JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 23, No. 6, November–December 2000

Active Flutter Suppression Using Recurrent Neural Networks

Franco Bernelli-Zazzera,* Paolo Mantegazza,† Giovanni Mazzoni,‡ and Matteo Rendina‡
Politecnico di Milano, 20158 Milan, Italy

The application of a recurrent dynamic neural network to the problem of adaptive active flutter suppression is described. A significant feature of such a neural controller is its application to an unstable and nonminimum phase system. Its development and experimental verification are carried out on a wing model with two tip control surfaces, one at the leading edge and one at the trailing edge. It combines a network for the identification of the variables to be controlled to a network carrying out the control task. The output generated by the first network is used to command the deflection of the control surfaces. The controlled variables are two appropriate wing accelerations, at the leading and trailing edges. Adaptivity is obtained by maintaining an online training of both neural networks. Extensive preliminary numerical simulations allow tuning the values of the fundamental design parameters required for an optimal compromise between computational load and system performances. The results of the application of such an approach to a real wing model have demonstrated its effectiveness in adapting to different operational conditions. The flutter free envelope of the wing model has been extended to a speed up to 34% higher than that corresponding to the uncontrolled flutter onset.

Introduction

A PROBLEM in modern aerospace vehicles is flutter because it can strongly limit the entire flight envelope. It is a dynamic instability caused by a variation of the global system damping: being fast it can be very dangerous and, if not properly controlled, even destroy the entire structure.

Consequently, active flutter suppression is a rewarding research area in aeroservoelasticity. The active control must work with a large number of varying configurations and operational parameters, for example, flight speed, mass, and/or variable wing geometry. This can generate some problems to actively controlled systems designed with classical tools.

In the last few years, the use of adaptive controllers has seemed to be the appropriate solution for the design of more flexible controllers, requiring less attention for the in-flight tuning and being capable to adapt to significant system parameters variations.[1]

Neural networks can be used to realize good adaptive controllers. They are used in many applications, such as character recognition, elaboration of signals, associative memory, identification and control.

The use of a neural network in control problems has been developed during the last 10 years, but for the most part, the results available deal with numerical simulations and static networks.[2–4] In the recent past, adaptive neural networks have been successfully applied in an experimental control of the vibrations of a large structure.[5]

Different neural networks can be used as controllers. One operates with static networks, where the free parameters (synaptic weights) are fixed and there is no online learning. Another possibility is the use of dynamic networks, which can follow time-varying systems with an active learning and, in this way, realize an adaptive control. The term active learning means that the weights of the network are automatically updated during the controller activity. The third possibility is the use of hybrid control, where the neural networks help in improving a classical controller.

The aim of this work is the design and implementation of a neural controller using a dynamic network characterized by online learning.[5,6] One network is used to identify the acceleration of some particular points of the dynamic system, and the information obtained by the first network is used in a mating controller network. The group of identification-control nets designed is like a black box. It does not need any information about the system except the measurements of the accelerometers, both for the identification and control. This paper pursues the objective of realizing this neural scheme of identification and control. The work is divided in two parts: the first is numerical and aims at the design of the network and at the choice of its correct architecture and topology. The second part is experimental and tests the system applied on the existing wing model. The tests have been carried out in a wind tunnel on a wing model with fixed geometry and mass, but the test conditions included variable wind-tunnel speed.

Physical and Numerical Wing Model

The model used is a cantilever wing (Fig. 1) of aspect ratio 8.25, taper ratio 2, wing surface 0.31 m^2, root chord 0.365 m, and span 1.22 m, with Wortmann FX L V-152 K 25 airfoil sections and two control surfaces ranging from 64 to 90% of the wing span and covering 20 and 25% of the chord, respectively, for the leading- and trailing-edge control surface. Each control surface can be deflected up to ± 20 deg. The most significant vibration modes and frequencies of the wing, with blocked control surfaces, are shown in Table 1.

Two brushless dc motors, placed at the wing root, actuate the control surfaces through connecting shafts. The motors are controlled by closed-loop current power drives that, independently from the actual motor state, allow a direct control of the actuating torque with a bandwidth greatly exceeding the one envisioned for the position servos. To emulate a real situation in which control surfaces are often controlled through a position servo, the control of the motor torque T_c is based on a proportional–integral–derivative (PID) command combined with a low-pass, second-order shaping filter with passband ω_s and is given by

$$T_c = \frac{as^2 + bs + c}{s(s + A)} \left(\frac{\omega_s^2}{s^2 + 2\xi\omega_s s + \omega_s^2} \delta_c - \delta_a \right) \tag{1}$$

where δ_a is the controlled surface deflection, as sensed by potentiometers placed at the wing tip side of each aileron, and δ_c is the related commanded position. The PID compensator is usually sufficiently robust against aerodynamic forces and unmodeled disturbances to allow its design by approximating the response of an isolated free control surface, as given by $s^2 J_a \delta_a = T_c$, with J_a the moment of inertia around the hinge axis, so that its closed-loop response is governed by

$$\delta_a = \frac{as^2 + bs + c}{\left(J_a s^4 + J_a A s^3 + as^2 + bs + c\right)} \times \frac{\omega_s^2}{\left(s^2 + 2\xi\omega_s s + \omega_s^2\right)} \delta_c \tag{2}$$

Received 31 July 1998; revision received 1 September 1999; accepted for publication 3 March 2000.

*Associate Professor, Department of Aerospace Engineering, Via La Masa 34. Member AIAA.
†Professor, Department of Aerospace Engineering, Via La Masa 34.
‡Graduate Student, Department of Aerospace Engineering, Via La Masa 34.

Table 1 Relevant vibration modes of wing model

Mode	NASTRAN frequency, Hz	Experimental frequency, Hz
First bending	3.07	3.1
In-plane bending	5.95	6.0
First torsion	8.2	8.5
Second bending	18.45	18.6

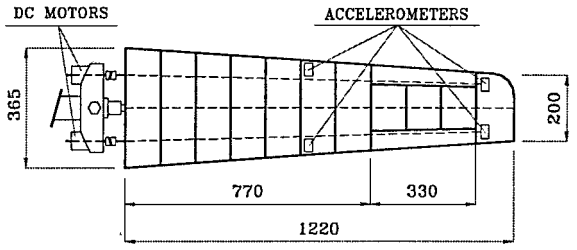

Fig. 1 Wing model.

