Active Control of Wind-Tunnel Model Aeroelastic Response Using Neural Networks

Robert C. Scott*
NASA Langley Research Center, Hampton, VA 23681

Under a joint research and development effort conducted by the National Aeronautics and Space Administration and The Boeing Company (formerly McDonnell Douglas) three neural-network based control systems were developed and tested. The control systems were experimentally evaluated using a transonic wind-tunnel model in the Langley Transonic Dynamics Tunnel. One system used a neural network to schedule flutter suppression control laws, another employed a neural network in a predictive control scheme, and the third employed a neural network in an inverse model control scheme. All three of these control schemes successfully suppressed flutter to or near the limits of the testing apparatus, and represent the first experimental applications of neural networks to flutter suppression. This paper will summarize the findings of this project.

Keywords: neural network, adaptive control, aeroelasticity, flutter suppression

Introduction

ACTIVE control of aeroelastic phenomena will become more prevalent on future flight vehicles. One of the key issues in gaining acceptance of such systems is reliability. Systems that can reliably adapt to sensor and actuator failures or plant changes will improve system reliability. Since neural networks can be trained to model dynamic systems, their use has been suggested for adaptive control and many studies proposing a variety of control system architectures have been studied. Some common types of algorithms include model reference control where the neural network is used to model the plant and inverse control where the neural network is used to model the inverse of the plant. Studies to investigate the aeronautical applications of neural networks have been much more limited. References 1, 2, and 3 describe analytical studies where neural networks were applied to flight controls in either fixed wing or rotorcraft applications. These studies have focused on utilizing neural networks to achieve improvements in trajectory tracking. Studies to investigate the application of neural networks to controlling aeroelastic response (flutter and gust load alleviation, for example) have, however, been even more limited. The purpose of the present research is to investigate the use of neural networks for controlling aeroelastic response.

The work described in this paper is part of the Adaptive Neural Control of Aeroelastic Response (ANCAR) project. ANCAR was a joint research and development effort conducted by the National Aeronautics and Space Administration, Langley Research Center and The Boeing Company (formerly McDonnell Douglas) under a Memorandum of Agreement. The goal of the program was to develop and demonstrate neural-network based adaptive control systems using the Benchmark Active Controls Technology (BACT) wind-tunnel model. The ANCAR program consisted of two phases. Phase I was the development and demonstration of a neural network gain scheduled flutter suppression system. Under Phase II, two adaptive neural-network based control systems were to be developed and demonstrated. These systems used predictive control and inverse model control methodologies. This paper will summarize all the accomplishments of the ANCAR project; however, the majority of the paper will focus on the neural-network based inverse model system which has not previously been reported.

The work presented in this paper was an exploratory study. Additional research is required to determine the merits of using neural networks for aeroelastic control. In addition to the neural-network based control systems, there were several other control systems tested using the BACT wind-tunnel model. These control systems include those described in references 4 and 5. The control systems described in these papers were designed using classical or modern methods. While the performance of these control systems was not directly compared with the neural-network based control systems, it was apparent that they were significantly more robust than either the neural predictive or inverse model control systems. The neural gain scheduled system is based on classical

*Research Engineer, Aeroelasticity Branch, Structures and Materials Competency.

control law design and its performance was qualitatively similar to the systems in references 4 and 5.

Apparatus

Transonic Dynamics Tunnel

Wind-tunnel testing was conducted in the NASA Langley Transonic Dynamics Tunnel (TDT).[6] The TDT is a single-return variable-density transonic wind tunnel. The slotted test section is 16 ft by 16 ft square with cropped corners. The speed and pressure are independently controllable over a range of Mach number from 0.0 to 1.2 (unblocked), and a range of stagnation pressures from near zero to one atmosphere. Either air or a heavy gas can be used as the test medium. The heavy gas used in these wind-tunnel tests was R-12, but the TDT has since been modified to use R134a as the heavy gas. The TDT is also equipped with quick-opening bypass valves which can be activated to rapidly reduce test-section dynamic pressure and Mach number when flutter occurs. The combinations of large scale, high speed, high density, variable pressure, and the bypass-valve system make the TDT ideally suited for aeroelastic testing.

Wind-Tunnel Model

The BACT wind-tunnel model is a rigid, rectangular wing with an NACA 0012 airfoil section. It is equipped with a trailing-edge (TE) control surface and upper- and lower-surface spoilers, all independently controllable. The model is attached to a flexible mount system, the Pitch And Plunge Apparatus (PAPA), that allows both pitch and plunge degrees-of-freedom. An image of the model mounted in the TDT and a separate image showing only the model and mount system are shown in figure 1. The model is extensively instrumented with pressure transducers and accelerometers to measure surface pressures and model dynamic response, and the mount system is instrumented with strain gauges to measure normal force and pitching moment. Parameters which could be varied during the test include Mach number, dynamic pressure, model angle-of-attack, and control surface deflection. Reference 7 contains a more detailed description of this wind-tunnel model.

Neural Networks

Neural networks have been studied for many years in a variety of fields. These fields include speech and image recognition, credit and insurance policy evaluation, and trajectory control to name a few. They have also been studied extensively for use in controlling dynamic systems. The MATLAB Neural Network Toolbox Manual[8] provides an excellent overview of neural network applications with many references. While numerous control system architectures and network types have been investigated, this section of the paper will provide a brief summary of only the neural networks applied in this paper.

