JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 16, No. 4, July–August 1993

Neurocontrol Design and Analysis for a Multivariable Aircraft Control Problem

Terry Troudet*
Sverdrup Technology, Inc., Brook Park, Ohio 44142
and
Sanjay Garg† and Walter Merrill‡
NASA Lewis Research Center, Cleveland, Ohio 44135

The feasibility of using artificial neural networks as control systems for modern, complex aerospace vehicles is investigated via an aircraft control design study. The problem considered is that of designing a controller for an integrated airframe/propulsion longitudinal dynamics model of a modern fighter aircraft to provide independent control of pitch rate and airspeed responses to pilot command inputs. An explicit model-following controller using H_∞ control design techniques is first designed to gain insight into the control problem as well as to provide a baseline for evaluation of the neurocontroller. Using the model of the desired dynamics as a command generator, a multilayer feedforward neural network is trained to control the vehicle model within the physical limitations of the actuator dynamics. This is achieved by minimizing an objective function that is a weighted sum of tracking errors and control input commands and rates. To gain insight into the neurocontrol design, linearized representations of the neurocontroller are analyzed along a commanded trajectory. Linear robustness analysis tools are then applied to the linearized neurocontroller models and to the baseline H_∞ based controller. Robustness issues of the neurocontrol design are identified and addressed in the context of neural training. Future areas of research are identified to enhance the practical applicability of neural networks to flight control design.

Introduction

IN the past few years, there has been an increasing interest within the control community in exploiting the promise of artificial neural networks to solve difficult control problems.[1,2] However, most of the neural network applications to control design that have appeared in the literature either dealt with robotic systems or with control problems that are mainly of academic interest, such as the inverted pendulum problem. Only more recently have neural networks been applied to the control design of more complex problems, e.g., manufacturing process[3] and aircraft systems.[4–6] The objective of this paper is to investigate the applicability of neural networks as controllers for aerospace vehicles with special emphasis on piloted flight. Toward this objective, results are presented from a preliminary study of neurocontrol design for an integrated airframe/propulsion model of a modern fighter aircraft for the piloted longitudinal landing task. To gain insight into the characteristics of the neurocontroller, linear analysis tools are applied to linearized representations of the neurocontroller and to a baseline H_∞ based controller. Closed-loop system performance and robustness of the neurocontroller are evaluated and discussed in relation to the H_∞ based controller.

The paper is organized as follows. The vehicle model and the desired closed-loop dynamics are first discussed, and an explicit model-following H_∞ based control design is presented. The architecture used to train the neurocontroller is then presented, and the results of the neurocontroller design are evaluated. A performance and robustness analysis is then presented for the neurocontroller and the H_∞ based controller.

Presented as Paper 91-2715 at the AIAA Guidance, Navigation, and Control Conference, New Orleans, LA, Aug. 12–14, 1991; received April 8, 1992; revision received Aug. 28, 1992; accepted for publication Sept. 4, 1992.

*Senior Research Engineer, Structures and Materials Department. Associate Member AIAA.
†Aerospace Engineer, Advanced Controls Technology Branch. Senior Member AIAA.
‡Chief, Advanced Controls Technology Branch. Senior Member AIAA.

Vehicle Model

The vehicle model consists of an integrated airframe and propulsion system state-space representation for a modern fighter aircraft powered by a two-spool turbofan engine and equipped with a two-dimensional thrust-vectoring and reversing nozzle.

The flight condition used in this application is representative of the short takeoff and landing (STOL) approach-to-landing task, with an airspeed of $V_0 = 120$ kt, a flight-path angle of $\gamma_0 = -3$ deg, and a pitch attitude of $\theta_0 = 7$ deg. The linearized dynamics of the vehicle model are of the form

$$\dot{\bar{x}} = A\bar{x} + B\bar{u}_a, \qquad \bar{z} = C\bar{x} \tag{1}$$

where the state vector is

$$\bar{x} = [u, w, Q, \theta, h, N2, N25, P6, T41B]^T \tag{2}$$

where u is the aircraft body axis forward velocity (ft/s), w the aircraft body axis vertical velocity (ft/s), Q the aircraft pitch rate (rad/s), θ the pitch angle (rad), h the altitude (ft), $N2$ the engine fan speed (rpm), $N25$ the core compressor speed (rpm), $P6$ the engine mixing plane pressure (psia), and $T41B$ the engine high-pressure turbine blade temperature (°R). The control input vector $\bar{u}_a$ is

$$\bar{u}_a = [WF, \delta TV]^T \tag{3}$$

where WF is the engine main burner fuel flow rate (lbm/h), and δTV the nozzle thrust vectoring angle (deg). The vehicle outputs to be controlled are

$$\bar{z} = [V, Q]^T \tag{4}$$

where V is the aircraft velocity (ft/s), and Q is the pitch rate (deg/s). The system matrices A, B, and C are available in

Table 1 Desired response transfer functions

Notation: $\{k(1/\tau)/[\xi;\ \omega_n] \equiv k(s+1/\tau)/(s^2+2\xi\omega_n s+\omega_n^2)\}$
$\frac{V_c}{V_{SEL}} = \frac{0.04(3.13)}{[0.89;\ 0.36]} : \frac{V_c}{Q_{SEL}} = 0$
$\frac{Q_c}{V_{SEL}} = 0; \quad \frac{Q_c}{Q_{SEL}} = \frac{35.12(0.5)}{[0.89;\ 2.24]}$

Ref. 7. The open-loop vehicle eigenvalues are as follows:
Airframe modes:

$$\lambda_1 = 0.07, \quad \lambda_{2,3} = -0.09 \pm j0.23, \quad \lambda_4 = 1.06, \quad \lambda_5 = -1.47$$

Propulsion modes:

$$\lambda_6 = -1.40, \quad \lambda_7 = -3.57, \quad \lambda_8 = -6.96, \quad \lambda_9 = -89.28$$

Note that the airframe is statically unstable with a highly unstable pitch mode. Open-loop analysis also indicated a strong coupling in the response of the controlled outputs $\bar{z}$ to control inputs $\bar{u}_a$.

The control design objective is to design a control system that provides decoupled command tracking of velocity and pitch rate from pilot control inputs with aircraft responses compatible with level I handling qualities requirements.[8] The desired response dynamics are selected to be of the form

$$\dot{\bar{x}}_m = A_m \bar{x}_m + B_m \bar{z}_{SEL}, \qquad \bar{z}_c = C_m \bar{x}_m \tag{5}$$

with $\bar{z}_{SEL} = [V_{SEL}, Q_{SEL}]^T$ where V_{SEL} is the pilot velocity command, Q_{SEL} is the pilot pitch rate command, and $\bar{z}_c = [V_c, Q_c]^T$, where the subscript c refers to the ideal response in V and Q. The system matrices A_m, B_m, and C_m are the state-space representation of the ideal response transfer functions listed in Table 1.

Actuator models were also used in control design and evaluation. The fuel-flow actuator was modeled as

$$G_{WF}(s) = \frac{10}{s+10} \cdot \frac{50}{s+50} \tag{6}$$

with a maximum fuel flow rate $|WF|_{\max} = 10{,}000$ lbm/h, and a rate limit $|\dot{WF}|_{\max} = 20{,}000$ lbm/h/s. The thrust-vectoring actuator was modeled as

$$G_{\delta TV}(s) = \frac{15}{s+15} \tag{7}$$

with a maximum thrust vector angle $|\delta TV|_{\max} = 10$ deg, and a rate limit $|\dot{\delta TV}|_{\max} = 20$ deg/s. As a result, *nonlinearities* appear in the control design and evaluation in the form of actuator position and rate limits.

H_∞ Control Design

Recent advances in H_∞ control theory[9] and computational algorithms to solve for H_∞ optimal control laws[10] have enabled the application of this theory to practical complex multivariable control design problems. Many example applications of H_∞ based control designs for aerospace vehicles have appeared in recent literature.[11-13] Before applying a neural network approach to control design for the example vehicle under study, an H_∞ based control law was obtained as a baseline for the performance and robustness analysis of the neurocontroller.