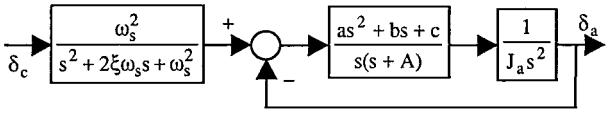

Fig. 2 Block diagram of control surface command.

The parameters a, b, c, and A are determined by prototyping the denominator, that is, the closed-loop poles, of the first term of Eq. (2) according to a fourth-order, low-pass Thomson filter (see Ref. 7). The command preshaping passband is chosen in a way to avoid the substantial bandwidth increase and significant step response overshoot caused by the zeros of the closed-loop transfer function, that is, to make $(as^2 + bs + c)/(s^2 + 2\xi\omega s + \omega^2)$ approximately constant. This command concept is shown in the block diagram of Fig. 2. The wing model is completed by four piezoresistive accelerometers, two mounted at the wing tip close to the control surfaces hinges and two at the midspan, whose signals are antialiazed by using two second-order Butterworth filters.

The wing response equations, to be used in the design simulations, have been determined by exploiting the aeroservoelastic modeling capabilities of NASTRAN.[8] These allow the inclusion of arbitrary transfer functions to connect the aeroelastic response to servos, sensors, and related filters, and lead to an equation of the type

$$\left(s^2[\boldsymbol{M}_g] + s[\boldsymbol{C}_g] + [\boldsymbol{K}_g] - q[\boldsymbol{H}_{am}(k, M)]\right)\{\boldsymbol{q}_g\} = [\boldsymbol{B}_c]\{\boldsymbol{\delta}_c\} \quad (3)$$

where $s = \sigma + j\omega$ is the complex frequency; $[\boldsymbol{M}_g]$, $[\boldsymbol{C}_g]$, and $[\boldsymbol{K}_g]$ are the generalized mass, damping, and stiffness matrices related to the generalized degrees of freedom vector $\{\boldsymbol{q}_g\}$, of order 12, including the 4 base structural modes earlier described, the 2 control surfaces rotations, the PIDs dynamics, the command shaping, and the antialiazing accelerometers filters. The aerodynamic transfer matrix $[\boldsymbol{H}_{am}(k)]$, with $k = \omega c/2V$, where c is the mean aerodynamic chord and V the asymptotic wind speed, is scaled by the dynamic pressure $q = \rho V^2/2$ and is available in tabulated form at different reduced frequencies. Because of the low operational wind speeds, it is assumed independent from the Mach number.

Equation (3) represents the system behavior in the complex frequency domain. Its direct translation into the time domain would lead to an integro-differential equation due to the convolution term arising from its aerodynamic part. It is, therefore, not amenable for simulating the system in real time with standard state of the art computational tools, generally related to differential equations. Therefore, a formulation of the unsteady aerodynamic forces in the time domain has to be found, to carry out the simulations required during the design phase.

To this end, a finite state approximation[9] could be used, but we prefer the simpler approach of setting

$$[\boldsymbol{H}_{am}(k)] \cong (jk)^2[\boldsymbol{M}_a] + jk[\boldsymbol{C}_a] + [\boldsymbol{K}_a] \quad (4)$$

with matrices $[\boldsymbol{M}_a]$, $[\boldsymbol{C}_a]$, and $[\boldsymbol{K}_a]$ being obtained through a least-squares fit of the available $[\boldsymbol{H}_{am}(k)]$, which, to cover the range of frequencies related to the flutter onset, extend up to $k = 2$.

Performing the inverse Laplace transform, we simply approximate Eq. (3) in the time domain with

$$\left([\boldsymbol{M}_g] - q(c/2V)^2[\boldsymbol{M}_a]\right)\{\ddot{\boldsymbol{q}}_g\} + ([\boldsymbol{C}_g] - q(c/2V)[\boldsymbol{C}_a])\{\dot{\boldsymbol{q}}_g\} + ([\boldsymbol{K}_g] - q[\boldsymbol{K}_a])\{\boldsymbol{q}_g\} = [\boldsymbol{B}_c]\{\boldsymbol{\delta}_c\} \quad (5)$$

so that, after defining $[\boldsymbol{M}_{AE}] = [\boldsymbol{M}_g] - q(c/2V)^2[\boldsymbol{M}_a]$, $[\boldsymbol{C}_{AE}] = [\boldsymbol{C}_g] - q(c/2V)[\boldsymbol{C}_a]$, and $[\boldsymbol{K}_{AE}] = [\boldsymbol{K}_g] - q[\boldsymbol{K}_a]$, it can be readily put into the state form:

$$\begin{Bmatrix}\dot{\boldsymbol{q}}_g \\ \ddot{\boldsymbol{q}}_g\end{Bmatrix} = \begin{bmatrix}[\boldsymbol{0}] & [\boldsymbol{I}] \\ -[\boldsymbol{M}_{AE}]^{-1}[\boldsymbol{K}_{AE}] & -[\boldsymbol{M}_{AE}]^{-1}[\boldsymbol{C}_{AE}]\end{bmatrix}\begin{Bmatrix}\boldsymbol{q}_g \\ \dot{\boldsymbol{q}}_g\end{Bmatrix} + \begin{bmatrix}[\boldsymbol{0}] \\ [\boldsymbol{M}_{AE}]^{-1}[\boldsymbol{B}_c]\end{bmatrix}\{\boldsymbol{\delta}_c\} + \begin{bmatrix}[\boldsymbol{0}] \\ [\boldsymbol{B}_n]\end{bmatrix}\{\boldsymbol{n}\} \quad (6)$$

to be used for the simulations. The noise term $\{\boldsymbol{n}\}$ is added to verify the adaptive flutter suppression system behavior in relation to realistic external disturbances, for example, turbulence and control noise.

A V–g presentation of the results of the stability analysis of Eq. (6) did not show any substantial discrepancy between the eigenvalues of Eq. (6) and those of a p–k solution of Eq. (3), thus demonstrating that Eq. (6) is an adequate model for the prediction of the flutter condition. This has been the only verification carried out to validate the approximation introduced by Eq. (4) because the actual adaptive control should be quite insensitive to the model used to simulate numerically its behavior in the preliminary design phase; otherwise, it will be doomed to failure during its implementation.

Dynamic Neural Network

A neural network (NN) is composed of elements called neurons, characterized by a set of weighted inputs. The links between the neurons and the external environment are called synapses. The total sum of the weighted inputs defines the internal activity of the neuron whose output is controlled by a neuron activation function, which usually limits the amplitude of the output. The mathematical model of a neuron can be synthesized in the following pair of equations:

$$v_k = \sum_{j=1}^{p} w_{kj} u_j, \qquad y_k = \varphi(v_k) \quad (7)$$

where $\boldsymbol{u}$ is the input vector, w the synaptic weight matrix, v the internal activity, φ the activation function, and y the output.