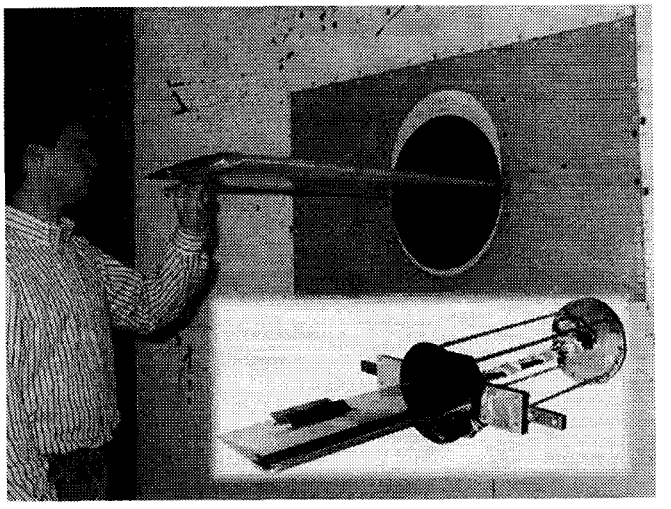


Fig. 1 BACT wind-tunnel model.

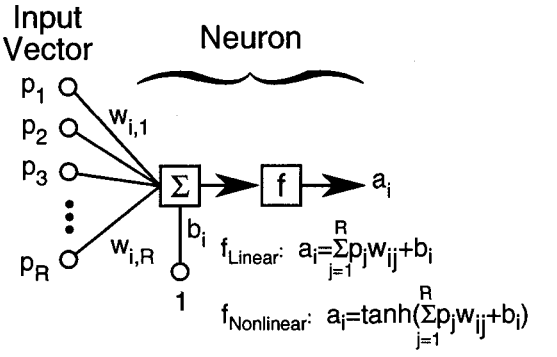


Fig. 2 Neural network computational element or neuron.

The name neural network comes from the fact that the networks emulate the structure of the brain. In biological nervous systems, the output of one neuron is connected to many other neurons. It is these connections that determine the function of the network. In practice, neural networks are composed of many computational elements or neurons operating in parallel. The general structure of an individual neuron is shown in figure 2. The function of each neuron is to sum the weighted inputs (w_{ij}) and the bias (b_j) and process this sum through a transfer function (f). The transfer functions considered in this study will be limited to the two shown in figure 2: a linear transfer function or a sigmoid transfer function (tanh). The use of a tanh or tan-sigmoid transfer function allows the modeling of nonlinear effects.

Two or more neurons can be used to form a layer and one or more layers forms a neural network. An example of the network structure used in this paper is shown in figure 3. This network consists of an input vector and two layers of neurons, a hidden layer and an output layer.

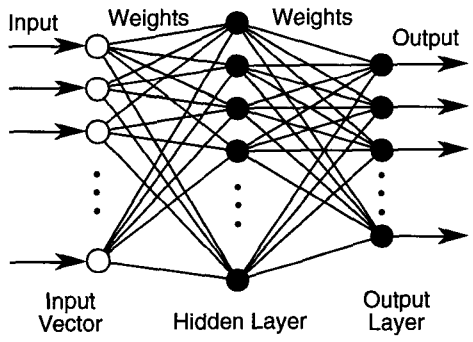


Fig. 3 Neural network with one hidden layer.

Typically the output layer uses linear transfer functions and the hidden layer uses linear or tan-sigmoid transfer functions. This network structure is the simplest form of multilayer feedforward network. More hidden layers can be used, but no more than one hidden layer was used in this paper. Given sufficient neurons on the hidden layer, this type of network can approximate most functions arbitrarily well.

Neural networks must be trained prior to use as predictive models. Training is a common term in the field of neural networks and simply refers to the process by which the network weights and biases are selected. The training process begins by selecting or acquiring training data, a set of input and output data for the plant or function to be approximated with the network. During the training process the weights and biases are adjusted until the error between the training data output and the network output is minimized. There are several methods or algorithms for adjusting the network weights and biases. A common training method, and the one used in this paper, is backpropagation. Backpropogation is a gradient descent optimization algorithm that can be used on multilayer networks as long as the neurons have differentiable transfer functions.

Gain Scheduler

This section of the paper will summarize the implementation and findings of the neural network gain scheduled flutter suppression system described in reference 9. The objective of this system was to use a neural network to schedule flutter suppression control laws. The approach taken in this study was to use a series of state-space models of the BACT wind-tunnel model to design classical single input single output (SISO) flutter suppression control laws. The state-space models were generated using the Integrated Structures, Aerodynamics, and Controls (ISAC) code.[10] The state-space models all used the same structural and aerodynamic models with Mach number (M) and dynamic pressure (q) being varied. In all, fifty six state-space models were used where Mach number varied from 0.3 to 0.9 and dynamic pressure varied from 75 to 250 psf. These models were used to design a fixed-gain control law and to design a series of control laws optimized to minimize accelerometer output for each combination of M and q. A neural network was used to schedule the series of 56 optimized control laws.

Fixed-Gain Control Law

The fixed-gain feedback control law was designed to stabilize and minimize the wing response over the entire range of the state-space models. A washout filter was included in the compensation to eliminate any drift due to bias errors in the accelerometers. Root-locus pole and zero placement methods were used to design the fixed-gain robust feedback control law given below.

$$\frac{s(s^2 + 12s + 520)}{s^4 + 27s^3 + 491s^2 + 4515s + 13050}$$

Neural-Network Scheduled Control Laws

For the neural-network scheduled control laws, the control law described above was tailored for the pole-zero dynamics of each state-space model. Fifty-six custom designs were generated for the various M and q condition state-space models. All of the resulting control laws had the same order numerator (3 zeros) and denominator (4 poles) as the robust fixed-gain control law. The general form of the these control laws is given below:

$$k\frac{s^3 + a_2s^2 + a_1s}{s^4 + b_3s^3 + b_2s^2 + b_1s + b_0}$$

The neural network, shown in the lower portion of figure 4, was trained using backpropagation to output the two numerator coefficients, the four denominator coefficients, and the overall gain as functions of M and q. The control law parameters used to train the neural network were in the continuous domain, rather than the discrete domain. This was required because continuous-domain coefficients vary smoothly as a function of M and q and do not require the high numerical precision of discrete-domain control law coefficients. The neural network outputs were transformed into the discrete domain before being transferred to the digital controller.

