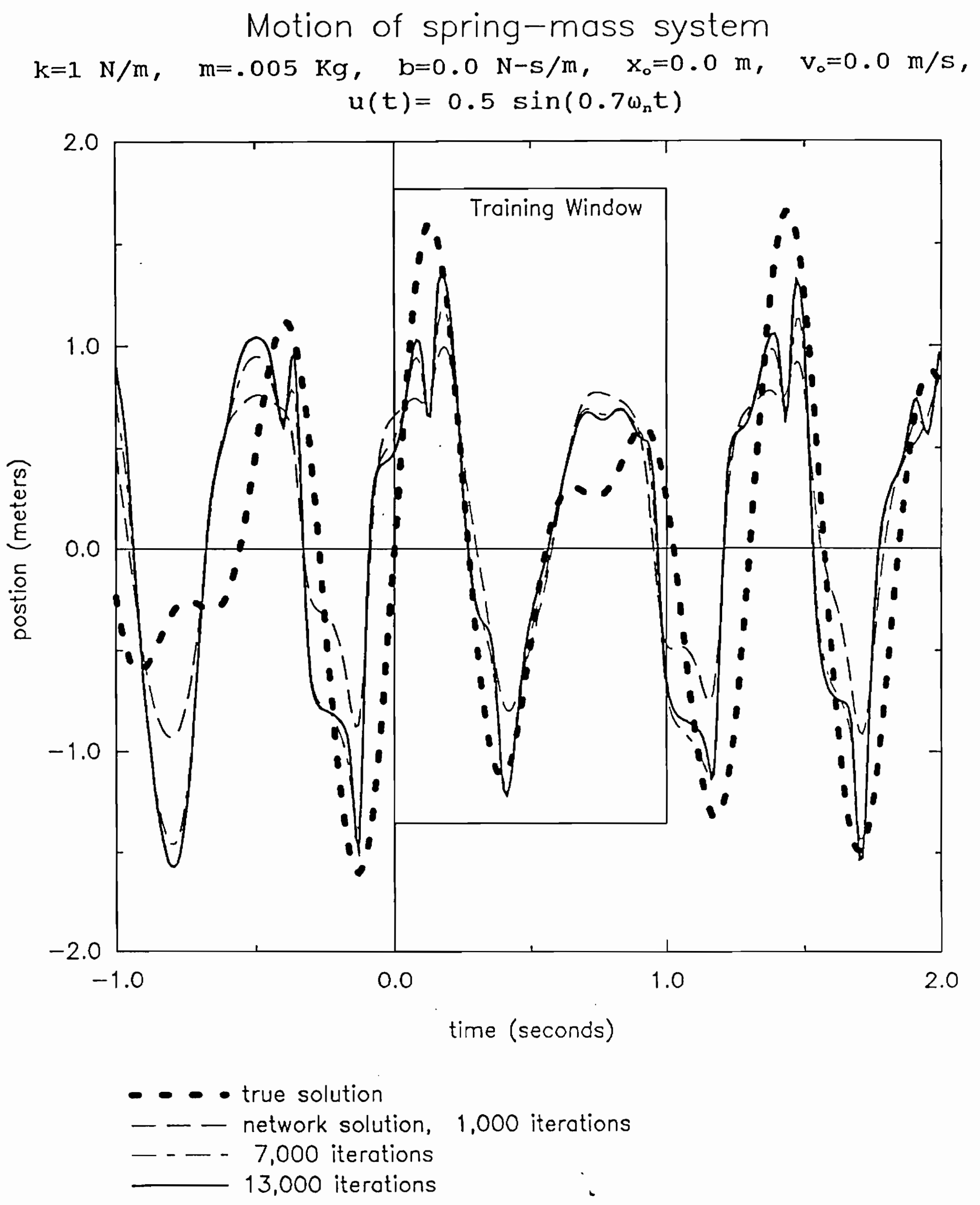


Figure 3-6: Training Spring Mass System
Inputs = x'/20, u(t), u(t-.3)

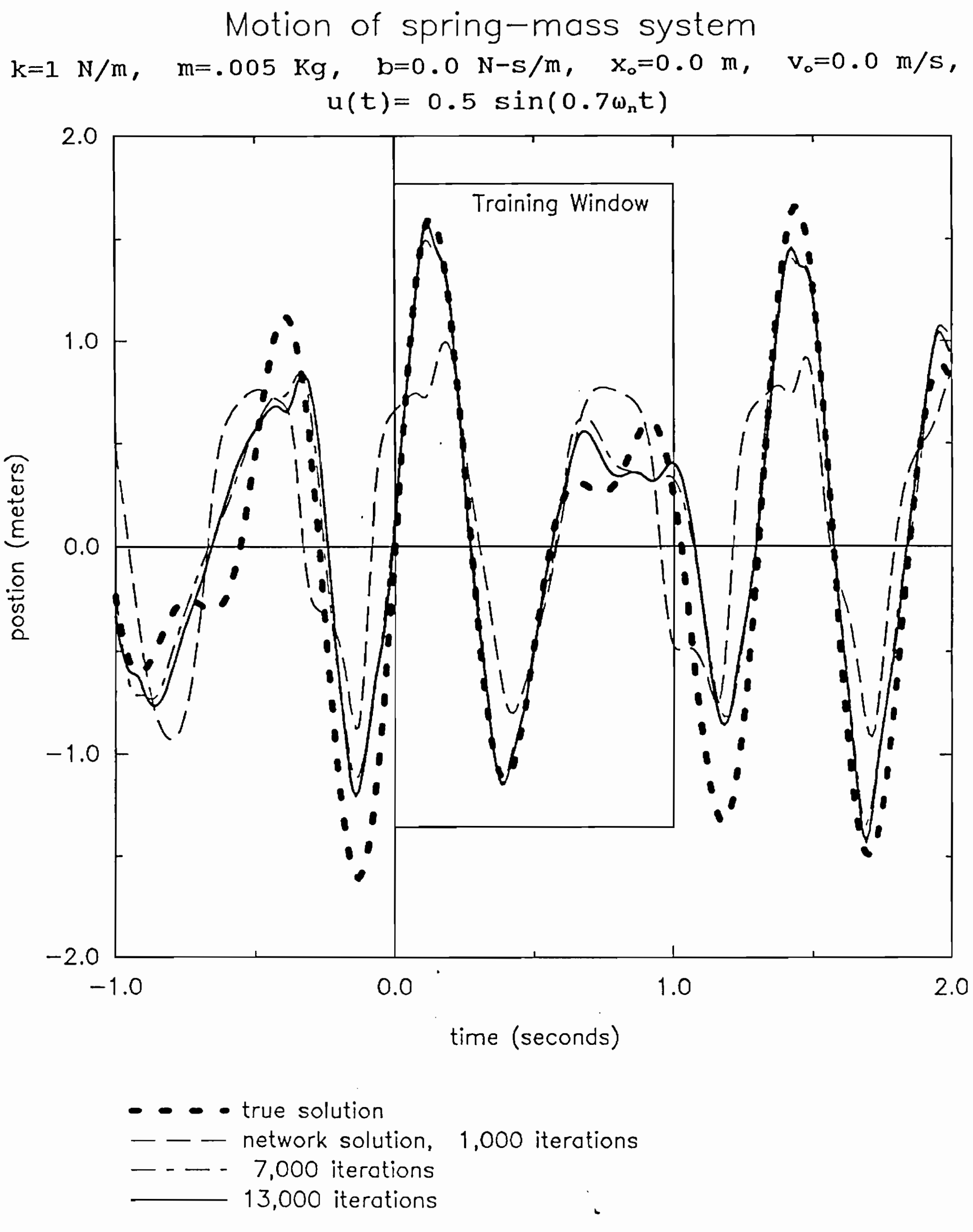


Figure 3-7: Training Spring Mass System
Inputs = x'/20, u(t), x'(t-.3)

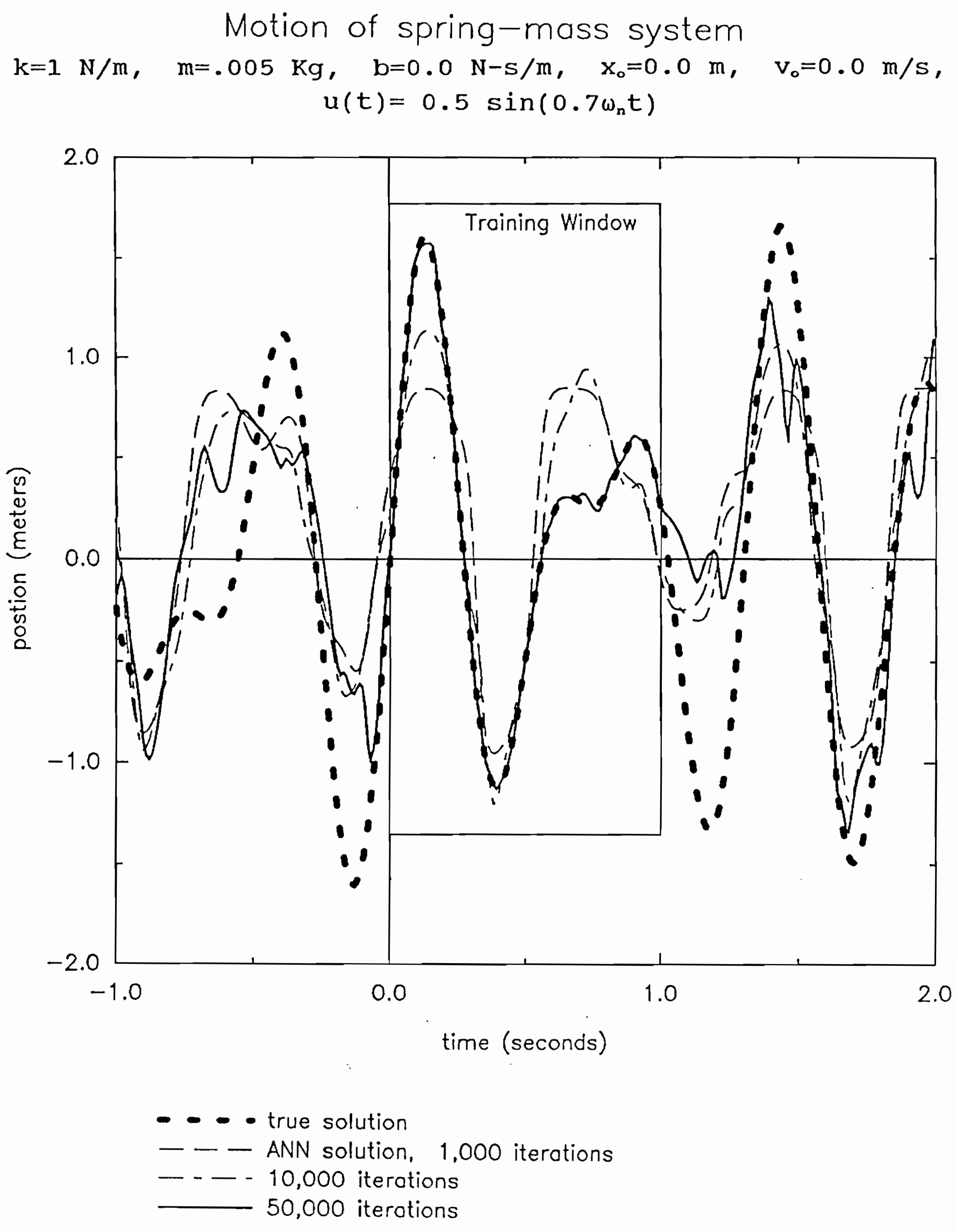


Figure 3-8: Training Spring Mass System
Inputs =.x'/20, u(t), u(t-.3), x(t-.3)