The literature reports different algorithms to structure a dynamic NN.[10] One possibility considers each neural weight matrix as a set of filters with a finite-duration impulse response instead of a scalar value. Other possibilities introduce time delays in the input vector of the NN or propagate the signals in time in a multilayer feedforward network scheme just as in the back propagation through time algorithm (BPTT).[10] These approaches are characterized by a large number of neurons, which entails a slow learning rate, so that they are effective only for slowly time-varying systems. The solution adopted in the present case modifies the architecture of the NN to make it intrinsically dynamic, and the topology used is that of a recurrent NN (RNN), where the output of the neurons is delayed by one time step and then fed back.

An RNN has many advantages over the other network schemes already presented, for example, smaller dimensions, reduced computational costs, and a faster learning: all characteristics that are very important for real-time control problems.[5,6]

The use of RNNs can, however, generate its own problems, the most important being the short memory caused by its small dimension, which leads to difficulties in the training phase.

Identification Network

In this paper a recurrent identification network (RIN) is used to identify the acceleration of the system at particular points of the wing. These points are connected with the location of the accelerometers in the real model. At the instant k, the RIN determines a prediction of the measurement $[\hat{y}(k + 1)]$ and allows carrying out

a predictive control of the system. The use of the one step delayed output in the prediction phase also acts in some sense as a first-order filter on the measured data, with beneficial effects because the accelerometers outputs are significantly corrupted by noise.

The RIN is essential in providing information about the dynamic system used by the control network. It is trained online through a real-time recurrent learning (RTRL), which proves to be very fast.[5,6,10]

The training is based on the minimization of a particular expression that changes according to the problem. The error function used for the RIN is the total sum of the average square errors referred to the nm output neurons.

$$E^{\mathrm{id}}(n) = \frac{1}{2}\sum_{j=1}^{nm}\left[e_j^{\mathrm{id}}(n)\right]^2 \tag{8}$$

The single error $e(n)$ is the difference between the real acceleration, that is, either the measured acceleration in the real system or the numerical output during the simulations, and the identified RIN output at the nth time step:

$$e_j^{\mathrm{id}}(n) = y_j(n) - \hat{y}_j^{\mathrm{id}}(n) \tag{9}$$

The output of the neurons is evaluated by

$$\hat{y}_j^{\mathrm{id}}(n+1) = \alpha^{\mathrm{id}} \cdot \tanh\left[\beta^{\mathrm{id}} \cdot v_j^{\mathrm{id}}(n)\right] = \varphi^{\mathrm{id}}\left[v_j^{\mathrm{id}}(n)\right] \tag{10}$$

$$v_j^{\mathrm{id}}(n) = \sum_{i=1}^{nid} w_{ij}^{\mathrm{id}}(n) \cdot u_i^{\mathrm{id}}(n) \tag{11}$$

where nid is the total number of inputs to the network, $\hat{y}^{\mathrm{id}}$ is the output of all of the neurons, and $\tanh(\cdot)$ is the adopted activation function. A $\tanh(\cdot)$ type activation function has been used for all of the neurons, both for identification and control. The motivation of this choice is that it matches the saturated positive and negative values of the real accelerations.

Equation (11) represents the internal activity of the single neuron. The input vector $\boldsymbol{u}$, composed by nid elements, contains, in the order, nm measured accelerations, ni control inputs, the one-step delayed output, related to nm predicted accelerations and nhm extra hidden outputs, and the threshold -1, as shown in Fig. 3.

The identification process aims at finding the distribution of the weights, that is, the design parameters of the RIN, that minimizes the error E^{id} at each time step. From a numerical point of view a gradient technique is used, so that the variation of the weights in the direction of the maximum slope proportional to the gradient function is

$$w_{ij}^{\mathrm{id}}(n+1) = w_{ij}^{\mathrm{id}}(n) + \Delta w_{ij}^{\mathrm{id}}(n) \tag{12}$$

$$\Delta w_{kl}^{\mathrm{id}}(n) = -\eta^{\mathrm{id}}\frac{\partial E^{\mathrm{id}}(n)}{\partial w_{kl}^{id}(n)} \tag{13}$$

where η^{id} represents the learning rate of the network. The indexes k and l range from 1 to the number of concatenated inputs and from 1 to the number of neurons, respectively.

In mathematical terms, the RTRL formulation is summarized by the following equations[6,10]:

$$\frac{\partial E^{\mathrm{id}}(n)}{\partial w_{kl}^{\mathrm{id}}(n)} = \sum_{j=1}^{nm} e_j^{\mathrm{id}}(n) \cdot \frac{\partial e_j^{\mathrm{id}}(n)}{\partial w_{kl}^{\mathrm{id}}(n)} = -\sum_{j=1}^{nm} e_j^{\mathrm{id}}(n) \cdot \frac{\partial \hat{y}_j^{\mathrm{id}}(n)}{\partial w_{kl}^{\mathrm{id}}(n)} \tag{14}$$

$$\frac{\partial \hat{y}_j^{\mathrm{id}}(n+1)}{\partial w_{kl}^{\mathrm{id}}(n)} = \frac{\partial \hat{y}_j^{\mathrm{id}}(n+1)}{\partial v_j^{\mathrm{id}}(n)} \cdot \frac{\partial v_j^{\mathrm{id}}(n)}{\partial w_{kl}^{\mathrm{id}}(n)} = \varphi'^{\mathrm{id}}\left[v_j^{\mathrm{id}}(n)\right] \cdot \frac{\partial v_j^{\mathrm{id}}(n)}{\partial w_{kl}^{\mathrm{id}}(n)} \tag{15}$$

$$\frac{\partial v_j^{\mathrm{id}}(n)}{\partial w_{kl}^{\mathrm{id}}(n)} = \sum_{i=1}^{nid}\left[\frac{\partial w_{ij}^{\mathrm{id}}(n)}{\partial w_{kl}^{\mathrm{id}}(n)} u_i^{\mathrm{id}}(n) + w_{ij}^{\mathrm{id}}(n)\frac{\partial u_i^{\mathrm{id}}(n)}{\partial w_{kl}^{\mathrm{id}}(n)}\right] \tag{16}$$

$$\sum_{i=1}^{nid}\frac{\partial w_{ij}^{\mathrm{id}}(n)}{\partial w_{kl}^{\mathrm{id}}(n)} u_i^{\mathrm{id}}(n) = \delta_{ki} \cdot u_l^{\mathrm{id}}(n) \tag{17}$$

$$\sum_{i=1}^{nid} w_{ij}^{\mathrm{id}}(n)\frac{\partial u_i^{\mathrm{id}}(n)}{\partial w_{kl}^{\mathrm{id}}(n)} = \sum_{i=1}^{nm+nhm} w_{ij}^{\mathrm{id}}(n)\frac{\partial \hat{y}_i^{\mathrm{id}}(n)}{\partial w_{kl}^{\mathrm{id}}(n)} \tag{18}$$

where $\varphi'[v_j(n)]$ is the first derivative, with respect to v_j, of the sigmoidal activation function, and δ_{ki} is the Kronecker delta.