Experimental Results

The control system architecture that was implemented in the 1995 BACT wind-tunnel test is shown in figure 4. The trained neural network and Tustin (continuous-to-discrete) transformation was implemented on a Macintosh computer. As M and q were varied, the Macintosh computer transferred control laws to the real time system. The Active Digital Controller (ADC)[11] was used as the real-time digital control system operating at a rate of 200 Hz. The fixed-gain control law was also implemented on the ADC.

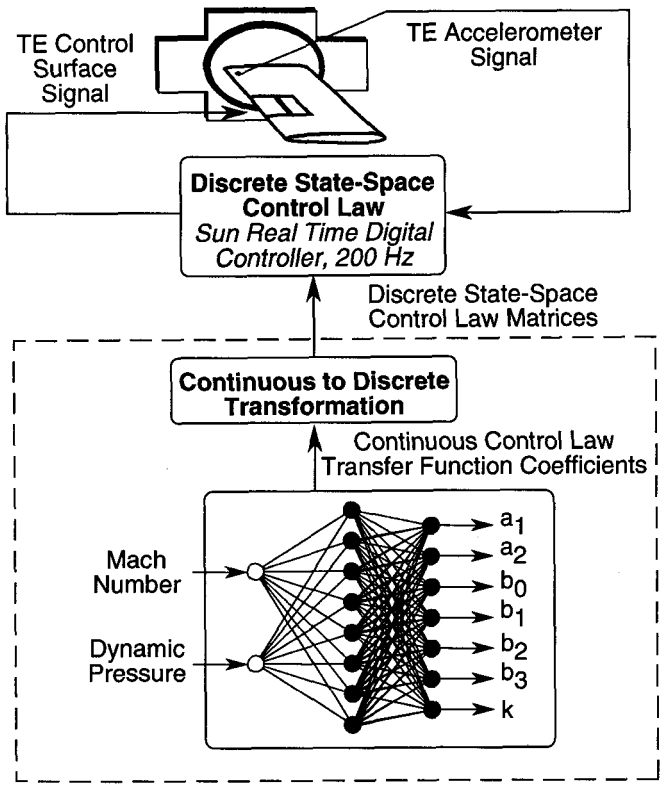


Fig. 4 Neural network gain scheduling system architecture.

The control systems were tested along four constant total-pressure lines which define test paths within the TDT. These H-lines and the BACT model open-loop flutter boundary are shown in figure 5. Each H-line is determined by the total pressure of the gas in the tunnel, and conditions are varied along each H-line by varying the motor RPM. For three of the four H-lines, open-loop unstable conditions were encountered and the control laws successfully suppressed flutter. For each H-line, data was collected for open-loop, fixed-gain closed-loop, and neural-network gain-scheduled closed-loop configurations. The open-loop tests were terminated as soon as classical flutter was encountered.

While both the fixed-gain and neural-network gain scheduled controllers provided effective flutter stabilization, the neural network system was able to achieve lower Root Mean Square (RMS) wing acceleration across the range of M and q conditions. The relative performance improvement is illustrated in figure 6 which plots the RMS wing acceleration versus dynamic pressure for the third H-line. The open-loop flutter boundary is indicated here by the sharp increase in open-loop response around q=160 psf.

Predictive Control

The neural predictive control system described in reference 12 will be summarized in this section. The goal

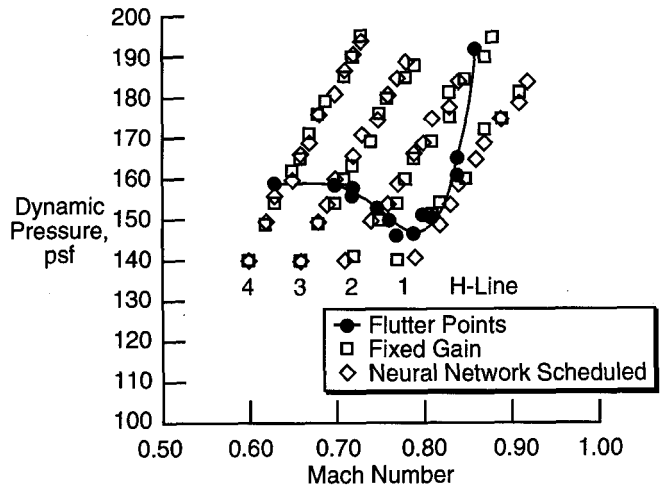


Fig. 5 Data acquired for fixed and gain scheduled systems.

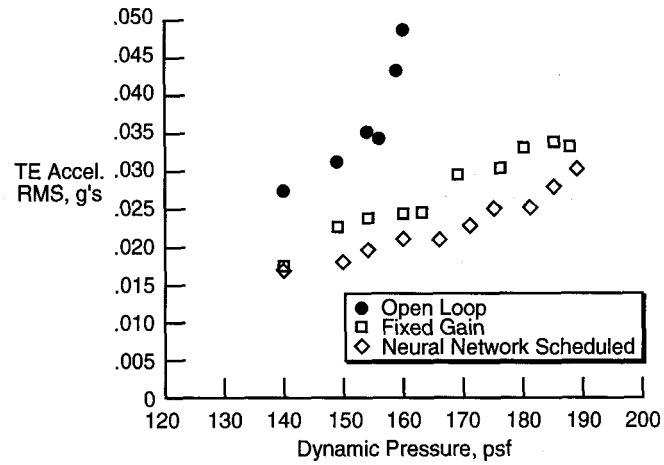


Fig. 6 Fixed and gain scheduled system response.

of this control system was to demonstrate the use of a neural network in a Model-Based Predictive Control (MBPC) system. MBPC utilizes a model of the plant to predict future physical model responses to future control inputs. A search algorithm is typically used to select the best control input. In this application, the objective of the search algorithm was to minimize model response. The plant model can be a linear or a nonlinear network, and the control system can include the ability to update the plant model. The system implemented in this study used a linear plant model. The control system trains the plant model using experimental data and the plant model can be retrained to adapt to changes in the physical plant. This algorithm is called Neural Predictive Control (NPC).