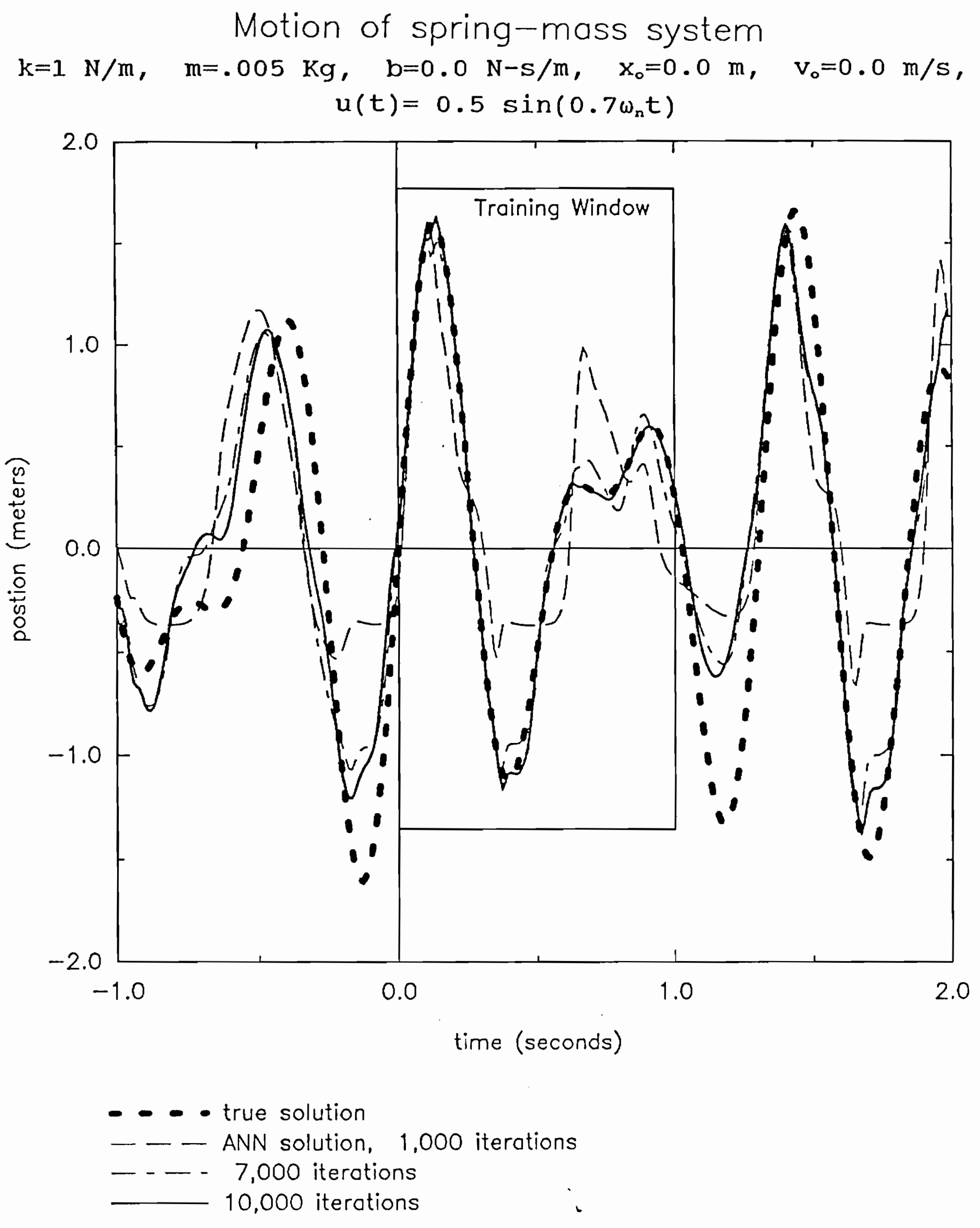


Figure 3-9a: Training Spring Mass System
Inputs = x′/20, u(t), x′(t−.3), u(t−.3), x(t−.3)

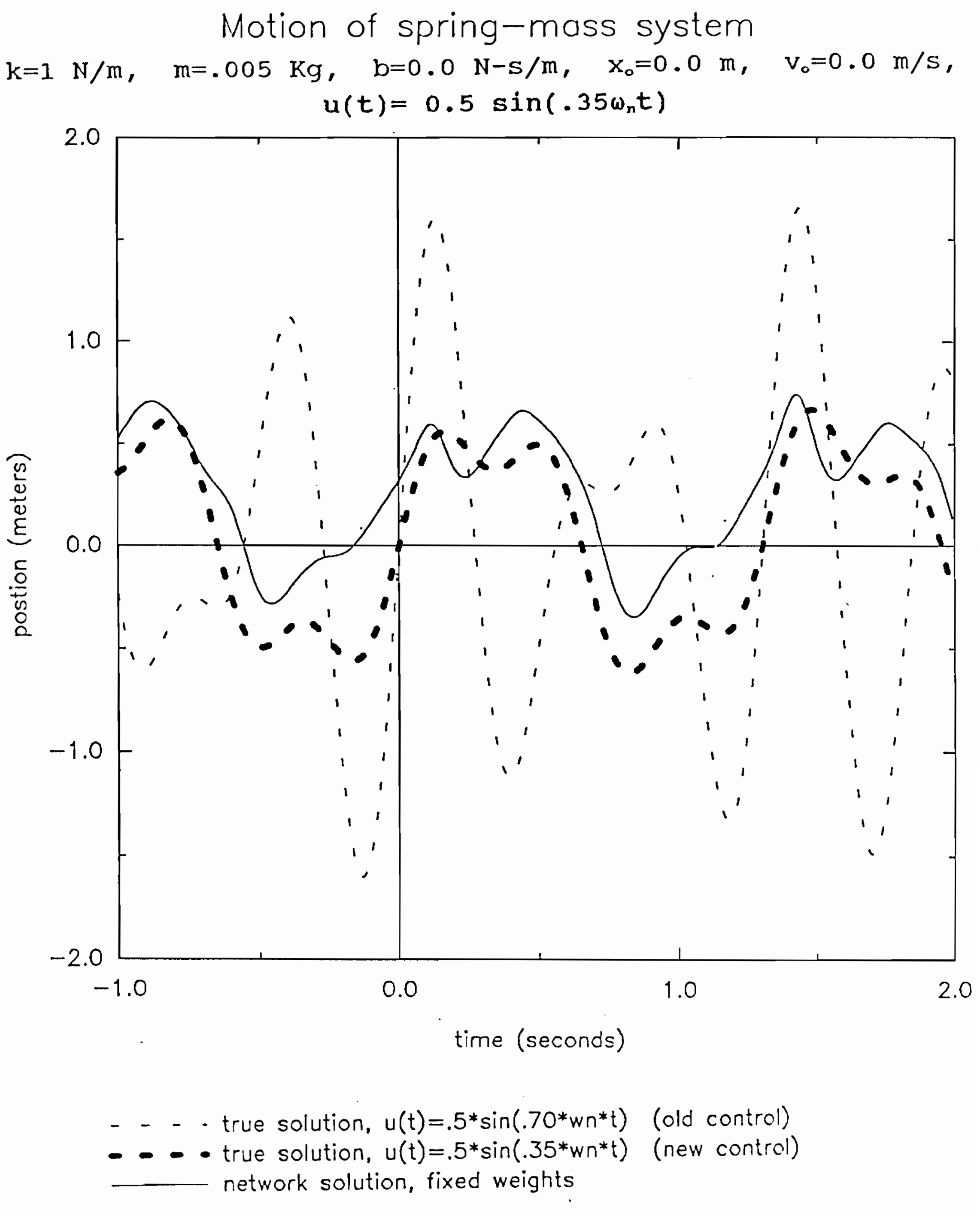


Figure 3-9b: Testing Spring Mass System
Inputs = x′/20, u(t), x′(t−.3), u(t−.3), x(t−.3)

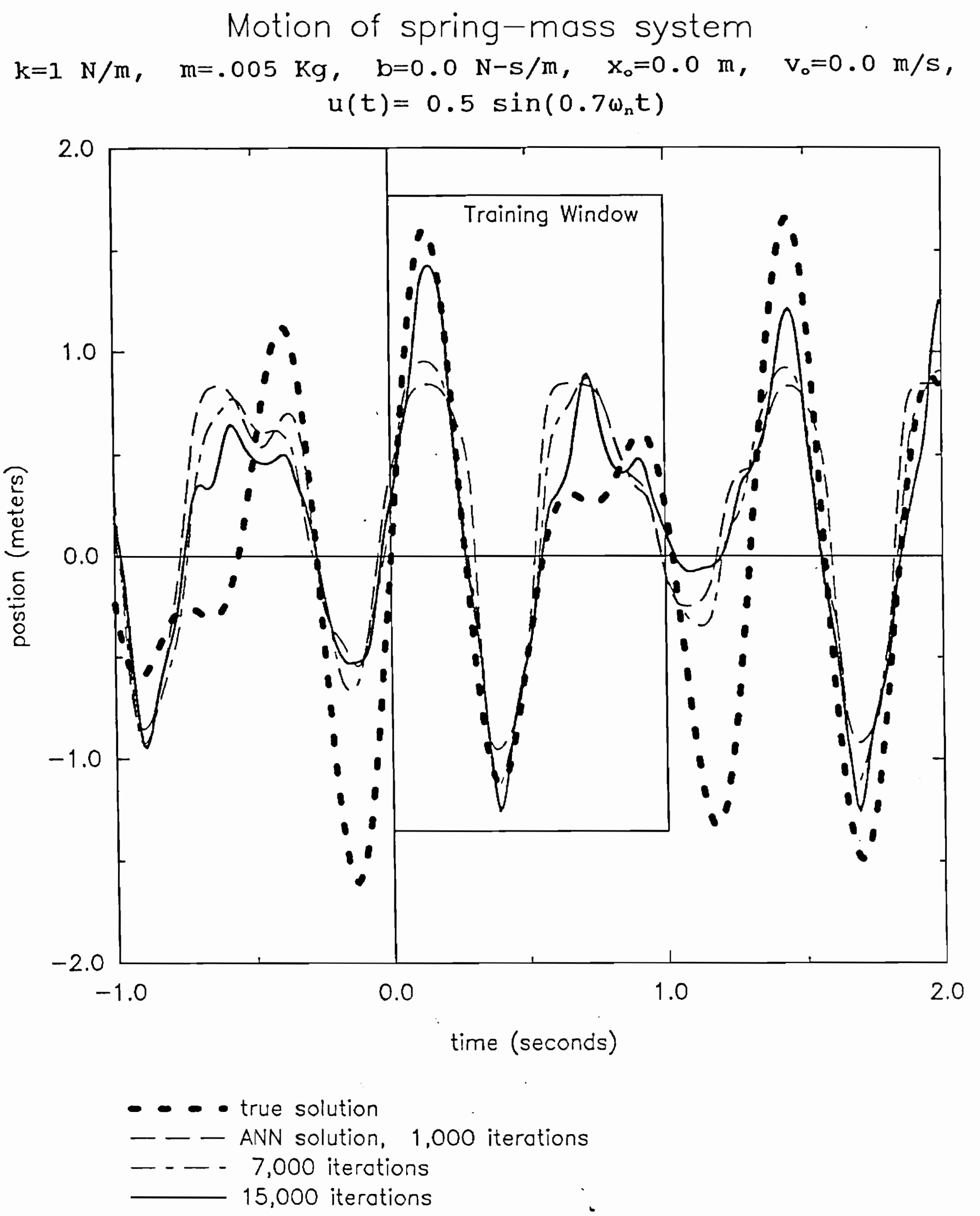


Figure 3-10a: Training Spring Mass System
Inputs = x'/20, u(t), x"(t)/400, x'(t-.3), u(t-.3), x(t-.3)

