JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 20, No. 1, January–February 1997

Nonlinear Flight Control Using Neural Networks

Byoung S. Kim* and Anthony J. Calise†
Georgia Institute of Technology, Atlanta, Georgia 30332-0150

The theoretical development of a direct adaptive tracking control architecture using neural networks is presented. Emphasis is placed on utilization of neural networks in a flight control architecture based on feedback linearization of the aircraft dynamics. Neural networks are used to represent the nonlinear inverse transformation needed for feedback linearization. Neural networks may be first trained off line using a nominal mathematical model, which provides an approximate inversion that can accommodate the total flight envelope. Neural networks capable of on-line learning are required to compensate for inversion error, which may arise from imperfect modeling, approximate inversion, or sudden changes in aircraft dynamics. A stable weights adjustment rule for the on-line neural network is derived. Under mild assumptions on the nonlinearities representing the inversion error, the adaptation algorithm ensures that all of the signals in the loop are uniformly bounded and that the weights of the on-line neural network tend to constant values. Simulation results for an F-18 aircraft model are presented to illustrate the performance of the on-line neural network based adaptation algorithm.

I. Introduction

CONTROL of nonlinear systems by feedback linearization is well known and has been applied to control of a wide variety of nonlinear dynamic systems. In particular, this approach has gained a great deal attention in aircraft flight control through theoretical and applied investigations, culminating with a number of actual flight tests.[1,2] These applications have employed separate inner and outer loop inversions based on a two time scale approximation normally inherent in aircraft dynamics. This allows the inverse design to be performed without state transformation because the dynamics for each loop (when considered separately) are square (number of controls equals the number of degrees of freedom). More recent studies have incorporated robust synthesis methods for the outer loop.[3,4] From a design viewpoint, the main advantages are that the nonlinear dynamics of the aircraft are transformed to an equivalent linear system, which has a standard form. The linear model is used to design a controller to achieve a desired response to pilot commands, based on handling qualities criteria and maneuvering objectives. Hence, the cost associated with the design of a flight control system can be significantly reduced in comparison to a gain scheduled design. Performance can also be enhanced because many of the uncertainties associated with small perturbation linearized models are removed.

The primary difficulty associated with the use of feedback linearization for aircraft flight control is that a detailed knowledge of the nonlinear plant dynamics is required. A secondary issue is that accurate full-envelope nonlinear inversion is computationally intensive. A high-fidelity nonlinear force and moment model must be constructed and inverted in real time. Neural networks offer the potential to overcome these difficulties through a combination of both off-line and on-line (i.e., in-flight) learning.

Recently, neural networks have been proposed as feedforward inverse dynamics controllers. Sanner and Slotine[5] developed a direct adaptive tracking control architecture using Gaussian radial basis function networks to adaptively compensate for plant nonlinearities. Gomi and Kawato[6] have also proposed similar learning control schemes using neural networks. These neural network controllers look very much like adaptive elements. The connection between neural network controllers and adaptive controllers was also exploited by Narendra and Parthasarathy.[7] They view neural networks as highly nonlinear control elements that offer distinct advantages over conventional linear parameter adaptive controllers in achieving desired system performance.

There are a number of flight control applications reported that illustrate the on-line learning capability of neural networks. Troudet et al.[8] designed a model-following neurocontroller for an integrated airframe/propulsion linearized model of a modern fighter aircraft. Only the piloted longitudinal landing task was considered. Sing and Wells[9] used a radial basis network with on-line weight adaptation to suppress wing rock for a slender delta wing configuration, using a simplified analytical model for wing rock. Sadhukhan and Feteih[10] combined neural control with dynamic inversion. Their study was limited to examining the linearized longitudinal dynamics of an A8 aircraft. Napolitano and Kincheloe[11] employ an extended back propagation algorithm to evaluate various neural network-based designs of autopilot systems for implementing a variety of flight control functions. They used a six-degree-of-freedom aircraft simulation code in their study.

This paper presents the design and evaluation of a flight controller using neural networks and demonstrates the feasibility of applying this approach to an aircraft flight control design. Emphasis is placed on 1) use of a command and stability augmentation system architecture based on feedback linearization using an off-line trained network to invert the nonlinearities and 2) development of an on-line adaptive architecture that employs a second neural network to compensate for inversion error. A stable weight adjustment rule for the second neural network is derived using a Lyapunov-like function. We prove stability in the presence of bounded realization error of the off-line trained network. Under mild assumptions on the nonlinearities representing the inversion error, the adaptation algorithm ensures that all of the signals in the adaptive loop are uniformly bounded when a dead zone is applied, and that the weights of the second neural network tend to constant values. The on-line adaptation approach used employs concepts from Ref. 5, wherein the dynamics are linear in the control variable. The main extension here is to the case where the dynamics are nonlinear in both the states and the controls. Nonlinearity in the control is an important issue with the advent of low-visibility aircraft and the increasing need for agility in fighter aircraft. The second major contribution of this paper lies in a detailed application to design of an adaptive command augmentation system that treats the six-degree-of-freedom nonlinear dynamics of aircraft motion. Although separate autopilot functions are design for each degree of freedom, the on-line network accounts for nonlinear coupling effects. The dead zone approach to the adaptation employed here was first developed in Ref. 12. References 13 and 14 provide some background and definitions for readers who are unfamiliar with neural networks and the related terminology employed.

Received June 13, 1994; presented as Paper 94-3646 at the AIAA Guidance, Navigation, and Control Conference, Scottsdale, AZ, Aug. 1–3, 1994; revision received Sept. 17, 1996; accepted for publication Sept. 17, 1996.

*Research Assistant, School of Aerospace Engineering; currently Senior Researcher, Agency for Defense Development, Yuseong, P.O. Box 35-3, Taejon 305-6000, Republic of Korea.

†Professor, School of Aerospace Engineering. Fellow AIAA.

The paper is organized as follows. Section II presents an approach for combining feedback linearization and neural networks and especially focuses on the theoretical background of a direct adaptive tracking control architecture. A stability proof is presented along with an analysis that shows that all signals within the resulting closed-loop system are uniformly bounded. Section III outlines an aircraft application of the approach developed in Sec. II, and Sec. IV summarizes an evaluation of the design from Sec. III for a high-g turn maneuver. Conclusions are given in Sec. V.

II. Nonlinear Control Using Feedback Linearization and Neural Networks

A. Derivation of the Linearizing Transformation

Consider an n-degree-of-freedom nonlinear dynamic system of the form

$$\ddot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{\delta}) \tag{1}$$

where $x(t), \dot{x}(t) \in R^n$ are the state variables, $\delta(t) \in R^m$ is the control variable, and f is a mapping from a domain $R^n \times R^n \times R^m$ into R^n. The simple case arises when $n = m$, which is called a square system. If $f(x, \dot{x}, \cdot)$ is invertible and $x, \dot{x}$ are measurable, the preceding system can be transformed to the form

$$\ddot{\boldsymbol{x}} = \boldsymbol{u} \tag{2}$$

$$\boldsymbol{u} = \boldsymbol{f}(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{\delta}) \tag{3}$$

where $u(t) \in R^n$ is the pseudocontrol variable. Equation (3) is the linearizing transformation using state feedback. Note that in this transformation only the control variable is transformed.

The inverse transformation of Eq. (3) is expressed in the form

$$\boldsymbol{\delta} = \boldsymbol{f}^{-1}(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{u}) \tag{4}$$

To implement the control technique a means to compute the inverse transformation in Eq. (4) is required. A neural network may be used to realize this inverse transformation and produce an approximation $(\hat{f}^{-1})$ of the inverse transformation when implemented in hardware. Initial training of the neural network may be conducted prior to flight with input–output pairs generated from a mathematical model. Such model-based training is referred to as off-line training. The block diagram shown in Fig. 1 depicts an off-line trained neural network (NN1) being used to implement the inversion process of Eq. (4).

