JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 24, No. 5, September–October 2001

Adaptive Neurocontroller for a Nonlinear Combat Aircraft Model

Piero A. Gili* and Manuela Battipede†
Politecnico di Torino, 10129 Turin, Italy

This paper introduces an adaptive controller, based on neural networks use, for a nonlinear six-degrees-of-freedom combat aircraft model. This controller is based on the determination of the inverse dynamics of aircraft through a state feedback, taking advantage of the neural network online learning ability in dealing with any changes of the aircraft dynamics during the flight. By comparing the online and offline training, how effective the neural controller is in adaptation is investigated and highlighted in situations involving highly demanding maneuvers as well as sudden environmental disturbances. The neural controller is designed according to the reference model adaptive direct inverse scheme. The behavior of this controller is compared with that of a conventional linear stability and control augmentation system (normal acceleration limiter), implemented under military handling qualities and high maneuverability requirements. The online training of the nonlinear neural controller is based on a recursive prediction error algorithm, whose performance results from a proportional derivative performance index formulation. The stability analysis demonstrates how the extra degree of freedom, provided by the derivative term, makes the algorithm more robust than the standard recursive least-squares method. Performance is verified through numerical simulations.

Nomenclature

a, b	=	reference model matrices
a_n	=	normal acceleration
C_L	=	lift coefficient
E	=	quadratic error
e_y	=	error signal
F_{long}	=	stick force
F_z	=	total force component along the z body axis
g	=	gravity acceleration
h	=	altitude
$\boldsymbol{K}$	=	gain matrix
K_d	=	derivative gain coefficient
K_p	=	proportional gain coefficient
k	=	time step
M	=	Mach number
m	=	aircraft mass
P	=	covariance matrix
PI	=	performance index
Q	=	pitch rate
t	=	time
u	=	control signal
$\hat{u}$	=	approximate control signal
V	=	Lyapunov function
$\boldsymbol{W}_1$	=	weights matrix from input layer to hidden layer
$\boldsymbol{W}_2$	=	weights matrix from hidden layer to output layer
x_a	=	accelerometer longitudinal position
y	=	output signal
$\hat{y}$	=	approximate output signal
α	=	angle of attack
β	=	sideslip angle
Δt	=	sample time
δ	=	deflection angle
$\dot{\delta}$	=	deflection rate
$\hat{\delta}$	=	approximate deflection rate
$\boldsymbol{\Theta}$	=	weights vector
λ	=	forgetting factor
ϕ, θ, ψ	=	Euler angles
$\boldsymbol{\Psi}$	=	derivative vector

Subscripts

a	=	ailerons
h	=	stabilator
lef	=	leading-edge flaps
max	=	maximum
r	=	rudder
ref	=	reference
th	=	throttle

Presented as Paper 98-4485 at the AIAA Guidance, Navigation, and Control Conference, Boston, MA, 10–12 August 1998; received 28 February 2000; revision received 31 October 2000; accepted for publication 9 February 2001.

*Associate Professor, Dipartimento di Ingegneria Aeronautica e Spaziale, C.so Duca degli Abruzzi 24; gili@polito.it. Member AIAA.

†Ph.D Researcher, Dipartimento di Ingegneria Aeronautica e Spaziale, C.so Duca degli Abruzzi 24; gili@polito.it. Member AIAA.

Introduction

TRADITIONAL controller design usually involves complex and extensive mathematical analysis, which implies high cost and cannot guarantee a good performance level in the whole flight envelope. Neural control has recently been put forward as a valid means to overcome most of the typical limitations of the classical control techniques. The increasing enthusiasm of the scientific community was promptly followed by concrete interest from the aviation industry and the armed forces, which recognized in neural control a way to obtain a more cost-effective performance.

A renowned contribution to research in flight dynamics neural control comes from the research group of A. J. Calise, who proposed a control scheme based on inversion of a linearized plant model (feedback linearization) combined with a multilayer perceptron neural network (NN) that compensate adaptively for the effects of the inversion errors. This scheme has proven to be effective over a wide application range: 1) systems operating in regimes characterized by highly nonlinear aerodynamics,[1] 2) systems displaying multitime scale behavior and thus rapidly varying nonlinear dynamics,[2,3] 3) systems characterized by a high degree of uncertainty,[4] and 4) systems demanding the maintenance of a certain level of the handling qualities even after failures in the actuation channels.[5,6] Furthermore, the same method proved to be effective in improving performance of an existing gain-scheduling autopilot, designed for an agile antiair missile,[7] both in conditions between design points and beyond the field covered by the gain scheduling.

The augmentation of an existing proportional-integral controller through an adaptive NN has been investigated also by KrishnaKumar and Kulkarni,[8] who have developed an inverse adaptive neural controller for a nonlinear engine model: The NN is trained to allow the controller to keep stationary performance in the presence of big engine changes. The uniqueness of their approach is in the NN architecture, which is fully forward connected; the learning rule for

online tuning of the weight is the standard back-propagation algorithm (SBPA).

Another strong stimulus in this field has been given by the group of Napolitano, who focused the last ten years of their research activity on the fixed-wing aircraft neural control. The stress is placed upon the role of the extended back-propagation algorithm (EBPA), which was demonstrated[9] to be particularly effective in terms of accuracy and learning speed for the online learning of NNs. To a great extent, applications concern restructurable flight control systems. Both actuator and sensor failures are addressed: the reconstruction of the control law is performed online to bring the aircraft back to a new equilibrium condition, after failure to vital control surfaces[10] (AFDIA) and after failure to sensors,[11] unsupplied with physical redundancy (SFDIA). The real-time capabilities and the integration between the AFDIA and SFDIA schemes have also been investigated for a fault-tolerant flight control system featuring hardware based on an online learning parallel NNs system.[12]

In this paper, we adopt a predictor–corrector control scheme, which is based on the specialized learning proposed by Psaltis et al.[13] and belongs to the direct inverse adaptive control strategy. In the early 1990s, many papers have detailed the calculation of the unknown plant Jacobian, which has been defined as the main obstacle in direct inverse adaptive control. A complete survey of the relative literature is given in Ref. 14 (pp. 43–77). The idea of using an NN to model the forward dynamics of the plant and approximate the plant Jacobian was first proposed by Jordan and Jacobs,[15] who named their scheme "forward and inverse modeling." Afterward, many authors tackled the problem of improving performance in the identification of the forward model to enhance the online estimation of the plant Jacobian.[16,17] In this paper, the recursive prediction algorithm, proposed by Chen et al.[18] for the forward model, is used to implement the back propagation through the model, typical of the direct inverse adaptive control strategy. As for the back-propagation algorithm, the recursive identification takes advantage of an adaptive learning rate that allows faster convergence rates, with the drawback of more stringent convergence requirements.[14]

In connection, demontration will be given of how the introduction of a derivative term, in the classical proportional performance index, greatly improves the convergence features, broadening the stability margins and reducing the overall system response error.

The controller is used as a normal-acceleration control augmentation system (CAS) for an F-16 combat aircraft model in six degrees of freedom (DOF). The performance analysis of the controller as a normal-acceleration tracking is particularly interesting because normal-acceleration is a nonlinear function of the longitudinal state and control variables.

Controller Classification

KrishnaKumar[19] suggests that intelligent controllers can be classified into four classes or levels, from level 0 (i.e., nonintelligent or nonadaptive) to level 3 (i.e., emergency decision generator):

1) Level 0 (Ref. 20): self-improvement (minimization) of the error defined as $Y_{act} - Y_{des}$, where Y_{act} is the actual output of the system and Y_{des} is the desired one.