Within the framework of H_∞ optimization, the control design problem for this example study was formulated as the model-following problem shown in Fig. 1. The three transfer functions that are of interest for such a problem are the sensitivity function $S(s)$, the complementary sensitivity function $T(s)$, and the control transmission function $C(s)$. These represent the transfer functions from the reference commands to tracking errors, controlled variables, and commanded control inputs, respectively, i.e., $\bar{e}_z(s) = S(s)\bar{z}_c(s)$, $\bar{z}(s) = T(s)\bar{z}_c(s)$, and $\bar{u}_c(s) = C(s)\bar{z}_c(s)$. To be able to influence both the low-frequency and high-frequency properties of the closed-loop system, it is desirable to find a controller $K(s)$ that minimizes a weighted norm of a combination of these three transfer functions, i.e.,

$$\min \|H(j\omega)\|_\infty \quad \text{with } H(j\omega) = \begin{bmatrix} W_S(j\omega).S(j\omega) \\ W_T(j\omega).T(j\omega) \\ W_C(j\omega).C(j\omega) \end{bmatrix} \tag{8}$$

The weighting functions $W_S(j\omega)$, $W_T(j\omega)$, and $W_C(j\omega)$ are the "knobs" used by the control designer to "tune" the controller $K(s)$ such that the design objectives are met. For instance, choosing W_S to be large at low frequencies insures good command tracking performance, and choosing W_T to be large at high frequencies insures robustness to high-frequency unmodeled dynamics; W_C is chosen to insure that control actuation bandwidths, as well as rate and deflection limits, are not exceeded in the control design.

For the aircraft example, the integrated design model $P(s)$ in Fig. 1 consisted of the vehicle model (1) and the actuator models (6) and (7). The ideal response model $R(s)$ in Fig. 1 consisted of the desired model dynamics (5) with a high-pass filter $[s/(s+0.1)]$ on the pilot pitch rate command. This high-pass filter is added to reflect the fact that pitch rate cannot be commanded in steady state. The outputs $\bar{z}$ and the errors $\bar{e}_z$ were scaled by their approximate maximum values to be commanded by the pilot, with $V_c^0 = 20$ ft/s and $Q_c^0 = 3$ deg/s, to account for magnitude disparities due to differences in measurement units. The sensitivity weights W_S, the complementary sensitivity weights W_T, and the control weights W_C were chosen on the basis of the performance and robustness arguments discussed earlier, and they are listed in Ref. 14. Note that the combination of tracking errors $\bar{e}_z$ and aircraft outputs $\bar{z}$ is used as a controller input instead of $\bar{e}_z$ and ideal response $\bar{z}_c$ to avoid control saturation due to large pilot inputs and undue amplification of inadvertent pilot command noise.

The H_∞ control design plant as discussed earlier is of 21st order, consisting of the 9th-order aircraft model, 2nd-order WF actuator model, 1st-order δTV actuator model, 5th-order ideal response model, and 1st-order W_S and W_T for the two controlled variables. The resulting 21st-order H_∞ optimal controller obtained using the solution algorithm of Ref. 9 was reduced to 13th order by residualization of the high-order modes. The maximum eigenvalue of the reduced-order controller is $|\lambda|_{\max} = 6.83$ rad/s, which implies that the controller can be implemented digitally with reasonable sampling rates. With this reduced-order controller, the performance results in terms of closed-loop response and control requirements are shown in Fig. 2 for the following pilot command input: $V_{SEL} = 20$ ft/s for $t > 0$, $Q_{SEL} = 3$ deg/s for $0 < t \le 3$ s, and

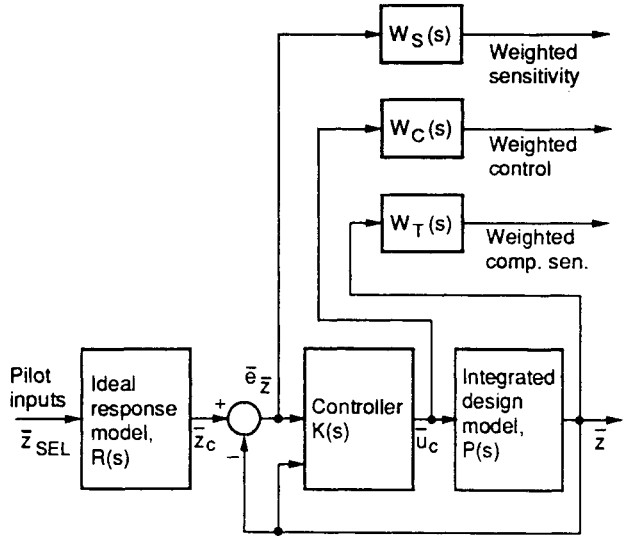

Fig. 1 Block diagram for the H_∞ control design.

$Q_{SEL}=0$ for $t>3$ s. This type of input command was chosen to illustrate the system performance because it is quite demanding in that the pilot is commanding the aircraft to pitch up as well as to accelerate to higher velocity. As shown in Fig. 2, the maximum fuel-flow rate is indeed commanded by the controller for an extended period of time to track the ideal response. The system performance is further illustrated and discussed in Ref. 14 by the closed-loop response to a less demanding pilot command input: $V_{SEL}=-20$ ft/s for $t>0$, $Q_{SEL}=3$ deg/s for $0<t\leq 3$ s, and $Q_{SEL}=0$ for $t>3$ s. The velocity response of the controller to the latter input command is quite close to the ideal response, and the control input commands and rates are reasonable. The pitch rate responses are quite similar for both types of input commands.[14]

Neurocontrol Design

Although the strength of neural networks lies in their ability to handle nonlinearities in the controlled dynamics, the control design for a linear aircraft model is being considered in this paper to gain insight into the neural network characteristics by using linear analysis tools. As discussed earlier, nonlinearities of concern for practical control design, such as actuator position and rate limits, are included in the design criteria.

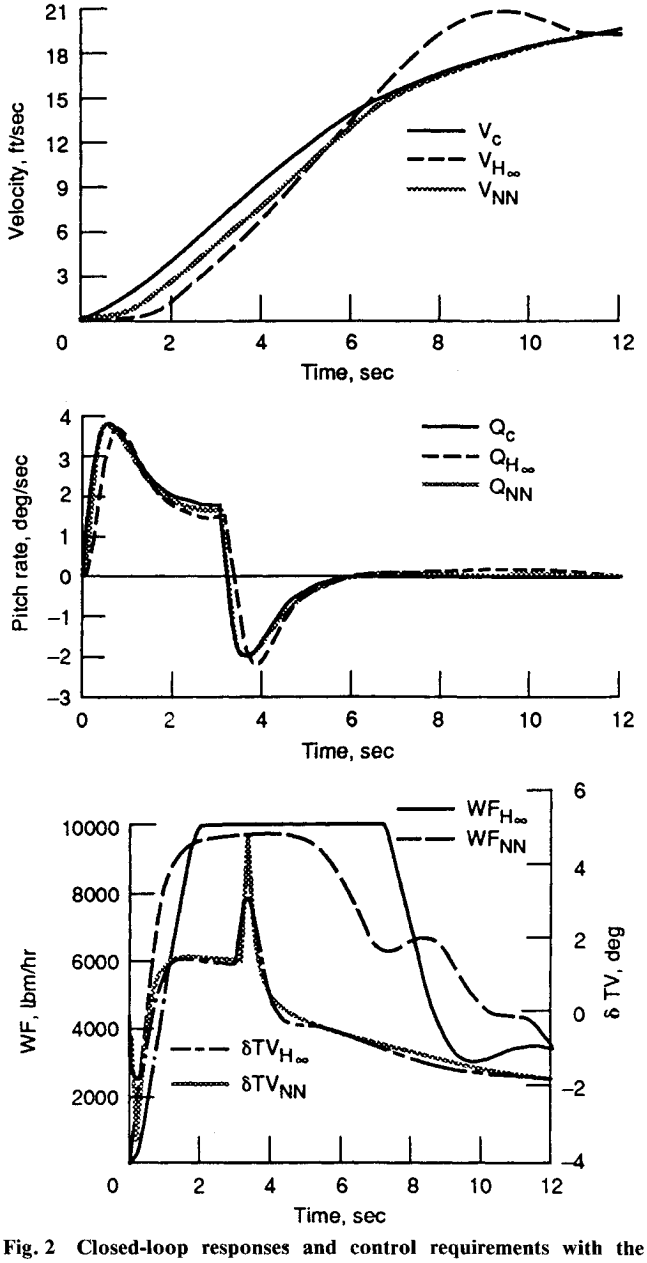

Fig. 2 Closed-loop responses and control requirements with the neurocontroller and the H_∞ reduced-order controller.