The RTRL algorithm appears quite similar to the traditional BPTT, but it is simpler because the actual memory of the network is limited to a single time step of the predicted output. Unlike the BPTT, no multilayer network is devised and no memory is required for the measurements and input. The errors are never backpropagated, but the recurrent connections allow explicit evaluation of the temporal effects of the errors in the cost function and its gradient.

Control Network

Different techniques can be used to control a system with a neural controller (NC). The most often used method is based on the inversion of the identified model, especially if the work is online and an adaptive approach is required. However, the system to be controlled in this work is unstable and nonminimum phase. The second characteristic is very important because it makes the inverted model unstable. The NC devised is, thus, based on a predictive control approach, but a pseudoinversion technique has been used, introducing in the controller cost function a penalty term due to the control action, to make the whole controller dynamics stable.[11] To adapt to time-varying conditions, a recurrent control network (RCN) has been devised also for the controller, as shown in Fig. 3.

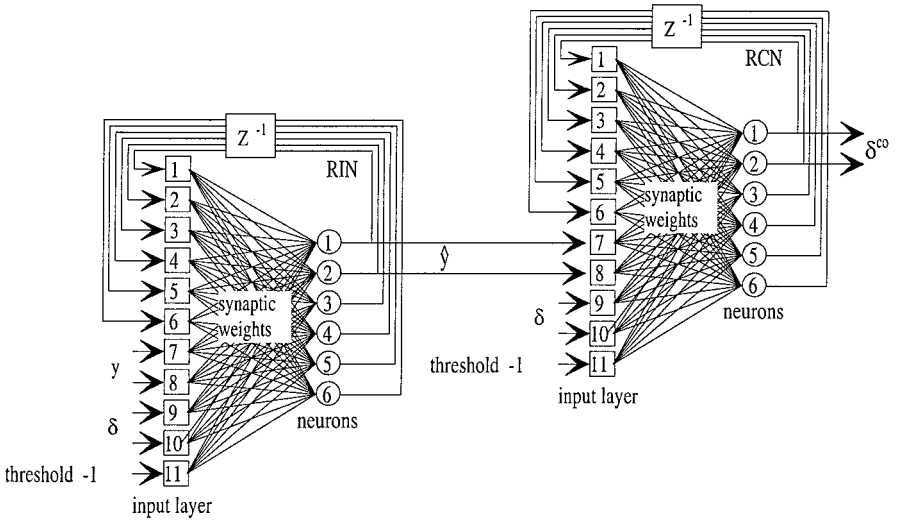

Fig. 3 RNNs (RIN and RCN).

The output of each neuron in the RCN is expressed as

$$\delta_s^{\mathrm{co}}(n+1) = \alpha^{\mathrm{co}} \cdot \tanh\left[\beta^{\mathrm{co}} \cdot v_s^{\mathrm{co}}(n)\right] = \varphi^{\mathrm{co}}\left[v_s^{\mathrm{co}}(n)\right] \quad (19)$$

$$e_j^{\mathrm{co}}(n) = y_j^{\mathrm{des}}(n+1) - \hat{y}_j^{\mathrm{id}}(n+1) \quad (20)$$

where $\hat{y}^{\mathrm{id}}$ is the output of the RIN, that is, the acceleration prediction, and y^{des} is the desired output, set to zero because the aim of flutter suppression requires maintaining a structural acceleration close to zero.

The cost function to minimize is

$$E^{\mathrm{co}}(n) = \frac{1}{2}\sum_{j=1}^{nm} e_j^{\mathrm{co}^2}(n) + \frac{1}{2}\gamma \sum_{j=1}^{ni} \delta_j^{\mathrm{co}^2}(n) \quad (21)$$

where the second term defines the penalty element depending on the same output of the RCN, nm is the number of the RIN outputs, and ni is the number of the RCN outputs.

Note the splitting of Eq. (21) in two different terms. The first depends on the difference between the actual and desired output of the system, whereas the second one is necessary to limit the use of controls, where $\delta^{\mathrm{co}}(n)$ are the outputs of the RCN, that corresponds physically to the commanded deflection of the control surfaces.

The term γ represents the weight of the penalty term and is a free parameter optimizing the activity of the controller.

Again, the minimization of the control cost function E^{co} is performed by the steepest descent rule

$$\Delta w_{tv}^{\mathrm{co}}(n) = -\eta^{\mathrm{co}} \frac{\partial E^{\mathrm{co}}(n)}{\partial w_{tv}^{\mathrm{co}}(n)} \quad (22)$$

where η^{co} is the learning rate of the control. The cost function gradient is evaluated as

$$\frac{\partial E^{\mathrm{co}}(n)}{\partial w_{tv}^{\mathrm{co}}(n)} = \sum_{j=1}^{nm} e_j^{\mathrm{co}}(n) \cdot \frac{\partial e_j^{\mathrm{co}}(n)}{\partial w_{tv}^{\mathrm{co}}(n)} + \gamma \sum_{j=1}^{ni} \delta_s^{\mathrm{co}}(n) \cdot \frac{\partial \delta_s^{\mathrm{co}}(n)}{\partial w_{tv}^{\mathrm{co}}(n)}$$

$$= E_1'(n) + \gamma \cdot E_2'(n) \quad (23)$$

There are different terms to be determined: The first one depends on the desired output and the second on the control penalty.