2) Level 1 (Ref. 9): self-improvement of the error and of a control parameter (adaptive, able to work in non-nominal conditions).

3) Level 2 (Ref. 21): self-improvement of the error, of the control parameter, and of a cost function over time (adaptive optimal control).

4) Level 3: self-improvement of the error, of the control parameter, of the cost function over time, and of the planning of an alternative cost function in emergency situations.

Following this classification, the controller dealt with here can be classified as level 1.

As we can see from the multi-level categorization,[14] that has been reported in Fig. 1, the neural control strategies are manifold. Most of these control structures and their terminology are borrowed from the field of conventional adaptive control. The adopted control strategy belongs to the direct inverse control schemes category, implemented in a version that also considers a reference model, so we refer to it as the "reference model adaptive direct inverse" controller. *Direct* means that the overall error is calculated by comparing the plant output and the desired response; the term *inverse* implies that an NN is trained to emulate the inverse dynamics of the plant. Another NN is trained to emulate the forward dynamics of a plant for the calculation of the plant Jacobian: It is placed in cascade with the previous one, which acts as the actual controller, leading to the scheme of Fig. 2. As shown in Fig. 2, the overall system error is back propagated through the plant emulator up to the controller, which is finally trained. The forward path is made up by the actual controller and the emulator (Fig. 3) and is the heart of the system: also successfully used in the past for various applications (see Ref. 22 for an extensive bibliography), it is considered one of the most attractive employments of the back propagation.

The technique can be used both for single-input/single-output (SISO) and multi-input/multi-output (MIMO)[23–25] control systems.

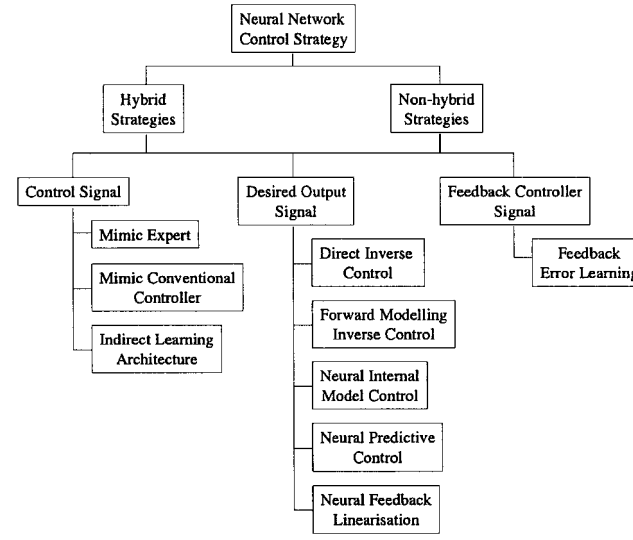

Fig. 1 Neural control strategies.

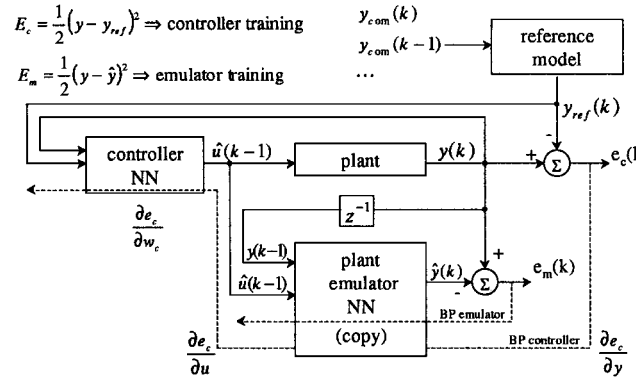

Fig. 2 Scheme of an adaptive inverse serial predictor–corrector neural controller.

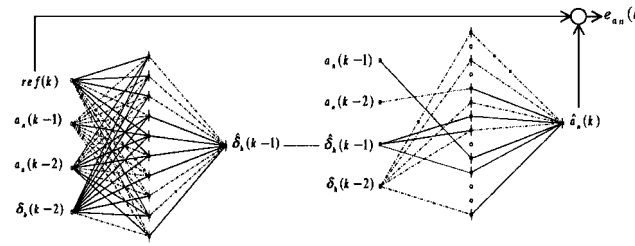

Fig. 3 Forward path.

Aircraft Mathematical Model

This section briefly describes the plant, which is a combat aircraft, flying in a wing-level, steady-state condition. The mathematical

model is nonlinear in six DOF: the aerodynamic model[26] is based on the wind tunnel data obtained in 1979 by Nguyen et al.[27] The limits of reliability of the aerodynamic database and the control surfaces mechanical limits can be approximately

$$\begin{aligned} -20\,\text{deg} \le \alpha \le 60\,\text{deg} \\ -30\,\text{deg} \le \beta \le 30\,\text{deg} \\ -25\,\text{deg} \le \delta_h \le 25\,\text{deg} \qquad & \dot{\delta}_{h\max} = 60\,\text{deg/s} \\ -21.5\,\text{deg} \le \delta_a \le 21.5\,\text{deg} \qquad & \dot{\delta}_{a\max} = 80\,\text{deg/s} \\ -30\,\text{deg} \le \delta_r \le 30\,\text{deg} \qquad & \dot{\delta}_{r\max} = 120\,\text{deg/s} \end{aligned} \tag{1}$$

The limits set by the postcombustor turbojet engine model, in terms of altitude and Mach number, are summarized in Fig. 4 and compared with the flight envelope diagram of the real aircraft.

The model is fitted out with leading-edge flaps whose deflection is automatically set by the values of the angle of attack and the Mach number, according to the limits

$$-25\,\text{deg} \le \delta_{\text{lef}} \le 25\,\text{deg} \qquad \dot{\delta}_{\text{lef}_{\max}} = 25\,\text{deg/s} \tag{2}$$

Model Longitudinal Dynamics

Because the controller is mainly concerned with the aircraft longitudinal dynamics, the lateral-directional dynamics were not considered. The main longitudinal characteristics of the aircraft are summarized as follows:

1) The center of gravity nominal position is selected to minimize the trim drag. However, in this balance configuration, the aircraft exhibits a light, static longitudinal instability at low Mach numbers, and this causes the establishment of an anomalous dynamic longitudinal behavior. To satisfy the flying qualities' requirements, the model was equipped with a longitudinal stability augmentation system (SAS) based on the angle-of-attack feedback: the neural stability and control augmentation system (SCAS) is supposed to act on the closed loop made up by the aircraft and the SAS.

2) Hypothetically, the aircraft could be trimmed up to $\alpha \cong 66$ deg, that is, up to the upper limit of the stabilator deflection. However, for $\alpha > 25$ deg, the stabilator starts losing its effectiveness, as it is almost completely stalled. In that case, a suitable CAS is advisable to prevent the angle of attack exceeding this limit.

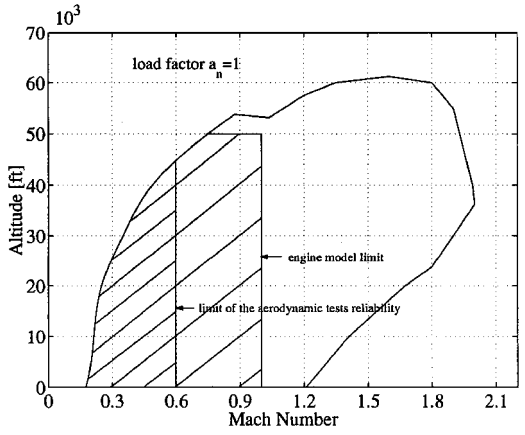

Fig. 4 Altitude–Mach flight envelope.