NPC Algorithm Description

A simplified block diagram of the NPC architecture is shown in figure 7. The NPC system can use either

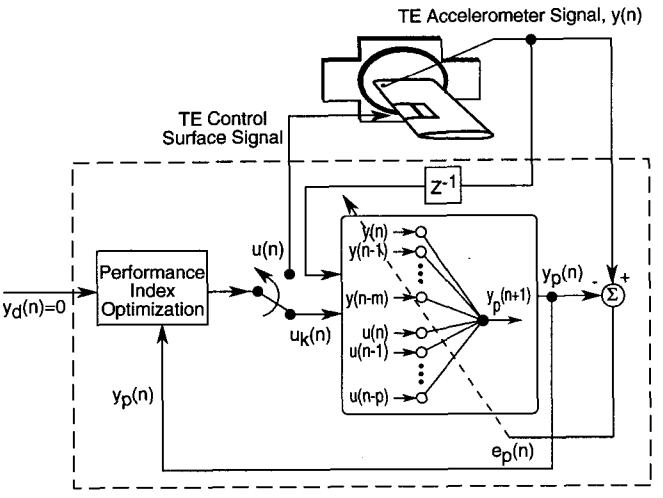


Fig. 7 Neural predictive control system architecture

a linear single layer network or neural-network plant model. As depicted in this figure, the NPC system implemented in this study used a linear plant model to capture the input/output dynamics of the BACT model. This predictive plant model is then utilized in an on-line optimization scheme to select the optimal control values for each control cycle. This system used a sample rate of 100 Hz, and each cycle had a duration of 0.01 seconds. The plant model can initially be trained by exciting the actuators, measuring the sensor response, and training the plant model. This model can then be updated on-line to handle with changing conditions and time-varying plant dynamics. The plant model training portion of this system is performed in parallel with the closed-loop portion and is depicted by the error feedback to the plant model in figure 7.

The method implemented for flutter suppression involves minimizing a cost function. The cost function, or performance index, is typically a quadratic function of the regulation/tracking error and required control input power. Consequently, NPC is an optimal controller in the same sense that the Linear Quadratic Regulator (LQR) is optimal. The advantage of NPC over LQR lies in its capability to be easily extended to nonlinear systems and to explicitly account for plant constraints in real time.

A step-by-step description of the NPC algorithm is given below:

1. Generate a reference trajectory, $y_d(n)$, which represents the desired value of the future plant output ($y_d(n) = 0$ for flutter suppression).

2. Predict the future plant output, $y_p(n+1)$, using the plant model. This prediction is based on the current and past values of the plant input and output, $u(n)$ and $y(n)$, and a new trial input value, $u_k(n)$. For the BACT model u is the TE control surface command and y is the TE accelerometer signal.

3. Evaluate the performance of the trial input value according to a performance index based on the cost of regulation/tracking error and the required control input power.

4. Repeat steps 2 and 3 with an optimization scheme until the termination criteria (desired performance or iteration limit) is achieved.

5. Output the trial control value selected by the optimization process to the plant. This is where the switch in figure 7 is moved from the down position to the up position so that the TE control surface command can be sent to the physical model. After the signal is sent the switch is returned to the down position for the next control cycle.

6. If on-line learning is engaged, update the plant model using a set of input/output data and an appropriate training algorithm.

7. Repeat the entire process for each control cycle.

The inputs to the plant model consist of time-delayed samples of the plant inputs, $u(n)$, and outputs, $y(n)$, as shown in figure 7. For nonlinear control applications, a multi-layer neural network architecture with backpropagation training can be used. For linear plants, a linear autoregressive moving average (ARMA) model is used. The NPC architecture was implemented using a Pentium 60 MHz PC with a plug-in Alacron neural accelerator board and analog input/output boards. The Pentium host CPU is responsible for all NPC computations and data transfer to the input/output and neural accelerator boards. The neural accelerator board performs the on-line model adaptation which occurs in parallel with the host CPU control loop computations. Due to several limitations of this hardware and software, only the linear ARMA plant models were considered in this study.

Simulation Results

The simulation studies were performed to design the appropriate plant model architecture, evaluate the robustness of the controller, and validate the real time control software. Successful adaptation and control was demonstrated across the range of simulated wind tunnel conditions, with the NPC system running at a rate of 100 Hz. One representative simulation time history is illustrated in figure 8 for an open-loop unstable condition, M=0.75 and q=175 psf. Starting with an untrained network, a white noise excitation signal was sent to the TE control surface for four seconds and the accelerometer response recorded providing 400 data points for neural

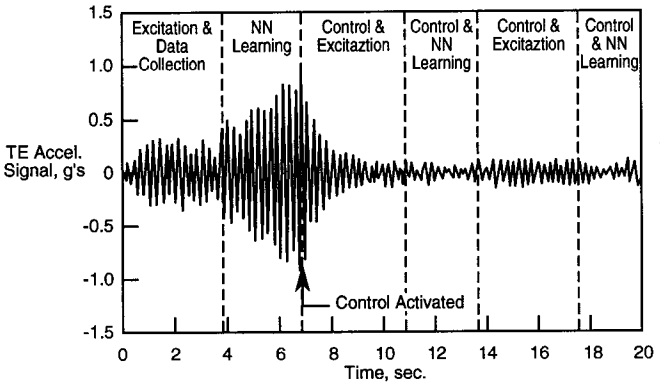


Fig. 8 NPC simulation time history.

network learning. The learning then occurred during the next 2.7 seconds, allowing control to be activated at about 6.7 seconds. As shown in the figure, from t=3.9 to t=6.7, the wing response grew steadily due to open-loop flutter until the controller was initiated for flutter suppression. Once the system was activated, learning and control occur simultaneously, allowing model updates to occur every 6.7 seconds. The length of this time interval is determined by the speed of the processors, the control cycle rate, and the amount of data needed for accurate plant modeling.