The architecture for training the neurocontroller is shown in detail in Fig. 3. For each pilot-selected trajectory $\bar{z}_{SEL}(t)$, a commanded trajectory $\bar{z}_c(t)$ is generated from Eq. (5). Before training, the commanded variables $\bar{z}_c(t)$ are sampled every $\delta t=0.02$ s and scaled to $\bar{z}_c^s(t_k)$ using the same scaling as for the H_∞ design. Likewise, the dynamics of the actuators and the vehicle model are discretized and scaled after normalizing the control input vector by its maximum value ($|WF|_{\max}, |\delta TV|_{\max}$). As for the H_∞ design, the *tracking error* at time t_k is the error between the scaled vehicle output vector and its desired scaled value at the same time t_k, i.e., $\bar{e}_z(t_k)=\bar{z}_c^s(t_k)-\bar{z}^s(t_k)$. However, because of the time discretization of the actuator dynamics and the vehicle model dynamics within the training loop, a commanded control input vector generated at time t_k by the neurocontroller will only affect the aircraft output at time t_{k+2}. Consequently, the tracking error at time t_{k+2} defines the magnitudes of the neurocontroller weight increments at time t_k. To be consistent with the time discretization of the dynamics, the internal representation of the neurocontroller was updated at time t_k on the basis of information becoming available only at a later time t_{k+2}. During training, knowledge of the anticipated commanded vehicle output at time t_{k+2}, $\bar{z}_c^s(t_{k+2})$, was therefore explicitly provided to the neural network at time t_k by means of the *commanded error* $\bar{\epsilon}_z(t_k)=\bar{z}_c^s(t_{k+2})-\bar{z}^s(t_k)$. This procedure insured that the proper action would be commanded by the neurocontroller at time t_k to achieve the desired tracking at time t_{k+2}. That no adverse effect on closed-loop performance results from this time discretization of the dynamics was demonstrated by the closed-loop evaluation of the neurocontroller operating in continuous time domain. These closed-loop evaluation results will be presented in the next subsection.

It is also noted that this *off-line* training scheme is amenable to forms that are more appropriate to on-line training by, for example, forwarding the desired dynamics filter of Fig. 3 with a neural time-series predictor that would have been previously trained to predict the future filtered command $\bar{z}_s^c(t_{k+2})$ on the basis of only present and past values of $\bar{z}_s^c(t)$, i.e., $t\leq t_k$. Another approach is to substitute $\bar{\epsilon}_z(t_k)$ by $\bar{e}_z(t_k)$ in Fig. 3, which amounts to training the neurocontroller to track the commanded trajectory with a small delay of two time steps, yet on the basis of information that is instantaneously available. The latter training procedure was found to lead to the same closed-loop neurocontrol performance as the training procedure of Fig. 3. In spite of this potential for growth, it is essential to realize that the training architecture of Fig. 3 was designed for an off-line synthesis of the neurocontrol and that the full analysis of the feasibility of an on-line neurocontrol synthesis would require addressing important issues such as on-line identification of the vehicle model, global/local learning capabilities of the backpropagation training algorithm, adaptive selection of the training parameters such as the steepest descent coefficient(s), computational performance of present neurohardware/neurosoftware, etc. Such issues lie beyond the primary scope of this work, and they are the object of ongoing research throughout the neurocontrol community.

As shown in Fig. 3, the two commanded control inputs are calculated by a two hidden-layer feedforward neural network with eight input units (four pairs of input units associated to the Q and V variables) and two neurons in the output layer. These pairs consist of the scaled output vector $\bar{z}^s(t_k)$, the commanded error $\bar{\epsilon}_z(t_k)$ between the scaled vehicle output vector at time t_k and its desired scaled value at time t_{k+2}, the discrete time derivative of the tracking error $\dot{\bar{e}}_z(t_k)$, and the time average of the tracking error $1/t_k\int_0^{t_k}\bar{e}_z(t)\,dt$. The structure of the neurocontroller is therefore that of a nonlinear multivariable output error "PID" (proportional + integral + derivative) and nonlinear output feedback controller. As in the H_∞ design, the motivation behind using the combination of $\bar{z}^s(t_k)$ and $\bar{\epsilon}_z(t_k)$ as inputs to the neurocontroller, instead of $\bar{z}^s(t_k)$ and $\bar{z}_c^s(t_{k+2})$, is to allow the neural network to reconstruct the command without direct feedforward of the command. The role of the error rates $\dot{\bar{e}}_z(t_k)$ is to provide the neural network

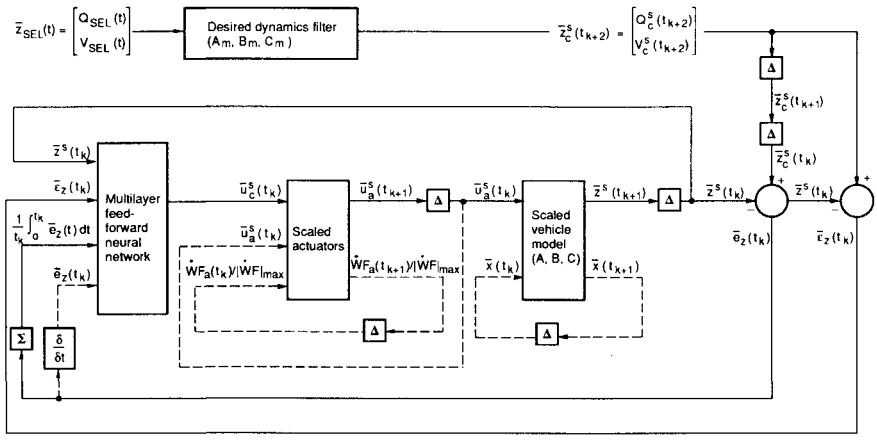

Fig. 3 Training architecture.

with lead information, and the time-averaged error feedback $1/t_k \int_0^{t_k} \bar{e}_z(t)\,dt$ is to minimize the steady-state tracking error for step command inputs. The motivation behind scaling the integral error $\int_0^{t_k} \bar{e}_z(t)\,dt$ into its time average is to improve backpropagation learning by bounding the corresponding input to the neural network. In the case of long trajectories, however, the proposed *uniform* time averaging may decrease the sensitivity to the recent tracking errors in comparison with the initial tracking errors. A *weighted* time averaging of the type $\int_0^{t_k} W(t)\bar{e}_z(t)\,dt / \int_0^{t_k} W(t)\,dt$, with $W(t) = (t/t_k)^{1/\lambda}$ and $1 > \lambda > 0$, would be more sensitive to recent events. Other alternatives would be to low-pass filter the integral error itself or to remove the scaling factor $1/t_k$ from the uniform time-averaged error as learning takes place. Because of their potential to improve steady-state tracking, such approaches should be considered in future neurocontrol designs.

In Fig. 3, the symbol Δ represents a latch that is clocked every δt seconds to update the inputs to the neurocontroller, the actuators, and the vehicle model. A network configuration of 15 neurons in the first hidden layer and 10 neurons in the second hidden layer is chosen for the neurocontroller. Each neuron of the neurocontroller has the activation function:

$$y = \tanh(x) \tag{9}$$

which limits its output y to the interval $[-1, +1]$ for any input signal x. For a given set of weights of the neural network, the two output neurons yield the normalized commanded control input vector

$$\bar{u}_c^s(t_k) = \left[WF_c/|WF|_{\max},\quad \delta TV_c/|\delta TV|_{\max}\right]^T \tag{10}$$

which is applied to the scaled actuators. After a small time interval $\delta t = t_{k+1} - t_k$, the actuators yield the normalized actuator control output vector $\bar{u}_a^s(t_{k+1})$ as defined by Eqs. (6) and (7). The normalized actuator control output vector $\bar{u}_a^s(t_{k+1})$ is subsequently applied as input to the scaled vehicle model over the time interval $[t_{k+1}, t_{k+2}]$ and changes the state vector of the vehicle model from $\bar{x}(t_{k+1})$ to $\bar{x}(t_{k+2})$.