Reference CAS Model

The longitudinal fly-by-wire SCAS system, used as the term of comparison with the neural nonlinear controller, is shown in Fig. 5. It has been designed by making use of the linear systems classical control techniques, which are made effective on the whole flight envelope through the gain scheduling of the dynamic pressure. Basically, the SAS system is coupled with the normal acceleration CAS that has a twofold task:

1) It allows the pilot to perform sudden and rough maneuvers (pull up or turn), making sure to not exceed the structural limits of the airframe as well as his or her own physical limits, thanks to the angle-of-attack feedback, the commanded normal acceleration can be limited by either the dynamic pressure or the angle of attack.

2) It limits the wing-level, steady-state flight at the trim value of $\alpha = 25$ deg.

The lateral-directional dynamics have been left uncontrolled, but constant attention was paid to the trend of all state variables, to ensure that the lateral-directional ones are not seriously involved in the performed maneuvers. To sum up, the control vector will have a hybrid form, between a completely controlled aircraft and an uncontrolled one:

$$\boldsymbol{U} = (F_{\text{long}}, \delta_a, \delta_r, \delta_{\text{th}}) \tag{3}$$

for both the conventional linear SCAS system and for the neural nonlinear controller, which must perform exactly the same tasks as the conventional one.

Neural CAS

In analogy with the conventional SCAS presented in the preceding section (i.e., the normal acceleration SCAS system), the adaptive neural controller was planned as a SISO system that controls a six-DOF plant.

Referring to Fig. 2, the following subsection will provide a brief description of the main elements of the system along with a general explanation about the training procedures.

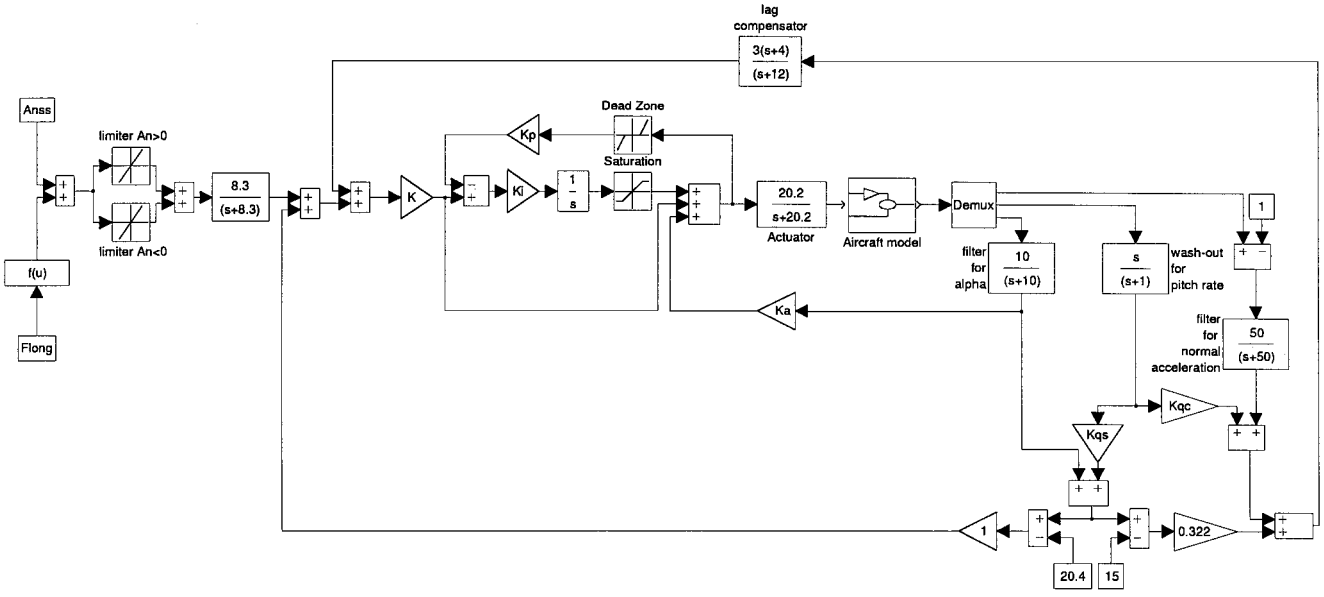

Fig. 5 SCAS linear system.

Plant

The plant, namely the aircraft model previously illustrated, is represented by 13 differential equations of the first order, nonlinear in the state and control variables:

$$\dot{\boldsymbol{X}} = f(\boldsymbol{X}, \boldsymbol{U}) \tag{4}$$

The output is the normal acceleration calculated with respect to body axes and depurated by the gravity component:

$$a_n = [(-F_z/m + x_a \dot{Q})/g] - \cos\theta \cos\phi \tag{5}$$

NN Emulator

The emulator has been modeled as a second-order system, according to the model IV scheme proposed by Narendra and Parthasarathy.[28] It is represented by the three-layer NN shown in Fig. 3 (right). The input layer is the second-order regressor vector, the hidden-layer neurons have a bipolar hyperbolic tangent activation function:

$$f(x) = 1 - 2/(e^{2x} + 1) \tag{6}$$

and the output-layer neuron has a linear activation function. This structure is commonly applied in neural modeling and identification problems for nonlinear dynamic systems;[16] under this aspect, no intentional action was made to optimize the shape[29] and the slope[30] of the activation functions, nor the order of the system.[31]

The emulator has been trained offline with the Levenberg-Marquardt method[32] according to the version described by Fletcher.[33] Because the analysis of the controller behavior concerns a single steady-state flight condition ($M = 0.6$ and $h = 30{,}000$ ft), the emulator is not trained online even if in specific situations, such as in a failure event,[25] the online training of the emulator is an important ingredient in the controller architecture.

The network gave off good generalization features thanks to the pruning procedure carried out using the so-called optimal brain surgeon (OBS) proposed by Hassibi and Stork.[34] (This technique consists of eliminating a set number of connections for every iteration and evaluating the network's subsequent capability of generalizing, with the constraint that the hidden layers cannot be eliminated.) In longitudinal maneuvers up to $a_n = 6\,g$, the maximum relative errors are lower than 10^{-4}.

NN Controller

The idea involves training the controller to approximate the inverse model:

$$\hat{\delta}_h(k-1) = \hat{f}^{-1}\left[a_{n_{\text{ref}}}(k), a_n(k-1), a_n(k-2), \delta_h(k-2)\right] \tag{7}$$

The architecture of the NNs is identical to that of the forward model, but unlike the procedure used to identify the forward model (global method), the inverse model was trained through the technique of the recursive identification,[35] which requires processing data online as they become available. The real-time identification makes the nonlinear inverse control feasible because the inverse model identification is performed locally for a narrow neighborhood of the flight condition, and the weights updating allows overstepping of the neighborhood boundaries, specializing the inverse model for the actual operative condition in the flight envelope. Moreover, the adaptability of the inverse model allows dynamic control of either minimum-phase or nonminimum-phase plants. In fact, numerical experiments[36] on a discrete linear second-order system show that the adaptive direct inverse controller exhibits a continuum behavior as the zero is moved from inside the unit circle in the z-plane to outside the circle itself, and it seems to work reasonably well even with the zero exactly on the unit circle.

