Neural network identification and control of a parametrically excited structural dynamic model of an F-15 tail section

Ayman A. El-Badawy, Ali H. Nayfeh* and Hugh VanLandingham
Virginia Polytechnic Institute and State University, Blacksburg, VA 24061, USA

Received 24 August 1999
Revised 24 July 2000

We investigated the design of a neural-network-based adaptive control system for a smart structural dynamic model of the twin tails of an F-15 tail section. A neural network controller was developed and tested in computer simulation for active vibration suppression of the model subjected to parametric excitation. First, an emulator neural network was trained to represent the structure to be controlled and thus used in predicting the future responses of the model. Second, a neurocontroller to determine the necessary control action on the structure was developed. The control was implemented through the application of a smart material actuator. A strain gauge sensor was assumed to be on each tail. Results from computer-simulation studies have shown great promise for control of the vibration of the twin tails under parametric excitation using artificial neural networks.

1. Introduction

In aeroelastic investigations of aircraft, the structural behavior of aircraft components is assumed to be linear [1]. However, in reality nonlinearities are present in one form or another. It is always important to develop and verify several techniques to predict flight-vehicle dynamic and aeroelastic behavior to prevent vibration damage and aeroelastic instabilities. One of the available techniques that can be used to solve critical structural problems on fixed-wing aerospace vehicles is neural networks. During the past several years, neural networks have emerged as one of the most active areas in system identification and control of nonlinear systems. Also, neural networks are becoming more and more popular for a variety of applications including electronics, dynamic modeling of chemical process systems, speech recognition, telecommunications, and transportation. Lisboa [9] summarizes some of the widespread use of neural networks. The reason for the wide variety of applications of neural networks in almost all fields of study is their usefulness in avoiding the complexity involved in directly modeling the system dynamics.

Another interesting application for neural networks is active vibration control of smart structures. The design of controllers to suppress vibrations in smart structures is a challenging problem due to the presence of nonlinearities in the structural system and the actuators as well as the limited availability of control forces. One of the main objectives is to deal with imprecise mathematical models due to unmodeled dynamics and hence remove the requirement of having an exact detailed mathematical model for the system which can be a very time consuming process. This is where the power of neural networks is stressed; it can identify the system using the true input/output data without any prior model information.

There are several types of neural networks that have evolved over the last decade and half of them have proven to be efficient tools in identifying nonlinear systems. Some of these are Volterra series models, group method of data handling models (GMDH), self organizing neural networks, and radial basis functions (RBN) [5,6,13,14]. Also, it has been shown that multilayer perceptrons (MLPs) are universal function approximators [8]. Later this model type has been used to train neurocontrollers to suppress the vibrations of nonlinear smart structures. The linear and nonlinear mapping properties of neural networks have been extensively utilized in the design of multi-layered feedfor-

*Corresponding author: Ali H. Nayfeh, ESM Dept., MC 0219, Virginia Tech, Blacksburg, VA, 24061. Tel.: +1 540 231 5453; Fax: +1 540 231 2290; E-mail: anayfeh@vt.edu.

Shock and Vibration 7 (2000) 355–361
ISSN 1070-9622 / $8.00

ward neural networks for the implementation of control algorithms [10,11].

The vibration controller used in this paper is similar to an indirect linear model following control (LMFC) system [2,15]. It is also similar to the backpropagation-through-time neural controller (BTTNC), which is presented by Chen et al. [3]. The BTTNC was used in the active control of structures under strong dynamic loadings. To train the neurocontroller, one needs the difference between the network output and the ideal input to the plant. Since nothing is known about the ideal input, the necessary data must be provided in a different way. It is possible to model the plant by a neural net. After finding an appropriate model, one can train the controller network by backpropagating the error through the plant model. Thus, the present study consists of two steps. First, a model of the plant was trained. After achieving a well performing plant model, we fixed its weights. Second, the plant was replaced by its model. The cascade of both networks was then used to train the neurocontroller. Only the controller network weights were updated. Here, the overall system is tested and the performance of the network is discussed.