To maximize the tracking performance while minimizing the costs associated with high control effort and high control rate requirements, the neural network is trained to minimize an objective function that includes tracking errors, control effort, and control rate requirements

$$J(t_k) = \tfrac{1}{2}\left[\bar{e}_z^T(t_{k+2})\cdot\bar{\bar{\rho}}\cdot\bar{e}_z(t_{k+2}) + \bar{u}_a^{s\,T}(t_{k+1})\cdot\bar{\bar{\lambda}}\cdot\bar{u}_a^s(t_{k+1}) + \dot{\bar{u}}_a^{s\,T}(t_{k+1})\cdot\bar{\bar{\mu}}\cdot\dot{\bar{u}}_a^s(t_{k+1})\right] \tag{11}$$

where $\bar{e}_z(t_{k+2})$ is the error between the scaled commanded vector $\bar{z}_c^s(t_{k+2})$ and the scaled vehicle output $\bar{z}^s(t_{k+2})$. The matrices $\bar{\bar{\rho}}$, $\bar{\bar{\lambda}}$, and $\bar{\bar{\mu}}$ are 2×2 diagonal matrices whose coefficients can be adapted so as to modify the characteristics of the neurocontroller to achieve a practical performance/control-effort tradeoff. Expression (11) is of the same form as the objective function used in Ref. 15 to design a neurocontroller for the same airframe/propulsion system, but without simulating the actuator dynamics within the training loop. In Ref. 15, it was found that training the neural network to minimize only the tracking error led to high control effort and high control rate requirements. When the actuator dynamics were included in the closed-loop evaluation, this resulted in a highly oscillatory pitch rate response and a limit cycle behavior in velocity/fuel-flow response. However, a satisfactory tradeoff between tracking performance and control effort could be achieved with finite values of $\bar{\bar{\lambda}}$ and $\bar{\bar{\mu}}$. Since the bandwidth limiting effect of the actuators is now explicitly taken into account within the training loop, much improvement in performance/control-effort tradeoff is expected from the minimization of Eq. (11).

The backpropagation algorithm[16] was used to find the set of weights of the neurocontroller that minimize the objective function of Eq. (11) over the set of pilot input commands. To backpropagate Eq. (11), the linear dynamics of the vehicle model were time discretized and emulated as a linear system of 11 inputs (i.e., the two normalized control outputs of the actuators and the nine state variables of the vehicle model) and 9 outputs (i.e., the nine state variables of the vehicle model). The nonlinear dynamics of both actuators were time discretized and emulated as two nonlinear neural networks. The second-order dynamics of the fuel-flow actuator were simulated by a three-layer network of linear and *linear-thresholding* neurons. Constraining fuel-flow effort and fuel-flow rate requirements was achieved by thresholding the linear neurons of the two last layers.[14] The first-order dynamics of the thrust-vectoring actuator were simulated by a two-layer neural network of linear-thresholding neurons.[14] Constraining the effort and rate requirements of the thrust-vectoring actuator is also achieved by means of linear-thresholding neurons.

The layers of an $(N+1)$-layer neural network can be labeled by an index p from 0 to N, $p=0$ denoting the input layer. Layer p has $\nu(p)$ elements consisting of $[\nu(p)-1]$ neurons and one unit that is permanently "on" and used to define the thresholds of the neurons of the $(p+1)$th layer. With symmetric activation functions of the type of Eq. (9), the threshold of a neuron is defined as the value of its input signal above which its output is positive and below which its output is negative. During training, the thresholds are updated with backpropagation in a manner similar to the updating of the weights.[16]

The weight connecting the ith neuron of the pth layer to the jth neuron of the $(p+1)$th layer is denoted as $w_{j,(p+1);i,p}$. The threshold of the jth neuron of the $(p+1)$th layer thus corresponds to $w_{j,(p+1);\nu(p),p}$. For a single feedforward pass of the neural network, a weight increment is given by

$$\delta w_{j,(p+1);i,p} = \alpha o_{i,p}\Delta_{j,(p+1)} \tag{12}$$

where α is the steepest descent coefficient, $o_{i,p}$ is the output of the ith neuron of the pth layer, and $\Delta_{j,(p+1)}$ is the effective error at the output of the jth neuron of the $(p+1)$th layer. The effective errors $\Delta_{k,(p+2)}$ in the $(p+2)$th layer are backpropagated to the $(p+1)$th hidden layer to give the effective errors in the $(p+1)$th layer, as

$$\Delta_{j,(p+1)} = f'[x_{j,(p+1)}] \times S_{j,(p+1)}$$

$$S_{j,(p+1)} = \left[\sum_{k=1}^{\nu(p+2)} \Delta_{k,(p+2)} w_{k,(p+2);j,(p+1)} \right] \tag{13}$$

where $f'[x_{j,(p+1)}]$ is the value of the derivative of the neural activation function for the net input $x_{j,(p+1)} = \Sigma_{k=1}^{k=\nu(p)} \times w_{j,(p+1);k,p} o_{k,p}$ of the jth neuron in the $(p+1)$th layer. In the output layer, the effective errors $\Delta_{j,N}$ are the gradients of the objective function of Eq. (11)

$$\Delta_{j,N} = -f'(x_{j,N}) \frac{\partial J}{\partial o_{j,N}} \tag{14}$$

Whenever the neural activation is not differentiable over the range of all possible neuron input values (as is the case for the linear-thresholding neurons used for emulating the actuators), f' should be constructed to preserve the characteristics of a monotonous continuous function. For example, the linear-thresholding activation function that is defined as

$$f_{l\text{th}}(x) = x \quad \text{if} \quad |x| \le 1, \quad \text{and} \quad f_{l\text{th}}(x) = x/|x| \quad \text{if} \quad |x| > 1 \tag{15}$$

is clearly not differentiable over $[-\infty, +\infty]$. Since $f_{l\text{th}}$ is piecewise differentiable, it would seem a priori natural to define $f_{l\text{th}}'$ as $f_{l\text{th}}'(x) = 1$ if $|x| \le 1$, and $f_{l\text{th}}'(x) = 0$ if $|x| > 1$. With this definition of $f_{l\text{th}}'$, however, any time the net input x_0 of a neuron would take a value outside of $[-1, +1]$ during training, the neuron output would remain trapped to 1, if $x_0 > 1$, or -1, if $x_0 < -1$. For such neuron input values, the weights of the incoming connections would remain frozen, and this would bias the learning. To permit the neurons full access to the output state space during training, $f_{l\text{th}}'$ is thus defined as

$$f_{l\text{th}}'(x_{i,p}) = 1 \quad \text{if} \quad \left(|x_{i,p}| \le 1 \quad \text{or} \quad x_{i,p} \cdot S_{i,p} < 0\right)$$

$$\text{and} \quad f_{l\text{th}}'(x_{i,p}) = 0 \quad \text{otherwise} \tag{16}$$

which will insure that the weights will be properly incremented during training. The $S_{i,p}$ that appears in Eq. (16) is defined as in Eq. (13). The serial arrangement of the neurocontroller, the neuro-emulators of the actuators, and the neuro-emulator of the vehicle model constitutes a larger neural network through which the objective functions $J(t_k)$, Eq. (11), can be backpropagated through time[1] using Eqs. (13–16). The interconnections between neurocontroller and neuro-emulators that were used as backpropagating channels can be found in Ref. 14.

The commanded trajectories used to train the neural network were generated as follows. The pilot-selected pitch rate was a doublet centered at a time t_c randomly selected between 2.5 and 5 s, with the following characteristics: $Q_{SEL}(t) = Q_0$ for $t \le t_c$, $Q_{SEL}(t) = -Q_0$ for $2t_c \ge t > t_c$, and $Q_{SEL}(t) = 0$ for $t > 2t_c$. The pilot-selected airframe velocity was a step function characterized by $V_{SEL}(t) = 0$ for $t \le 0$ and $V_{SEL}(t) = V_0$ for $t > 0$. The maximum intensities $|Q_0|$ and $|V_0|$ of the randomly selected input commands were bounded by $Q_{\max} = 3$ deg/s and $V_{\max} = 20$ ft/s. Random sets of input trajectories were generated from uniform distributions of Q_0, V_0, and t_c over $[-Q_{\max}, Q_{\max}]$, $[-V_{\max}, V_{\max}]$, and [2.5 s, 5 s], respectively. The commanded variables $Q_c(t)$ and $V_c(t)$ were filtered from $Q_{SEL}(t)$ and $V_{SEL}(t)$ over a period of 12 s with a time step $\delta t = 0.02$ s. These types of commanded trajectories represent typical pilot command inputs.