The control strategy has been articulated in two phases, according to the method proposed by Psaltis et al.[13] The first phase is the generalized training, which is done using the same gradient descent algorithm adopted to train the forward model. The second phase is the specialized training, which acts on the overall system. This expedient helps to solve the problem of minimizing the nonlinear function

$$E = \tfrac{1}{2}(u - \hat{u})^2 \tag{8}$$

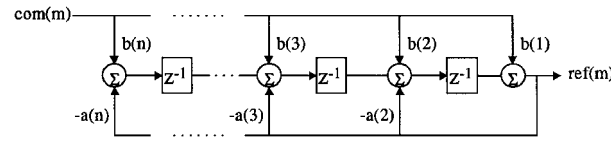

Fig. 6 Transposed direct form 2 filter.

because it applies the gradient descent algorithm on a function that features milder nonlinearities:

$$E = \tfrac{1}{2}(y_{\text{ref}} - y)^2 \cong \tfrac{1}{2}(y_{\text{ref}} - \hat{y})^2 \tag{9}$$

Actually, Ref. 13 presents a simulation example that shows how the generalized learning has a beneficial effect on the specialized learning as it provides improved initial conditions while the specialized learning allows escape from possible local minima, changing the descent direction that was previously chosen by the generalized training.

Suppose, to redefine the cost function in Eq. (9) as

$$E = \tfrac{1}{2}K_p(y_{\text{ref}} - y)^2 = \tfrac{1}{2}K_p e_y^2 \tag{10}$$

the specialized training is based on the step-by-step updating of the $\boldsymbol{\Theta}$ vector, which groups in vector shape the couples of matrices $\boldsymbol{W}_1$, $\boldsymbol{W}_2$. Equation (11) represents the kth time step of the recursive pseudolinear regression algorithm[37] (RPLR):

$$\begin{aligned}
\boldsymbol{K}(k) &= \boldsymbol{P}(k-1)\boldsymbol{\Psi}(k)\left[\lambda I + K_p \boldsymbol{\Psi}^T(k)\boldsymbol{P}(k-1)\boldsymbol{\Psi}(k)\right]^{-1} \\
\boldsymbol{\Theta}(k) &= \boldsymbol{\Theta}(k-1) + K(k)K_p e_y(k) \\
\boldsymbol{P}(k) &= (1/\lambda)\left[I - \boldsymbol{K}(k)K_p\boldsymbol{\Psi}^T(k)\right]\boldsymbol{P}(k-1)
\end{aligned} \tag{11}$$

where the $\boldsymbol{\Psi}$ vector determines the gradient descent direction:

$$\boldsymbol{\Psi}(k) = \nabla_\Theta \hat{y}(k) = \frac{\partial \hat{y}(k)}{\partial u(k-1)} \cdot [\nabla_\Theta u(k-1)] \tag{12}$$

The recursive scheme to update the inverse model weights comes from the mean square theorem application. The stability conditions of this method will be investigated in the following section.

Reference Model

The neural CAS acts, making the plant track the reference signal as closely as possible. Even if this appears to be its most attractive feature, it may happen that the reference signal is overdemanding, meaning that the plant runs the risk of being damaged in the attempt to track it. It is also unthinkable to expect that the aircraft will be able to accomplish sharp maneuvers such as the step: The inertial effects and the mechanical limits of the control surfaces impose limited maneuver rate. The controller would try to force the aircraft to track the signal overmaneuvering it, causing dangerous overshoot of the variable under control.

To prevent this problem, the reference model acts as a digital filter, smoothing the signal before it is used as a reference. The adopted scheme is the one sketched in Fig. 6. In the examples reported in the next section, this structure has been generally used with zero order.

This concept is analogous to that used in the classic linear control theory, where the reference model is defined by a single pair of complex dominant poles whose parameters are set by the handling qualities requirements. Under this aspect, the digital filter used is equivalent to the most demanding situation in the range imposed for level 1 by the MIL-STD-1797A.[38] A milder reference model makes the controller task easier, but the response time increases in accordance with the reference model parameters.

Stability Analysis

In terms of convergence speed and steady state error, the controller behavior is very sensitive to the error function that has to be minimized. Intuitively, the training algorithm could benefit from a further piece of information indicating if the error is increasing or decreasing.

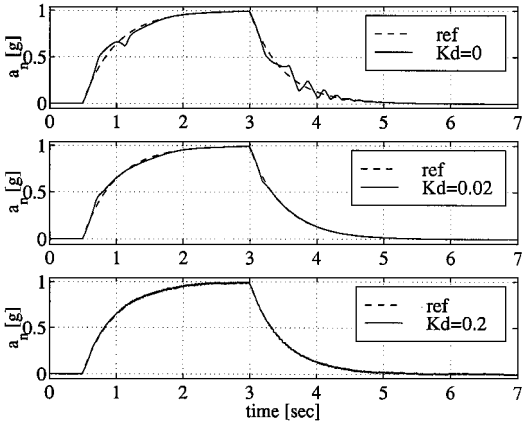

Fig. 7 Examples of response to the double-step signal as a function of K_d.

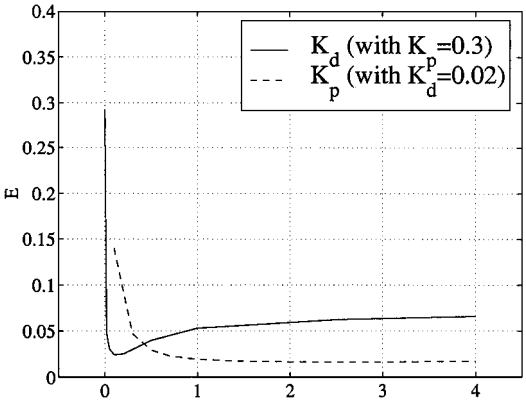

Fig. 8 Trend of the overall error as a function of K_p (with K_d constant), and K_d (with K_p constant), in the double-step test case.

Once it has been ascertained that the cost function

$$E = \tfrac{1}{2}K_p(y_{\text{ref}} - y)^2 + \tfrac{1}{2}K_d(\dot{y}_{\text{ref}} - \dot{y})^2 = \tfrac{1}{2}K_p e_y^2 + \tfrac{1}{2}K_d \dot{e}_y^2 \quad (13)$$

is minimized by the same Θ solution as the cost function in Eq. (10), good results can be obtained by adding the derivative term directly in the definition of the performance index:

$$PI = K_p(y_{\text{ref}} - y) + K_d(\dot{y}_{\text{ref}} - \dot{y}) \quad (14)$$

This is used in the iterative updating of vector $\boldsymbol{\Theta}$, according to

$$\boldsymbol{\Theta}(k) = \boldsymbol{\Theta}(k-1) + K(k)PI(k) \quad (15)$$

It should be noted that when engaging the derivative term in Eq. (15), noise may occur in the iterative process as a result of objective difficulties in surveying the $\dot{y}$ signal, as in the present application. However, the theoretical analysis reveals that the proportional derivative performance index (PDPI) enhances the system behavior, as shown in Figs. 7 and 8, and broadens the system stability field, as will be demonstrated in the following. Figure 8 is representative of the step maneuver of Fig. 7, but similar trends can be traced for any kind of maneuver.