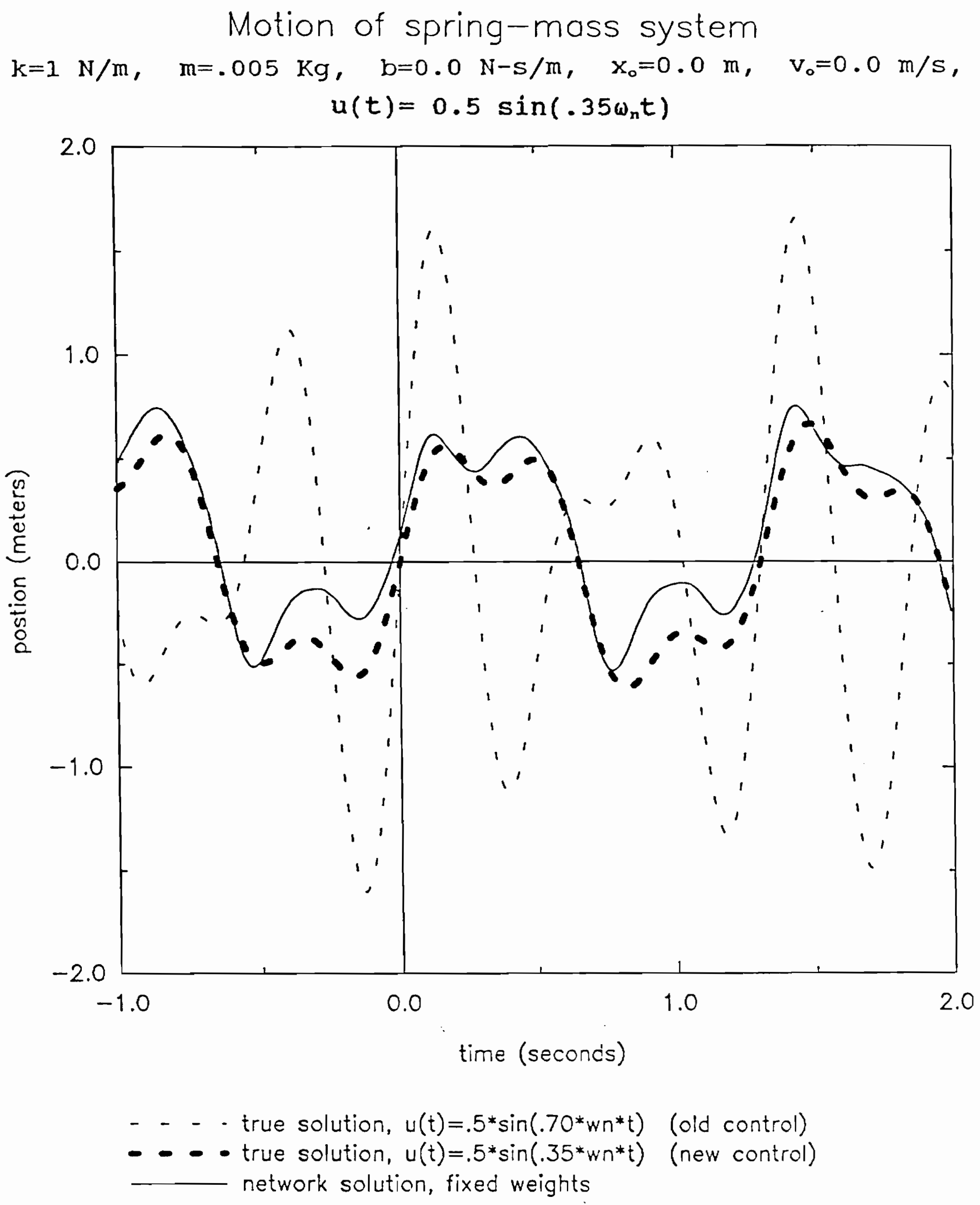


Figure 3-10b: Testing Spring Mass System
Inputs = x'/20, u(t), x"(t)/400, x'(t-.3), u(t-.3), x(t-.3)

4 ESTIMATION OF HELICOPTER MAIN ROTOR DYNAMIC LOADS AND FATIGUE

To date, attempts to improve helicopter fatigue estimation have included computer simulations, and fitting the aircraft with "fatigue meters" [9]. Computer simulation of dynamic loading is limited by the difficulty in developing the governing equations which estimate such loading. Particular difficulties arise when discontinuities in the aircraft structure are present, such as hinges or joints in the main rotor mechanism. Also, computer simulations have encountered problems with excessive run times and have not indicated significant increases in accuracy over other methods [10].

"Fatigue meters" entail using on-board devices to estimate dynamic flight loadings. It is desirable to develop a system capable of calculating the fatigue in real time by on-board data processors, giving a read-out of life consumed for major components after each flight [9]. The alternative is to record the necessary data, and at a later time, estimate the fatigue by processing the recorded information. The latter technique requires a significant amount of data storage capability, and does not provide the efficient means of directly logging the consumed life immediately after each flight.

One method of in-flight monitoring involves placing strain gauges on the all major components of the aircraft.

Care is taken to place the strain gauges in areas well away from local stress concentrations. Records from these gauges then enable the loading to be found directly [9].

Another method of in-flight monitoring involves using flight parameters to estimate the dynamic loading. This approach analyzes the measured loads and flight parameters on fully instrumented test aircraft, from which equations are empirically developed relating the desired loads to the known flight parameters. These equations are then used to estimate fatigue life on all aircraft by measuring the flight parameters. (This method is limited in how closely each aircraft behaves like the flight test helicopter.) One method of developing these equations involves using statistical regression techniques [9]. More recently, emphasis has been placed on modeling these loads using artificial neural networks [7].

4.1 OVERVIEW OF PROBLEM

The following sections describe specifically the problem objective and the various ANN techniques used in an attempt to model the dynamic loads for a particular flight test helicopter. These techniques incorporate lessons learned from Chapter 3, where ANNs were used to model the familiar dynamic system of the spring-mass oscillator. The dynamic load models are then evaluated according to their potential to estimate fatigue.

The flight test helicopter which provides the data for the following analysis is a US Navy SH-60B Seahawk. Built by Sikorsky, the helicopter functions in the role of LAMPS, or Light Airborne Multi-Purpose System. Aside from its ship-to-ship utility role, other missions of the helicopter include search-and-rescue, anti-submarine warfare, and anti-ship surveillance [11].

The computer algorithm which was used to develop, train, and test neural network models was written by Michael Tran of Virginia Polytechnic Institute and State University [12].

4.2 OBJECTIVE

The objective of this study is to accurately estimate the dynamic loads and fatigue of the main rotor components, using readily available information, such as data from the flight recorder. The "known" or "available" information takes the form of 28 separate parameters, including altitude, attitude, angular and translational rates and accelerations, relative wind direction, and deflections of control surfaces, servos, and pilot controls. These 28 parameters, listed in Table 4-1, comprise the "inputs" to the neural network models. Information on structural internal loads takes the form of 5 separate parameters, including forces, stresses, and moments at different locations in the main rotor assembly. These 5 parameters, also listed in Table 4-1, comprise "unknown" or "unavailable" information for fleet helicopters. As such, they are the "outputs" which the neural network model attempts to estimate. For the purpose of training and testing various neural network models, the structural loads are known through the use of sensors on the flight test helicopter. For the purpose of applying the neural network models to fleet helicopters, the output information is not directly available, and would be obtained through the "available" input information and the neural network simulation and prediction of the dynamic system.

For the purpose of training neural network models for estimating loads on fleet helicopters, the full range of data available from the flight test helicopter would be used. For the purpose of evaluating and comparing the performance of various networks, only a portion of the data is used for training, and the remainder is used for testing and evaluation.

Table 4-1 :
Available Input/Output Information from SH-60B Flight Tests

AVAILABLE INFORMATION
(INPUTS)

#1	Velocity (knots)
#2	Boom Altitude (ft)
#3	Boom Rate of Climb (fpm)
#4	Pitch Attitude (deg)
#5	Roll Attitude (deg)
#6	Main Rotor Speed, Nr (%)
#7	Weight (lb)
#8	Collective Stick Position (%)
#9	Lateral Stick Position (%)
#10	Longitudinal Stick Position (%)
#11	Pedal Position (%)
#12	Aft Primary Servo Output Position (%)
#13	Fwd Primary Servo Output Position (%)
#14	Lateral Primary Servo Output Position (%)
#15	Pitch Rate (deg/s)
#16	Roll Rate (deg/s)
#17	Yaw Rate (deg/s)
#18	Sideslip Angle (deg)
#19	Angle of Attack (deg)
#20	Tail Rotor Impressed Pitch (deg)
#21	Stabilator Angle Position (deg)
#22	Load Factor (G's)
#23	Longitudinal Acceleration (G's)
#24	Lateral Acceleration (G's)
#25	Pitch Acceleration (deg/s^2)
#26	Roll Acceleration (deg/s^2)
#27	Yaw Acceleration (deg/s^2)
#28	Fuel Totalizer (Total Fuel Used, lbs)

DESIRED LOAD INFORMATION
(OUTPUTS)

#1	Main Rotor Push Rod Load (lb)
#2	Main Rotor Blade Normal Bending (Sta X) (in-lb)
#3	Main Rotor Spar Stress Trailing Edge Top At 40% Radius(psi)
#4	Main Rotor Rotation Scissors(lb)
#5	Main Rotor Shaft Bending Moment (in-lb)

4.3 DATA PREPARATION

In attempting to model the dynamic loads with neural networks, it is important to manipulate the data to allow for proper training using the back propagation algorithm, to aid in identification of the dynamic system, and to place the data in the proper format for use by the computer code. Data manipulation for these purposes includes scaling, eliminating various input parameters, including time delays, randomizing the data, and constructing training and testing files. This sequence is shown schematically in Figure 4-1a.

The computer code expects the data to exist in a file consisting of N rows and I + O columns, where I represents the number of inputs to the neural network, O represents the number of outputs, and N represents the number of data sets to be used for training or testing purposes. A "data set" or "data group" refers to the combination of an input vector and its associated output vector, represented by any row in the sample data file as shown below:

$$\left[\begin{array}{ccccc|cccc}
X_{11} & X_{12} & X_{13} & \cdots & X_{1I} & Y_{11} & Y_{12} & \cdots & Y_{1O} \\
X_{21} & \cdots & & & & Y_{21} & \cdots & & \\
\vdots & & & & & \vdots & & & \\
\vdots & & & & & \vdots & & & \\
X_{N1} & & & \cdots & X_{NI} & Y_{N1} & & \cdots & Y_{NO}
\end{array}\right]$$

Inputs *Outputs*

4.3.1 RAW DATA

As mentioned previously, available data for dynamic load modeling comes from flight tests of a US Navy SH-60B helicopter in four separate flight maneuvers. The data takes the form of time-sequential group-readings at intervals of approximately 1/10 of a second, and each group, or data set, consists of the 33 simultaneous recordings of the flight variables and structural internal loads (28 inputs and 5 outputs listed in Table 4-1). The flight maneuvers included are level flight, a climbing turn, a rolling pullout, and a symmetric pullout, which contain 60, 209, 138, and 96 data sets, respectively, for a total of 503 data sets.