Experimental Results

The NPC system was tested at several M and q conditions along two H-lines. Testing began at the lower end of the H-line, in the open-loop stable region, with plant model learning activated. Dynamic pressure and Mach number were then gradually increased to conditions well beyond the open-loop flutter boundary. The continuous adaption of the plant model, which was one goal of the project, was not reliable enough to use for long periods of time. Periodically, the plant model generated by the on-line learning algorithm was not accurate enough and resulted in an unstable control system. Additional analysis is necessary to isolate the root of the problem, but it may be related to the level of random noise used to excite the wing dynamics, the amount of data used for training, or tuning of the performance index used by the NPC system.

After several unsuccessful attempts to operate in a fully adaptive mode, it was decided to activate learning only when a new model was required. Thus, for each H-line the plant model was generated at the low end of the H-line at an open-loop stable condition, and then learning was turned off, thereby freezing the plant model parameters. This arrangement was used to successfully suppress flutter along two H-lines as shown in figure 9.

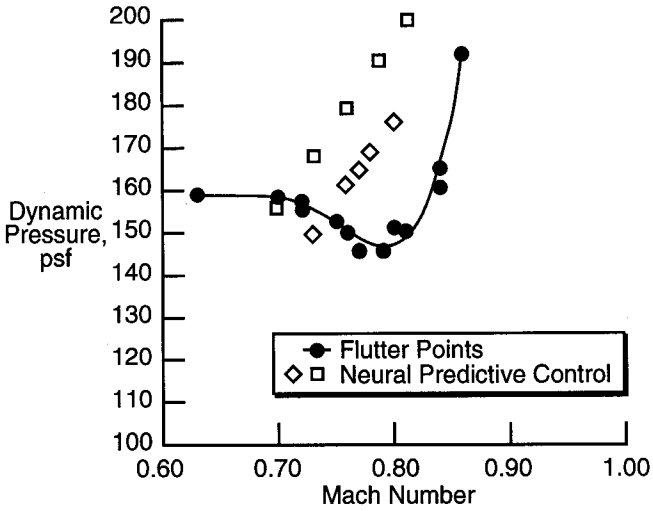


Fig. 9 NPC wind-tunnel data points.

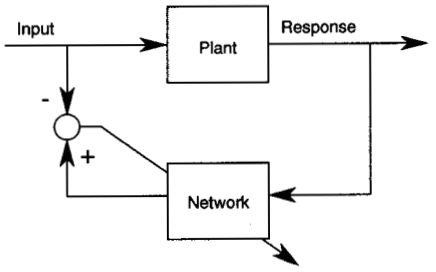
Inverse Model Control

This section of the paper will provide a detailed description of the application of neural networks and inverse modelling to the control of aeroelastic response.

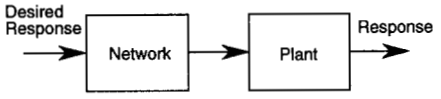
Architecture

The inverse modelling control architecture used in this study borrows elements from several proposed control schemes. This section of the paper will describe the relevant aspects of these systems and the system architecture implemented here. Inverse modelling control has generally been applied to trajectory tracking applications. References 13, 14, and 15 describe several approaches to inverse modelling control and introduce several elements applicable to the present problem. In general, the key assumption in inverse model control is that an unknown plant can be made to track an input command signal when this signal is applied to a controller whose transfer function approximates the inverse of the plant's transfer function. An adaptive control system can be created by applying an adaptive inverse modelling process to the plant. A simple form of such a system is depicted in figure 10. The upper portion of this figure shows the training procedure. Here the arrow passing through the network box indicates the feedback of the error for training of the network using back propagation. The lower portion of figure 10 shows the implementation of the inverse model. In practice the adaptive inverse model will be continuously updated. Thus, no direct feedback is used, except that the plant output is monitored and utilized to adapt the parameters of the controller.

The type of system just described is only suitable for minimum phase systems. The introduction of a time delay on the plant input used in the inverse modelling process allows one to obtain an approximate delayed in-



a) Inverse modelling process.



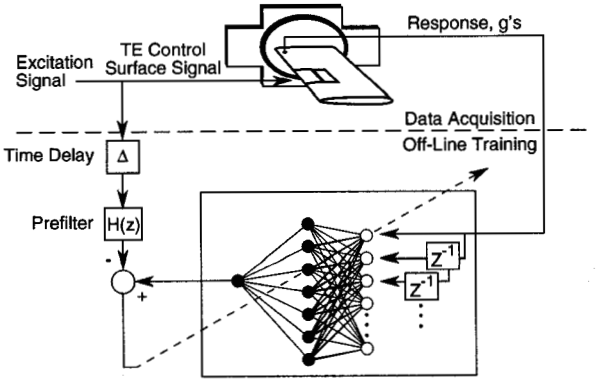
b) Control process.

Fig. 10 Generic inverse model control block diagram.

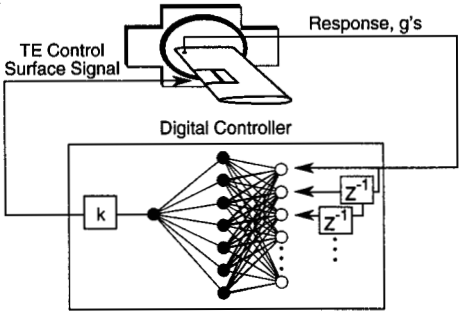
verse model to both minimum and non-minimum phase plants. Reference 14 also suggests using a prefilter on the plant input signal. Both the time delay and prefilter concepts will be employed in this study to generate inverse plant models. As the aeroelastic control application investigated here is not a trajectory following problem, the actual implementation of the inverse model in the control system will be different than that proposed by Widrow.[13–15] The implementation of the inverse model in a closed-loop system is discussed next.