2. Backpropagation neural network

A prerequisite of the system identification task is to acquire as much information about the plant behavior as possible using input signals that excite the relevant dynamic processes of the structure under consideration. In addition, the signal amplitude has to cover the stationary operating range of interest. In this paper we use the backpropagation algorithm to train both the emulator and the controller multilayer perceptron (MLP)networks. An input to the network generates some output; then, the error between the desired output and the network output is used in the backpropagation learning algorithm. Each neuron in the network receives its own error factor and adjusts its weights accordingly. This training process continues until a desirable level of performance is reached. This can be done by specifying the sum-squared error between the network output and the desired target output. That is, the network seeks to minimize the following cost function:

$$E = \sum_{i=1}^{n} (d_i - y_i)^2 \tag{1}$$

where d_i is the target output, y_i is the neural network output, and n is the number of input-output training pairs. These forward and backward sweeps are continued until a desirable level of performance is reached. After the network has been sufficiently trained, only the feedforward paths are used. Illustrative examples of the backpropagation algorithm can be found in [7].

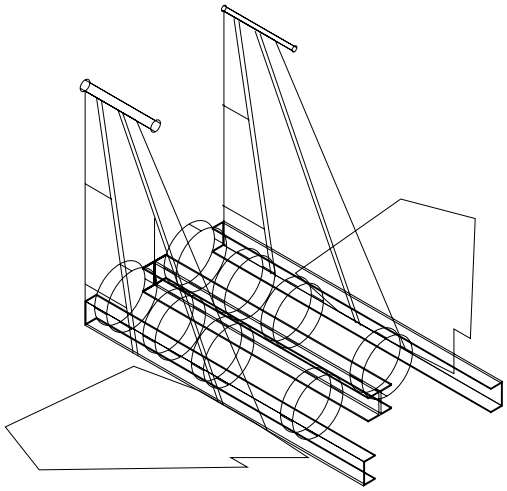

Fig. 1. Three-dimensional view of the twin-tail assembly.

3. Mathematical model of the twin tails

The problem at hand is to suppress the vibrations of a structural dynamic model (1/16 dynamically scaled) of the twin tails of the F-15 fighter plane. The tail section construction includes a series of aluminum channels, brass rings, composite plates, metal masses, and various adhesives. Figure 1 presents a three-dimensional view of the tail section used to acquire the training and validation data sets and to verify the on-line predictive control scheme. The model is approximately 0.355 m long, 0.228 m tall and 0.482 m wide. A series of bolts is used to fix the model and several positioning blocks to a vertical shaker table.

The dynamics of the twin tails can be described by the following two nonlinear coupled differential equations. If u_1 and u_2 denote the generalized coordinates of the first modes of the isolated right and left tails, respectively, then we assume that the first modes of the twin-tail assembly are governed by the following two mass-normalized second-order coupled differential equations:

$$\begin{aligned} &\ddot{u}_1 + \omega_1^2 u_1 + 2\epsilon\mu_1\dot{u}_1 + \epsilon\alpha_1 u_1^3 + \epsilon\mu_3\dot{u}_1 \mid \dot{u}_1 \mid \\ &-\epsilon k(u_2 - u_1) - \epsilon u_1\eta_1 F\cos(\Omega t + \tau_1) + K f_1 \\ &= 0 \end{aligned} \tag{2}$$

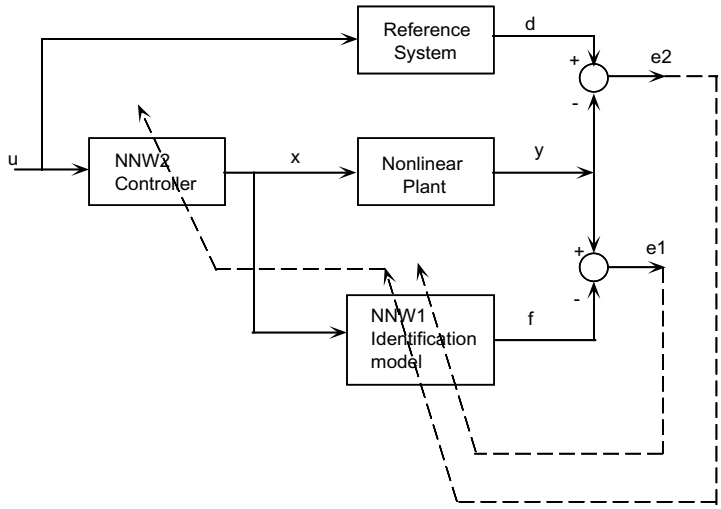

Fig. 2. Flowchart.