The RTRL algorithm applied to RCN learning is synthesized in the following equations, based on the imposition that the variation of the RIN weights is independent from the variation of the RCN weights.[6]

The dependence on the desired output is evaluated with

$$E_1'(n) = -\sum_{j=1}^{nm} e_j^{\mathrm{co}}(n) \cdot \frac{\partial \hat{y}_j^{\mathrm{id}}(n+1)}{\partial w_{tv}^{\mathrm{co}}(n)} \quad (24)$$

$$\frac{\partial \hat{y}_j^{\mathrm{id}}(n+1)}{\partial w_{tv}^{\mathrm{co}}(n)} = \frac{\partial \hat{y}_j^{\mathrm{id}}(n+1)}{\partial v_j^{\mathrm{id}}(n)} \cdot \frac{\partial v_j^{\mathrm{id}}(n)}{\partial w_{tv}^{\mathrm{co}}(n)} = \varphi'^{\mathrm{id}}\left[v_j^{\mathrm{id}}(n)\right] \cdot \frac{\partial v_j^{\mathrm{id}}(n)}{\partial w_{tv}^{\mathrm{co}}(n)} \quad (25)$$

$$\frac{\partial v_j^{\mathrm{id}}(n)}{\partial w_{tv}^{\mathrm{co}}(n)} = \sum_{i=1}^{nid}\left[\frac{\partial w_{ij}^{\mathrm{id}}(n)}{\partial w_{tv}^{\mathrm{co}}(n)} \cdot u_i^{\mathrm{id}}(n) + w_{ij}^{\mathrm{id}}(n) \cdot \frac{\partial u_i^{\mathrm{id}}(n)}{\partial w_{tv}^{\mathrm{co}}(n)}\right] \quad (26)$$

$$\sum_{i=1}^{nid} \frac{\partial w_{ij}^{\mathrm{id}}(n)}{\partial w_{tv}^{\mathrm{co}}(n)} \cdot u_i^{\mathrm{id}}(n) = 0 \quad (27)$$

$$\sum_{i=1}^{nid} w_{ij}^{\mathrm{id}}(n) \cdot \frac{\partial u_i^{\mathrm{id}}(n)}{\partial w_{tv}^{\mathrm{co}}(n)} = \sum_{s=1}^{ni} w_{(s+nm+nhm)j}^{\mathrm{id}}(n) \cdot \frac{\partial \delta_s^{\mathrm{co}}(n)}{\partial w_{tv}^{\mathrm{co}}(n)} \quad (28)$$

$$\frac{\partial \delta_s^{\mathrm{co}}(n+1)}{\partial w_{tv}^{\mathrm{co}}(n)} = \frac{\partial \delta_s^{\mathrm{co}}(n+1)}{\partial v_s^{\mathrm{co}}(n)} \cdot \frac{\partial v_s^{\mathrm{co}}(n)}{\partial w_{tv}^{\mathrm{co}}(n)} = \varphi'^{\mathrm{co}}\left[v_s^{\mathrm{co}}(n)\right] \cdot \frac{\partial v_s^{\mathrm{co}}(n)}{\partial w_{tv}^{\mathrm{co}}(n)} \quad (29)$$

$$\frac{\partial v_s^{\mathrm{co}}(n)}{\partial w_{tv}^{\mathrm{co}}(n)} = \sum_{r=1}^{nco}\left[\frac{\partial w_{rs}^{\mathrm{co}}(n)}{\partial w_{tv}^{\mathrm{co}}(n)} \cdot u_r^{\mathrm{co}}(n) + w_{rs}^{\mathrm{co}}(n) \cdot \frac{\partial u_r^{\mathrm{co}}(n)}{\partial w_{tv}^{\mathrm{co}}(n)}\right] \quad (30)$$

$$\sum_{r=1}^{nco} \frac{\partial w_{rs}^{\mathrm{co}}(n)}{\partial w_{tv}^{\mathrm{co}}(n)} \cdot u_r^{\mathrm{co}}(n) = \delta_{tr} \cdot u_v^{\mathrm{co}}(n) \quad (31)$$

$$\sum_{r=1}^{nco} w_{rs}^{\mathrm{co}}(n) \cdot \frac{\partial u_r^{\mathrm{co}}(n)}{\partial w_{tv}^{\mathrm{co}}(n)} = \sum_{r=1}^{ni+nhi} w_{rs}^{\mathrm{co}}(n) \cdot \frac{\partial \delta_r^{\mathrm{co}}(n)}{\partial w_{tv}^{\mathrm{co}}(n)} \quad (32)$$

with nco the number of RCN inputs, δ_{tr} the Kronecker symbol, and nhi the number of hidden inputs, that is, the fictitious outputs of the RCN that are fed back to the network.

The dependence on the control is evaluated with

$$E_2'(n) = \sum_{s=1}^{ni} \delta_s^{\mathrm{co}}(n) \cdot \frac{\partial \delta_s^{\mathrm{co}}(n)}{\partial w_{tv}^{\mathrm{co}}(n)} \quad (33)$$

where the derivative of the control with respect to the weights is given by Eqs. (29–32).

Finally, Eq. (34) represents the complete variation of the weight matrix:

$$\Delta w_{tv}^{\mathrm{co}} = -\eta^{\mathrm{co}}(E_1' + \gamma \cdot E_2') \quad (34)$$

Preliminary Verification Through Numerical Simulations

Extensive numerical simulations must be carried out to understand the behavior of the presented networks, to decide the correct architecture and topology in terms of number of neurons, and to tune the numerical parameters.

In detail, the objectives of such numerical simulations are the determination of the number of hidden and output neurons for the RIN and RCN, the parameters of the activation function for each neuron, the learning rates, the penalty terms, the input vector in the RIN and RCN, the identification and the control frequency, and the combination of accelerations to identify.

These numerical simulations can be divided in two different groups: the first is characterized by a constant wind speed and is used to build a robust set of networks, whereas the second is characterized by an accelerating flow, and it is used to verify the new flutter speed with the activation of the controller.

To make the numerical simulations closer to the experimental activity the simulations included the following:

1) A disturbance on the accelerometers signal is included in the form of a zero mean white noise of 0.01 g intensity. The noise level has been inferred from the analysis of series of measurements taken by the real accelerometers in previous wind tunnel tests.[1]

2) A square wave is included to force the control surfaces motion and to maintain the numerical model permanently excited. The amplitude of the square wave is a parabolic decreasing function of the airspeed, from ±3-deg amplitude at zero speed to 0.5 deg at and above the uncontrolled flutter speed of 25 m/s. This to have a constant maximum force as the dynamic pressure varies. The amplitude is either positive or negative according to a randomly generated binary sequence.