The inverse model can take numerous forms including the one used here, a neural network. The implementation of the neural-network inverse model used in this study is similar to that proposed in reference 16. Unlike many other studies, reference 16 suggests a relatively simple implementation where the neural network inputs and outputs are connected directly to the plant like a standard feedback controller. The present implementation will use this approach. However, unlike reference 16 which used perceptron neural networks trained using genetic algorithms, the present application will use a two-layer feed-forward neural network trained using back-propagation.

The present implementation of the inverse modelling approach is shown in figure 11. The upper part of the figure shows the acquisition of training data and the off-line network training procedure. The input to the network is one of the BACT model's accelerometer signals. As indicated on the figure, the input to the network is made up of the current and delayed values of this signal. The number of time-delayed inputs used can be varied. Also, as discussed earlier, the model input signal used to train the network is delayed and filtered prior to used. The data acquisition was performed using the ADC[11] sampling at 200 Hz. The network training was performed off-line using the MATLAB software. The time delay



a) Data acquisition and controller training.



b) Controller implementation.

Fig. 11 Inverse model control system architecture.

(Δ) was an integer number of time steps. The prefilter was implemented using the MATLAB FILTFILT function. This function digitally filters the data in both the forward and reverse directions. The result has zero phase distortion with a magnitude modified by the square of the filter's magnitude response. The prefilter is tailored to have a peak magnitude near the frequency of the physical phenomenon to be controlled. For flutter suppression, the flutter frequency is used.

The lower portion of figure 11 shows the implementation of a trained network. Here the network is inserted in the control loop with the the BACT model's TE control surface command as the output of the network. This control system also was implemented on the ADC[11] sampling at 200 Hz. The gain, k, on the output signal could be varied on-line.

The following steps summarize the operation of this system.

1. Send excitation signal to the BACT model recording both the excitation (TE control surface command) and the model response (accelerometer signal).

2. Train the network using delayed and filtered excitation signal and BACT accelerometer response time history.

3. Implement trained network on the real-time system.

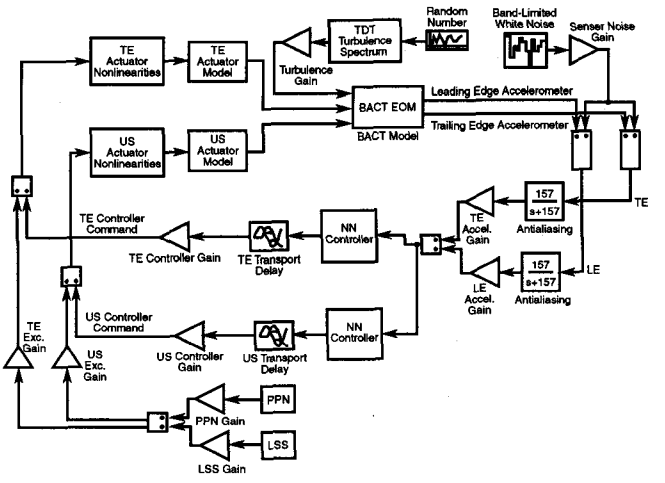


Fig. 12 Inverse model control system simulation block diagram.

Prior to implementation of this system, many parameters were explored using the simulation model discussed in the next section.

Simulation Model

The simulation model developed by Waszak[17,18] was used in this study. The model was formed by combining the equations of motion for the BACT wind-tunnel model with actuator models and a model of wind-tunnel turbulence. Wherever possible, the numerical model parameters were determined experimentally. The mass and inertia parameters were obtained by measuring the mass, stiffness, and damping properties of the BACT flexible mount. The static aerodynamic parameters were determined from experimental data when the BACT model was mounted to a five-degree-of-freedom balance.[7] The dynamic derivatives were obtained computationally using ISAC.[10] The numerical values for the static and dynamic stability and control derivatives are only valid at a single Mach number of 0.77; however, the dynamic pressure for the model could be changed. The analytical flutter boundary for this simulation model occurs at a dynamic pressure of 150.8 psf. Figure 12 shows the version of the simulation model used in this study. The BACT equations of motion and actuator models developed by Waszak are contained in the appropriately labeled blocks. The turbulence model in the upper left portion of the model is based on a Dryden spectrum with parameters tuned to match power spectrum data obtained in the TDT. The other elements of the block diagram were added for this investigation.

The primary changes to the model developed by Waszak included the addition of the neural-network control blocks and numerous other gain blocks. The control system studies in this investigation were SISO so the gain blocks were included to allow the same SIMULINK[19] model to be used for either accelerometer signal (trailing edge or leading edge) or either control surface (TE control surface of upper spoiler) without changing the model. For instance, if the gains on the upper spoiler control signal and the gain on the leading edge accelerometer signal are both zero, the resulting system would be SISO utilizing the TE control surface and the TE accelerometer. This was the configuration considered in this investigation. In addition, excitation generators and their associated gains were introduced for obtaining training data and for studying the effects of noise and turbulence.

Several parameters that could also be varied but are not shown in figure 12, include the number of hidden layers on the neural network and the number of time-delayed network inputs. The hidden layer could be either linear or nonlinear.

Parametric Variations

This section of the paper will present results from several parametric variations using the simulation model described above. First, a few words about how these parametric studies were performed. The primary objective of this system is flutter suppression and the performance of the control system will be evaluated at a dynamic pressure where the model is open-loop unstable, 175 psf. The data for training the network must be obtained initially at a dynamic pressure were the model is open-loop stable. The initial training was obtained from an open-loop simulation using a dynamic pressure of 133 psf where a pseudo-random noise (PPN) or linear sign sweep (LSS) input was used to drive the TE control surface. Both excitation signals had frequency content between 0 and 12 Hz. The prefilter for the flutter suppression applications was selected to have peak magnitude of unity in the vicinity of the flutter frequency of the BACT model. The prefilter used for flutter suppression control laws is shown below.

$$\frac{-431.1s}{s^3 + 19.6s^2 + 811.6s + 2529.8}$$

It has a peak magnitude of unity at 4 Hz and significant washout below and rolloff above this target frequency.