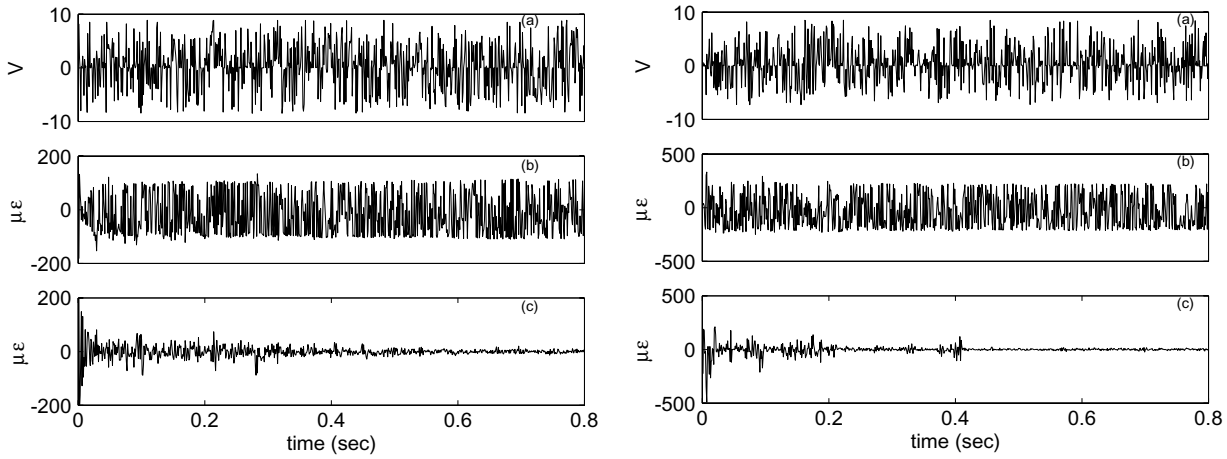

Fig. 3. Training of the twin tails (a) input forcing, (b) tail response, (c) error between the desired and the network responses.

$$\ddot{u}_2 + \omega_2^2 u_2 + 2\epsilon\mu_2\dot{u}_2 + \epsilon\alpha_2 u_2^3 + \epsilon\mu_4\dot{u}_2 \mid \dot{u}_2 \mid$$
$$-\epsilon k(u_1 - u_2) - \epsilon u_2\eta_2 F\cos(\Omega t + \tau_2) + K f_2$$
$$= 0 \tag{3}$$

where ω_1 and ω_2 are the natural frequencies of the isolated right and left tails, $2\mu_1$and $2\mu_2$ are the linear damping coefficients, α_1 and α_2 are the coefficients of the cubic nonlinearity, μ_3 and μ_4 are the aerodynamic damping coefficients, k is the coupling term between the twin tails, $\eta_1 F\cos(\Omega t + \tau_1)$and $\eta_2 F\cos(\Omega t + \tau_2)$ are the parametric excitation forces applied to the tail section, K is a proportionality constant (between volts and $\mu\epsilon$, measured experimentally), and f_1 and f_2 are the control action forces. Here, η_1 and η_2 are transmissibility terms that make the units of the whole equation consistent and ϵ is a bookkeeping parameter which is set equal to unity in the final analysis.

This model was provided by a system identification process [4] and all of the parameters were identified by techniques adopted from nonlinear dynamics theory [12]. Having this model facilitated the design of the neural networks for the identification and the control of the system since, instead of collecting real data from an experiment, we used this mathematical model to provide the necessary data for network training purposes.