4.3.2 DATA SCALING

For purposes of training the network, it is crucial to scale the data such that the magnitudes of the numbers (inputs and outputs) are on the order of unity or less, thus preventing saturation of the learning algorithm, as discussed in Section 2.4.

In the following description of scaling methods, "global" refers to data from all four flight maneuvers discussed above, and "regional" refers to data from one of the four maneuvers. Global scaling of the data was achieved by considering data from all available flight maneuvers, and re-assigning a value of 0.15 to the algebraically smallest value of each of the 33 parameters (listed in Table 4-1). The largest value is re-assigned a value of 0.85. The rest of the data is scaled linearly between 0.15 and 0.85. Thus, the scaling equation takes the form,

$$X_{i\ scaled} = \frac{X_i - X_{i\,\min}}{X_{i\,\max} - X_{i\,\min}} \times (.7) + .15$$

and i represents one of the 33 parameters listed in Table 4-1.

Scaled data is also available in which each of the four flight maneuvers is considered separately for the purpose of finding the minimum and maximum values. The data for each flight maneuver is thus scaled in the same manner as described above, except that regional, rather than global maximum and minimum values are used.

4.3.3 REMOVING INPUT DATA

For various reasons, it may be desirable to eliminate certain input parameters listed in Table 4-1, as will be discussed in further detail in Section 4.4.6. This is accomplished by eliminating the desired column of data from the data file, and then shifting the remaining columns to fill the void. In this manner, the proper format of the data file is maintained and can be implemented in the computer algorithm.

4.3.4 ADDING TIME DELAYS

Because of the dynamic nature of the helicopter loads, time delays are included in various networks in an attempt to better capture the system dynamics. These attempts are described in detail in Section 4.4.5. Delays are implemented by augmenting the input vector to include past values of selected inputs, and shifting the output columns to the right to "make room" for the additional input columns.

As a consequence of including delays, the number of data sets available for testing decreases. This reduction is due to the fact that past information is not available for data groups measured at the beginning of the sequence. Of course, it is important to include the delays <u>before</u> randomization of the data.

4.3.5 DATA RANDOMIZATION

The data is randomized to prevent the back-propagation algorithm from arriving at local solutions to the minimization routine of adjusting the weights, as discussed in Section 2.4. After scaling, rows of data are selected at random from each maneuver, and then randomized again by "shuffling" the selected data sets among each other, as one would shuffle a deck of cards. The shuffled data then comprises the training set for the neural network model.

The following describes in detail how the above randomization procedure is accomplished. In an analogy to card shuffling, each data group, consisting of the 33 parameters listed in Table 4-1, is referred to as a "card." A collection of "cards", or data groups is referred to as a "deck". The "top card" of the deck refers to the first data group, whereas the "bottom card" refers to the last.

A deck is shuffled by first assigning each card an integer value, assigning the top card (first data group) a value of 1, and the bottom card a value of N, where N is the number of cards contained in the deck. An integer J is then randomly chosen between 1 and N, inclusive, by generating a pseudo-random number between 0 and 1, multiplying that number by N+1, and truncating the decimal. The J'th card then exchanges places with the N'th card by reassigning the values accordingly. The card that initially occupied the J'th position is now placed at the bottom of the deck and is not moved for the remainder of the shuffling routine. The procedure is then repeated for the remaining N-1 cards, until all cards have been selected and assigned a new position.

In order to select a portion of cards from a deck at random, the deck is first shuffled, and then the desired portion of cards removed from the top.

The pseudo-random number generator for selecting values between 0 and 1 uses the following equation,

$$R(k+1)=4\,R(k)\,(1-R(k))$$

where k is an integer. Starting with an initial value assigned for R(0) between 0.0 and 0.5, the remaining random numbers R(k) for k>0 are calculated according to the equation. For the purposes of testing various ANNs, the initial arbitrary value of R(0)=0.315 was consistently used so that the training and testing files were comprised of the same data sets for each ANN.

4.4 ESTIMATING DYNAMIC LOADS

Several different neural network models are constructed to estimate the main rotor loadings from the 28 known flight parameters listed in Table 4-1. Emphasis is placed on developing a model that identifies the underlying dynamics of the system, as opposed to merely "memorizing" the input/output data. As seen from the spring-mass system, neural networks are capable of both. By identifying the dynamics, it is expected that the model achieves better prediction accuracy, especially outside of the training range of data. For this reason, the testing and evaluation of the neural networks in this chapter are based only on the data that has <u>not</u> been used for training purposes. (See Figure 4-1b)

It is expected that a network will better estimate the loads of the fleet helicopters if the underlying dynamics are identified. In contrast, the network which has merely identified the input/output data is more likely to have learned the specific characteristics of the flight test helicopter, and is probably not as capable of estimating the loads of fleet helicopters.

4.4.1 EVALUATING VARIOUS NEURAL NETWORK MODELS

The following sections describe various neural network models which were developed for the purpose of capturing the system dynamics. The neural networks are evaluated on the basis of the error parameter obtained from the testing data. The error is determined according to the following equation:

$$E=\frac{\sum_{i=1}^{5}\sum_{j=1}^{N}\left|Y_{ij_{\substack{network,\\ scaled}}}-Y_{ij_{\substack{measured,\\ scaled}}}\right|}{5\times N}$$

where Y_{ij} represents the value of the ith output for the jth data set, and N represents the number of data sets which comprise the testing set. Thus, the error parameter, E, is the average of the error magnitudes of all the outputs for the scaled testing data.

Because the error is based on the scaled data, it does not readily provide an accurate indication of the performance of the neural network model. Rather, the error is used as a basis for comparing the performance of the various neural networks using the same scaled data.

Even when using the average error solely as a means of comparison, conclusions must be made cautiously. Problems with comparing trained models arise from the fact that neural network training is not a precise science, since frequent human interaction is often required, as discussed at the end of Section 2.2.3.

In addition to evaluating the performance on the basis of the error, consideration is also given to the practicality of implementing the neural network structure as a data processor in the fleet. Considerations include data-processing time, data storage requirements, and data availability.

4.4.2 NOMINAL NETWORK MODEL CONFIGURATION

For the purpose of comparing the different neural network models, a nominal configuration is established below. The nominal configuration was arrived at after experimenting with variations of the network training and finding a model that appeared to work well. The resulting network, shown in Figure 4-2, possesses 29 input nodes--one for each of the 28 input parameters listed in Table 4-1, plus a bias node. The single hidden layer consists of 17 nodes, including 1 bias node. The squashing function used is hyperbolic tangent. The output layer contains 5 nodes--one for each of the 5 output parameters listed in Table 4-1b.

The training and testing sets are compiled from the 443 regionally scaled data sets from the climbing turn, rolling pullout, and symmetric pullout maneuvers. The training set is created by randomly selecting 30 data sets from each of maneuver except level flight, for a total of 90 data sets. These 90 data sets are then randomized before feeding to the neural network for training. (See Figure 4-1b)

After training the network with the 90 data sets, the weights were frozen, and the network was then tested using the remaining 353 data sets, from which a normalized error of .0432 was obtained. Again, this number does not provide much information in the present form, but serves as a basis for comparing other networks.

Except where otherwise noted, the following networks use the same geometry as the nominal case, and use the same data sets for training and testing.

4.4.3 VARYING THE STRUCTURE OF THE NEURAL NETWORK MODEL

It was speculated that training could be improved by reducing the network to single-output design. The back-propagation algorithm would then have fewer parameters to vary, and could seek to adjust the weights for one particular output, without risk of interference from weight adjustments prompted by other outputs. Instead of attempting to model all five outputs with one network, five separate networks were created. The resulting test showed no improvement however-- the error did not change significantly. The training time for a single neural network dropped, but when considering that there were now five separate networks, the overall training time increased. (Especially if one accounts for the extra time required to manipulate the data into 5 separate files.)

Other changes to the neural network structure included varying the number of hidden nodes. Increasing the number of nodes increases the adaptability of the network to learn complex systems. Yet, increasing the number too much complicates the learning procedure, and unnecessarily increases the training time. The number of hidden nodes was varied between 8 and 24. Increasing the number of nodes beyond 16 did not significantly improve the test results.

4.4.4 SCREENING THE DATA FOR NOISE

Noise in the training data provides no pertinent information about the dynamic system, and can only interfere with the learning process. In order to reduce the effects of noise present in the training phase, the data was screened by discarding the data sets with outputs below a threshold of 0.3 for the scaled data. Relatively speaking, noise (from vibration, for example) is likely to be more pronounced in the smaller values of the output. Furthermore, the outputs of greatest interest are the relatively large loads and stresses. It is these large loads which are critical to the determination of the fatigue undergone by the hardware components. By concentrating on the larger loads for the purpose of training, it was hoped that the network model could better estimate the critical data in the testing regime. Despite this attempt, the results showed no improvement.

4.4.5 INCLUDING INPUT DELAYS

The available information of the 28 inputs in Table 4-1 includes aircraft position, velocities, accelerations, and controls. Defining the state variables as those which describe the position and orientation of the helicopter, the structure of the dynamic system can be represented as a function of state and control variables:

$$Dynamic\ Load = f(x,\dot{x},\ddot{x},u,\ldots)$$

$$where$$

$$x \equiv \begin{bmatrix} position \\ orientation \end{bmatrix}$$

$$\dot{x} \equiv \begin{bmatrix} translational\ velocities \\ angular\ velocities \end{bmatrix}$$

$$\ddot{x} \equiv \begin{bmatrix} translational\ accelerations \\ angular\ accelerations \end{bmatrix}$$

$$u \equiv \begin{bmatrix} stick/pedal\ position \\ servo\ position \\ external\ controls \end{bmatrix}$$

From the experience of the spring-mass oscillator explored in Chapter 3, it was found that the second order terms provide the necessary information for identifying the dynamics of a second order system. If the second order terms were not included, they could be estimated through the use of delays.

Although certainly more complex than the spring mass system, the physics governing the behavior of the helicopter dynamics involve, at its most basic level, Newton's second law relating forces and moments to translational and angular accelerations (i.e. second-order terms). Because second order terms for each degree of freedom are available among the 28 inputs, delays of the velocities and positions are deemed unnecessary. Rather, delays of the controls are implemented in an attempt to improve performance. The input vectors of the nominal case are augmented to include past values of the controls in various combinations. The different input structures attempted and their results are shown in Table 4-2. The first column lists the control inputs which were delayed. The second column indicates how many past samples were included for each of the delayed inputs. The third column lists the testing results of each attempt.

Table 4-2:
The Effect of Including Delays in the Dynamic Loads Model

INPUTS DELAYED	NUMBER OF DELAYS FOR EACH INPUT	AVERAGE ERROR
NONE (Nominal Case)	X	.0432
#8-Collective stick position #9-Lateral stick position #10-Longitudinal stick position #11-Pedal position	1	.0590
	2	.0600
	4	.0767
#8,#9,#10,#11, and #12-Aft primary servo output position #13-Fwd primary servo output position #20-Tail rotor impressed pitch #21-Stabilator angle position	1	.0573
	2	.0573
	4	.0594

As seen in Table 4-2, including the input delays did not have the desired effect, for the error increased. The results of Table 4-2 indicate that significant delays do not exist between the control inputs and the output parameters. For the case of the helicopter dynamics, it appears that adding control delays only serves to hinder the learning process by adding information unnecessarily. (A similar situation occurred with the spring-mass system, Figure 3-10.)