If f is perfectly known and the realization of f^{-1} with the neural network is perfect, then one has a linear transformed system exactly as in Eq. (2). Unfortunately, more often than not, the information about f and the realization of the neural network are not perfect.

Now, consider a representation of the inversion error in Eq. (2) as follows:

$$\ddot{\boldsymbol{x}} = \boldsymbol{u} + \boldsymbol{\Delta}'(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{u}) \tag{5}$$

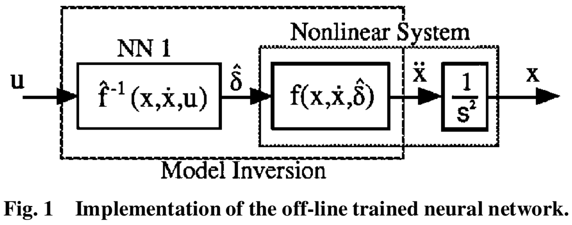

Fig. 1 Implementation of the off-line trained neural network.

where

$$\boldsymbol{\Delta}'(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{u}) = \boldsymbol{f}(\boldsymbol{x}, \dot{\boldsymbol{x}}, \hat{\boldsymbol{\delta}}) - \boldsymbol{f}(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{\delta}) \tag{6}$$

and $\boldsymbol{\Delta}' : R^n \times R^n \times R^n \to R^n$ is a mapping representing the inversion error. In Eq. (6) $\hat{\boldsymbol{\delta}} = \hat{\boldsymbol{f}}^{-1}(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{u})$ is the NN1 realization of f^{-1}. Equation (5) is shown in the dotted box in Fig. 2 for the ith control channel, which also depicts a conventional proportional-plus-derivative (PD) linear controller architecture. Note that the unknown nonlinear term $\boldsymbol{\Delta}'$ is a function of the pseudocontrol variable $\boldsymbol{u}$ in addition to the state variables. Thus, the choice of the control structure affects the inversion error.

B. Direct Adaptive Control Using Neural Networks

The adaptive control structure for each channel is chosen as

$$u_i(t) = u_{\text{pd}_i}(t) + \ddot{x}_{c_i}(t) - \hat{u}_{\text{ad}_i}(t), \qquad i = 1, 2, \ldots, n \tag{7}$$

where u_{pd_i} is a PD control and the $\hat{u}_{\text{ad}_i}$ is an adaptive control. The PD control is used to shape system response, and the adaptive control is used to compensate for the inversion error during operation. Let u_{pd_i} be defined by

$$u_{\text{pd}_i}(t) = k_{p_i}\left[x_{c_i}(t) - x_i(t)\right] + k_{d_i}\left[\dot{x}_{c_i}(t) - \dot{x}_i(t)\right] \qquad i = 1, 2, \ldots, n \tag{8}$$

where x_{c_i} and $\dot{x}_{c_i}$ are the position and velocity commands. The parameters k_{p_i} and k_{d_i} are the proportional and derivative control gains, which are designed so that the system in Eq. (5) is stable and performance specifications (settling time, overshoot, etc.) are met for $\Delta_i' = 0$. The adaptive term $\hat{u}_{\text{ad}_i}$ is used to compensate for the inversion error Δ_i' during operation. It will be realized by a second neural network (NN2$_i$), also depicted in Fig. 2, where $u_{(-i)}$ denotes u with the u_i element deleted.

Using Eq. (7) in Eq. (5) the error dynamics become

$$\ddot{\tilde{x}}_i + k_{d_i}\dot{\tilde{x}}_i + k_{p_i}\tilde{x}_i = \hat{u}_{\text{ad}_i} - \Delta_i'(x, \dot{x}, u), \qquad i = 1, 2, \ldots, n \tag{9}$$

where $\tilde{x}_i = x_{c_i} - x_i$. In state-space form, we have

$$\dot{\boldsymbol{e}}_i = A_i\boldsymbol{e}_i + \boldsymbol{b}\left[\hat{u}_{\text{ad}_i} - \Delta_i'(x, \dot{x}, u)\right], \qquad i = 1, 2, \ldots, n \tag{10}$$

where $\boldsymbol{e}_i^T = [\tilde{x}_i \;\; \dot{\tilde{x}}_i]$, and

$$A_i = \begin{bmatrix} 0 & 1 \\ -k_{p_i} & -k_{d_i} \end{bmatrix} \qquad \boldsymbol{b} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{11}$$

Note that in general Eq. (10) is nonautonomous due to the dependence of Δ_i' in Eq. (9) on $\boldsymbol{x}$, $\dot{\boldsymbol{x}}$, and $\boldsymbol{u}$, which in turn depend on $\boldsymbol{x}_c(t)$, $\dot{\boldsymbol{x}}_c(t)$, and $\ddot{\boldsymbol{x}}_c(t)$. The ideal purpose of $\hat{u}_{\text{ad}_i}$ is to cancel the inversion error Δ_i' so that the error states go to zero asymptotically.

Let $\hat{\Delta}_i'$ be an realization of Δ_i' when neural networks with a finite number N of basis functions β_{ij} are used:

$$\hat{\Delta}_i'(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{u}) = \sum_{j=1}^{N} \hat{w}_{ij}(t)\beta_{ij}'(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{u}) = \hat{\boldsymbol{w}}_i^T(t)\boldsymbol{\beta}_i'(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{u}) \qquad i = 1, 2, \ldots, n \tag{12}$$

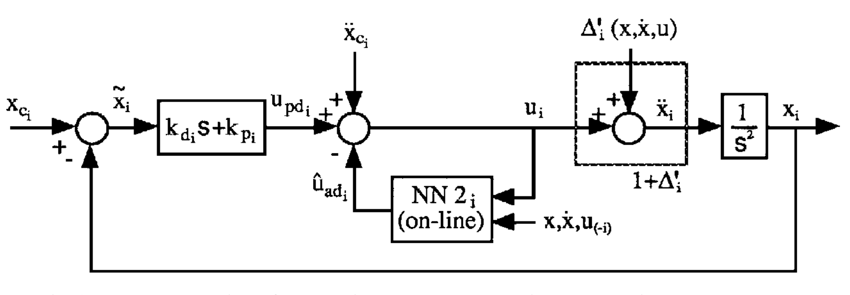

Fig. 2 Implementation of the on-line neural network in the adaptive control scheme.

Define

$$\hat{\Delta}_i'^*(x,\dot{x},u) = \sum_{j=1}^{N} w_{ij}^* \beta_{ij}'(x,\dot{x},u) = w_i^{*T}\beta_i'(x,\dot{x},u)$$

$$i = 1,2,\ldots,n \quad (13)$$

where w_i^* is the value obtained when $\hat{w}_i$ is optimized over some compact domain $\mathcal{D}$ of $\{x,\dot{x},u\} = d$ of interest. For example, we may define

$$w_i^* = \arg\left\{ \min_{\hat{w}_i} \max_{d\in\mathcal{D}} |\Delta_i'(d) - \hat{\Delta}_i'(d)| \right\} \quad (14)$$

If $\varepsilon_i > 0$ is the upper value of the min–max problem just defined, then

$$\left|\Delta_i'(d) - \hat{\Delta}_i'^*(d)\right| \le \varepsilon_i \qquad \forall d \in \mathcal{D} \quad (15)$$

Note that for fixed N, ε_i depends on the choice of the basis functions and the accuracy of NN1, which determines $|\Delta_i'|$, $\forall d \in \mathcal{D}$.