Conditions on the K_p and K_d parameters are gained by using the Lyapunov method to ensure the local stability of the closed-loop system. Define the discrete Lyapunov function as

$$V(k) = \tfrac{1}{2}e_y^2(k) \quad (16)$$

where

$$e_y(k) = y_{\text{ref}}(k) - y(k) = a_{n_{\text{ref}}}(k) - a_n(k) \quad (17)$$

is the prediction error. According to the second theorem of Lyapunov, the closed-loop local stability is guaranteed by the condition

$$\Delta V(k) = V(k+1) - V(k) = \tfrac{1}{2}\left[e_y^2(k+1) - e_y^2(k)\right] < 0 \quad (18)$$

The error difference can be expressed as

$$e_y(k+1) = e_y(k) + \left[\nabla_{\theta(k-1)}e_y(k)\right]^T \Delta\boldsymbol{\Theta}(k-1) \quad (19)$$

where $\Delta\boldsymbol{\Theta}(k-1)$ is the updating of the weights vector at the time step k:

$$\Delta\boldsymbol{\Theta}(k-1) = \boldsymbol{\Theta}(k) - \boldsymbol{\Theta}(k-1) = \boldsymbol{P}(k)\boldsymbol{\Psi}(k)[K_p e_y(k) + K_d \dot{e}_y(k)] \quad (20)$$

Suppose that, in a restricted neighborhood of the solution

$$\dot{e}_y(k) = [e_y(k) - e_y(k-1)]/\Delta t \cong [e_y(k+1) - e_y(k)]/\Delta t \quad (21)$$

thus

$$\begin{aligned}\Delta\boldsymbol{\Theta}(k-1) &= \boldsymbol{P}(k)\boldsymbol{\Psi}(k)\{K_p e_y(k) + K_d[e_y(k+1) - e_y(k)]/\Delta t\} \\ &= \boldsymbol{P}(k)\boldsymbol{\Psi}(k)(K_d/\Delta t)e_y(k+1) \\ &\quad + \boldsymbol{P}(k)\boldsymbol{\Psi}(k)(K_p - K_d/\Delta t)e_y(k) \end{aligned} \quad (22)$$

Substituting Eq. (22) in Eq. (19) results in

$$\begin{aligned}e_y(k+1) &= e_y(k) - \boldsymbol{\Psi}^T(k)\boldsymbol{P}(k)\boldsymbol{\Psi}(k)(K_d/\Delta t)e_y(k+1) \\ &\quad - \boldsymbol{\Psi}^T(k)\boldsymbol{P}(k)\boldsymbol{\Psi}(k)(K_p - K_d/\Delta t)e_y(k)\end{aligned} \quad (23)$$

This gives the expression for the error at the time step $k+1$:

$$e_y(k+1) = \frac{1 + \boldsymbol{\Psi}^T(k)\boldsymbol{P}(k)\boldsymbol{\Psi}(k)[(K_d/\Delta t) - K_p]}{1 + \boldsymbol{\Psi}^T(k)\boldsymbol{P}(k)\boldsymbol{\Psi}(k)(K_d/\Delta t)}e_y(k) \quad (24)$$

Hence, substituting Eq. (24) in Eq. (18):

$$\begin{aligned}&\Delta V(k) \\ &= \frac{1}{2}\left\{\frac{\left[1 + \boldsymbol{\Psi}^T(k)\boldsymbol{P}(k)\boldsymbol{\Psi}(k)(K_d/\Delta t - K_p)\right]^2}{\left[1 + \boldsymbol{\Psi}^T(k)\boldsymbol{P}(k)\boldsymbol{\Psi}(k)(K_d/\Delta t)\right]^2}e_y^2(k) - e_y^2(k)\right\} \\ &= \frac{e_y^2(k)}{2\left[1 + \boldsymbol{\Psi}^T(k)\boldsymbol{P}(k)\boldsymbol{\Psi}(k)(K_d/\Delta t)\right]^2}\boldsymbol{\Psi}^T(k)\boldsymbol{P}(k)\boldsymbol{\Psi}(k) \\ &\quad \times\left[\boldsymbol{\Psi}^T(k)\boldsymbol{P}(k)\boldsymbol{\Psi}(k)\left(K_p^2 - 2\frac{K_d K_p}{\Delta t}\right) - 2K_p\right] \\ &= -e_y^2(k)\frac{\boldsymbol{\Psi}^T(k)\boldsymbol{P}(k)\boldsymbol{\Psi}(k)}{\left[1 + \boldsymbol{\Psi}^T(k)\boldsymbol{P}(k)\boldsymbol{\Psi}(k)(K_d/\Delta t)\right]^2} \\ &\quad \times\left\{K_p\left[1 - \left(\frac{K_p}{2} - \frac{K_d}{\Delta t}\right)\boldsymbol{\Psi}^T(k)\boldsymbol{P}(k)\boldsymbol{\Psi}(k)\right]\right\}\end{aligned} \quad (25)$$

It follows that the inequality in Eq. (18) is met if

$$K_p\left[1 - (K_p/2 - K_d/\Delta t)\boldsymbol{\Psi}^T(k)\boldsymbol{P}(k)\boldsymbol{\Psi}(k)\right] > 0 \quad (26)$$

Suppose to have $K_d = 0$, stability would be guaranteed by condition

$$K_p\left[1 - (K_p/2)\boldsymbol{\Psi}^T(k)\boldsymbol{P}(k)\boldsymbol{\Psi}(k)\right] > 0 \quad (27)$$

which, due to positive definiteness of covariance matrix $\boldsymbol{P}$, becomes

$$K_p > 0, \qquad 0 < \boldsymbol{\Psi}^T(k)\boldsymbol{P}(k)\boldsymbol{\Psi}(k) < 2/K_p \quad (28)$$

according to similar results obtained in Ref. 14 (pp. 113–119). This condition can be translated in a limitation of the maximum eigenvalue of the $\boldsymbol{P}(k)$ matrix. In this respect, some numerical methods, such as the constant trace technique, have been developed to prevent the maximum eigenvalue of the $\boldsymbol{P}(k)$ matrix to exceed the upper bound set by Eq. (28).

Now supposing that $K_d \neq 0$, the inequality in Eq. (26) has a double solution:

$$K_p > 0, \qquad K_d < K_p \Delta t/2$$
$$\Rightarrow 0 < \Psi^T(k)\boldsymbol{P}(k)\Psi(k) < 1/(K_p/2 - K_d/\Delta t) \tag{29}$$
$$K_p > 0, \qquad K_d \geq K_p \Delta t/2$$
$$\Rightarrow \Psi^T(k)\boldsymbol{P}(k)\Psi(k) > -1/|K_p/2 - K_d/\Delta t| \tag{30}$$

In the first case, it can be noted that the stability criterion is less stringent than in Eq. (28), provided K_d is positive. Nevertheless, the effects of the K_d coefficient on the algorithm robustness are much more evident in the second case. In fact, by taking the second set of conditions on the K_p and K_d coefficients, stability is always guaranteed, because the condition given by Eq. (30) on the $\Psi^T(k)\boldsymbol{P}(k)\Psi(k)$ matrix is always verified.

Numerical Results

All the numerical results refer to the same trimmed flight initial condition, namely wings-level horizontal flight at $M = 0.6$ at the altitude of 30,000 ft. The aircraft is equipped with an SAS, so that stability is always ensured. Another common aspect of all the numerical results is that the maneuver is accomplished only through the action of the stabilator while the throttle is kept constant ($\delta_{th} = \text{cost} = 0.315$). The upper diagram of each figure refers to the controller performance in terms of comparison between the reference and the actual a_n signal, obtained from both the adaptive neural and the gain-scheduling controller, and the lower diagram represents the required stabilator maneuver. Both the preliminary networks trainings and the simulations have been carried out with a sample time of 0.01 s.

The first test case deals with the CAS used as a steady-state-hold autopilot. Figure 9 shows that the nonadaptive controller (trained offline) is unable to hold the trim condition, but the adaptive version allows the trim condition to be maintained indefinitely.