Training the neurocontroller with the backpropagation algorithm requires knowledge of the gradient of the global objective function that is to be minimized, i.e., the gradient of the infinite sum of all of the time integrals $\int_0^{12} J_c(t)\, dt$, where $J_c(t)$ is the generic expression of Eq. (11) for every possible commanded trajectory "c" already defined. In view of the impossibility/impracticality of calculating the gradient of the global objective function, the following computationally effective alternative, which is based on *estimates* of the true gradient, was used to train the neurocontroller. Since the weights are initially randomly distributed, *gross* estimates of the true gradient are sufficient to enable learning at the beginning of the training. In this gross-tuning phase of the training, the synaptic weights were updated after calculating the gradient of $J_c(t_k)$ at every time $t_k = k\delta t$ between 0 and 12 s, for every commanded trajectory that was randomly generated. The training data set consisted of 4000 random commanded trajectories, and the steepest descent coefficient was $\alpha = 0.001$. Although this gross-tuning phase is computationally efficient in initiating the learning, successive weight adjustments based on single-iteration gradients do not average out within the precision required to meet the control objectives, even for very small values of the steepest descent coefficient. As a result, more accurate estimates of the true gradient become necessary to find a set of weights that provide the neurocontroller with a satisfactory closed-loop performance. In this next fine-tuning phase of the training, the weights were therefore updated according to the following moving window scheme based on multi-iteration gradients. At every time t_k sampled over a period of 12 s, the true gradient was approximated by that of the time integral $\int_{t_k}^{t_k + n_w \delta t} J_c(t)\, dt$ corresponding to a random commanded trajectory "c", n_w being the width of the time window. To improve learning, the width of the time window was progressively increased from a single point, i.e., $n_w = 1$, to an entire commanded trajectory, i.e., $n_w = 12\text{ s}/0.02\text{ s} = 600$, whereas the steepest descent coefficient was progressively reduced from its initial value of 0.001 to 0.0001. The motivation behind reducing the steepest descent coefficient in conjunction with increasing the time window is to provide *small* successive batched weight adjustments that average out into a consistent direction toward a local minimum and within the accuracy needed to achieve the desired control objectives.

Neurocontrol Performance

The neurocontroller was tested in closed loop on *pulse* pitch rate input commands, different from the *doublets* used in training. The input commands chosen to illustrate the neurocontrol performance were defined by the step pitch rate command $Q_{SEL}(t) = 3$ deg/s for $t \le 3$ s and $Q_{SEL}(t) = 0$ for $t > 3$ s, applied simultaneously with the step velocity command $V_{SEL}(t > 0) = 20$ ft/s. Additional results concerning the closed-loop response with the same pitch rate command but with the step velocity command $V_{SEL}(t > 0) = -20$ ft/s are given in Ref. 14. When training the neural network without giving any consideration to the cost associated with large control efforts and large control rates, i.e., $\bar{\bar{\lambda}} = \bar{\bar{\mu}} = \bar{\bar{0}}$ in Eq. (11), the neurocontroller learns very satisfactorily to track the commanded outputs. However, the fuel flow is quite irregular, and both control input commands generated by the neurocontroller ride the actuator rate limits. A study of the tradeoff between tracking performance and control effort requirement was conducted by training the neural network with $\bar{\bar{\lambda}}$ and $\bar{\bar{\mu}}$ of the form $\bar{\bar{\lambda}} = \text{diag}[\lambda_{WF}, \lambda_{\delta TV}]$ and $\bar{\bar{\mu}} = \text{diag}[\mu_{WF}, \mu_{\delta TV}]$, with the same training characteristics and the same matrix elements of $\bar{\bar{\rho}} = \text{diag}[\rho_V, \rho_Q]$ used earlier to minimize the tracking errors.

The results of this tradeoff study are shown in Fig. 2 for the same pilot input commands used to illustrate the closed-loop performance of the H_∞ controller, with the choice of parameters $\bar{\bar{\rho}} = \text{diag}[\rho_V, \rho_Q] = \text{diag}[2000, 20]$, $\bar{\bar{\lambda}} = 0.01\bar{\bar{I}}$, and $\bar{\bar{\mu}} = 0.1\bar{\bar{I}}$. The pitch rate response follows the commanded trajectory very smoothly, in spite of the thrust-vectoring requirement δTV reaching the actuator rate limit at the initiation and

at the end of the command. However, within the proposed training scheme, any attempt to lower the rate of thrust vectoring by increasing $\mu_{\delta TV}$ resulted in a loss of tracking performance. Neurocontrol tracking is still very satisfactory in pitch rate response but is slightly less satisfactory in velocity response due to the physically demanding effort of simultaneously increasing aircraft speed and pitch attitude.

To estimate the effect of providing the neurocontroller with lead information during training, the previous process was repeated without feeding the discrete time derivative of the tracking error, i.e., $\dot{e}_z(t_k)$, to the neural network during training. Without constraining control efforts and rates ($\bar{\lambda} = \bar{\mu} = \bar{0}$), the tracking performance deteriorated significantly with the appearance of some ringing in the pitch rate response and a limit cycle behavior in the velocity/fuel-flow response. The fuel-flow requirement and fuel-flow rate were both much more oscillatory than when lead information was provided to the neurocontroller during training. The fuel-flow rate oscillated between the maximum and minimum rate limit during and beyond the 12-s training period. A more oscillatory behavior was also noted for the control effort and rate of the thrust vectoring. However, the situation improved significantly when constraints on control efforts and rates were applied during training. In this case, a satisfactory tradeoff between performance and control effort was reached for values of $\bar{\lambda}$ and $\bar{\mu}$ in the vicinity of $\lambda_{WF} = \lambda_{\delta TV} = 0.02$, $\mu_{WF} = 0.2$, and $\mu_{\delta TV} = 1.0$. The results showed a similar velocity/fuel-flow response with and without lead information but showed a noticeable degradation in the pitch-rate/thrust-vectoring response in comparison with the situation where lead information was provided to the neurocontroller. This degradation in tracking performance resulted from the large value of the pitch-rate constraint $\mu_{\delta TV}$ (one order of magnitude larger than before), which was needed to decrease the tracking overshoots. In summary, lead information enabled the neurocontroller to overcome ringing and limit cycle behavior while increasing tracking performance. Thus, within the present scheme of neural computation, any dynamic characteristics required to achieve desirable performance had to be incorporated into the neural network with an appropriate choice of inputs. An extension of the present neural architecture to generate such dynamic characteristics could be a *feedforward* neural network with *intermediate feedback inputs*, i.e., a recurrent neural architecture as a dynamic neurocontroller.

Analysis of the Controllers

From a comparison of the closed-loop response with the H_∞ based reduced-order controller and the neurocontroller (Fig. 2 and Ref. 14), it is evident that the neurocontroller, as designed, provides improved command tracking, although at the expense of increased control rate activity, both for $\delta\dot{T}V$ and $\dot{W}F$. Also, the pitch vectoring control requirements are higher and the fuel-flow activity exhibits oscillatory behavior for the neurocontroller.

Note that the results presented so far have been with the nominal vehicle model used for control design. Since this model is only a simplified version of the vehicle dynamics, an important criterion for the design of controllers for flight vehicles is that of robustness. Robustness is defined here as maintaining performance and stability in the presence of uncertainties associated with the modeling process. Modeling uncertainties are due to neglected high-order dynamics, parameter changes due to change in flight conditions, and the margin of error associated with estimating model parameters based on analytical tools and experimental data. A classic specification for robustness, also used in the military specifications for design of flight control systems,[8] is that of stability margins, specifically gain and phase margins.[18] The tools to determine these margins are fairly well developed for linear systems: classical Bode analysis for single-input/single-output systems[17] and modern singular value and structured singular value analysis for multi-input/multi-output systems.[18,19] For nonlinear systems, one way to determine robustness is to conduct Monte Carlo type simulations using all possible combinations of modeling uncertainties that can be expected. Another approach is to linearize the closed-loop system at various points along a given trajectory and then apply the linear analysis tools. The latter approach is less time consuming and provides more insight into the characteristics of the nonlinear system. Furthermore, this latter approach allows a similar analysis to be performed for the linear H_∞ based reduced-order controller and the nonlinear neurocontroller for small perturbations along a given trajectory. However, and as will be further demonstrated in this section, it is clear that linear analysis tools cannot provide a full description of the nonlinear neurocontroller and that further theoretical advances are needed to analyze the stability and robustness of such nonlinear systems.