Assumption 1: There exists a fixed point $\hat{u}_{\text{ad}_i}$ of $\hat{\Delta}_i'$ for all $d \in \mathcal{D}$ such that

$$\hat{u}_{\text{ad}_i} - \hat{\Delta}_i'\big(x,\dot{x},u_{(-i)},u_{\text{pd}_i} + \ddot{x}_{c_i} - \hat{u}_{\text{ad}_i}\big) = 0, \qquad i = 1,2,\ldots,n \quad (16)$$

This ensures the existence of $\hat{u}_{\text{ad}_i}$ (see Fig. 2). Because Eq. (16) is a scalar condition, it is easily guaranteed by any set of basis functions (β_{ij}) that are continuous and bounded functions of u_i. Based on Eq. (12), the adaptive control signal of NN2$_i$ is chosen as

$$\hat{u}_{\text{ad}_i}(t) = \sum_{j=1}^{N} \hat{w}_{ij}(t)\beta_{ij}'(x,\dot{x},u) = \hat{w}_i^T(t)\beta_i'(x,\dot{x},u) \quad (17)$$

where the $\hat{w}_i(t) \in R^N$ comprise the weights $\hat{w}_{ij}(t)$ of NN2$_i$ to be updated on line. Note that the adaptive control $\hat{u}_{\text{ad}_i}$ in Eq. (17) is linearly parameterized. This feature makes it possible to derive a suitable on-line adaptation rule. In the following development, only neural networks whose weights (parameters to be updated) appear linearly in their output description are considered for NN2 in Fig. 2.

The estimation error is represented as

$$\hat{u}_{\text{ad}_i} - \hat{\Delta}_i'^* = \tilde{w}_i^T(t)\beta_i'(x,\dot{x},u) = \tilde{w}_i^T(t)\beta_i(t,e,\tilde{w})$$

$$i = 1,2,\ldots,n \quad (18)$$

where

$$\tilde{w}_i(t) = \hat{w}_i(t) - w_i^* \quad (19)$$

and $\tilde{w}_i(t) \in R^N$ is the parameter estimation error vector for NN2$_i$. Recall that Δ_i' and β_i' are functions of x, $\dot{x}$, and u, which in turn depend on t, e, and $\tilde{w}$ through Eqs. (7), (17), and (19). In Eq. (18), the first functional dependence is used in actual neural network implementation as illustrated in Fig. 2 and the second is used in the stability analysis.

From Eq. (18) the forcing term in Eq. (10) satisfies

$$\hat{u}_{\text{ad}_i} - \Delta_i'(x,\dot{x},u) = \big(\hat{u}_{\text{ad}_i} - \hat{\Delta}_i'^*\big) + \big[\hat{\Delta}_i'^*(x,\dot{x},u) - \Delta_i'(x,\dot{x},u)\big]$$

$$= \tilde{w}_i^T\beta_i(t,e,\tilde{w}) + \big[\hat{\Delta}_i^*(t,e,\tilde{w}) - \Delta_i(t,e,\tilde{w})\big] \quad (20)$$

Under Assumption 1 expressed in Eq. (16), an equivalent expression for Eq. (10) is

$$\dot{e}_i = A_i e_i + b\tilde{w}_i^T\beta_i + b\big(\hat{\Delta}_i^* - \Delta_i\big), \qquad i = 1,2,\ldots,n \quad (21)$$

The adaptation rule is chosen as follows:

$$\dot{\tilde{w}}_i = \begin{cases} -\gamma_i e_i^T P_i b \beta_i(t,e,\tilde{w}), & \text{when} \quad \|e_i\|_{P_i} > E_i \\ 0, & \text{when} \quad \|e_i\|_{P_i} \le E_i \end{cases}$$

$$i = 1,2,\ldots,n \quad (22)$$

where $\|e_i\|_{P_i}$ denotes $\sqrt{(e_i^T P_i e_i)}$, $\gamma_i > 0$ is the adaptation gain (or learning rate), β_i is an equivalent expression for the basis function vector as shown in Eq. (18), and P_{0_i} is a (2×2) symmetric, positive-definite matrix satisfying

$$P_i A_i + A_i^T P_i = -I, \qquad i = 1,2,\ldots,n \quad (23)$$

the solution of which is given by

$$P_i = \begin{bmatrix} \dfrac{k_{d_i}}{2k_{p_i}} + \dfrac{k_{p_i}}{2k_{d_i}}\left(1 + \dfrac{1}{k_{p_i}}\right) & \dfrac{1}{2k_{p_i}} \\ \dfrac{1}{2k_{p_i}} & \dfrac{1}{2k_{d_i}}\left(1 + \dfrac{1}{k_{p_i}}\right) \end{bmatrix} \quad (24)$$

In the adaptation rule of Eq. (22) the parameters are adjusted only when the norm of the state errorvector lies outside a dead zone, whose size is defined by E_i. A larger E_i implies that a larger error state is tolerated and that adaptation takes place over shorter total time.

Whereas the analysis of the adaptive system is carried out using Eq. (22), implementation of the adaptive law uses the first functional dependence of Eq. (18), which means that the adaptation rule is implemented as

$$\dot{\hat{w}}_i = \begin{cases} -\gamma_i e_i^T P_i b \beta_i'(x,\dot{x},u), & \text{when} \quad \|e_i\|_{P_i} > E_i \\ 0, & \text{when} \quad \|e_i\|_{P_i} \le E_i \end{cases}$$

$$i = 1,2,\ldots,n \quad (25)$$

Note that the adaptive law in Eq. (25) only requires on-line quadrature and that it can be implemented in parallel.

Theorem 1: All signals in the $(2+N)n$-dimensional system, described by Eqs. (21) and (22), are uniformly bounded if the following conditions are satisfied:

1) Assumption 1 is satisfied.

2) The compact domain $\mathcal{D}$ for which Eq. (15) holds is sufficiently large.

3) The size of dead zone for each channel is determined by $E_i \ge 2\varepsilon_i[\bar{\lambda}(P_i)]^{3/2}$, where $\bar{\lambda}(X)$ denotes the maximum eigenvalue of X. Moreover, for $\varepsilon_i = 0$ (no NN2$_i$ approximation error), the error dynamics Eq. (21) are asymptotically stable.

Proof: See Appendix.

Remark: The proper choice for the size of the dead zone is crucial for establishing uniform stability. From condition 2 and Eq. (24), this is dependent on the PD control gains k_{p_i} and k_{d_i}, and on the bound ε_i in Eq. (15). Hence, the prior knowledge needed to estimate the size of the dead zone is the bound ε_i.

Theorem 2: If conditions 1–3 of Theorem 1 are satisfied, then the total time during which adaptation takes place is finite, and $\|\tilde{w}_i\|_2$ decreases a finite amount every time the trajectory leaves and re-enters the dead zone. Moreover, in the limit as $t \to \infty$, $e_i(t)$ lies inside the dead zone and $\tilde{w}_i(t) \to$ a constant.

Proof: See Appendix.

Remark: The nature of the trajectories is illustrated in a two-dimensional space as shown in Fig. 3 (Ref. 12). This figure illustrates the properties guaranteed by Theorem 2, including several intervals during which adaptation takes place, ultimately culminating in a

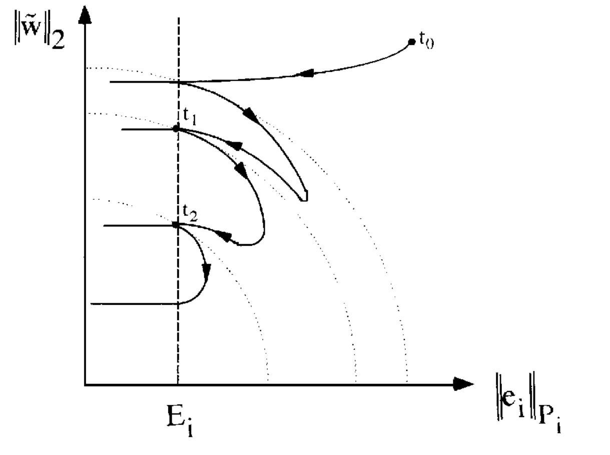

Fig. 3 Illustration of the properties shown in Theorem 2.

steady-state condition in which $\|e_i\|_{P_i} \le E_i$ and no adaptation occurs ($\tilde{w}_i = 0$). In Fig. 3 the dotted arcs represent contours of constant $V_i = [\|e_i\|^2_{P_i} + \|\tilde{w}_i\|^2_2/\gamma_i]/2$, and the break line parallel to the vertical axis depicts the boundary of the dead zone. From Theorem 1, if $\varepsilon_i = 0$, then E_i may be reduced to zero and $e_i(t) \to 0$ as $t \to \infty$.