In all the parametric results presented, the RMS values of the TE accelerometer and the TE control surface will be plotted against the parameter being varied. Lower is better and values off the scale of the plot indicate that the system is not stable. The value of Δ, the number of time delays used in training the network, is a very important parameter and will often be used as the independent variable in these plots. The nominal simulation parameters are as follows

Network Controller:

Hidden Nodes = 6

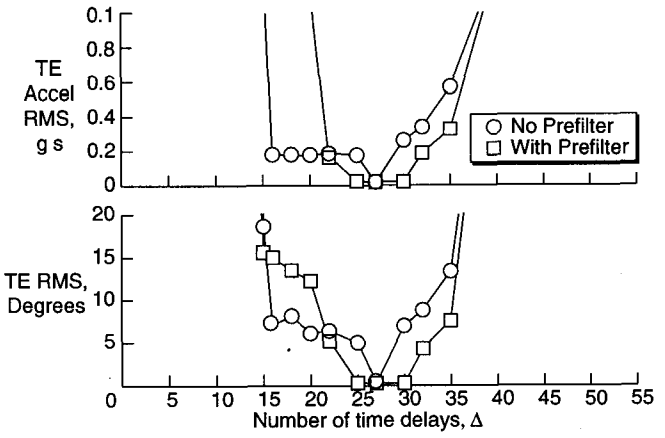


Fig. 13 Comparison of linear inverse model control system response with and without the use of the prefilter.

Time Delayed Inputs = 25
$\Delta = 25$

Training Data Simulation:

q = 133.0 psf
Turbulence Gain = 3.0
Simulation Time = 30 seconds
TE Control Surface Excitation = PPN

Controller Evaluation Simulation:

q =175.0 psf
Turbulence Gain = 3.0
Simulation Time = 30 seconds
Control Law Gain, $k = 2.0$

The first issue to be explored is the use of the prefilter. Figure 13 shows a comparison of linear controller performance where the linear network controllers were trained with and without the prefilter. The comparison shows that a controller that will suppress flutter can be obtained without using the prefilter, but it will have poor robustness properties. Typically, an acceptable flutter suppression control system will have control surface RMS values significantly below unity. Without the prefilter, only one value of Δ achieves acceptable control system performance.

The next parametric variation considered a linear network where Δ and the turbulence gain were varied. The turbulence gain was varied for the open-loop simulations where the training data was acquired. A gain of three was used to evaluate the closed-loop performance. This data is shown in figure 14. From these data, it is apparent that a time delay between 25 and 35 achieves the best performance. It can also be observed that moderate levels of turbulence improve the performance of the network controllers.

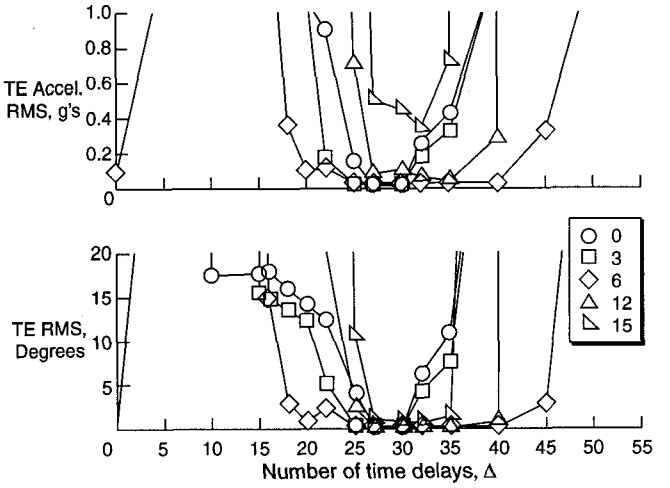


Fig. 14 Linear network inverse model controller performance for varying Δ and turbulence gain.

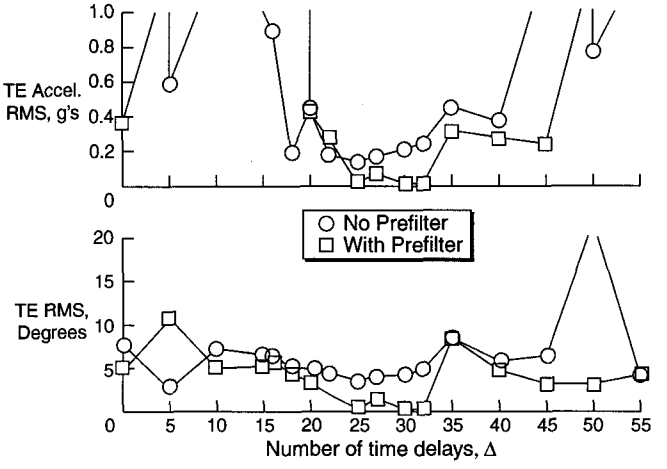


Fig. 15 Comparison of nonlinear inverse model control system response with and without the use of the prefilter.

The use of a nonlinear network (sigmoid transfer function on the hidden layer) is now considered. Figure 15 shows a comparison between nonlinear networks trained with and without a prefilter as a function of Δ. An important feature of the nonlinear result is that while the system may become unstable, the control command is limited by the saturation of the sigmoid transfer functions. Otherwise, the linear and nonlinear network controllers achieve similar levels of performance. To further explore the use of nonlinear network controllers, the turbulence gain and prefilter were varied. Figure 16 shows a comparison of the performance of nonlinear network controllers where the training data was obtained using various values of the turbulence gain. As with the linear case, moderate levels of turbulence were found to be desirable.