Since the vehicle model used in this analysis is linear, only linear small perturbation models of the neurocontroller at different points along a given trajectory are needed to perform the type of robustness analysis discussed earlier. Considering the closed-loop system response with the neurocontroller, as illustrated in Fig. 2, the linear neurocontroller models were generated at times $t = 0.5$, 2, 4, 6, 8, and 10 s. The first three points in time correspond to transient control activity, whereas the last three represent steady-state type command tracking with monotonically decreasing tracking error. Note that in closed-loop evaluation the inputs to the neurocontroller consist of the four sets of scaled data values: the time-averaged errors $1/t\int_0^t \bar{e}_z(\tau)\,d\tau$, the error rates $\dot{\bar{e}}_z(t)$, the errors $\bar{e}_z(t)$, and the controlled outputs $\bar{z}^s(t)$. The scaling, the time averaging, and the differentiation of the input error were embedded within the neurocontroller during the linearization process to be input/output-wise compatible with the H_∞ based controller that has only the errors $\bar{e}_z$ and the controlled outputs $\bar{z}$ as inputs. This was done by introducing an integrator block, a gain block, and a differentiator block between the $\bar{e}_z$ inputs and the neurocontroller and by defining the modified neurocontroller as consisting of these three blocks and the neurocontroller as designed. The frequency response Bode plots of the linearized neurocontroller models were obtained to gain insight into the characteristics of the control action. Bode gain plots for the thrust-vectoring angle (δTV) response to all of the inputs to the neurocontroller linearized at $t = 0.5$ s (500 ms) are shown in Fig. 4. The corresponding Bode gain plots for the H_∞ based controller are shown in Fig. 5. The results shown in Figs. 4 and 5 will be discussed in detail later in this section. Variations in the linearized neurocontroller characteristics with the change in magnitude of the inputs to the controller along the trajectory were also analyzed. The comparison of Bode gain plots for pitch rate error e_Q to thrust-vectoring angle δTV response for the six linearized neurocontrollers given in Ref. 14 indicated that the neurocontroller gains decrease with time. This type of behavior was exhibited by all of the other

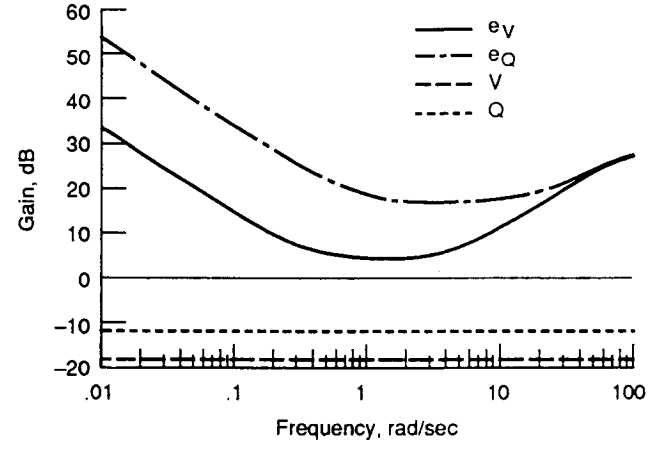

Fig. 4 Bode gain plots for the neurocontroller linearized at $t = 500$ ms, with all inputs to δTV.

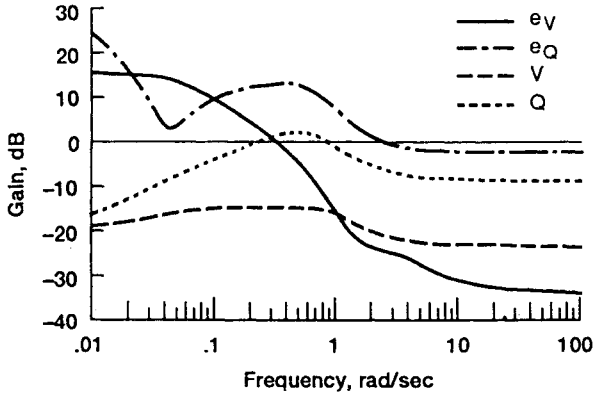

Fig. 5 Bode gain plots for the H_∞ reduced-order controller, with all inputs to δTV.

input/output Bode plots of the linearized neurocontroller models. So, in effect, the neurocontroller can be thought of as a set of linear controllers with the controller parameters being a strong function of the magnitude and direction (relative magnitude) of the inputs to the controller. Note that since the H_∞ based controller is linear, its dynamics are independent of the magnitudes of the controller inputs.

From Fig. 4 we note that the neurocontroller exhibits PID control-type behavior from the error inputs e_V and e_Q to the thrust-vectoring angle δTV output. This was also the case for the e_V and e_Q to WF response and was true all along the trajectory as illustrated in Ref. 14 for e_Q input. This dynamic behavior of the neurocontroller for the error inputs is directly due to allowing feedback of the integral and derivative errors. Since no such dynamics were added to the feedback of V and Q to the neurocontroller, the neurocontroller exhibits only proportional-type behavior from these inputs.

Comparing Figs. 4 and 5, we first note that the magnitude of the e_V and e_Q to δTV response is much lower for the H_∞ based controller compared with the particular linearized neurocontroller models. This was also true for the error inputs to WF response. This result is a further confirmation that the control effort and control rate requirements to track a given set of commands will be higher for the neurocontroller. It is noted that the structural complexity of the linear H_∞ based controller lies in its high-order dynamics, whereas that of the neurocontroller results from its nonlinearity. From Fig. 5, some integral and derivative action is evident in the e_Q to δTV response for the H_∞ controller. The integral action was built into the H_∞ based controller through the choice of the sensitivity weighting; unlike for the neurocontrol design, however, the error rate information was not explicitly provided in the H_∞ controller. The H_∞ control synthesis procedure is such that it naturally builds in the amount of lead (error rate) information into the controller that is necessary to meet the control design objectives specified through the weighted quantities. As evident from Figs. 4 and 5, the H_∞ based controller provides lead at a lower frequency in the e_Q to δTV response as compared with the linearized neurocontroller.

Another difference between the H_∞ based controller and the neurocontroller is the compensation from the measurements of the controlled plant outputs (V and Q) to the control inputs (WF and δTV). As mentioned earlier, this compensation is a "constant" (varying with input magnitude) gain from the controller inputs to outputs for the linearized neurocontroller. However, as seen from Fig. 5, the H_∞ based controller has dynamics associated with this part of the control compensation and also has higher compensation gains than the linearized neurocontroller (Fig. 4). The controller structure used for the H_∞ and the neurocontrol design is consistent with the classical approach of flight control design wherein an inner loop compensation ($\bar{z} \rightarrow \bar{u}$) is designed first to provide stability augmentation and place the augmented plant dynamics within the handling qualities specifications, and then the outer loop compensation ($\bar{e}_z \rightarrow \bar{u}$) is designed to provide decoupled command tracking to reduce pilot workload. The significance of the difference between the H_∞ based controller and neurocontroller "inner loop" compensation was studied further by considering failures in the outer compensation loops, i.e., failures in the error loops. Eigenvalue analysis showed that the closed-loop system with H_∞ based controller will remain stable for failures in any or both of the error loops, whereas the closed-loop system with the neurocontroller linearized at $t = 500$ ms was *predicted* (by linear analysis) to be unstable for failure in either or both of the error loops. The velocity response of the closed-loop system with failure in the e_Q loop is illustrated in Fig. 6 for both the H_∞ based controller and the nonlinear neurocontroller. With e_Q error loop failure the H_∞ based controller still tracks the velocity command as shown in Fig. 6 and provides stable response in pitch rate as shown in Ref. 14, whereas the neurocontroller gives a highly unstable response as predicted by linear analysis. With e_V error loop failure, however, closed-loop system simulations indicated that the nonlinear neurocontroller still tracks the pitch-rate command with a bounded velocity response.

Therefore, the H_∞ based controller and the neurocontroller are using the velocity measurements V in a manner consistent with the classical idea of providing inner loop plant augmentation. How to formulate the neurocontrol design problem such that the resulting controller exploits pitch-rate measurement information Q to provide inner loop stability augmentation will be discussed in the next section.

Stability margin analysis was performed for the linearized neurocontroller models and the H_∞ based controller to quantify robustness of the control designs. Among the linearized neurocontroller models, stability margins were worst for the one linearized around $t = 500$ ms, and so only those results are discussed here. Structured singular value analysis[20] showed

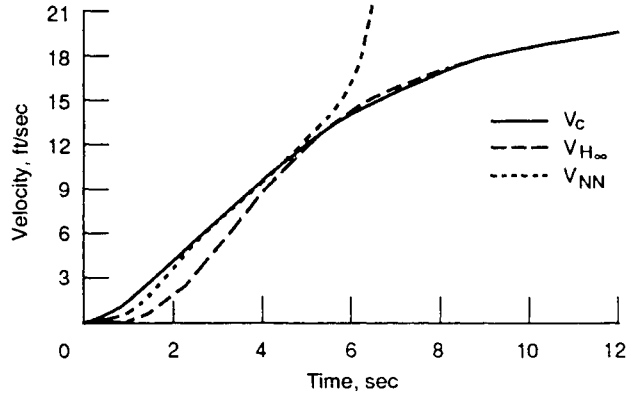

Fig. 6 Closed-loop velocity response with e_Q error loop failure.