III. Application to Aircraft Flight Control

A. Command Argumentation System Design

Figure 4 depicts the architecture of a command augmentation system (CAS) design based on feedback linearization for an aircraft. The pilot inputs to the CAS are chosen as roll rate command and normal acceleration command. The CAS is required to stabilize the airframe while following normal acceleration and roll rate commands from the pilot. In addition to these functions, the CAS has responsibility for maintaining zero angle of sideslip during maneuvers. The aircraft of interest is an F-18, for which the following controls were selected: $\delta h_L(\delta h_R)$ = left (right) stabilator and δr = rudder. Reference 15 details the design of the command augmentation logic and the command transformation from body rates to Euler angle rates. Reference 15 also provides baseline performance results for a set of maneuvers using a nearly exact (but nonadaptive) inverse of the aircraft moment equations.

The role of neural networks in this architecture is divided to two functions: 1) an off-line learning intended to provide nominal input–output mapping using the mathematical model of an aircraft and 2) an adaptive network that compensates for imperfect inversion and in-flight changes in the actual aircraft dynamics. The theory developed in Sec. II is applied only to the attitude loop depicted in Fig. 4, with the slower dynamics of the outer loop regarded as frozen. The command augmentation logic has been designed to ensure that this two time scale property is preserved at all flight conditions.[16]

B. Off-Line Neural Network Design

The key purpose of the off-line neural network (NN1) is to approximately invert the vehicle attitude dynamics using an implementible architecture. By inversion we mean given the desired angular accelerations (U_i in Fig. 4), determine the required control deflection. It is important to optimize the topology of NN1 for this purpose. Training of the neural networks to represent the inverse mapping of plant dynamics from moment coefficients to control deflections, instead of from angular accelerations to control deflections, facilitates a representation of the mapping that is less sensitive to altitude and Mach number variation. This considerably reduces network complexity and the effort required in training. The detailed computational flow for the inversion using neural networks is given in Fig. 5, where $\delta e = (\delta h_L + \delta h_R)/2$ is the effective elevator deflection and $\delta t = (\delta h_L - \delta h_R)/2$ is the effective differential tail deflection.

A blind application of the network architecture may require a prohibitive number of neurons. Careful examination of the dependence of the aerodynamic moment data on flight condition and control deflections allowed development of an efficient network architecture made up of a combination of radial basis functions units and sigma-pi units in this application. Reference 16 provides a more detailed justification for the design of the off-line network architecture, which is defined in equation form as follows.

Longitudinal network:

$$\delta e = \sum_{j=1}^{N+1} \big(x_{1j} + x_{2j}h + x_{3j}Q + x_{4j}C_M + x_{5j}hC_M + x_{6j}Q^2 + x_{7j}QC_M + x_{8j}C_M^2\big)\phi_j(\alpha, M) \tag{26}$$

Lateral network:

$$\delta t = \sum_{j=1}^{N+1} (y_{1j} + y_{2j}h + y_{3j}P + y_{4j}R + y_{5j}C_L + y_{6j}C_N + y_{7j}hP + y_{8j}hR + y_{9j}hC_L + y_{10j}hC_N)\phi_j(\alpha, M) \tag{27}$$

$$\delta r = \sum_{j=1}^{N+1} (z_{1j} + z_{2j}h + z_{3j}P + z_{4j}R + z_{5j}C_L + z_{6j}C_N + z_{7j}hP + z_{8j}hR + z_{9j}hC_L + z_{10j}hC_N)\phi_j(\alpha, M) \tag{28}$$

In Eqs. (26–28) x_{ij}, y_{ij}, and z_{ij} are the weights, ϕ_j are radial basis functions (two-dimensional Gaussian functions), and N is the number of radial basis function units.

Training data (U_i and δ_i pairs) were artificially generated at different flight conditions from simulation software (4275 for longitudinal and 30,240 for lateral network training). An example envelope of flight conditions is presented in Fig. 6 for an F-18 aircraft model that was used for concept demonstration. Training data were generated uniformly over the domain of flight conditions and control surface deflection indicated in the figure. Because the weights appear linearly as shown in Eqs. (26–28), a standard linear least square approximation method was used as the training method. The centers and sigma values of the Gaussian functions, ϕ_j, were adjusted experimentally to improve the approximation.

C. On-Line Neural Network Design and Direct Adaptive Control

Under the assumption that the inversion error Δ is decoupled, that is, $\Delta_i = \Delta_i(M, \alpha, h, x_i, \dot{x}_i, u_i)$, $i = \phi, \theta, \psi$, all channels of roll, pitch, and yaw motions are decoupled. This assumption is not

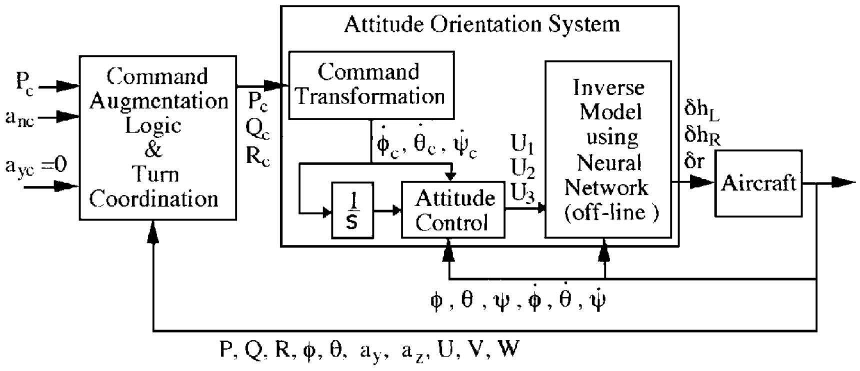
Attitude Orientation System
P_c
a_{nc}
$a_{yc}=0$
Command Augmentation Logic & Turn Coordination
P_c Q_c R_c
Command Transformation
$\dot\phi_c, \dot\theta_c, \dot\psi_c$
$\frac{1}{s}$
Attitude Control
U_1 U_2 U_3
Inverse Model using Neural Network (off-line)
δh_L δh_R δr
Aircraft
$\phi, \theta, \psi, \dot\phi, \dot\theta, \dot\psi$
P, Q, R, ϕ, θ, a_y, a_z, U, V, W

Fig. 4 Command augmentation system for an aircraft.

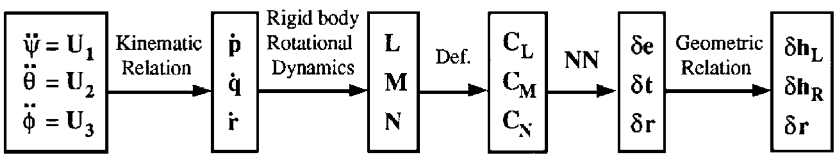
$\ddot\psi = U_1$
$\ddot\theta = U_2$
$\ddot\phi = U_3$
Kinematic Relation
$\dot p$ $\dot q$ $\dot r$
Rigid body Rotational Dynamics
L M N
Def.
C_L C_M C_N
NN
δe δt δr
Geometric Relation
δh_L δh_R δr

Fig. 5 Computational flow for neural network based inversion.

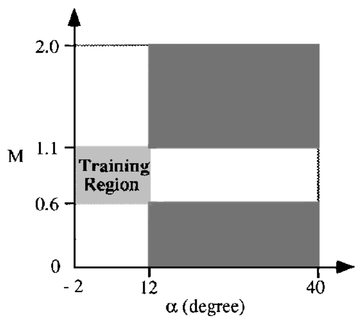

Fig. 6 Off-line training envelope: $0 \le H \le 40{,}000$, $-1.5 \le p \le 1.5$, $-1.5 \le q \le 1.5$, $-1.0 \le r \le 1.0$, $-24 \le \delta e \le 10.5$ deg, $-21 \le \delta t \le 21$ deg, and $-30 \le \delta r \le 30$ deg.