Furthermore, the inclusion of the delays increases the computational requirements of the learning algorithm, and training time almost doubles in some cases. For purposes of implementing an on-board data processor for in-flight use, including delays will significantly complicate the problem of calculating the outputs in real-time. A significant increase in accuracy would most likely be required to justify adding control delays.

4.4.6 ELIMINATING INPUTS THAT MAY HINDER THE LEARNING PROCESS

As mentioned in Chapter 3, one might expect that the algorithm would adjust the weights such that the multipliers of the irrelevant inputs would be set to zero. Yet it was also found from the spring-mass system in Chapter 3 that having more information than necessary may make it more difficult for the ANN to learn the system and reduce the testing performance.

To find a combination of inputs that appears to yield the best performance, the 28 inputs are categorized in accordance with their expected importance (level of contribution) to the modeling of the system. The most important inputs are expected to be those which are most directly related to forces and moments acting on the helicopter. These inputs include the translational and angular accelerations, which are related to the forces and moments through Newton's second law. The controls are rated next in importance because they bring about direct changes in the forces and moments acting on the aircraft. Relative wind direction is rated next, for it is related to the forces through aerodynamic laws. Angular and translational rates may be related through viscous terms, and through cross terms associated with the aircraft kinematics. Orientation is related to the forces only by describing the relative direction of the gravity vector, and altitude is related only through atmospheric conditions such as density. Finally, weight of the aircraft and weight of the fuel used are relatively constant parameters and provide little information about the dynamics of the system that is not already provided by the accelerations.

Starting with the perceived most important parameters, the subsequent parameters are added and their contribution to the ANN performance noted. Table 4-3 summarizes the results of varying the combinations of inputs fed to the neural network. The left most column lists the inputs in order of relative importance. The remaining columns each represent a separate network, defined by the input parameters which are used. An 'x' indicates that the network uses the inputs associated with that row. The bottom row reveals the training error achieved for each network.

Table 4-3 :
Results of Varying Flight Variable Inputs

INPUT PARAMETERS	A	B	C	D	E	F	G	H	I	NOM.
Load Factor/Accelerations	x	x	x	x	x	x		x	x	x
Control Surfaces, Rotor Speed		x	x	x		x	x	x	x	x
Servos			x	x		x	x	x	x	x
Stick/Pedal Position				x	x	x	x	x	x	x
Relative Wind						x	x	x	x	x
Angle Rates								x	x	x
Translational Rates									x	x
Orientation									x	x
Altitude										x
Weight / Fuel Used										x
AVERAGE NORMALIZED ERROR	.0631	.0529	.0462	.0410	.0478	.0389	.0463	.0398	.0425	.0432

Table 4-3 suggests that the most relevant inputs are the acceleration terms, the control inputs and the relative wind direction, for these inputs used together produce the smallest error. Admittedly, the differences in these errors are small, and this conclusion is made tentatively. Cases B through E show that steady improvement takes place as more control information is added. Comparison of case H and I tends to support the claim that less relevant inputs may hinder the learning process. Again these conclusions are made cautiously, especially when considering the "human factor" involved when training neural networks, as user interaction into the learning process is often required.

Despite the uncertainty in comparing the networks on the basis of their relative errors, certain conclusions can be made more confidently. Case A shows that the acceleration terms alone can be used to determine the structural loads within a reasonable degree of accuracy. Case G demonstrates that the acceleration terms may be excluded, and a reasonable dynamic model still maintained with information from controls and relative wind. It is clear that the acceleration and control inputs provide useful information for identifying the dynamics of the system. Furthermore, there appears to be an overlap of information provided by cases A and G, despite the fact that the two cases contain no common inputs.

On the basis of the results summarized in Table 4-3, it may be desirable to exclude certain inputs from the network model, even if a slight degree of accuracy is lost. Depending on the expense or technical difficulty of obtaining each of the 28 inputs, it may be justifiable to sacrifice the accuracy in favor of cutting costs. Eliminating inputs also facilitates the recording of information by reducing the data storage requirments. For the purpose of implementing an on-board data processor, the elimination of inputs reduces the computational requirements, and would simplify the problem of calculating the outputs in real-time.

4.4.7 RESULTS

For demonstration of the performance of neural network training in estimating the helicopter dynamic loads, the input layer is structured according to case F in Table 4-3 above, because this case yielded the smallest average error for all the attempted variations from the nominal training case. Thus, the input layer is comprised of loads/accelerations, controls, and relative wind direction, corresponding to 18 inputs of the 28 inputs listed in Table 4-1, numbered 6, 8-14, and 18-27. The training file was constructed in the same manner as the nominal case, except that 30 level flight data were also included, as shown in Figure 4-1a. The testing file included all available data arranged sequentially for each flight maneuver, so that the loads estimated by the ANN could be graphed as a function of time after de-scaling the output data. The ANN load-time history for each of the 5 outputs listed in Table 4-1 are graphed in Figures 4-3a through 4-3e, and compared with the load history obtained from direct measurements using sensors on the flight test helicopter. The neural network model is evaluated for each output according to the average relative error for all 503 data sets. Thus, the following equation is used:

$$E_{\substack{average \\ relative}} = \frac{\sum_{i=1}^{503} \frac{|Y_i - O_i|}{O_i}}{503} \times 100\%$$

The average relative errors for the load history for each output are, 3.99%, 7.33%, 6.5%, 8.1%, and 15.0%, respectively.

4.5 ESTIMATING FATIGUE

After experimenting with the neural network models that would achieve the smallest error, the performance of the models are further evaluated according to the potential for the neural network to estimate the fatigue of the hardware components.

The idea behind such performance evaluation is to focus attention on the critical loads applied to the helicopter components, for it is these loads which contribute to fatigue and shorter life spans. The error evaluations provide a good indication of how well the ANN has modeled the system dynamics; however, the "critical load count" is a better gauge to judge the success of the ANN for estimating fatigue of the helicopter components. The "critical load count" is defined as the number of output responses above a certain threshold. If the output is above that threshold, it is defined as a "critical load." The thresholds used in the following analysis are not chosen on the basis of fatigue analysis of the hardware components. Rather, they are selected arbitrarily as a means for determining the potential for the neural network models to predict the outputs in such a binary manner, as required for fatigue analysis.

After training the network to model the system dynamics, the testing input data is fed into the neural network model and the output responses recorded, noting whether or not the output is a "critical load." The success of the network is determined by how many of the outputs have been correctly determined to be "critical" or "non-critical." For each output, there are four possibilities:

Table 4-4 : Identifying Critical Loads

	Definition	True Output	Network Output
1	"Hit"	Critical	Critical
2	"Miss"	Critical	Non-Critical
3	"Fake"	Non-Critical	Critical
4	"Fair"	Non-Critical	Non-Critical

In evaluating the performance of neural network models in terms of the "critical load count", four figures of merit are established.

1. The "hit fraction", defined as the number of outputs which were correctly estimated as critical, divided by the true number of critical outputs.

$$hit\ fraction = \frac{hits}{hits + misses}$$

2. The "pseudo-hit fraction", defined as the number of outputs (correctly or incorrectly) estimated as critical, divided by the true number of critical outputs. This "pseudo-fraction" takes into account that errors resulting from "misses" may tend to be cancelled out by errors resulting from "fakes".

$$\textit{pseudo-hit fraction} = \frac{\textit{hits} + \textit{fakes}}{\textit{hits} + \textit{misses}}$$

3. The "total fraction", defined as the number of outputs which were estimated correctly, divided by the total number of outputs studied.

$$\textit{total fraction} = \frac{\textit{hits} + \textit{fairs}}{\textit{total number of outputs studied}}$$

4. The "pseudo-total fraction", defined as follows:

$$1 - \frac{|\textit{misses} - \textit{fakes}|}{\textit{total number of outputs studied}}$$

Again, this figure of merit takes into account that "miss" errors and "fake" errors may tend to cancel each other.

The "hit fraction", the "total fraction", and the "pseudo total fraction" will always be less than or equal to unity, of course. If the "pseudo hit fraction" is greater than unity, then the network is conservative in its critical load count errors. (i.e. more critical loads have been predicted than actually exist) Conversely, a "pseudo hit fraction" of less than unity demonstrates that the network has predicted fewer critical loads than actually exist.

Table 4-5 shows a typical "critical load count" performance evaluation of a neural network model. Table 4-6a summarizes the "critical load count" performance evaluations for several of the models already discussed in the evaluation of dynamic loads prediction.

Table 4-5 :
Typical Critical Load Count Performance Evaluation

Critical Load Count (Threshold = 0.6, scaled data)									
Output	# hits	# misses	# fairs	# fakes	# total	hit fraction	total fraction	pseudo hit fraction	pseudo total fraction
1	34	6	313	0	347	0.850	0.983	0.850	0.983
2	19	19	314	1	333	0.500	0.943	0.514	0.949
3	22	6	322	3	344	0.786	0.975	0.880	0.992
4	13	4	334	2	347	0.765	0.983	0.867	0.994
5	21	10	303	19	324	0.677	0.918	1.750	0.975
ALL	109	45	1586	25	1695	0.708	0.960	0.845	0.989

Table 4-6 :
Critical Load Count Results for Various ANNs

INPUT PARAMETERS	D	F	H	NOM.
Load Factor/Accelerations	X	X	X	X
Control surfaces	X	X	X	X
Servos	X	X	X	X
Stick/Pedal Position	X	X	X	X
Relative Wind		X	X	X
Angle Rates			X	X
Translational Rates				X
Orientation				X
Altitude				X
Weight / Fuel Used				X
AVERAGE NORMALIZED ERROR	.0410	.0389	.0398	.0432

Table 4-6a :
Critical Load Count Results

CRITICAL LOAD COUNT Threshold = 0.6, scaled	D	F	H	NOM
Hit Fraction	0.812	0.777	0.825	0.708
Total Fraction	0.968	0.952	0.979	0.960
Pseudo Hit Fraction	0.987	0.929	0.890	0.845
Pseudo Total Fraction	0.999	0.991	0.990	0.989

Table 4-6b :
Weighted Critical Load Count Results

WEIGHTED CRITICAL LOAD COUNT, Threshold = 0.6, Scaled "Boost" = 0.15	D	F	H	NOM
Hit Fraction	0.909	0.877	0.857	0.883
Total Fraction	0.969	0.971	0.965	0.960
Pseudo Hit Fraction	1.175	1.084	1.117	1.221
Pseudo Total Fraction	0.985	0.993	0.990	0.981

Looking at the critical load count evaluations in Table 4-6a, it is apparent that the network has trained more effectively with respect to the lower values of the outputs, since the total fraction is consistently higher than the hit fraction.

In an attempt to improve the ability of the neural network to correctly estimate critical loads, the following technique is implemented. The data is pre-processed such that all outputs above the critical threshold are increased by a constant value. The data is then fed to the neural network for training. Having boosted the critical data, the back propagation algorithm then attempts to train the neural network model to match the values of the boosted outputs. As a consequence of trying to match the increased values of the critical loads, the network will predict a higher value of the output, increasing its chance of being placed above the threshold.