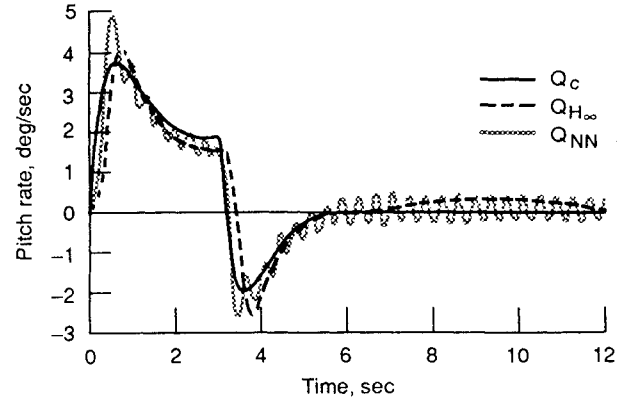

Fig. 7 Closed-loop responses with a time delay of 50 ms in both neurocontrol channels.

that the H_∞ based controller has guaranteed multivariable gain margins of -3.7 to 6.6 dB (gain factor of 0.65 to 2.1) and phase margins of ± 30 deg for simultaneous loop gain or phase changes at the plant output (V and Q) and margins of -3.8 to 7.2 dB and ± 32.5 deg at the plant input (WF and δTV). For the linearized neurocontroller, these multivariable margins were only -0.6 to 0.6 dB and ± 3.4 deg for loop gain/phase variations at the plant output and -0.9 to 1.1 dB and ± 6.6 deg at the plant input. Since the multivariable margins can sometimes be conservative, the stability robustness of the closed-loop system was further evaluated using the more classical approach of "breaking" one loop at a time, i.e., one loop open and other loops closed. This one-loop-at-a-time analysis confirmed the poor stability margins of the neurocontroller as trained using the procedure described earlier. The closed-loop pitch-rate response of the system with the H_∞ based controller and the nonlinear neurocontroller for an added delay of $\tau_d = 50$ ms in the two control channels (WF and δTV) is shown in Fig. 7. This value of τ_d corresponds to a phase loss of 8 deg at a frequency of 3 rad/s, which is the frequency that corresponds to the guaranteed multivariable phase margin of 6.6 deg for the linearized neurocontroller, and it is quite representative of the kinds of time delays to be expected in practical implementation of complex flight control designs. From Fig. 7 we note that the H_∞ based control shows very little degradation in tracking performance in the presence of time delay, whereas the neurocontroller exhibits a limit cycle behavior in the pitch-controlled variable, as predicted by linear analysis. Although the analysis of the linearized representations of the nonlinear neurocontroller indicated very poor stability margins with gain changes at the plant output (V and Q), closed-loop simulations with the *nonlinear* neurocontroller showed stable response with good command tracking for various cases that were tried with gain factors of plant output measurements between 0.5 and 2 (corresponding to the required gain margins from flying qualities specifications[8]). Clearly then, an accurate analysis of the neurocontroller robustness would require the development of advanced tools for nonlinear analysis.

In the neurocontrol design, the weights of the neural network (the *internal representation* of the neurocontroller) were chosen so as to minimize the objective function of Eq. (11) over an exhaustive set of pilot input commands to the nominal vehicle model by using the backpropagation algorithm. No information on modeling uncertainties and no constraint on "off-nominal" actuator dynamics were provided to the neural network during training. Without any constraint other than control effort and rate limits, the trained neural network learned to control the nominal vehicle model as efficiently as possible. Consequently, the robustness of the neurocontroller as trained using the procedure described earlier is mostly subject to the ability of the neural network to provide stable control for off-nominal vehicle model dynamics that were not used during training. Because feedforward neural networks

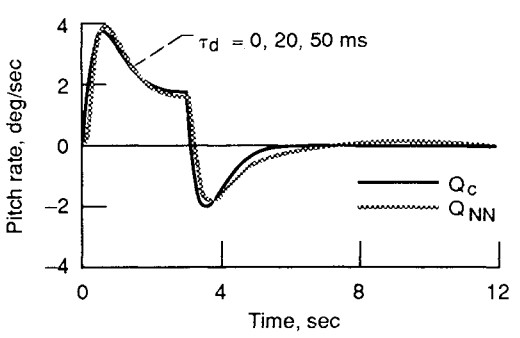

Fig. 8 Closed-loop pitch-rate response with time delays $\tau_d = 0$, 20, and 50 ms in both neurocontrol channels after training the neurocontroller with $\tau_d = 60$ ms.

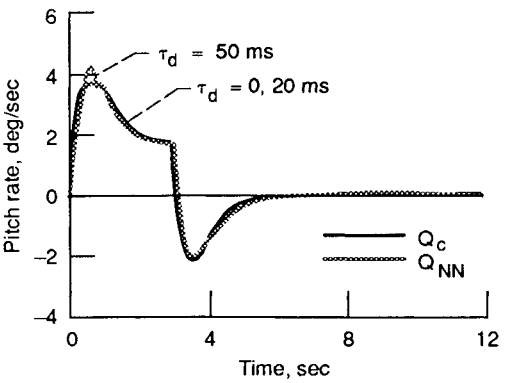

Fig. 9 Closed-loop pitch-rate response with time delays $\tau_d = 0$, 20, and 50 ms in both neurocontrol channels after training the neurocontroller with time delays τ_d uniformly distributed between 0 and 60 ms.

trained with backpropagation are known to have, in general, a limited ability to generalize beyond the training data set,[21] the robustness of the neurocontroller as trained here could be expected to be quite limited. Within the neural architecture of Fig. 3, it is therefore natural to expect to enhance the robustness of the neurocontroller by including modeling uncertainties in the training data set. The potential and limitations of this approach will be discussed in the next section in relation to phase robustness and error loop failure stability. Whether the training architecture of Fig. 3 can be extended to enhance the robustness of the neurocontroller through on-line learning of system uncertainties is contingent on resolving the major issues previously identified in the section dealing with the neurocontrol design.

Performance and Robustness Issues

To demonstrate the possibility of enhancing the phase robustness of the neurocontroller, the effect of the various time delays encountered by the signals throughout the closed-loop system was modeled by introducing in the training architecture a delay of adaptable length $\tau_d = n_d \delta t$ between plant and actuators. With this new training architecture, the control inputs $\bar{u}_a^s$ generated by the actuators at time t affect the plant outputs at time $[t + (n_d + 1)\delta t]$. It is proposed to increase the phase stability margin of the neurocontroller by training the feedforward neural network to track pilot input commands in the presence of the time delay(s) τ_d. There are two main strategies to achieve this goal, once an exhaustive set of commanded trajectories has been randomly generated following the sampling method previously described. In the first strategy, the neural network is trained to minimize the objective function of Eq. (11) over the whole training set in the presence of a fixed, a priori given delay $\tau_d > 0$ between plant and actuators. In the second strategy, the delay τ_d varies from one commanded trajectory of the training set to another according to an a priori given probability density function $1 > P(\tau_d) > 0$.

Using the first strategy, the neural network was trained to minimize Eq. (11) with $\tau_d = 60$ ms. Closed-loop system simulations showed that the performance of the trained neurocontroller was to a high degree insensitive to values of τ_d between 0 and 60 ms. This is illustrated in Fig. 8 by the closed-loop pitch-rate response of the neurocontroller with the time delays $\tau_d = 0$, 20, and 50 ms. Comparison with Fig. 7 shows the absence of limit cycle in pitch-rate response for $\tau_d = 50$ ms and thus an increased phase stability margin.

Using the second strategy, the neural network was trained to minimize Eq. (11) with time delays randomly distributed between 0 and 60 ms, i.e., $P(\tau_d) = 1/60$ if $0 < \tau_d < 60$ ms and $P(\tau_d) = 0$ otherwise. In contrast to the first strategy, closed-loop system simulations showed a significant degradation in pitch-rate response for large time delays. This is illustrated in

Fig. 9 by the closed-loop pitch-rate response of the neurocontroller with the time delays $\tau_d = 0$, 20, and 50 ms. Although pitch-rate tracking is essentially the same for $\tau_d = 0$ and 20 ms, small overshoots occur during pitch-rate transients for $\tau_d = 50$ ms. However, the pitch-rate response does not exhibit the limit cycle behavior of Fig. 7 any more, which indicates an increased phase stability margin.