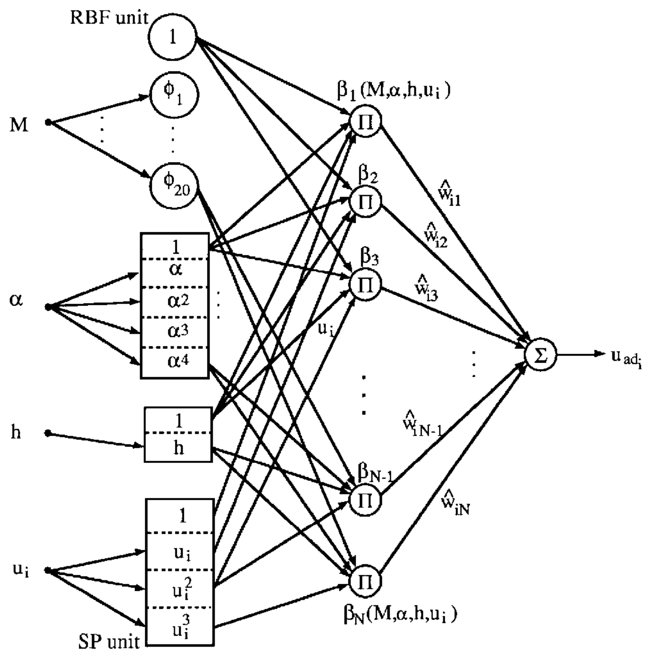

Fig. 7 Structure of the on-line adaptive neural network for each channel.

necessary for application of the theory, but is employed to reduce the size and complexity of NN2 for implementation.

The NN2$_i$ for each channel of attitude motion was chosen to satisfy the form expressed in Eq. (17) in which the weights appear linearly. The basis functions have a functional dependence on M, α, h, u_i, and corresponding attitude angle and attitude angular rate. The functional dependency of Δ_i on corresponding attitude angle and attitude angular rate was neglected to reduce the number of inputs to NN2$_i$. Because the dimension of the basis function neural network increases exponentially with the number of network inputs, reduction of the number of inputs from 6 to 4 results in a substantial reduction in the required size of the neural network. Thus, the adaptive control signals in Eq. (17) were represented as follows:

$$
\begin{aligned}
u_{\mathrm{ad}_\phi} &= \sum_{j=1}^{N} \hat{w}_{\phi,j}\beta_j(M,\alpha,h,u_\phi)\\
u_{\mathrm{ad}_\theta} &= \sum_{j=1}^{N} \hat{w}_{\theta,j}\beta_j(M,\alpha,h,u_\theta) \qquad (29)\\
u_{\mathrm{ad}_\psi} &= \sum_{j=1}^{N} \hat{w}_{\psi,j}\beta_j(M,\alpha,h,u_\psi)
\end{aligned}
$$

The structure of NN2$_i$ for each channel is shown in Fig. 7. For this network 21 radial basis function units and 40 sigma-pi units were employed to build the basis functions. Thus, the total number of weights for each neural network is 840. One-dimensional Gaussian functions were used as radial basis functions for M. The center and sigma values of the Gaussian functions were adjusted to improve the approximation, i.e., more dense centers were placed in the transonic region (i.e., near Mach 1). For α, sigma-pi units (polynomial representations) were used up to fourth order. For h, sigma-pi units were used up to first order, and for u_{c_i}, units up to third order were employed. The resulting description for each basis function of NN2$_i$ is as follows:

$$\beta_j(M,\alpha,h,u_i) = \varphi_k(M)\alpha^l h^m u_i^n, \qquad i=\phi,\theta,\psi \qquad (30)$$

where $k = 1,\ldots,21$; $l = 0,\ldots,4$; $m = 0,1$; and $n = 0,\ldots,3$. The prescribed architecture of NN2 does not necessarily represent the optimum network size for practical hardware implementation.

From Eq. (25), the adaptation rule for each neural network (NN2$_i$) is as follows:

$$
\begin{aligned}
\dot{\hat{w}}_\phi &= -\gamma_\phi s_\phi \beta_\phi(M,\alpha,h,u_\phi)\\
\dot{\hat{w}}_\theta &= -\gamma_\theta s_\theta \beta_\theta(M,\alpha,h,u_\theta) \qquad (31)\\
\dot{\hat{w}}_\psi &= -\gamma_\psi s_\psi \beta_\psi(M,\alpha,h,u_\psi)
\end{aligned}
$$

where

$$s_i = \boldsymbol{e}_i^T P_{0_i}\boldsymbol{b} = \left(1/2k_{p_i}\right)\tilde{x}_i + \left(1/2k_{d_i}\right)\left[1+\left(1/k_{p_i}\right)\right]\dot{\tilde{x}}_i \qquad i=\phi,\theta,\psi \qquad (32)$$

The elements of $\hat{\boldsymbol{w}}_i$ and $\boldsymbol{\beta}_i$, $i=\phi,\theta,\psi$, represent the weight vector and basis function vector in each on-line neural network. The quantity γ_i is an adaptation gain for each channel. No dead zone was applied, which amounts to the assumption that near perfect realization of the inversion error is attainable [$\varepsilon_i = 0$ in Eq. (15)]. The parameters k_{p_i} and k_{d_i} were chosen to provide a 1% settling time of 0.8 s and a damping ratio of 0.707 for the attitude loops. The 1% settling time for the outer command augmentation loop was designed to be 2.4 s. The outer loop designed was slightly modified from that of Ref. 15. The details may be found in Ref. 16.

IV. Simulation Results

The maneuver chosen to illustrate the CAS performance is a high-g, fixed throttle turn, taken from Ref. 15. The simulation model was obtained from NASA Dryden Research Center and contains a detailed aerodynamic representation. Simple first-order models were used to model actuator dynamics, and no sensor noise was modeled. Starting at $M_0 = 0.6$ and $h_0 = 10{,}000$ ft, a roll rate command of 10 deg/s is given until $\phi = 80$ deg. The normal acceleration is maintained at 1.0 g during the roll maneuver. This is followed by a 5.0-g normal acceleration command that is maintained till the end of the maneuver. The throttle is maintained at 100% throughout the maneuver. This is a challenging maneuver because the aircraft starts at a subsonic Mach number and reaches a supersonic value at the end. All simulations were conducted using a predictor-corrector method with $\Delta t = 0.0058$ s.

Figure 8 illustrates a sample of the statistical measures used to evaluate the performance of the NN1-based inversion, by showing the effect that initial altitude and Mach number have on the rms error of the network output for left stabilator. The errors were computed by simultaneously computing the near exact inversion using the method of Ref. 15 at each integration step and taking the rms over each trajectory. Only the NN1 output was actually applied to control the aircraft in the simulation. It can be seen that for $h_0 = 10{,}000$ ft, the worst-case rms error increases for both low and high initial Mach numbers. That is because some portion of the flight is outside the training range of Fig. 8. For $M_0 = 0.5$ the starting Mach number lies below the training range, and for $M_0 = 0.9$ the Mach number exceeds 1.2 during a middle portion of the maneuver. A moving average of the rms error is illustrated in the lower portion of Fig. 8 that exhibits this effect as a function of time. Also note the maneuver initiated at $M_0 = 0.7$ and $h_0 = 20{,}000$ ft exhibits a large rms error that is sustained over the last 15 s of flight. Over this portion of the trajectory the Mach number is nearly constant at near

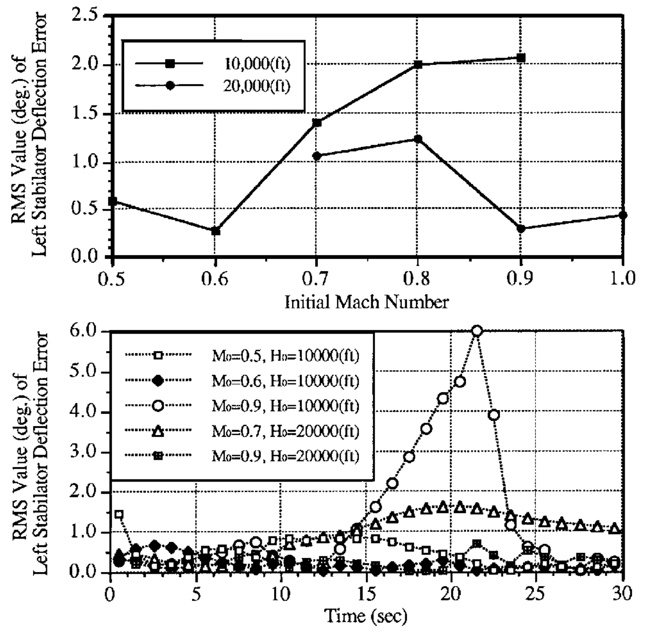

Fig. 8 Root mean square error of the NN1-based CAS for a series of fixed-throttle, high-*g* turn.