Table 4-6b summarizes the results of the critical load predictions after employing this technique. The results of weighting the critical outputs in such a manner are somewhat successful, increasing the percentage of "hits" in each case, but decreasing the total percentage in some cases. Although the number of hits increases in each case, so does the number of fakes. The false hit percentage increases above unity in each case as a result of this. Apparently, the method of weighting the critical loads merely adds conservatism to the study, without necessarily increasing the accuracy of the predictions.

4.6 FUTURE WORK / FOLLOW-UP PROCEDURES

To improve the accuracy of predictions with respect to the critical loads, emphasis should be placed on improving the identification of the system dynamics. The uneven testing performance for critical and noncritical loads indicates that the network may not have learned the full range of the system dynamics. That is, there may be dynamics associated with aircraft during critical loads which are otherwise negligible or nonexistent, especially when considering the fact that the system is highly nonlinear. To train the network to learn such dynamics, it is recommended that more data sets representing critical occurrences be included in the training set for the network. Because the available data contains a low percentage of "heavy" loads to begin with, the training files for this study also contain a small percentage, because the training data sets are selected at random from the available data. By actively selecting more sets containing heavy loads for the purpose of training, it is anticipated that the network will better learn the dynamic characteristics associated with the critical outputs.

To improve the accuracy of the dynamic loads modeling in general, it is suggested that the input vector be augmented to include the squares of certain inputs. Kinetic and kinematic governing relationships often include squares of angular and translational rates. (For example, the tension in a string is proportional to the velocity squared, for the case of swinging a string-tied object in circles) By explicitly including squared values of selected inputs, the neural network may be more capable of modeling the dynamics associated with such parameters.

The ANN training may be further aided by scaling the data differently. This study uses data scaled between 0.15 and 0.85 based on the algebraically smallest and largest values available. It is suggested to re-scale the data based on a logical reference value, and also to account for symmetry. For example, the lateral stick position, (input #9, Table 4-1) is measured in terms of a percentage from full left (0%) to full right (100%). Presently, full left and full right positions would assume values of 0.15 and 0.85, respectively. A logical reference value for this input is dead center (50%). By assigning full left, dead center, and full right positions new values of -0.85, 0.0, 0.85, respectively, the symmetry of the aircraft and its governing equations can be incorporated into the input data for the network. (This also assumes that the squashing function is symmetric about zero, such as the hyperbolic tangent.) The reference value of zero also aids the ANN in determining the magnitude of the input. For the case of dead center, the ANN is assured of receiving a null input from the control stick. A deflection of the control stick assures that the ANN will receive a larger magnitude input. In the previous scaling, a zero deflection corresponds to a non-zero input signal of 0.5. Even though a bias node may compensate for this, no assurances exist that the ANN will adjust the weights properly such that all non-zero input signals representing zero deflections will not adversely affect the calculated output values. By combining the squaring of inputs mentioned above with re-scaling of the data, the sense (+ or -) of the deflection is ignored, and symmetry is assured. Similarly, the logical reference value for the vertical load factor (input #22, Table 4-1) is 1.0 G, the value maintained when the aircraft is not accelerating. By scaling inputs in this manner, the ANN structure may be aided by our physical knowledge of the system. Though our knowledge about the system is limited, basic fundamentals such as symmetry and squaring of inputs may prove helpful in modeling of the dynamic characteristics of the system.