These two second-order differential equations were transformed into a set of four first-order differential equations to make use of the available numerical routines that are specifically used for first-order systems. To this end, we let $x_1(t) = u_1(t)$, $x_2(t) = \dot{u}_1(t)$, $x_3(t) = u_2(t)$ and $x_4(t) = \dot{u}_2(t)$, and thus rewrite

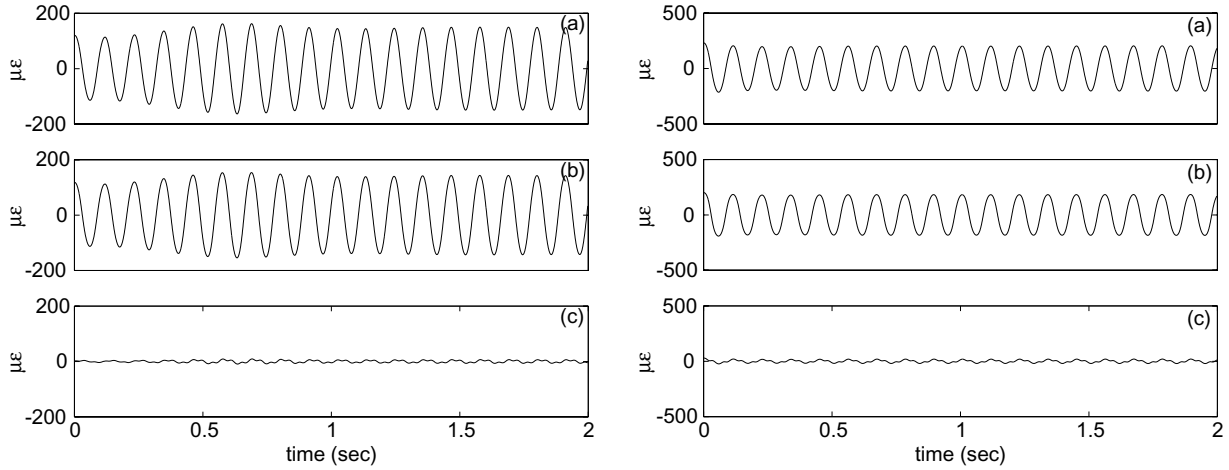

Fig. 4. Validation of the twin-tail neural networks: (a) model output signal, (b) neural network output signal, (c) error between theoutputs of the model and the neural network outputs.

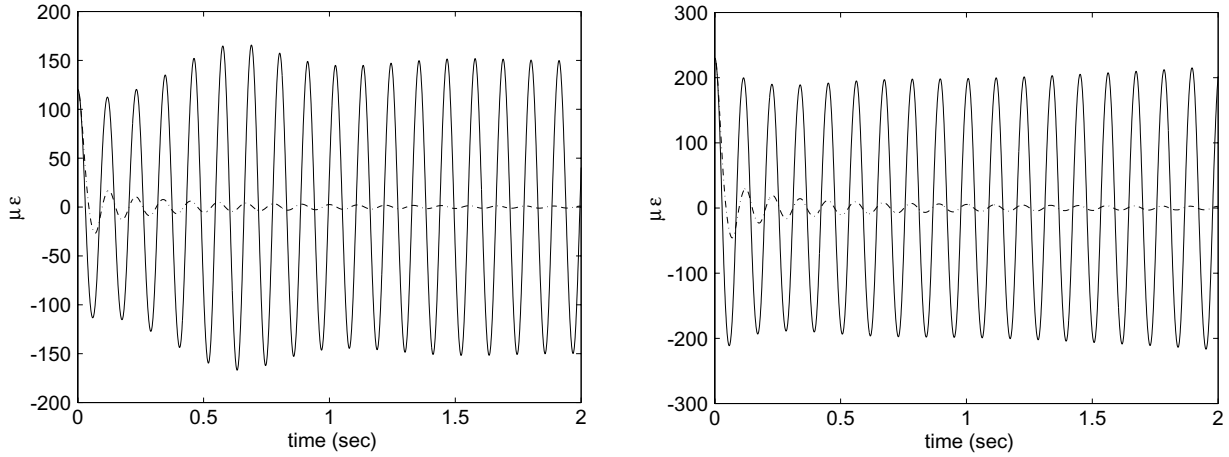

Fig. 5. Time histories of the responses of the tails (–) without control and (–.) with control.