The nonadaptive controller seems to work better in signal-tracking problems, as shown by the next two test cases (Figs. 10 and 11), even if its performance is not excellent. However, no final conclusion can be drawn about the nonadaptive controller because the offline training of a nonlinear system, starting from a random initial point, is a stochastic process: Depending on the the starting point, the training process may lead to different controllers, whose parameters affect the closed-loop behavior in a different way. Ultimately, the nonadaptive controller features poor or unpredictable stability, even in nominal operational conditions without any external disturbances, and is for this reason barely considered.

Figure 11 refers to a vertical gust of 10 m/s that suddenly occurs 3 s from controller activation, while the aircraft is performing a unit-step maneuver. The gust goes on for 0.5 s and then disappears instantaneously. In this event, the reference model role is highlighted: If environmental conditions change abruptly over a predefined margin, the reference model instantaneously modifies the reference signal to narrow the gap between the aircraft response and the input commanded signal. During the next instant, the reference signal trends

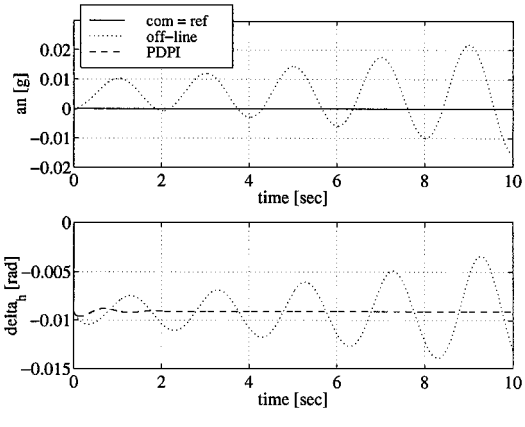

Fig. 9 Steady-state-hold autopilot (PDPI gain coefficients: K_p = 0.3, K_d = 0.04).

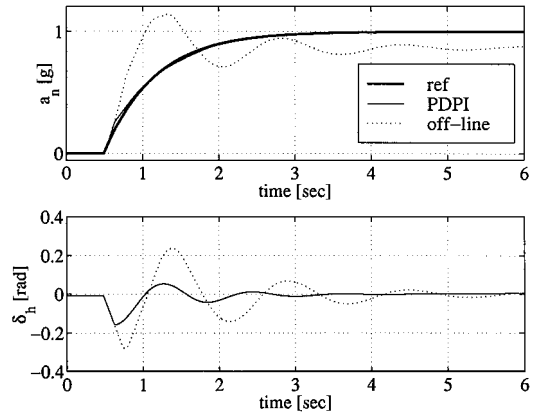

Fig. 10 Step signal [PDPI gain coefficients: K_p = 0.3, K_d = 0.07; reference model: a = (1 − 0.985), b = (0.015)].

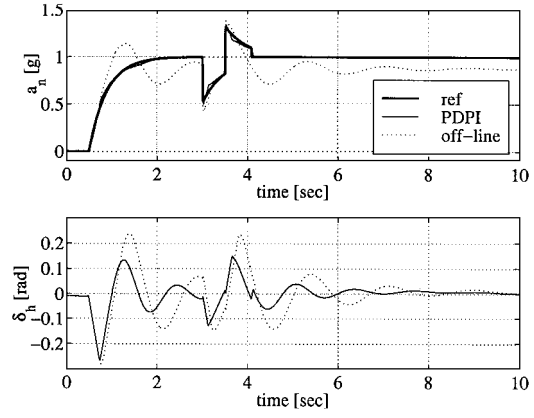

Fig. 11 Step signal with a sudden vertical gust of 10 m/s [PDPI gain coefficients: K_p = 0.3, K_d = 0.07; reference model: a = (1 − 0.975), b = (0.025)].

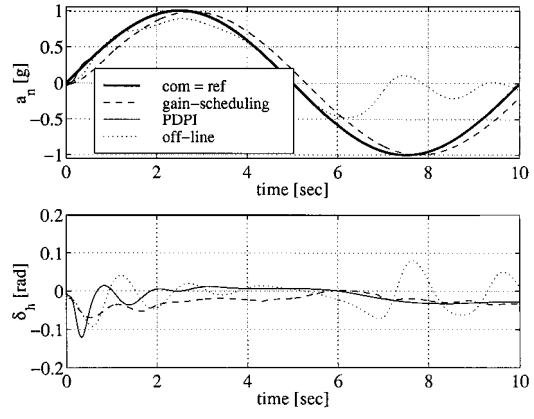

Fig. 12 Sinusoidal signal (PDPI gain coefficients: K_p = 0.3, K_d = 0.01).

asymptotically toward the commanded signal, acting as a moving target for the controller output. This means that although the input commanded signal is not affected by external disturbances, the reference signal $a_{n_{\text{ref}}}(k)$ is modified to maintain the Θ vector in the attractive basin of the global minimum. This expedient allows the system to be always close to optimal solution, under the cost function term in Eq. (9).

In the fourth (Fig. 12) and fifth (Fig. 13) test cases, the performance of the neural CAS is compared to those of the conventional gain-scheduling CAS. In Fig. 12, where the commanded signal is sinusoidal, the gain-scheduling CAS follows the signal with a slight delay, and the PDPI neural CAS follows the reference signal perfectly, apart from some minimal initial oscillations.

Corresponding conclusions can be drawn from Fig. 13, where a double-step signal with a maximum value of 1 g is considered. The gain-scheduling CAS presents a slight overshoot, and the PDPI neural CAS correctly follows the reference signal with a very smooth

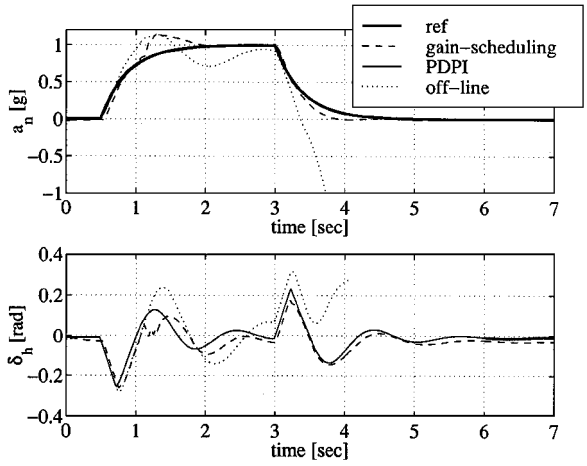

Fig. 13 Double-step signal [PDPI gain coefficients: $K_p = 0.3, K_d = 0.07$; reference model: $a = (1 - 0.975), b = (0.025)$].

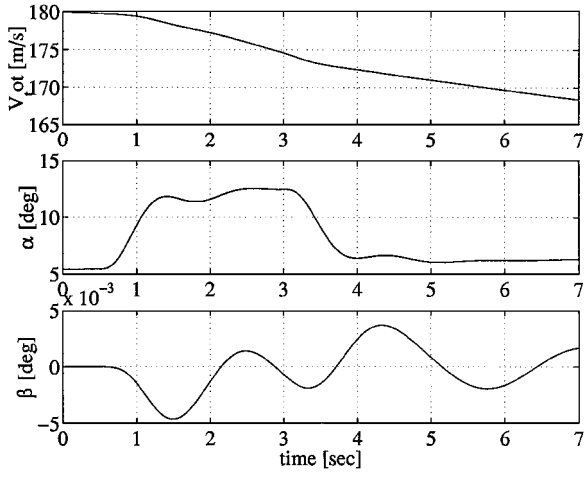

Fig. 14 Example of some state variables trend in the double-step test case 1.