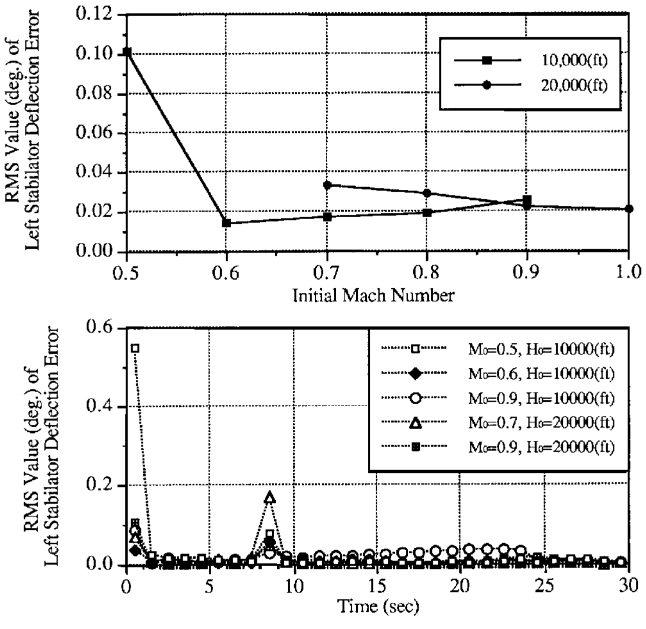

Fig. 9 Root mean square error of the NN1+NN2-based CAS for a series of fixed-throttle, high-*g* turn.

1.0. Although this lies within the training range, it corresponds to a region where the errors were larger than average due to the more complex variation that the aerodynamic data exhibits near Mach 1. That is, an off-line trained network with weights trained to provide good average performance over a given envelope of data can exhibit large errors within a portion of the envelope that has a high degree of complexity in its mapping.

Figure 9 examines statistical measures for the performance of the NN1+NN2-based inversion. Note that the rms errors in Fig. 9 are an order of magnitude lower than those of Fig. 8. The lower portion of Fig. 9 (moving average results) reveals that the errors (where they are larger) take on a totally different character in that they are of extremely short duration in comparison to the lower portion of Fig. 8. For example, the $M_0 = 0.5$, $h_0 = 10{,}000$ ft result initially starts outside the NN1 training envelope, which explains the higher initial error. However, this error is rapidly reduced by the on-line weight adjustment algorithm used in NN2. Rapid adaptation also occurs near $t = 8$ s into the maneuver when the high-*g* turn is initiated and roll rate is rapidly commanded to zero. The $M_0 = 0.7$,

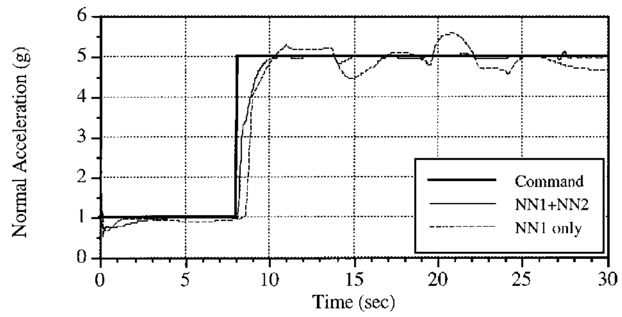

Fig. 10 Performance results of the NN2-based controller for the flight inside the off-line training region, $M_0 = 0.6$, $h_0 = 10{,}000$ ft.

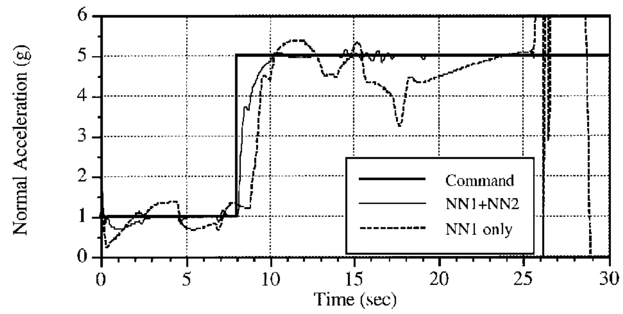

Fig. 11 Performance results of the NN2-based controller for the flight outside the off-line training region, $M_0 = 0.8$, $h_0 = 10{,}000$ ft.

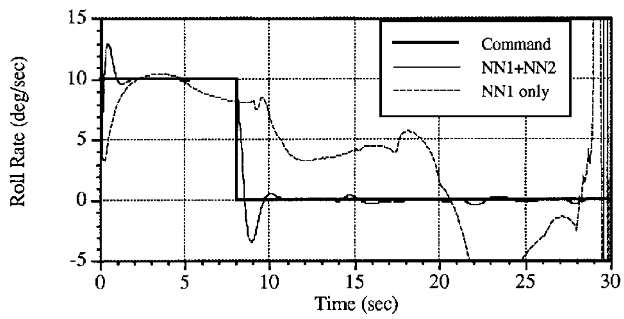

Fig. 12 Performance results of the NN2-based controller with 30% loss of left stabilator effectiveness, $M_0 = 0.6$, $h_0 = 10{,}000$ ft.

$h_0 = 20{,}000$ ft results demonstrate the on-line learning that occurs within the training envelope of NN1. Here, the sustained rms error of Fig. 8 (due to flight in the transonic region) is nearly completely eliminated. These results are typical of all of the statistical studies performed to date.

A sample simulation result for the high-*g* turn maneuver is given in Fig. 10. With full throttle, the flight speed reaches a Mach number of 1.1 at the end of the maneuver. Thus, this maneuver stays inside the NN1 training envelope. The resulting normal acceleration responses with NN1 and with NN1+NN2 are compared. Note that a significant improvement in tracking performance is achieved with the action of NN2. The NN1 tracking errors occur mainly as the aircraft passes through the transonic region. Figure 11 presents an example of what occurs when the aircraft is flown outside the training envelop for NN1, starting at $M_0 = 0.8$ and $h_0 = 10{,}000$ ft. Note that the control with NN1 alone becomes unstable. Figure 12 shows the roll rate response for the same case as in Fig. 10, but with an unexpected 30% loss in left stabilator effectiveness.

Another interesting feature is that all of the NN1+NN2 responses to the 5.0-*g* command (starting at 8.0 s in Figs. 10 and 11) are first order (no overshoot and a settling time of approximately 2.4 s), which is exactly what the outer-loop command augmentation logic of Fig. 4 was designed to accomplish. That is, if the inner-loop attitude orientation system response corresponds to that of exact inversion (and with knowledge of partial failure), we would expect a first-order tracking error response in normal acceleration. These responses show that NN2 is able to closely approximate perfect inversion without knowledge of the uncertainties due to imperfect inversion and sudden changes in aircraft configuration. Thus, robust performance is achieved in the sense that the aircraft not only remains stable, but also continues to satisfy handling qualities criteria. A predictable response (with minimal overshoot) is maintained in the presence of large-scale modeling uncertainties and partial failures. A possible explanation for such outstanding performance is that NN2 serves as a short-term memory device, which computes a best fit of its basis functions to cancel the NN1 inversion error for the local (current M, h, and α) conditions. Additional statistical and trajectory results evaluating NN1 and NN1+NN2 performance may be found in Ref. 16, including time histories of all the trajectory and control variables. Applications of this methodology to helicopter flight control and missile autopilot design can be found in Refs. 17 and 18. In these applications, the approximate inversions are based on a linearized aerodynamic representation of the force and moment maps at a single flight condition.