3) Step control surfaces deflections of about 10 deg are included.

The number of the neurons affects the memory of the networks. A large number of neurons increases its robustness and the precision of the identified output, but also the computational cost, while decreasing the learning speed. However, a very precise identification of the measured output is usually useless for control purposes because in this case the control network also would be fed by the sensor noise, causing dangerous chattering of the control surfaces. The compromise is, therefore, between learning speed, true output identification, and measurement noise filtering.

For conciseness, the simulated performances of the controller will not be shown but the considerations that led to the final design are briefly reviewed.

As mentioned, the activation function chosen in this application is the hyperbolic tangent, expressed in Eqs. (10) and (20), with α and β chosen to maximize the reaction of the output of each neuron as suggested in the literature.[10]

The learning rates, either for the RIN or the RCN, have been chosen as a compromise between the need of a fast learning (high rates) and the obvious requirement of stability (low rates).

The penalty term turns out to be effective in producing a stable inversion of the nonminimum phase aeroelastic system, thus avoiding the divergence of the control term.

The input vectors to the two networks in this application contain the feedback of all of the neurons, because the networks are fully recurrent, whereas other inputs as detailed in Fig. 3 have been chosen to provide information about the model to the networks.

The control frequency is restricted by the computational power available because for online learning the number of floating point operations is very high when the RIN and RCN are working. However, to follow the fairly fast dynamic system at hand, a relatively high control frequency is required.

The best combination of accelerations to be identified turned out to be those related to the midspan accelerometers. The use of wing tip accelerations, alone or combined with the midspan ones, lead to a controller extremely sensitive to any disturbance. In this case, the control activity becomes too strong to allow a safe model inversion. To overcome this problem, it could be useful to increase the control penalty in the RCN, but this opportunity has not been investigated.

The parameters selected on the basis of all of the numerical tests are summarized in Table 2.

The designed identification and control networks are each composed of six neurons, two of which have a visible output whereas the others have a hidden output. The schemes of the RIN and RCN and their connections with the wing model are presented in Figs. 3 and 4.

The numerical simulations proved that it was possible to increase the flutter speed from 25 to 34 m/s with a topology that can withstand all of the earlier listed disturbances by using a neural control and identification with a permanently active learning whose neuron weights start from a group of values obtained from simulations related to a lower wind speed. This preliminary learning phase is important to prepare the networks for the disturbances at increasing flow speeds.

The numerical simulation demonstrated also the capability to adapt to a variation of the model mass. This has been simulated by suddenly varying the mass matrix of the wing model, as in a store release case.

Table 2 Values of the networks free parameters

Parameter	Value
Learning rate RIN, RCN η	0.4
Activation function parameters α, β	1.716, $\frac{2}{3}$
Penalty γ	0.1
Frequency RIN, RCN	180 Hz

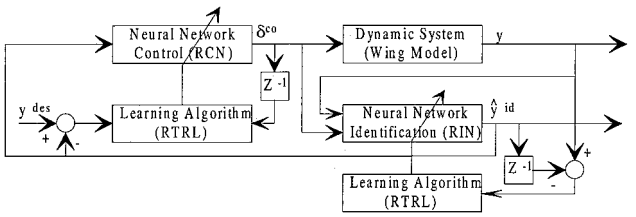

Fig. 4 Block diagram of adaptive active flutter suppression system.

Experimental Setup

The tests of the control system are carried out using an experimental setup (Fig. 5) composed of two personal computers, a 200-MHz Pentium dedicated to both of the neural networks and a 100-MHz Pentium implementing the servos. The use of two computers is due mainly to the workload generated by the online training of the RNN and partly to emulate realistic operating conditions with actuator servoing controlled independently. In this view, each computer is provided with a data acquisition card that also allows communication between the two personal computers using analog signals.

By the use of the networks tested during the numerical simulations, it is possible to reach a neural control frequency of 180 Hz, and the controls determined by the RCN are sent to the other Pentium at that rate. This control frequency is considered adequate for the flutter suppression because the unstable mode is the torsional one, at a frequency just above 8 Hz. The Pentium 100-MHz implements the PID digitally at a frequency of 400 Hz, reads the command input from the analog input channels of the card, and uses them to set the positions of the two control surfaces. Because the value calculated by the RCN is that at the $k + 1$ time step but is read shortly after step k, it is kept in memory and used only at the next loop of the servo's algorithm.[12]

Experimental Verification

After a preliminary verification of the whole system and the calibration of all of the parameters for the experimental setup, a verification of the correct operation of the RNN was made by investigating the filtering capability of the identifier and the stability of the controller with the penalty of the control output. The objectives of this experimental activity are as follows.

1) Verify the controller capability to increase the damping of the aeroservoelastic system at speed below the uncontrolled flutter onset, in presence of significant external disturbances represented by wind turbulence and control surfaces impulsive commands.

2) Demonstrate the effectiveness of the adaptive controller to suppress flutter at speeds higher than that of the uncontrolled system.

3) Demonstrate the adaptivity of the RNN to variation of the operational conditions, for example, speed variation.

All of these goals have been achieved, with an increase of the flutter speed from just above 25 up to 34.5 m/s. The initial weights of the two networks theoretically should not be influenced by the results because any test and any true application would start at zero speed and give the opportunity to train the networks with increasing airspeed. However, the limited size of the networks and the bounded learning rate, due to computer limitation and network stability problems, call for appropriate initial weights to maximize performances.

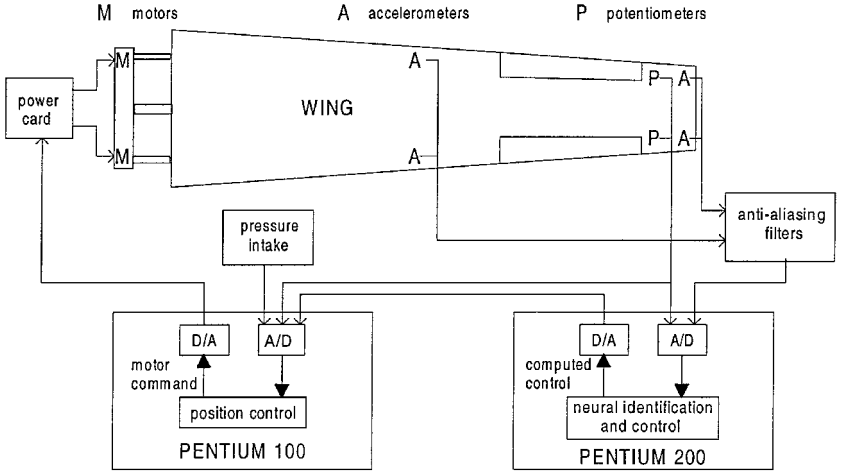

Fig. 5 Overall framework of the experimental setup.