The next parameter to be considered in this study

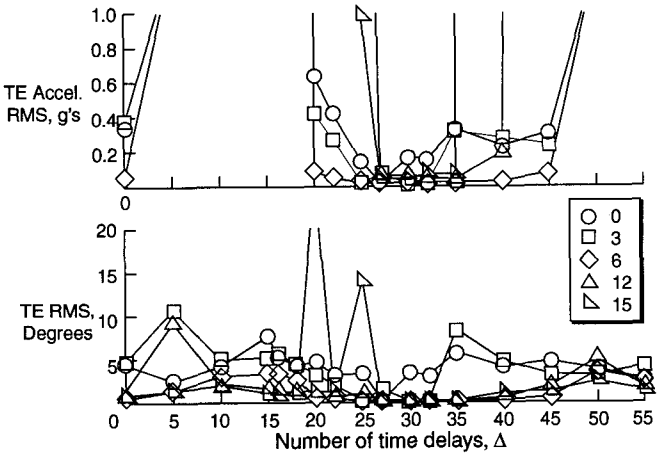


Fig. 16 Nonlinear network inverse model controller performance for varying Δ and turbulence gain.

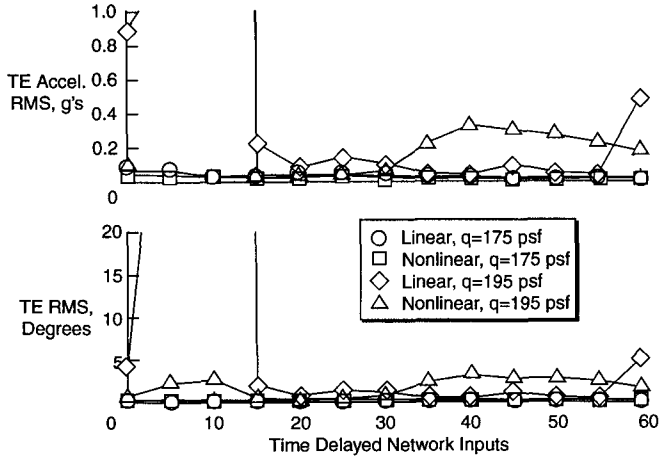


Fig. 17 Inverse model controller performance as a function of the number of delayed network inputs. Δ=25.

was the number of delayed inputs to use on the network controller. Figure 17 shows a comparison of performance of a linear and nonlinear network where the number of time-delayed inputs was varied from 0 to 60. The value of Δ in these analyses was 25. The systems were evaluated at two dynamic pressures, 175 psf and 195 psf. At 175 psf dynamic pressure, the performance of the system was insensitive to the number of time delays, but at 195 psf dynamic pressure, a minimum value of 15 time delays was required.

The final parameter considered is the excitation type used to obtain the training data. So far only a PPN excitation has been used. A comparison of PPN results with those obtained using a LSS excitation is presented in figure 18 for both linear and nonlinear controllers. Both types of excitations yield controllers with similar performance, but the PPN excitation appears to have a slight advantage over the LSS excitation.

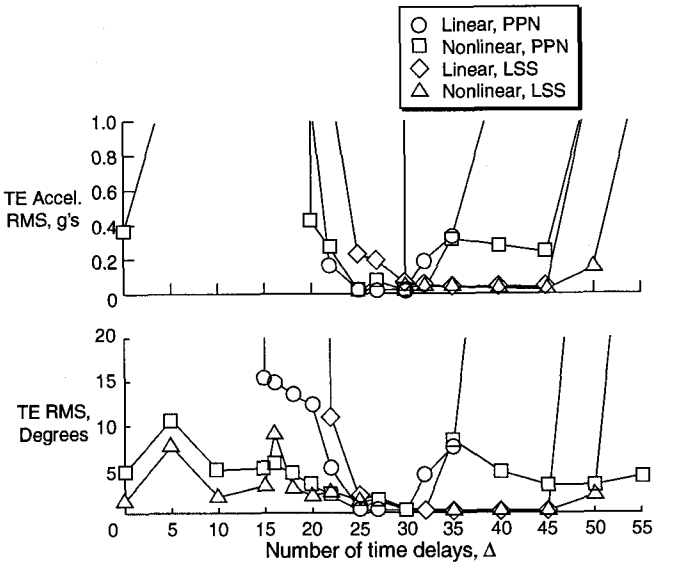


Fig. 18 Linear and nonlinear inverse model controllers trained using PPN and LSS excitation.

This completes the discussion on the inverse model control parametric studies. Note that not all parameters were varied. Future studies should consider the use of the leading accelerometer sensor and the use of the spoiler. The amplitude of the control surface excitation was not varied, and sensor noise was not considered at all. The only control systems considered were SISO, yet the simulation model can be modified to use both leading-edge and trailing-edge accelerometers as input signals to the controller. Finally, a truly adaptive system needs to continually re-acquire data and train new inverse model controllers as the plant changes. This was not considered in the present study.

Experimental Results

This section of the paper describes experimental results for application of inverse modelling control to the BACT model during the 1996 wind-tunnel test. The networks used during this wind-tunnel test all had 25 time-delayed inputs to the network and 6 nodes on the hidden layer. These controllers were SISO with the input coming from the TE accelerometer and the output being sent to the TE control surface. Unless otherwise noted, the prefilter used is the same as that described in the preceding subsection. Due to time constraints only a limited number of parameter variations were explored experimentally.

Two inverse model flutter suppression control systems were demonstrated. Figure 19 shows a portion of the TDT operating envelope with the BACT open-loop flutter boundary superimposed. The two inverse model control systems are designated A and B. System A was trained using data acquired at conditions well below the open-loop flutter boundary (M=0.65, q=133 psf).