Equations (2) and (3) in the following alternate form:

$$\dot{x}(t) = A(t)\ x(t) + n(t) \tag{4}$$

$$y(t) = Cx(t) \tag{5}$$

where

$$\dot{x}(t) = \begin{bmatrix} \dot{x}_1\ \dot{x}_2\ \dot{x}_3\ \dot{x}_4 \end{bmatrix}^T \tag{6}$$

$$A(t) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\omega_1^2 - \epsilon k + \epsilon\eta_1 F\cos(\Omega t + \tau_1) & -2\epsilon\mu_1 & \epsilon k & 0 \\ 0 & 0 & 0 & 1 \\ \epsilon k & 0 & -\omega_2^2 - \epsilon k + \epsilon\eta_2 F\cos(\Omega t + \tau_2) & -2\epsilon\mu_2 \end{bmatrix} \tag{7}$$

$$n(t) = \begin{bmatrix} 0 & -\epsilon\alpha_1 x_1^3 - \epsilon\mu_3 x_2 \mid x_2 \mid -Kf_1 & 0 \\ & -\epsilon\alpha_2 x_3^3 - \epsilon\mu_4 x_4 \mid x_4 \mid -Kf_2 \end{bmatrix}^T \tag{8}$$

It is very important to be able to discretize the system for simulation purposes; thus, a discretization scheme representing a zero-order-hold (ZOH) using the Euler formula (forward rule) was first tried. Also, a bilinear transformation (Tustin) scheme was also tried. Both schemes are considered to be very crude and require a very fast sampling frequency. Finally, a Runge-Kutta integration scheme was used with the period T of integration being specified. In this study, the data sampling period as well as the period T of integration were chosen to be $T = 0.001$ sec. The first bending mode of the right tail is 9.135 Hz and that of the left tail is 9.05 Hz. The state vector was updated after every step of inte-

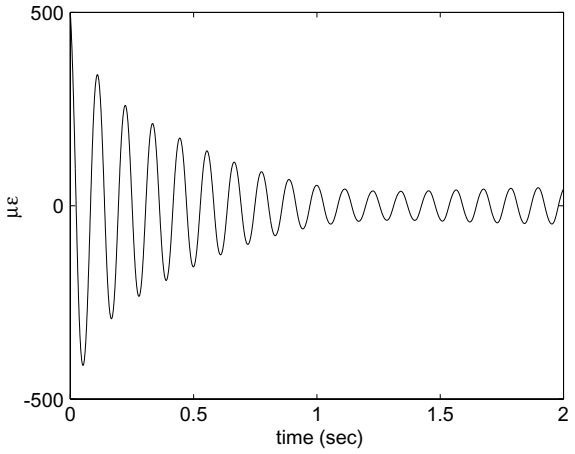

Fig. 6. Left-tail response after control using the ungeneralized trained neural net.

gration. The input to the system is f_1 and f_2 and the output is the strain. After discretization of the system, the governing differential equations can be rewritten as difference equations; that is,

$$x(k+1) = \Phi(k)x(k) + n(k) \tag{9}$$

$$y(k) = Cx(k) \tag{10}$$

Emulator Neural Network

To suppress vibrations using a neurocontroller, we had to train an emulator neural network first. The emulator learns to predict the response of the tails given the actuation forces. Thus, the emulator can be considered as representing the transfer function from the actuator signal to the sensor reading. This transfer function includes the nonlinear response of the tails as well as the effect of the actuator dynamics in the control loop. The emulator learns this relationship between the input and the output from the past history of the response given the actuation forces. The output $y(k)$ of the plant is a function of the input forcing function $f(k)$. Thus, the neural net requires $f(k)$ to be an input to the network. We chose to have an $f(k)$ as well as two delayed versions of it as inputs to the emulator network. Also, as may be seen from the equations describing the system, the nonlinear plant is of second order. To increase the accuracy of the emulator network, we also fedback the output of the network as inputs to the network; i.e. $y(k-1)$ and $y(k-2)$. The emulator was chosen to have one hidden layer. The input layer consisted of five units, which was found to be sufficient for a good model. The output layer had one unit that represented the current strain in the tail. Thus a 5-5-1 MLP was used. Figure 2 shows the detailed architecture of the emulator neural network together with the neurocontroller.