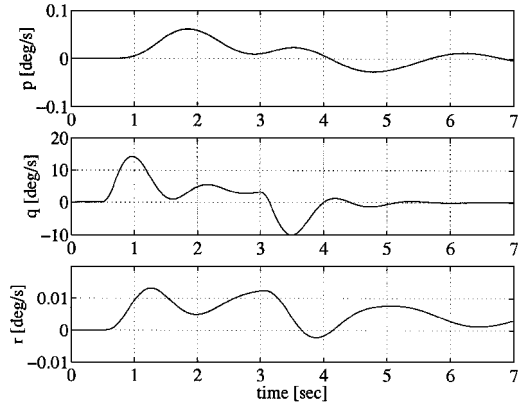

Fig. 15 Example of some state variables trend in the double-step test case 2.

maneuver. As far as the latter test case is concerned, Figs. 14 and 15 show a sample of the time histories of the most representative state variables. It can be noted that even if the latero-directional motion is left uncontrolled during the maneuver, the oscillations of the relative state variables are unimportant.

During the specialized learning, the reference signal can be repeated for several epochs. The controller is expected to enhance its performance, but the improvements do not usually justify the extra time needed and, moreover, the overtraining may lead to control signal oscillations that may diverge. Figure 16 shows the effect of the training epochs: the second epoch is beneficial, but during the third and the fourth epochs the overtraining problem occurs and, in fact, high-frequency oscillations are evident.

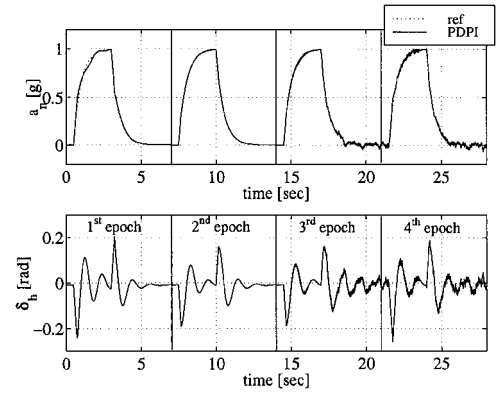

Fig. 16 Three training epochs for the double-step signal [PDPI neural CAS; reference model: $a = (1 - 0.98), b = (0.02)$].

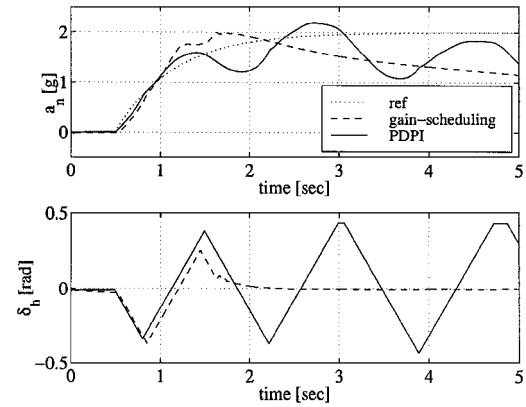

Fig. 17 Single-step signal with $\dot{\delta}_{h_{\max}} = 60$ deg/s [PDPI gain coefficients: $K_p = 0.01, K_d = 1.0$; reference model: $a = (1 - 0.985), b = (0.015)$].

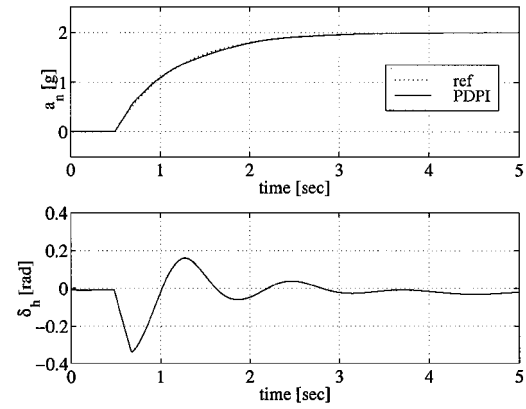

Fig. 18 Single-step signal with $\dot{\delta}_{h_{\max}} = 100$ deg/s [PDPI gain coefficients: $K_p = 0.3, K_d = 0.04$; reference model: $a = (1 - 0.985), b = (0.015)$].

The last test case is a single-step signal of 2 g (Fig. 17). The reference model is less strict, as the a and b matrices show, to allow the aircraft to accomplish the maneuver without excessive a_n overshoot. Nonetheless, neither the PDPI nor the gain-scheduling CAS is able to maintain the level of the commanded signal, except for about 2 s at most. (Note that the values of the gain coefficients used for the PDPI are quite different from values used in all the other test cases. However, in this event, they should not be considered as an effective solution, but just as a means to allow the plant not to go out of the aerodynamic database.) Moreover, the PDPI controller causes wide oscillations due to the repeatedly checked maneuver. The δ_h time history shows that the whole control action is accomplished with a constant $|\dot{\delta}_h|$ equal to the maximum value the actuator allows. This behavior can be traced back to a classical saturation problem; in fact, raising the maximum deflection rate of the stabilator to the value of 100 deg/s and leaving unchanged all the other limits of Eq. (1), the neural controller begins working again (Fig. 18).

Conclusions

This paper has highlighted the characteristics of a neural CAS controller that has been designed according to the reference model adaptive inverse scheme. The online training has been performed through a modified version of the standard recursive prediction error method based on a proportional derivative performance index, which enhances the algorithm convergence feautures. Proof was given that, provided the gain coefficients are properly set, the PDPI makes the plant response track the reference signal perfectly, reducing the plant output error.

The test cases presented here, although they involve simple and not binding maneuvers, show how the online training improves the performance of the neural controller over those of the gain-scheduling controller.

The role of the reference model in dealing with the neural controller has been highlighted. The reference model cannot be considered as an external element and, although the handling qualities requirements request this specifically, it should be always designed as an integrated part of the whole controller because this has a substantial effect on its performance. From a technical point of view, the advantange of using a reference model is well known: Even if parameters are imposed by the handling qualities, a certain choice margin is allowed, considering that a too-mild reference model does not allow the aircraft to reach high performance in terms of time response. However, it is not realistic to demand that the aircraft exceed its mechanical limits. From a mathematical point of view, the reference model is the stratagem that enables the algorithm to act constantly in the attractive basin of the optimal solution, with beneficial effects on the convergence properties.

Acknowledgment

Research for this study was performed with National Research Council contribution.

References

[1]Calise, A. J., Sharma, M., and Corban, J. E., "An Adaptive Autopilot Design for Guided Munitions," *AIAA Guidance, Navigation, and Control Conference*, Vol. 3, AIAA, Reston, VA, 1998, pp. 1776–1785.

[2]Leitner, J., Calise, A. J., and Prasad, J. V. R., "A Full Authority Helicopter Adaptive Neuro-Controller," *IEEE Aerospace Conference Proceedings*, Vol. 2, Inst. of Electrical and Electronics Engineers, New York, 1998, pp. 117–126.

[3]Nardi, F., Rysdyk, R. T., and Calise, A. J., "Neural Network Based Adaptive Control of a Thrust Vectored Ducted Fan," *AIAA Guidance, Navigation, and Control Conference*, Vol. 1, AIAA, Reston, VA, 1999, pp. 374–383.

[4]McFarland, M. B., and Calise, A. J., "Robust Adaptive Control of Uncertain Nonlinear Systems Using Neural Networks," *Proceedings of the American Control Conference*, American Automatic Control Council, Evanston, IL, Vol. 3, 1997, pp. 1996–2000.