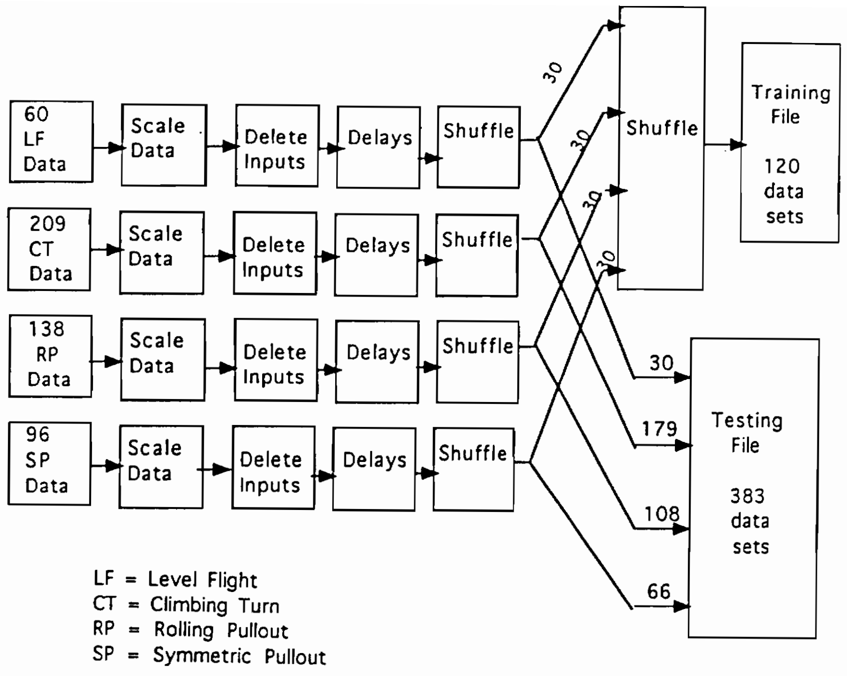


Figure 4-1a: Data Manipulation Schematic

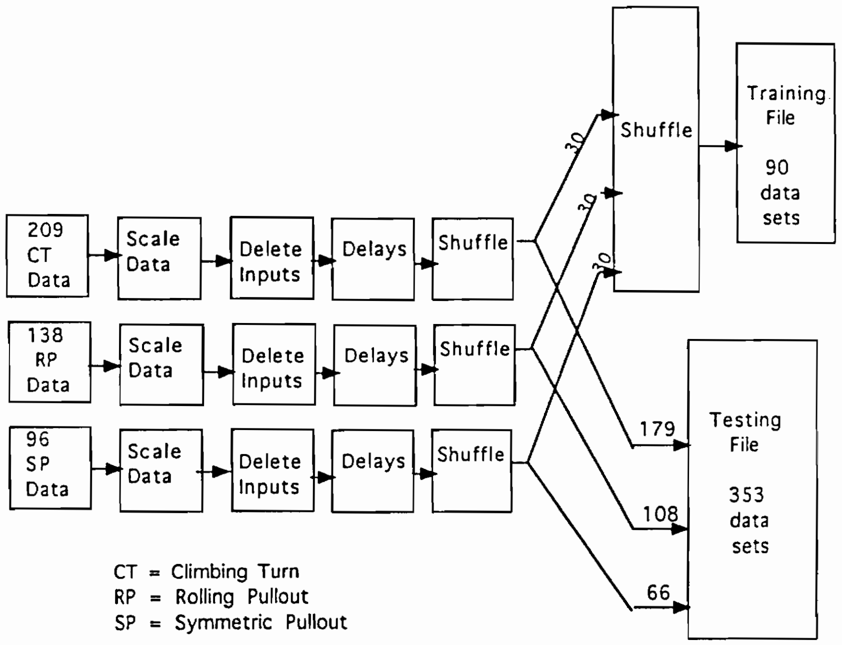


Figure 4-1b:
Data Manipulation Schematic for Nominal Neural Network

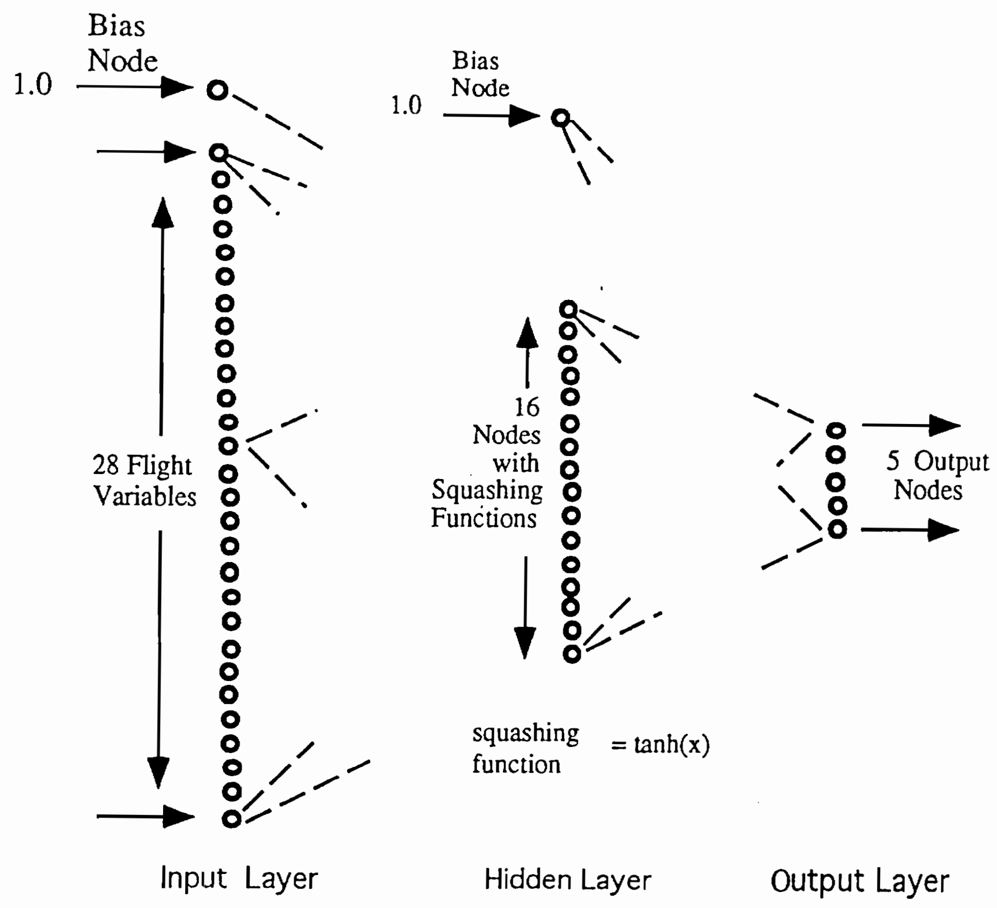
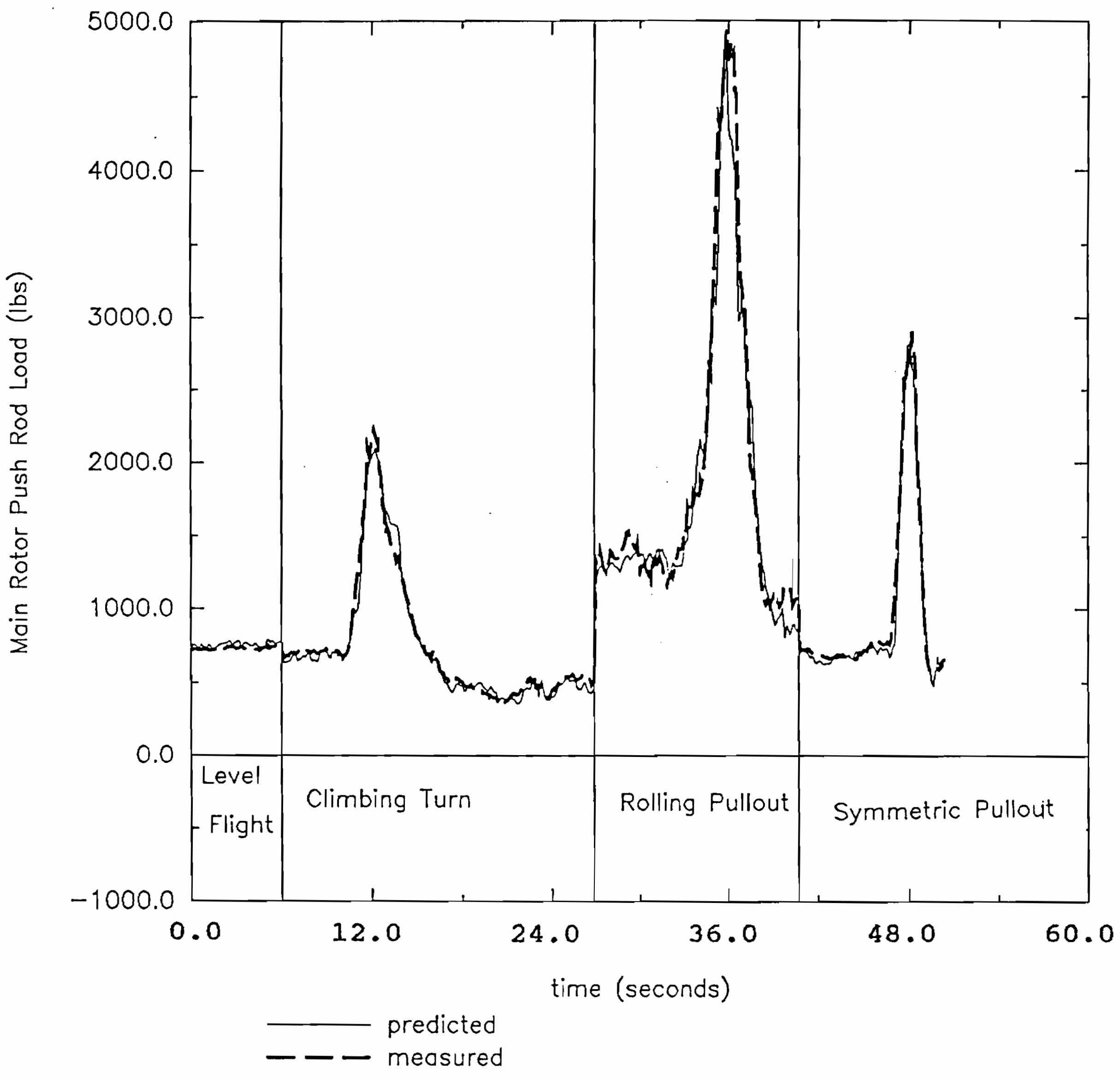
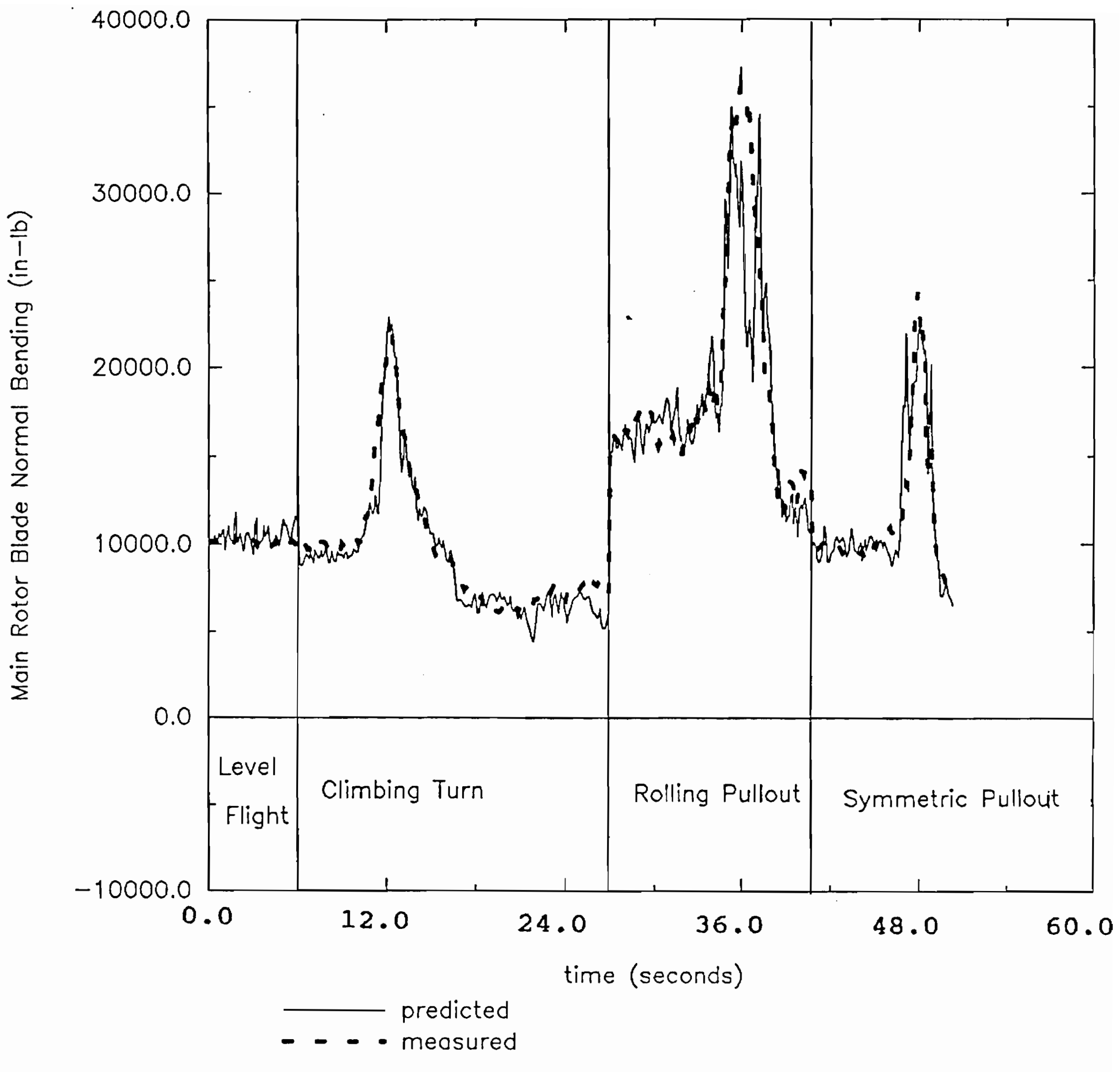


Figure 4-2:
Nominal Neural Network Configuration
for Modeling Helicopter Dynamic Loads