Two emulator networks were trained in this study. One network for the right tail, the other for the left tail. To train the network, since nothing is known about the ideal input, one must provide the necessary input data in a different way. A combination of uniformly distributed random data was used as input to the system. All desired values were simulated before the actual training process. In the real training process, the input vector was chosen randomly from the simulated data. The model was trained in one epoch, which was found to be enough. The values of the learning coefficients used in the training epoch are 0.8 and 0.5, respectively. The squashing functions used in the networks were hyperbolic tangent functions. Scaling of the input and output data pairs was necessary, since the effective range of the hyperbolic tangent function used is between -1 and 1. In this paper, the largest element in any input vector was used to scale the inputs to the specified range. The appropriate scaling factor was used before any neural network simulations. At the end, the output data was then scaled back to recover the original range of values of the signal. Figure 3 represents the results of this training for the right and left tails. These plots show the input to the system in volts, the output in microstrains, and the corresponding error between the desired and the actual responses of the network.

4. Model validation

In general, model validation addresses the verification of the dynamic model with respect to its prediction accuracy. Model validation is a significant step within the model development procedure. Validation tests ensure adequacy of the model be generalized to transient and stationary operating states, which were not introduced to the network during its training. Validation signals can be chosen arbitrarily, so that relevant process characteristics in certain ranges can be verified. Figure 4 shows the output signal from both the model and the neural net after fixing the weights. The input signal used was provided by a proportional derivative control law and it was used to validate the trained MLP network. We know that such an input signal will not affect the response of the tails since the external dis-

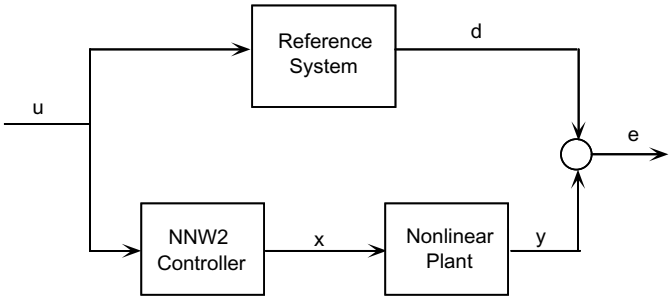

Fig. 7. Test of the controller.

turbance to the system is parametric. This signal possessed different characteristics compared to the signals used for the training.

Clearly, the MLP model neural network identified the dynamics of the twin tails and can predict the system response to a high degree of accuracy. A prime advantage of the MLP neural network is its capability to generate the steady-state characteristics of the dynamic process identified.

5. Model based control

The second step of the design was to train the controller network, having replaced the plant by its neural network model. Only strain can be measured from the twin tails. Therefore, the strain signal was chosen to be the input to the neurocontroller; that is two delayed versions of it $y(k-1)$ and $y(k-2)$. The outputs of the neurocontroller f_1 and f_2 were used as inputs to the neuroemulator (the plant model). Its output, the strain $y(k)$ of the tail, was computed and the error function e was calculated as the difference between the computed output strain of the neuroemulator and the model reference. The weights of the actuation network were adjusted based on the control command errors computed from the error function backpropagating through the emulator network, which was fixed. This process was done for both the right and left tails. The target response for these cases was chosen to be once a second-order oscillator with a high-damping ratio $\zeta = 0.707$ and the same natural frequency as the true system and the other time the target value was chosen to be zero. In both cases, the training process converged quickly with the model reference being slightly slower than the zero reference. The neurocontroller used was implemented as 2-5-1 MLP. Figure 5 displays the results of training the cascaded system. From this figure, it is clear that the neurocontroller did a good job in suppressing the steady-state vibrations of the two tails.