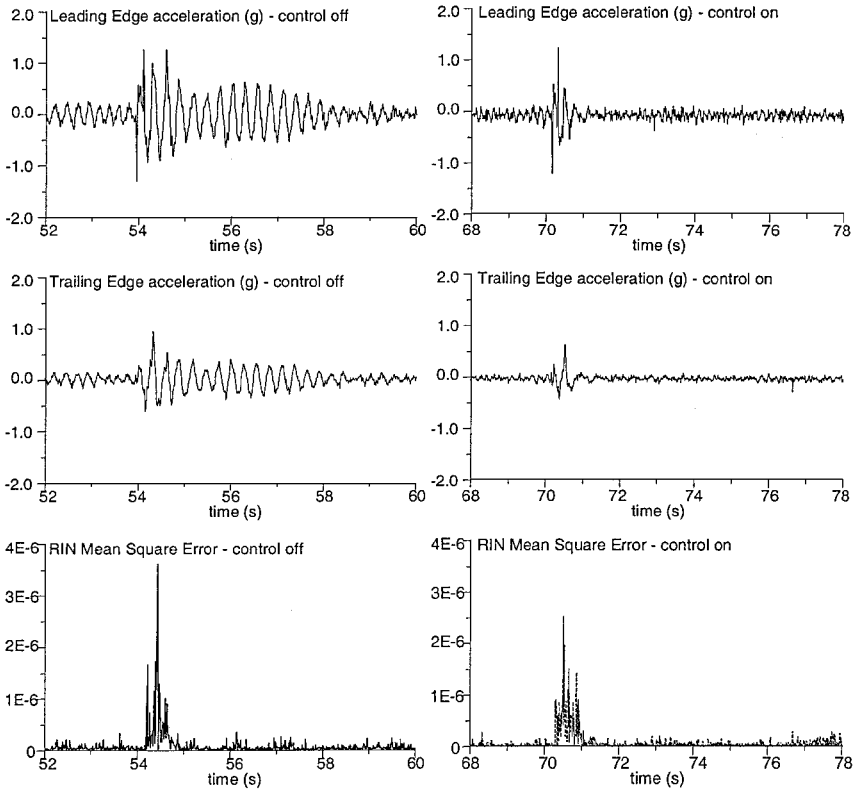

Fig. 6 Experiment at constant wind speed of 25 m/s.

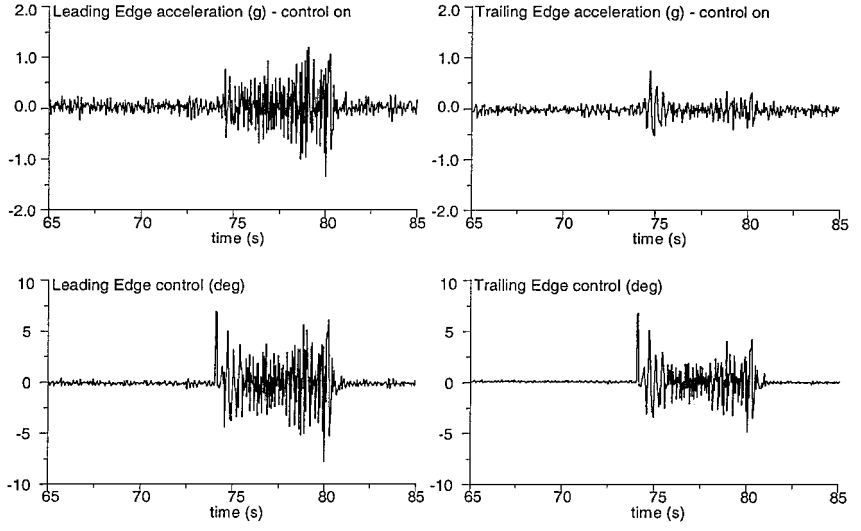

Fig. 7 Experiment at constant wind speed of 28.5 m/s.

For each experimental test, the two networks have been initialized with the weights determined during simulations at a constant speed of 25 m/s. This choice has proven to provide very good results regardless of the transient history of the airspeed.

Three significant test verifications are reported in Figs. 6–8. The capability to increase the damping of the system below the uncontrolled flutter speed is shown in Fig. 6, where a series of impulsive commands are applied to the system by deflecting each control surface by 6 deg, at a constant wind speed of 25 m/s. A first impulse is applied after 54 s, with the control off but the identification active, and a second impulse is applied after 70 s, when the previous transient is completely damped, with the identification active and the control on. The damping enhancement of the controlled acceleration with respect to the uncontrolled case is evident. Figure 6 reports also the RIN mean square error (the square of the difference between the identified and the real acceleration of the system). The system is always well identified, proving the suitable choice of network topology and initial weights, and the peaks in the error are always decreasing as more disturbances are applied to the wing.

Figure 7 reports the results of a test performed at a constant wind speed of 28.5 m/s, 15% higher than the uncontrolled flutter speed. The wind speed is progressively increased up to 28.5 m/s, with the control system already active and learning. Once the target speed is reached, a 6-deg impulse is commanded to both control surfaces. The system response is particularly interesting because it shows a fast adaptation to the new dynamic condition. The perturbation is completely damped in about 8 s. During the first 6 s, the system is basically controlling the trailing-edge acceleration and learning how to control the leading edge. Once the learning phase is completed, the wing motion is damped in about 2 s without excessive command power. This behavior is probably due to a less precise numerical modeling of the leading-edgecontrol effect in the numerical simulations, whose effects become relevant beyond the flutter speed and, therefore, deserve more time to be appropriately learned.