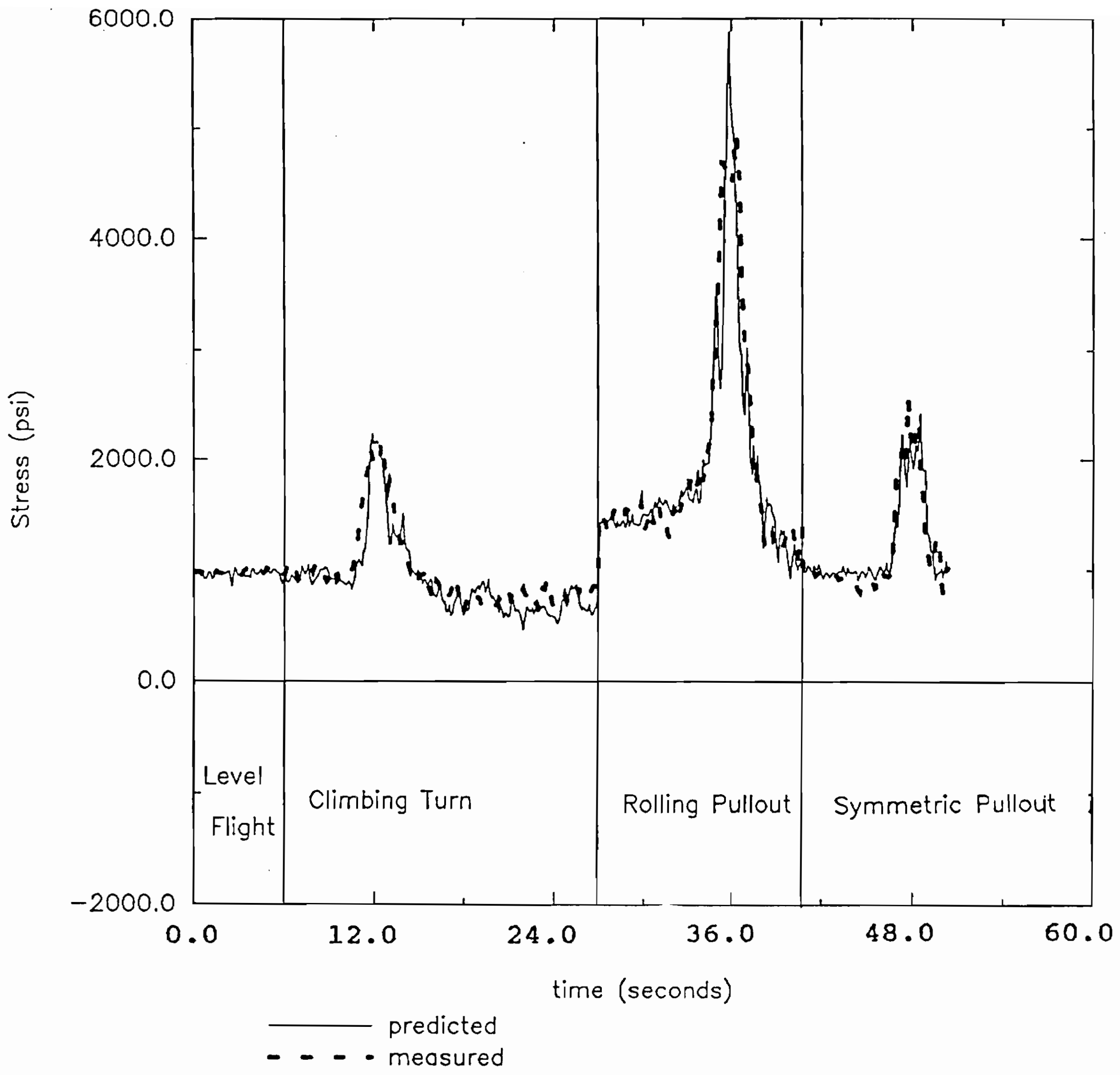
Average Error = 3.99%

Figure 4-3a: Main Rotor Push Rod Load



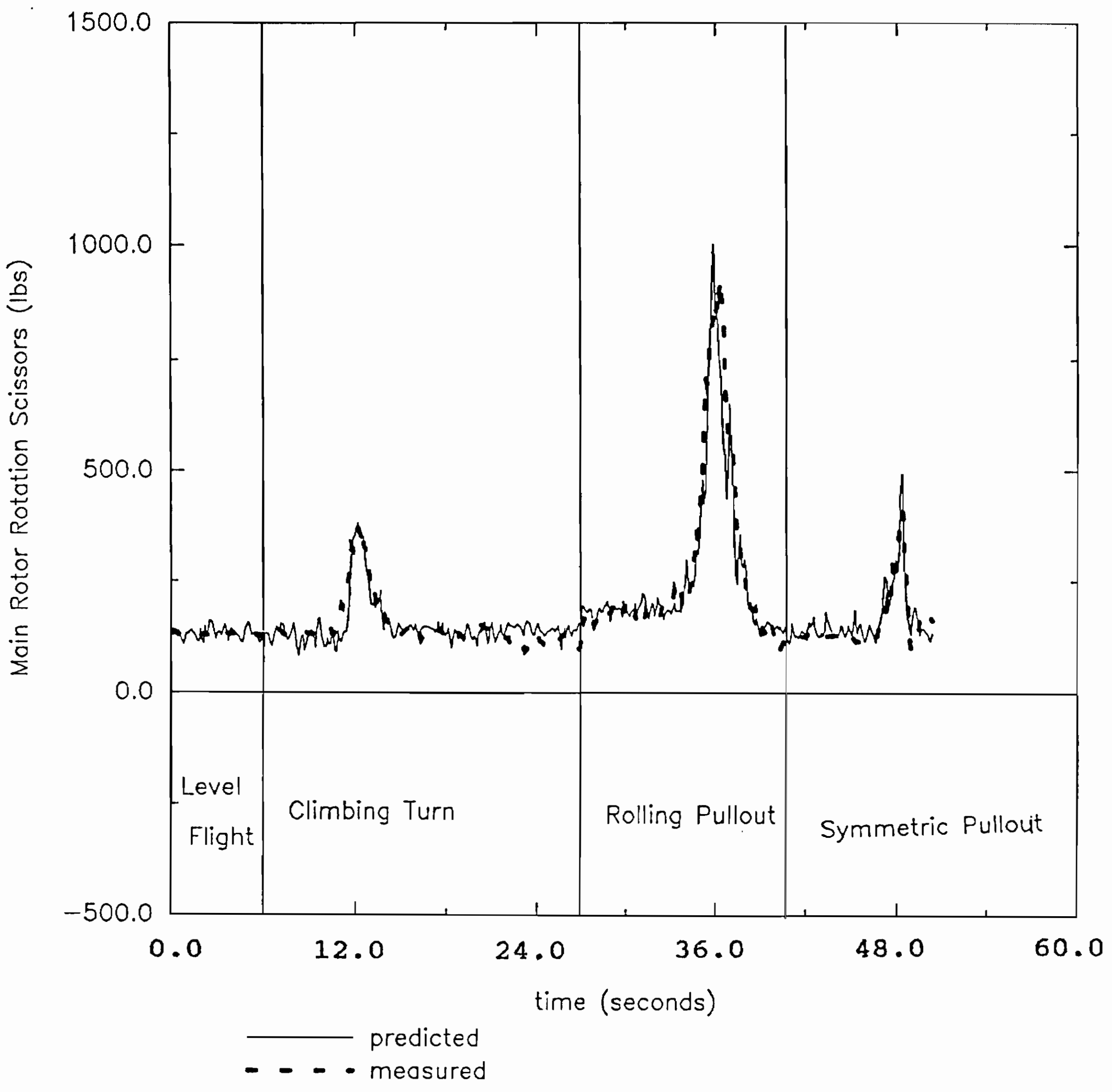
Average Error = 7.33%

Figure 4-3b: Main Rotor Blade Normal Bending



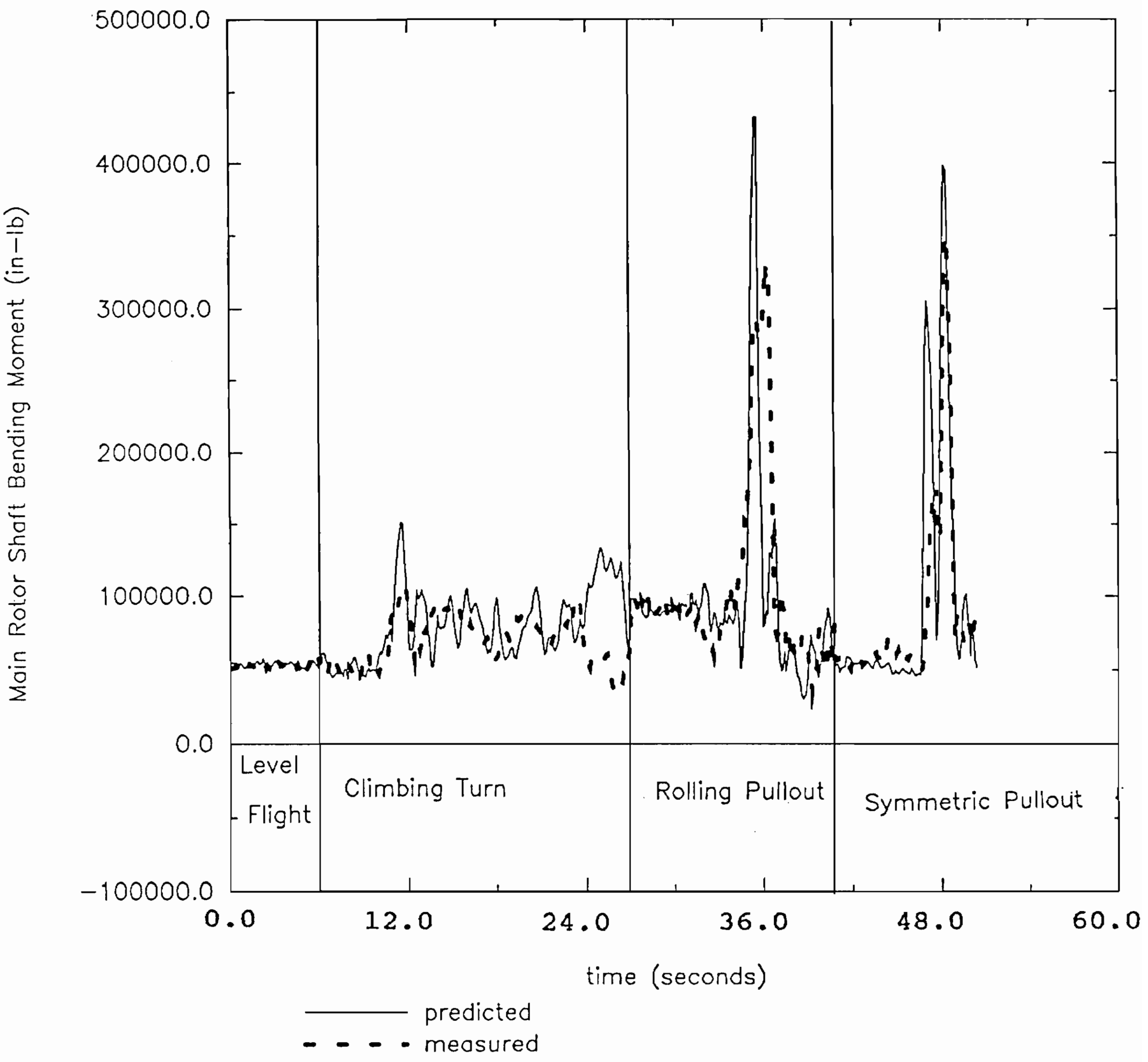
Average Error = 6.5 %

Figure 4-3c:
Main Rotor Spar Stress Trailing Edge Top at 40% Radius



Average Error = 8.1 %

Figure 4-3d: Main Rotor Rotation Scissors Load



Average Error =15.0 %

Figure 4-3e: Main Rotor Shaft Bending Moment

5 CONCLUSIONS

Artificial neural networks allow for modeling of complex, nonlinear systems with little or no physical knowledge about such systems.

For second-order dynamic systems identification using ANNs, best performance was obtained using control information and higher order states, namely, acceleration.

Based on study of the SH-60B flight test data, delayed control information does not help in determining the dynamic loads of the helicopter main rotor.

For estimating dynamic loads, a limited amount of information could be used to achieve reasonable results. Namely, acceleration data alone or control information combined with relative wind data achieved an acceptable error level. Combination of this information improved the results slightly, but additional information on lower order states and aircraft weight hindered the model performance.

ANNs demonstrate ability to predict fatigue of the main rotor components. They further demonstrate flexibility for practical implementation as a "fatigue meter" by maintaining a reasonable degree of accuracy while allowing for:

1. elimination of "costly" input information
2. reduction of data storage requirements
3. reduction of computational requirements for possible use as an in-flight, real-time fatigue predictor.

Attempts to improve fatigue estimation by weighting the critical loads did not significantly improve the prediction accuracy, but added conservatism to the predictions.

6 REFERENCES

1. Niu, M. C. Y.; Airframe Structural Design. Conmilit Press LTD, Los Angeles, CA 1990

2. "Applications in Neural Computing", NeuralWare, Inc. 1991

3. Antegnetti, P., and Milutinovíc, V.; Neural Network Concepts, Applications, and Implementation, Prentice Hall Inc., 1991

4. "Proceedings of the Second Workshop on Neural Networks" WNN-AIND 91. Auburn University Hotel and Conference Center, Auburn, Alabama. Omnipress, Madison, WI 1991

5. Lippmann, R. P.; "An Introduction to Computing with Neural Nets," IEEE ASSP Magazine, April 1987

6. Rumelhart, D., Hilton G., and Williams, R.; Learning Internal Representations by Error Propagation, MIT Press, Cambridge, MA 1986

7. Cook, A. B.; "Application of Neural Networks to Indirect Monitoring of Helicopter Loads from Flight Variables," M.S. Thesis. Virginia Polytechnic Institute and State University, Blacksburg, VA October 1991

8. Derrick, W. R., and Grossman, S. I., Introduction to Differential Equations with Boundary Value Problems, 3rd Edition, West Publishing Company, New York, NY 1987

9. "Helicopter Fatigue Life Assessment AGARD Conference Proceedings" No. 297. Papers presented at the 51st meeting of AGARD structures and materials panel held in Aix-en-Provence, France on 14-19 September 1980. Technical Editing and Reproduction Ltd, Copyright AGARD 1981

10. "Validation of the Rotorcraft Flight Simulation Program (c81) for Articulated Rotor Helicopters through Correlation with Flight Data," USAA MRDL-TR-76-4. United Technologies Corporation, Sikorsky Aircraft Division. Stratford Ct. May 1976

11 Jane's All the World's Aircraft 1991-1992, Butler and Tanner Ltd, London, UK, Copyright Jane's Information Group 1991.

12 Tran, Michael; "BKP.EXE", Back-propagation algorithm. Virginia Polytechnic Institute and State University, Blacksburg, VA 1992.

13 Gessow; Aerodynamics of the Helicopter. Frederick Ungar Publishing Company, New York, NY 1967

14 Khanna, T.; Foundations of Neural Networks, Addison-Wesley Publishing Co, New York, NY 1990

15 Hannan, E. J., and Deistler, M.; The Statistical Theory of Linear Systems, John Wiley and Sons, New York, NY 1988

16 Eykhoff; Trends and Progress in System Identification, Pergamon Press, New York, NY 1981

17 VanLandingham, H.F., Bingulac S., and Tran, M.; "A Comparison of Conventional and Neural Network Approaches to System Identification," The Bradley Department of Electrical Engineering, Virginia Polytechnic Institute and State University, Blacksburg, VA 1992

18 VanLandingham, H.F.; "Multivariable System Identification," The Bradley Department of Electrical Engineering, Virginia Polytechnic Institute and State University, Blacksburg, VA 1992

7 VITA

Michael George Nosek was born May 1st, 1970. Mike attended Beloit Catholic High School in Beloit, Wisconsin, graduating as valedictorian of his class in 1988. In May of 1992, Mike received a Bachelor of Science degree in aerospace engineering from University of Notre Dame, and a commission in the U.S. Navy. He then attended graduate school at Virginia Tech to pursue a Masters of Science degree in aerospace engineering. Upon graduating from Virginia Tech, Mike plans to pursue a career in Naval Aviation.

Mike Nosek

Mike Nosek