As expected, a comparison between Fig. 2 and Figs. 8 and 9 indicates a degradation in tracking performance for an increase in phase robustness. Interestingly, the second training strategy provides a *gradual* loss of tracking performance as τ_d increases, in contrast to the first training strategy where the tracking performance is quasi-independent of τ_d over the range of τ_d values that were used for training the neural network. For "small" time delays such as $\tau_d = 20$ ms, the neurocontroller trained with the second strategy performs better than the neurocontroller trained with the first strategy. However, for large time delays such as $\tau_d = 50$ ms, the neurocontroller trained with the first strategy performed better than the neurocontroller trained with the second strategy. Phase stability is insured in both types of training strategies. Based on the statistical nature of the neural training, these results suggest selective sampling of the training set [e.g., the probability density function $P(\tau_d)$ in the example discussed earlier] as a means of minimizing the loss of tracking performance around "expected" nominal conditions, while still providing the stability margins needed by the control system. Future neurocontrol designs could take advantage of this characteristic to optimize the tradeoff between neurocontrol performance and robustness.

To analyze the possibility of increasing the inner-loop compensation of the neurocontrol system, failures in the e_Q loop were simulated during training. The training data set consisted of an exhaustive set of commanded trajectories randomly generated following the sampling method described earlier. During training, error loop failures in the e_Q loop were simulated by training the neural network to only track the velocity response without providing the neural network with any information on pitch-rate error, i.e., $e_Q = 0$, $\dot{e}_Q = 0$, and $1/t \int_0^t e_Q(\tau)\, d\tau = 0$. This was done in the fine-tuning phase of the training with a 5% ratio of trajectories *with* error loop failures against trajectories *without* error loop failures. Although it was possible to satisfactorily track the velocity command, the pitch-rate response exhibited a highly oscillatory behavior before settling to steady state. In addition, this induced inner-loop stability augmentation resulted in a substantial loss of control smoothness, both with and without the e_Q error loop failure. How to modify the neurocontrol architecture to improve the tradeoff between performance and stability augmentation remains an open question that needs to be addressed in future work.

Conclusion

A neural architecture and training procedure for flight control design were analyzed through the process of designing a model-following neurocontroller for an integrated airframe/propulsion model of a modern fighter aircraft for the piloted longitudinal landing task. To gain insight into the differences between the neurocontroller characteristics and those of controllers designed using "traditional" multivariable control design techniques, an H_∞ based controller that is input/output-wise compatible with the neurocontroller was designed and evaluated concurrently. Although the neurocontroller showed a better nominal performance than the baseline H_∞ controller (designed for the same command tracking problem), the phase stability margin of the neurocontroller was poor in comparison to the H_∞ controller. The possibility of enhancing the neurocontroller robustness through simulation of modeling uncertainties during training was discussed, and a technique for improving the phase stability margin of the closed-loop system was demonstrated. The major differences between the two control designs were identified as being the objective functions of the design processes and the structures of the controllers. The objective function of the H_∞ design is minimized in the *frequency* domain, whereas that of the neurocontrol design is minimized in the *time* domain. The structure of the H_∞ controller is that of a linear dynamic controller whose dynamics are automatically synthesized through the synthesis procedure, whereas the trained neurocontroller is a nonlinear mapping whose dynamics result from an appropriate selection of the inputs. The comparative analysis of these two controllers has provided insight into developing procedures for synthesizing neurocontrollers with improved performance and robustness.

Acknowledgment

We would like to thank the reviewers for their valuable comments and suggestions.

References

[1]Nguyen, D., and Widrow, B., "Neural Networks for Self-Learning Control Systems," *IEEE Control Systems Magazine,* Vol. 10, No. 3, 1990, pp. 18–23.

[2]Narendra, K. S., and Parthasarathy, K., "Identification and Control of Dynamical Systems Using Neural Networks," *IEEE Transactions on Neural Networks,* Vol. 1, No. 1, 1990, pp. 4–27.

[3]White, D., and Sofge, D., "Neural Network Based Process Optimization and Control," *29th IEEE Conference on Decision and Control* (Honolulu, HI), Vol. 6, Inst. of Electrical and Electronics Engineers, Dec. 1990, pp. 3270–3276.

[4]Jorgensen, C. C., and Schley, C., "A Neural Network Baseline Problem for Control of Aircraft Flare and Touchdown," *Neural Networks for Control,* edited by W. T. Miller, R. S. Sutton, and P. J. Werbos, MIT Press, Cambridge, MA, 1990, pp. 403–425.

[5]Baker, W. L., and Farrell, J. A., "Learning Augmented Flight Control for High Performance Aircraft," *Proceedings of the AIAA Guidance, Navigation, and Control Conference* (New Orleans, LA), Vol. 1, AIAA, Washington, DC, Aug. 1991, pp. 347–358.

[6]Ha, C. M., "Neural Networks Approach to AIAA Aircraft Control Design Challenge," *Proceedings of the AIAA Guidance, Navigation, and Control Conference* (New Orleans, LA), Vol. 1, AIAA, Washington, DC, Aug. 1991, pp. 653–663.

[7]Garg, S., Mattern, D. L., and Bullard, R. E., "Integrated Flight/Propulsion Control System Design Based on a Centralized Approach," *Journal of Guidance, Control, and Dynamics,* Vol. 14, No. 1, 1991, pp. 107–116.

[8]"Military Specification—Flying Qualities of Piloted Airplanes," U.S. Air Force, MIL-F-8785C, Wright-Patterson AFB, OH, Nov. 1980.

[9]Doyle, J. C., Glover, K., Khargonekar, P. P., and Francis, B. A., "State-Space Solutions to Standard H_2 and H_∞ Control Problems," *IEEE Transactions on Automatic Control,* Vol. 34, No. 8, 1989, pp. 831–847.

[10]"MATRIXx Robust Control Module," Integrated Systems, Inc., Santa Clara, CA, Dec. 1989.

[11]Kaminer, I., Khargonekar, P. P., and Robel, G., "Design of Localizer Capture and Track Modes for a Lateral Autopilot Using H_∞ Synthesis," *IEEE Control Systems Magazine,* Vol. 10, No. 4, 1990, pp. 13–21.

[12]Reichert, T. R., "Application of H_∞ Control to Missile Autopilot Design," *Proceedings of the AIAA Guidance, Navigation, and Control Conference* (Boston, MA), Vol. 2, AIAA, Washington, DC, Aug. 1989, pp. 1065–1072 (AIAA Paper 89-3550).

[13]Garg, S., Mattern, D. L., Bright, M. M., and Ouzts, P. J., "H_∞ Based Integrated Flight/Propulsion Control Design for a STOVL Aircraft in Transition Flight," *Proceedings of the AIAA Guidance, Navigation, and Control Conference* (Portland, OR), Vol. 1, AIAA, Washington, DC, Aug. 1990, pp. 183–193 (AIAA Paper 90-3335).

[14]Troudet, T., Garg, S., and Merrill, W. C., "Neural Network Application to Aircraft Control System Design," *Proceedings of the AIAA Guidance, Navigation, and Control Conference* (New Orleans, LA), Vol. 3, AIAA, Washington, DC, Aug. 1991, pp. 993–1009.

[15]Troudet, T., Garg, S., Mattern, D. L., and Merrill, W. C., "Towards Practical Control Design Using Neural Computation," *IEEE/INNS International Joint Conference on Neural Networks* (Seattle, WA), Vol. 2, Inst. of Electrical and Electronics Engineers, July 1991, pp. 675–681.

[16]Rumelhart, D. E., McClelland, J. L., and the PDP Research

Group, *Parallel Distributed Processing—Exploration in the Microstructure of Cognition, Vol. I: Foundations,* MIT Press, Cambridge, MA, 1986.

[17]Ogata, K., *Modern Control Engineering,* Prentice-Hall, Englewood Cliffs, NJ, 1970.

[18]Lehtomaki, N. A., "Practical Robustness Measures in Multivariable Control System Analysis," Ph.D. Dissertation (LIDS-TH-1093), Massachusetts Inst. of Technology, Cambridge, MA, May 1981.

[19]Doyle, J. C., "Structured Uncertainty in Control System Design," *Proceedings of the 24th Conference on Decision and Control* (Ft. Lauderdale, FL), Inst. of Electrical and Electronics Engineers, Dec. 1985, pp. 260–265.

[20]Apkarian, P. R., "Structured Stability Robustness Improvement by Eigenspace Assignment Techniques: A Hybrid Methodology," *Journal of Guidance, Control, and Dynamics,* Vol. 12, No. 2, 1989, pp. 162–168.

[21]Troudet, T., and Merrill, W. C., "Neuromorphic Learning of Continuous Valued Mappings from Noise Corrupted Data," *IEEE Transactions on Neural Networks,* Vol. 2, No. 2, 1991, pp. 294–301.