[5]Rysdyk, R. T., and Calise, A. J., "Fault Tolerant Flight Control via Adaptive Neural Network Augmentation," *AIAA Guidance, Navigation, and Control Conference*, Vol. 3, AIAA, Reston, VA, 1998, pp. 1722–1728.

[6]Wise, K. A., Brinker, J. S., Calise, A. J., Elgersma, M. R., and Voulgaris, P., "Direct Adaptive Reconfigurable Flight Control for a Tailless Advance Fighter Aircraft," *International Journal of Robust and Nonlinear Control*, Vol. 9, No. 14, 1999, pp. 999–1012.

[7]McFarland, M. B., "Augmentation of Gain-Scheduled Missile Autopilot Using Adaptive Neural Networks," *AIAA Guidance, Navigation, and Control Conference*, Vol. 3, AIAA, Reston, VA, 1998, pp. 1786–1792.

[8]KrishnaKumar, K., and Kulkarni, N., "Inverse Adaptive Neuro-Control of a Turbo-Fan Engine," *AIAA Guidance, Navigation, and Control Conference*, Vol. 1, AIAA, Reston, VA, 1999, PP. 354–364.

[9]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.

[10]Napolitano, M. R., Naylor, S., Neppach, C., and Casdorph, V., "On-Line Learning Nonlinear Neurocontrollers for Restructurable Control Systems," *Journal of Guidance, Control, and Dynamics*, Vol. 18, No. 1, 1995, pp. 170–176.

[11]Napolitano, M. R., An, Y., Seanor, B., and Pispitsos, S., "Application of a Neural Sensor Validation Scheme of Actual Boeing 737 Flight Data," *AIAA Guidance, Navigation, and Control Conference*, Vol. 3, AIAA, Reston, VA, 1999, pp. 354–364.

[12]Napolitano, M. R., Molinaro, G., Innocenti, M., Seanor, B., and Martinelli, D., "A Complete Hardware Package for a Fault-Tolerant Flight Control System Using On-Line Learning Neural Networks," *Proceedings of the American Control Conference*, Vol. 3, American Automatic Control Council, Evanston, IL, 1999, pp. 2615–2619.

[13]Psaltis, D., Sideris, A., and Yamamura, A. A., "A Multilayered Neural Network Controller," *IEEE Control Systems Magazine*, Vol. 8, No. 2, 1988, pp. 17–21.

[14]Ng, G. W., *Application of Neural Networks to Adaptive Control of Nonlinear Systems*, Research Studies Press, Tauton, U.K., 1997.

[15]Jordan, M. I., and Jacobs, R. A., "Learning to Control an Unstable System with Forward Modeling," *Advances in Neural Information Processing Systems*, edited by R. P. Lippmann, S. E. Moddy, and D. S. Touretzky, Morgan Kaufmann, San Mateo, CA, 1990.

[16]Chen, S., and Billings, S. A., "Neural Networks for Nonlinear Dynamics System Modeling and Identification," *International Journal of Control*, Vol. 56, No. 2, 1992, pp. 319–346.

[17]Wang, D., and Bao, P., "Enhancing the Estimation of Plant Jacobian for the Adaptive Neural Inverse Control," *Neurocomputing*, Vol. 34, Nos. 1–4, 2000, pp. 99–115.

[18]Chen, S., Billings, S. A., and Grant, P. M., "Nonlinear System Identification Using Neural Networks," *International Journal of Control*, Vol. 51, No. 6, 1990, pp. 1191–1214.

[19]KrishnaKumar, K., "Levels of Intelligent Control: A Tutorial," *AIAA August Professional Development Short Course*, AIAA, Reston, VA, Aug. 1997.

[20]Shamma, J. S., and Athans, M., "Analysis of Gain Scheduled Control for Nonlinear Plants," *IEEE Transactions on Automatic Control*, Vol. 35, No. 8, 1990, pp. 898–907.

[21]Balakrishnan, S. N., and Biega, V., "Adaptive-Critic-Based Neural Networks for Aircraft Optimal Control," *Journal of Guidance, Control, and Dynamics*, Vol. 19, No. 4, 1996, pp. 893–898.

[22]Barto, A. G., "Connectionist Learning for Control: An Overview," *Neural Networks for Control*, edited by W. T. Miller III, R. S. Sutton, and P. J. Werbos, MIT Press, Cambridge, MA, 1990, pp. 5–58.

[23]Gili, P., and Battipede, M., "A MIMO Neural Adaptive Autopilot for a Nonlinear Helicopter Model," AIAA Paper 99-4219, Aug. 1999.

[24]Gili, P., and Battipede, M., "A Comparative Design of a MIMO Neural Adaptive Autopilot for a Nonlinear Helicopter Model," *Proceedings of the 8th European Symposium of Artificial Neural Networks*, D-Facto, Brussels, 2000, pp. 159–164.

[25]Gili, P., and Battipede, M., "Adaptive Features of a MIMO Full-Authority Controller," *AIAA Guidance, Navigation, and Control Conference*, [CD-ROM], AIAA, Reston, VA, 2000.

[26]Stevens, B. L., and Lewis, F. L., *Aircraft Control Simulation*, Wiley, New York, 1992, pp. 585–592.

[27]Nguyen, L. T., Ogburn, M. E., Gilbert, W. P., Kibler, K. S., Brown, P. W., and Deal, P. L., "Simulator Study of Stall/Post Stall Characteristics of a Fighter Airplane with Relaxed Longitudinal Static Stability," NASA TP-1538, Dec. 1979.

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

[29]Colla, V., Reyneri, L. M., and Sgarbi, M., "Training Activation Function in Parametric Classification," *Proceedings of the 8th European Symposium of Artificial Neural Networks*, D-Facto, Brussels, 2000, pp. 365–370.

[30]Napolitano, M. R., Chen, C. I., and Naylor, S., "Aircraft Failure Detection and Identification Using Neural Networks," *Journal of Guidance, Control, and Dynamics*, Vol. 16, No. 6, 1993, pp. 999–1009.

[31]Lendasse, A., Lee, J., Wertz, V., and Verleysen, M., "Time Series Forecasting Using CCA and Kohonen Maps—Application to Electricity Consumption," *Proceedings of the 8th European Symposium of Artificial Neural Networks*, D-Facto, Brussels, 2000, pp. 329–334.

[32]Norgaard, M., "Neural Network Based System Identification Toolbox," DTU-TR-97-E-851, Dept. of Automation, Technical Univ. of Denmark, Lyngby, Denmark, June 1997.

[33]Fletcher, R., *Practical Methods of Optimization*, Wiley, New York, 1987, pp. 100–107.

[34]Hassibi, B., and Stork, D. G., "Second Order Derivatives for Network Pruning: Optimal Brain Surgeon," *Proceedings of the Advances in Neural Information Processing Systems 5 Conference*, Morgan Kaufmann, San Mateo, CA, 1993, pp. 164–171.

[35]Ljung, L., *System Identification: Theory for the User*, Prentice-Hall, Englewood Cliffs, NJ, 1987, pp. 316–336.

[36]Widrow, B., and Walach, E., *Adaptive Inverse Control*, Information and Systems Sciences Series, Prentice-Hall, Englewood Cliffs, NJ, 1996, pp. 330–338.

[37]Norgaard, M., "Neural Network Based Control System Design Toolkit," DTU-TR-96-E-830, Dept. of Automation, Technical Univ. of Denmark, Lyngby, Denmark, June 1996.

[38]Military Specifications—Flying Qualities of Piloted Vehicles," MIL-STD-1797A, March 1987.