V. Conclusions

There are three main conclusions to be drawn from this research. First, feedback linearization is an alternative to gain scheduling, which holds the potential for simplifying the problem of flight control system design for high-performance fighter aircraft. Second, it is possible to design neural networks that are capable of internally representing the inversion needed to make feedback linearization a viable approach to real-time implementation of full envelope flight control systems. This requires that careful attention be given to the selection of the network architecture in concert with the demands of network training. Third, an on-line adaptive neural network is able to closely approximate perfect inversion without full knowledge of uncertainties. Experience with this approach has shown outstanding potential for rapid and accurate adaptation, even in the presence of partial actuation failure.

Appendix: Proofs of Theorems 1 and 2

Proof of Theorem 1: Define

$$V(\boldsymbol{e}, \tilde{\boldsymbol{w}}) = \sum_{i=1}^{n} V_i(\boldsymbol{e}_i, \tilde{\boldsymbol{w}}_i) \tag{A1}$$

$$V_i(\boldsymbol{e}_i, \tilde{\boldsymbol{w}}_i) = \begin{cases} \frac{1}{2}\boldsymbol{e}_i^T P_i \boldsymbol{e}_i + (1/2\gamma_i)\tilde{\boldsymbol{w}}_i^T \tilde{\boldsymbol{w}}_i, & \text{when } \|\boldsymbol{e}_i\|_{P_i} > E_i \\ E_i + (1/2\gamma_i)\tilde{\boldsymbol{w}}_i^T \tilde{\boldsymbol{w}}_i, & \text{when } \|\boldsymbol{e}_i\|_{P_i} \le E_i \end{cases} \tag{A2}$$

where $\boldsymbol{e}_i$ and $\tilde{\boldsymbol{w}}_i$ are the error state vector and the parameter estimation error vector for each channel, respectively. P_i is a (2×2) symmetric positive-definite matrix satisfying

$$P_i A_i + A_i^T P_i = -Q_i, \qquad Q_i = Q_i^T > 0 \tag{A3}$$

The existence of $P_i = P_i^T > 0$ is guaranteed by the fact that A_i is Hurwitz.

Outside the dead zone, the time derivative of V_i along any trajectory of Eqs. (21) and (22) is given by

$$\dot{V}_i = \tfrac{1}{2}\boldsymbol{e}_i^T Q_i \boldsymbol{e}_i + \boldsymbol{e}_i^T P_i \boldsymbol{b}\left\{\tilde{\boldsymbol{w}}_i^T \boldsymbol{\beta}_i + \left(\hat{\Delta}_i^* - \Delta_i\right)\right\} + (1/\gamma_i)\tilde{\boldsymbol{w}}_i^T \dot{\tilde{\boldsymbol{w}}}_i \tag{A4}$$

Using Eq. (15), it follows that

$$\dot{V}_i \le -\tfrac{1}{2}\boldsymbol{e}_i^T Q_i \boldsymbol{e}_i + \varepsilon_i\left|\boldsymbol{e}_i^T P_i \boldsymbol{b}\right| + \tilde{\boldsymbol{w}}_i^T\left[\boldsymbol{e}_i^T P_i \boldsymbol{b}\boldsymbol{\beta}_i + (1/\gamma_i)\dot{\tilde{\boldsymbol{w}}}_i\right] \tag{A5}$$

Adopting the form in Eq. (22) for the adaptation law reduces the last term in Eq. (A5) to zero. Thus, outside the dead zone we have

$$\dot{V} \le \sum_{i=1}^{n}\left(-\frac{1}{2}\boldsymbol{e}_i^T Q_i \boldsymbol{e}_i + \varepsilon_i\left|\boldsymbol{e}_i^T P_i \boldsymbol{b}\right|\right)$$

$$\le \sum_{i=1}^{n}\|\boldsymbol{e}_i\|_{P_i}\left\{-\frac{1}{2}\|\boldsymbol{e}_i\|_{P_i}\frac{\underline{\lambda}(Q_i)}{\bar{\lambda}(P_i)} + \varepsilon_i[\bar{\lambda}(P_i)]^{\frac{1}{2}}\right\} \tag{A6}$$

where $\|\boldsymbol{b}\|_2 = 1$ for $\boldsymbol{b}$ defined in Eq. (11). Each $\dot{V}_i$ is negative when the dead zone size is chosen to satisfy

$$E_i \ge \frac{2\varepsilon_i[\bar{\lambda}(P_i)]^{\frac{3}{2}}}{\underline{\lambda}(Q_i)} \tag{A7}$$

The ratio $\bar{\lambda}(P_i)/\underline{\lambda}(Q_i)$ is minimized with the particular choice $Q_i = I$ (Ref. 19). With this choice, Eq. (A7) reduces to

$$E_i \ge 2\varepsilon_i[\bar{\lambda}(P_i)]^{\frac{3}{2}} \tag{A8}$$

Inside the dead zone, each $\dot{V}_i = 0$ because $\dot{\tilde{\boldsymbol{w}}}_i = 0$. In summary, each V_i has the following properties:

1) $V_i \ge E_i > 0$ and V_i is continuous at the boundary of dead zone.

2) $\dot{V}_i < 0$ outside the dead zone; $\dot{V}_i = 0$ inside the dead zone. This implies that $\|\boldsymbol{e}_i(t)\|_2$ and $\|\tilde{\boldsymbol{w}}_i(t)\|_2$ are uniformly bounded.

For $\varepsilon_i = 0$ (no NN2 approximation error), the corresponding dead zone defined by Eq. (A8) shrinks to zero. The system equations (21) and (22) become

$$\dot{\boldsymbol{e}}_i = A_i \boldsymbol{e}_i + \boldsymbol{b}\tilde{\boldsymbol{w}}_i^T \boldsymbol{\beta}_i(t, \boldsymbol{e}, \tilde{\boldsymbol{w}}) \tag{A9}$$

$$\dot{\tilde{\boldsymbol{w}}}_i = -\gamma_i \boldsymbol{e}_i^T P_{0_i} \boldsymbol{b}\boldsymbol{\beta}_i(t, \boldsymbol{e}, \tilde{\boldsymbol{w}}), \qquad i = 1, 2, \ldots, n \tag{A10}$$

The origin $(\boldsymbol{e} = 0, \tilde{\boldsymbol{w}} = 0)$ is an equilibrium point of the system described by Eqs. (A9) and (A10). The Lyapunov function candidate

$$V = \sum_{i=1}^{n} V_i = \sum_{i=1}^{n}\left[\frac{1}{2}\boldsymbol{e}_i^T P_i \boldsymbol{e}_i + \frac{1}{2\gamma_i}\tilde{\boldsymbol{w}}_i^T \tilde{\boldsymbol{w}}_i\right] \tag{A11}$$

has a time derivative $\dot{V}$, which when evaluated along the solution of Eqs. (A9) and (A10) leads to

$$\dot{V} = \sum_{i=1}^{n}\dot{V}_i = \sum_{i=1}^{n}\left(-\frac{1}{2}\boldsymbol{e}_i^T \boldsymbol{e}_i\right) \le 0 \tag{A12}$$