6. Generalization of the neuro-controller

Since the governing equations of the twin tails are nonlinear, we can expect several steady-state responses in the phase plane as well as the trivial solution. So the task of any controller in this case would be similar to that of a sliding-mode controller where all we need is to put the states of the system in the basin of attraction of the trivial solution (i.e., the no-vibration case). For these two coupled differential equations, there are three different possible solutions. Two of them are of the same order of magnitude of vibration, but are different in phase, and the third one has a smaller amplitude. Instead of training a neurocontroller for each solution, we tried the solution we had. Although there were many differences in the peak amplitudes of both solutions, the performance of the neurocontroller trained for the first solution was also very good for controlling the lower-amplitude solution, but this was not the case for the other way around, as seen in Fig. 6 since the steady state amplitude of the tail converged to aout 40 $\mu\epsilon$. This is expected because, in the second case, the system was operating outside the range of the training region and was thus expected to behave badly. Consequently, it is important for the designer to train the networks in the full regions of operation of the structure so that the networks can be generalized. Another issue that was taken into consideration during the design of the networks was that the networks were trained under high-disturbance levels so that lower-disturbance levels could also be controlled through the same trained networks. For this reason the networks were trained assuming a disturbance level of 3.5 g, which was even higher than we could excite experimentally (3.2 g).

After training the controller, the model was replaced by the true plant again. The system was tested according to the signal flowchart shown in Fig. 7. Similar plots were realized for the control of the system, which is expected since the neural network behaves similar to the mathematical model of the system.

7. Conclusion

This study demonstrated that neural networks can accurately identify and control a nonlinear system. To control the system, we first trained a neural network emulator to learn the mapping between the control signal and the response of the tails. Then, the neurocontroller learns how to control the structure with the help of the emulator network, which was used as a gateway to backpropagate the errors between the emulator's output and a model reference. This error was used to modify the weights of the neurocontroller until the system converged to the desired output provided by the reference model.

Acknowledgment

This work was supported by the Air Force Office of Scientific Research under Grant No. F49620-98-1-0393 and the National Science Foundation under Grant No. CMS-9423774.

References

[1] R.L. Bisplinghoff, H. Ashley and R.L. Halfman, *Aeroelasicity*, Addison-Wesley, Cambridge, MA., 1955.
[2] V.V. Chalam, *Adaptive Control System-Techniques and Applications*, Marcel Dekker, New York and Basel, 1987.
[3] H.M. Chen, K.H. Tsai, G.Z. Qi, J.C.S. Yang and F. Amini, Neural network for structure control, *Journal of Computing in Civil Engineering* **9** (1995), 168–175.
[4] A.A. El-Badawy and A.H. Nayfeh, Nonlinear Identification of a scaled structural dynamic model of the F-15 tail section, in: *Proceedings of the 17th International Modal Analysis Conference*, Kissimmee, Fl, 1999, pp. 1175–1181.
[5] S.J. Farlow, *Self-Organizing Methods in Modeling; GMDH Type Algorithms*, Marcel Dekker, New York, 1984.
[6] F.C. Fu and J.B. Farison, On the Volterra-series functional identification of nonlinear discrete-time systems, *Int. J. Control* **18** (1973), 1281–1289.
[7] M.T. Hagan, H.B. Demuth and M. Beale, *Neural Network Design*, PWS, Boston, MA., 1996.
[8] K. Hornik, M. Stinchcombe and H. White, Multilayer feedforward networks are universal approximators, *Neural Networks* **2** (1989), 359–366.
[9] P. Lisboa, *Neural Networks: Current Applications*, Chapman and Hall, New York, 1992.
[10] K.S. Narendra and K. Parthasarathy, Identification and control of dynamic systems using neural networks, *IEEE Trans Neural Networks* **1** (1990), 4–27.
[11] K.S. Narendra and K. Parthasarathy, Neural networks in dynamical systems, *SPIE Intelligent Control and Adaptive Systems* **1196** (1989).
[12] A.H. Nayfeh and B. Balachandran, *Applied Nonlinear Dynamics*, Wiley-Interscience, New York, 1995.
[13] J. Park and I.W. Sandberg, Universal approximation using radial-basis function networks, *Neural Computation* **3** (1991), 246–257.
[14] M.F. Tenorio and W.T. Lee, Self-organizing neural nets for the identification problem, in: *Advances in Neural Information Processing System*, (vol. 1), D. Touretzky, ed., 1989.
[15] S.M. Yang and G.S. Lee, Vibration control of smart structures by using neural networks, *Journal of Dynamic Systems, Measurment and Control* **119** (1997), 34–39.