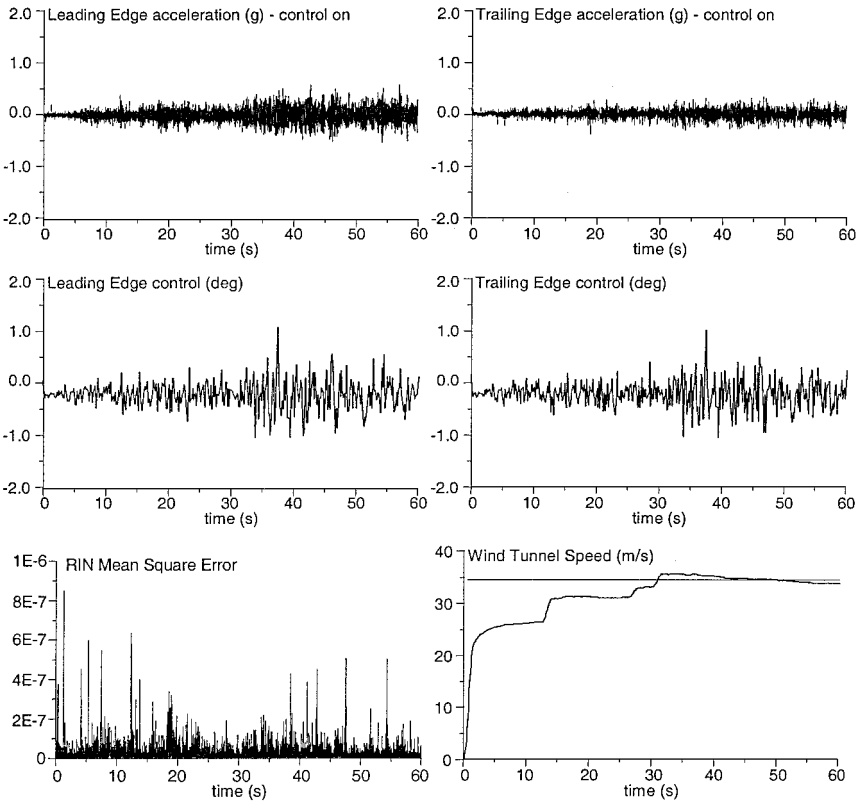

Fig. 8 Experiment at variable wind speed.

As a final example, in Fig. 8 it is possible to see the behavior of the neural controller when the wind speed is progressively increased up to 34.5 m/s. Once stabilized at the uncontrolled flutter speed, the airspeed is increased in three subsequent intervals with acceleration of about 5 m/s^2. The controller obtained in this work has shown a good capability to learn from the past perturbations, as proven by the mean square error of the RIN. Because this test follows the one presented in Fig. 6, the error is even lower, and its trend is toward a further decrease during the test even in presence of a perturbation, represented in this case by the wind-tunnel turbulence. As shown in Fig. 8, the control system is continuously working, but the control deflections are about 5% of the admissible ones even at the maximum speed, which would leave most of the control power still available for other tasks.

Comparing the results of the numerical simulations to those of the experiments, discrepancies were found in relation to the predicted robustness of the controller. In the simulations it was able to control a larger amount of perturbations, whereas during the tests the controller has not presented suitable control authority in the presence of wind-tunnel turbulence at speeds above 34 m/s. That may be due to two effects. First, the relevant noise present in the input signals to the RIN is larger than expected and larger than simulated in the network setup. Second, and most serious, the substantial change in the physical system at higher speeds is not compatible with the stability, which would require either a faster learning rate of the networks or a larger number of neurons, which at the moment is not compatible with the computational power available.

Conclusions

The feasibility of using neural networks for an adaptive flutter control has been verified, both for the identification tasks and for the control tasks. In view of the limitations imposed by the available hardware, the identification network carries out the first one with relative ease. The controller obtained has demonstrated correct behavior but some lack of robustness with respect to numerical simulations. This has somewhat limited its predicted operational capability. Nevertheless, a significant improvement (34%) of the flutter speed has been obtained, against varying operating conditions. The neural controller has proven to be applicable for the stabilization of unstable nonminimum phase systems, with some aspects that seem to make its application more appealing than other control techniques. It is worth continuing the study toward more robust algorithms and networks architecture for stabilization because they displayed a remarkable ease of use that seem apt to solve very complicated tasks. Furthermore, the suppression of flutter arising from sudden change in physical parameters should be investigated. This situation, due, for instance, to release of stores, can be more critical because of the very short training time.

References

[1]Andrighettoni, M., and Mantegazza, P., "Multi-Input/Multi-Output Adaptive Active Flutter Suppression for a Wing Model," *Journal of Aircraft*, Vol. 35, No. 3, 1998, pp. 462–469.

[2]Special Issue on Neural Networks in Control Systems, *IEEE Control Systems Magazine*, Vol. 10, No. 3, 1990.

[3]Levin, A. U., and Narendra, K. S., "Control of Nonlinear Dynamical Systems Using Neural Networks: Controllability and Stabilization," *IEEE Transactions on Neural Networks*, Vol. 4, No. 2, 1993, pp. 192–206.

[4]Levin, A. U., and Narendra, K. S., "Control of Nonlinear Dynamical Systems Using Neural Networks—Part II: Observability, Identification, and Control," *IEEE Transactions on Neural Networks*, Vol. 7, No. 1, 1996, pp. 30–42.

[5]Bernelli-Zazzera, F., and Lo-Rizzo, V., "Adaptive Control of Space Structures: An Experiment Based on Recurrent Neural Networks," *Proceedings of the Eleventh VPI&SU Symposium on Structural Dynamics and Control*, Virginia Polytechnic Inst. and State Univ. Press, Blacksburg, VA, 1997, pp. 605–614.

[6]Bernelli-Zazzera, F., and Lo-Rizzo, V., "Adaptive Control of Space Structures via Recurrent Neural Networks," *Dynamics and Control*, Vol. 9, No. 1, 1999, pp. 5–20.

[7]Nachtigal, C. L. (ed.), *Instrumentation and Control: Fundamentals and Applications*, Wiley, New York, 1990, pp. 487–535.

[8]"MSC/NASTRAN Handbook for Aeroelastic Analysis," MacNeal-Schwendler Corp., Los Angeles, 1987.

[9]Morino, L., Mastroddi, F., De Troia, R., Ghiringhelli, G. L., and Mantegazza, P., "Matrix Fraction Approach for Finite-State Aerodynamic Modeling," *AIAA Journal*, Vol. 33, No. 4, 1995, pp. 703–711.

[10]Haykin, S., *Neural Networks: A Comprehensive Foundation*, Maxwell Macmillan International, New York, 1994, pp. 121–235, 498–536.

[11]Isermann, R., Lachmann, K. H., and Matko, D., Adaptive Control Systems, Prentice–Hall, New York, 1992, pp. 189–242.

[12]Franklin, G. F., Powell, J. D., and Workman, M. C., *Digital Control of Dynamic Systems*, Addison Wesley, Reading, MA, 1992, pp. 133–237.