Thus, from Eqs. (A11) and (A12) the equilibrium state of the system described by Eqs. (A9) and (A10) is uniformly stable (i.e., $\boldsymbol{e}, \tilde{\boldsymbol{w}} \in L_\infty$). The basis functions $\boldsymbol{\beta}_i$ are bounded when the input commands $(x_c, \dot{x}_c, \ddot{x}_c)$ and $\tilde{\boldsymbol{w}}_i$ are bounded, and it follows from Eq. (A9) that $\dot{\boldsymbol{e}}_i \in L_\infty$. Because

$$\frac{1}{2}\int_{t_0}^{\infty}\boldsymbol{e}_i^T \boldsymbol{e}_i \,\mathrm{d}\tau = -\int_{t_0}^{\infty}\dot{V}_i \,\mathrm{d}\tau = V_i(t_0) - V_i(\infty) < \infty$$

$$i = 1, 2, \ldots, n \tag{A13}$$

we have $\boldsymbol{e}_i \in L_2$, that is, $\boldsymbol{e}_i$ is square integrable. Because $\dot{\boldsymbol{e}}_i \in L_\infty$ and $\boldsymbol{e}_i \in L_2 \cap L_\infty$, it can be shown from Barbalat's lemma[20] that $\boldsymbol{e}_i(t) \to 0$ as $t \to \infty$. Thus, for $\varepsilon = 0$, the error system Eq. (21) is asymptotically stable. □

Proof of Theorem 2: The proof follows a similar line to that used in Ref. 11. For the ith channel, let the trajectory of Eq. (21) be on the boundary of the dead zone at $t = t_1$ and $t = t_2$, as shown in Fig. 3. Then we have

$$V_i\left(t_1^+\right) = E_i + (1/2\gamma_i)\|\tilde{\boldsymbol{w}}_i(t_1)\|_2^2 \tag{A14}$$

$$V_i\left(t_2^-\right) = E_i + (1/2\gamma_i)\|\tilde{\boldsymbol{w}}_i(t_2)\|_2^2 \tag{A15}$$

Let the error states be outside the dead zone for the open interval (t_1, t_2). If $\|\tilde{\boldsymbol{w}}_i(t_1)\|_2$ is finite, the length of the interval (t_1, t_2) must be finite because $\dot{V}_i \le -\eta_i < 0$ and

$$t_2 - t_1 \le \frac{1}{2\gamma_i\eta_i}\|\tilde{\boldsymbol{w}}_i(t_1)\|_2^2 < \infty \tag{A16}$$

Because $\dot{V}_i < 0$ outside the dead zone, the total time during which adaptation takes place must be finite. Thus, the number of intervals like (t_1, t_2) is finite or the sequence of the time interval durations converges to zero so that the total adaptation time is finite. Further, this implies that as $t \to \infty$, $\boldsymbol{e}_i(t)$ lies inside the dead zone and

$\tilde{w}_i(t) \to$ a constant because from Eq. (22) $\dot{\tilde{w}}_i = 0$ inside the dead zone.

From $\dot{V}_i < 0$ outside the dead zone, it follows that

$$V_i\left(t_2^-\right) - V_i\left(t_1^+\right) = (1/2\gamma_i)\left(\|\tilde{w}_i(t_2)\|_2^2 - \|\tilde{w}_i(t_1)\|_2^2\right) < 0 \quad \text{(A17)}$$

Hence, the norm of the estimation error decreases a finite amount every time the trajectory leaves and re-enters the dead zone. □

Acknowledgments

This research was supported by the Naval Air Warfare Center under Contract N62269-91-C-0038, and by the Army Research Office under Grant DAAH04-93-G-0002. The authors are grateful to P. K. A. Menon of Optimal Synthesis, Palo Alto, California, for his assistance in developing the comparisons in this study.

References

[1]Meyer, G., Su, R., and Hunt, L. R., "Application of Nonlinear Transformations to Automatic Flight Control," *Automatica*, Vol. 20, No. 1, 1984, pp. 103–107.

[2]Menon, P. K. A., Badgett, M. E., Walker, R. A., and Duke, E. L., "Nonlinear Flight Test Trajectory Controllers for Aircraft," *Journal of Guidance, Control, and Dynamics*, Vol. 10, No. 1, 1987, pp. 67–72.

[3]Reiner, J., Balas, G. J., and Garrard, W. L., "Robust Dynamic Inversion for Control of Highly Maneuverable Aircraft," *Journal of Guidance, Control, and Dynamics*, Vol. 18, No. 1, 1995, pp. 18–24.

[4]Reigelsperger, W. C., Hammett, K. D., and Banda, S. S., "Lateral-Directional, Full Envelope Control Law Design for F-16 with Thrust Vectoring," *Proceedings of the AIAA Guidance, Navigation, and Control Conference* (Baltimore, MD), AIAA, Washington, DC, 1995, pp. 706–717.

[5]Sanner, R. M., and Slotine, J. J. E., "Gaussian Networks for Direct Adaptive Control," *IEEE Transactions on Neural Networks*, Vol. 3, No. 6, 1992, pp. 837–863.

[6]Gomi, H., and Kawato, M., "Neural Network Control for a Closed-Loop System Using Feedback-Error-Learning," *Neural Networks*, Vol. 6, No. 7, 1993, pp. 933–946.

[7]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.

[8]Troudet, T., Garg, S., and Merrill, W., "Neural Network Application to Aircraft Control System Design," *Journal of Guidance, Control, and Dynamics*, Vol. 16, No. 4, 1993, pp. 738–747.

[9]Sing, S. N., and Wells, W. R., "Direct Adaptive and Neural Control of Wing-Rock Motion of Slender Delta Wings," *Journal of Guidance, Control, and Dynamics*, Vol. 18, No. 1, 1995, pp. 25–30.

[10]Sadhukhan, D., and Feteih, S., "F8 Neurocontroller Based on Dynamic Inversion," *Journal of Guidance, Control, and Dynamics*, Vol. 19, No. 1, 1996, pp. 150–156.

[11]Napolitano, M. R., and Kincheloe, M., "On-Line Learning Neural-Network Controllers for Autopilot Systems," *Journal of Guidance, Control, and Dynamics*, Vol. 18, No. 6, 1995, pp. 1008–1015.

[12]Peterson, B. B., and Narendra, K. S., "Bounded Error Adaptive Control," *IEEE Transactions on Automatic Control*, Vol. AC-27, No. 6, 1982, pp. 1161–1168.

[13]Poggio, T., and Girosi, F., "Networks for Approximation and Learning," *Proceedings of the IEEE*, Vol. 78, No. 9, 1990, pp. 1481–1497.

[14]Rumelhart, D. E., Hinton, G. E., and Williams, R. J., "Learning Internal Representations by Error Propagation," *Parallel Distributed Processing*, edited by D. E. Rumelhart and J. L. McClelland, Vol. 1, MIT Press, Cambridge, MA, 1986.

[15]Menon, P. K. A., "Nonlinear Command Augmentation System for a High Performance Fighter Aircraft," *Proceedings of the AIAA Guidance, Navigation, and Control Conference* (Monterey, CA), AIAA, Washington, DC, 1993, pp. 720–730.

[16]Kim, B.-S., "Nonlinear Flight Control Using Neural Networks," Ph.D. Thesis, School of Aerospace Engineering, Georgia Inst. of Technology, Atlanta, GA, Dec. 1993.

[17]Leitner, J., Calise, A. J., and Prasad, J. V. R., "Analysis of Adaptive Neural Networks for Helicopter Flight Control," *AIAA Guidance, Navigation, and Control Conference* (Baltimore, MD), AIAA, Washington, DC, 1995, pp. 871–879.

[18]McFarland, M., and Calise, A. J., "Neural Networks for Stable Adaptive Control of Air-To-Air Missiles," *AIAA Guidance, Navigation, and Control Conference* (Baltimore, MD), AIAA, Washington, DC, 1995, pp. 871–879.

[19]Patel, R. V., and Toda, M., "Qualitative Measures of Robustness for Multivariable Systems," Joint Automatic Control Conf., Paper TP8-A, San Francisco, CA, Aug. 1980.

[20]Astrom, K. J., and Wittenmark, B., *Adaptive Control*, Addison–Wesley, Reading, MA, 1995, p. 205.