JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 25, No. 3, May–June 2002

Classical/Neural Synthesis of Nonlinear Control Systems

Silvia Ferrari* and Robert F. Stengel†
Princeton University, Princeton, New Jersey 08544

Classical/neural synthesis of control systems combines the most effective elements of old and new design concepts with the promise of producing better control systems. There is considerable precedent for applying gain-scheduled linear controllers to nonlinear systems, especially those that can be locally approximated as linear-parameter-varying systems; however, a means for transferring the insights gained from these linear controllers to nonlinear controllers remains to be identified. The approach taken is to design nonlinear control systems that take advantage of prior knowledge and experience gained from linear controllers, while capitalizing on the broader capabilities of adaptive, nonlinear control theory and computational neural networks. Central to this novel approach is the recognition that the gradients of a nonlinear control law must represent the gain matrices of an equivalent, locally linearized controller. The focus is on the initial specification for the control law, which consists of a set of hypersurfaces expressed as neural networks that represent satisfactory linear controllers designed over the plant's operating range. The neural networks are defined solely by solving algebraic linear systems of equations, and their effectiveness is demonstrated in a case study.

Introduction

NEURAL networks' ability to approximate unknown nonlinear mappings with high-dimensional input spaces and their potential for online learning make them excellent candidates for use in adaptive control systems. Extensive numerical studies[1–3] have shown that they are capable of dealing with those difficulties typically associated with complex control applications, such as nonlinearity and uncertainty. However, practical applications also call for a better understanding of the theoretical principles involved.[3] In particular, there is no simple way to apply the insights afforded by classical control design methods to the specification and preliminary design of neural network controllers. Furthermore, the issues of network structure, number of nodes required for satisfactory performance, data overfitting, and adaptation to changing operating conditions remain to be resolved. This paper tackles these issues through a novel technique for defining control hypersurfaces as neural networks that match linear control systems at an arbitrary number of operating points. The network weights are calculated algebraically, establishing an innovative framework for classical/neural control system design.

The advantages brought about by using neural approximators in conjunction with linear control theory have been recognized in the past by several authors.[2,4,5] Previous applications consisted of training a global neural network to approximate the linear gains provided by linear multivariable control. The objective often has been to determine an adaptive algorithm or rule that iteratively adjusts the parameters of a prespecified network architecture as input–output data pairs are presented to it. Backpropagation has been the most commonly used training method.[3]

Here the nonlinear control system comprises a network of neural networks whose architecture, parameters, and size, that is, number of hidden nodes, are determined from the initial specification of the control law solely by solving algebraic linear systems of equations. A critical observation is that the gradients of the neural networks must equal corresponding linear gain matrices at chosen operating points. Following this initialization phase, online learning by an adaptive-critic approach[6] can be used to improve control response for large, coupled motions, by accounting for differences between actual and assumed dynamic models and for nonlinear effects not captured in the linearizations. Optimizing the same metric during initialization and during online learning affords a systematic approach to design that is as conservative as the linear model and as effective as the global nonlinear controller. This paper addresses the classical/neural system synthesis that takes place in the initial phase of the proposed design method.

The initial phase alone defines a global nonlinear neural network controller and provides an excellent starting point for the online learning phase. The effectiveness of the novel technique is demonstrated by evaluating the performance of a neural control system, motivated by the proportional–integral (PI) controller, for the global control of the lateral dynamics of a transport aircraft. The nonlinear control structure is obtained by replacing the linear gains of a PI controller with nonlinear neural networks: a forward neural network, a feedback neural network, and a command-integral neural network replace the respective gains. The aircraft's nonlinear dynamic model is locally approximated as a linear-parameter-varying system. Linear control laws are designed for specific operating points in the aircraft's flight envelope, and they are subsequently learned and generalized by the neural controllers.

The linear control laws are used to derive a set of requirements for each neural network. By the imposition of these requirements on the networks' input–output and gradient relations, a set of nonlinear equations is obtained relating the adjustable parameters to the information available from the gain-scheduled design. These equations are then reduced to sets of linear algebraic equations that can be solved exactly for the networks' weights.

Foundations of the Nonlinear Control Problem

An initial assumption is that the dynamic properties of the plant to be controlled can be described by a nonlinear differential equation of the form

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}[\boldsymbol{x}(t), \boldsymbol{p}(t), \boldsymbol{u}(t)] \tag{1}$$

where $\boldsymbol{x}$ is the $n \times 1$ plant state, $\boldsymbol{p}$ is a $l \times 1$ vector of plant and observation parameters, and $\boldsymbol{u}$ is the $m \times 1$ control vector. The equation may represent a lumped-parameter system, or it may be the approximation to an unsteady partial differential equation. Plant motions and disturbances are sensed in the $e \times 1$ output vector $\boldsymbol{y}_s$,

$$\boldsymbol{y}_s(t) = \boldsymbol{h}_s[\boldsymbol{x}(t), \boldsymbol{p}(t), \boldsymbol{u}(t)] \tag{2}$$

Presented as Paper 2000-4552 at the AIAA Guidance, Navigation, and Control Conference, Denver, CO, 14–17 August 2000; received 20 October 2001; revision received 1 December 2001; accepted for publication 17 December 2001. Copyright © 2002 by Silvia Ferrari and Robert F. Stengel. Published by the American Institute of Aeronautics and Astronautics, Inc., with permission. Copies of this paper may be made for personal or internal use, on condition that the copier pay the $10.00 per-copy fee to the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923; include the code 0731-5090/02 $10.00 in correspondence with the CCC.

*Graduate Student, Department of Mechanical and Aerospace Engineering, D215 Engineering Quadrangle. Student Member AIAA.

†Professor, Department of Mechanical and Aerospace Engineering, D202 Engineering Quadrangle. Fellow AIAA.

Here, it is assumed that perfect output measurements are available. The design objective is to specify a control law of the general form

$$\boldsymbol{u}(t) = \boldsymbol{c}[\boldsymbol{y}_s(t), \boldsymbol{p}(t), \boldsymbol{y}_c(t)] \tag{3}$$

that has two properties: It achieves mission goals, as expressed by the $e_c \times 1$ command input $\boldsymbol{y}_c$, and it furnishes adequate stability and transient response, assuring that excursions from $\boldsymbol{y}_c$ caused by disturbance or measurement error are acceptably small and do not require excessive control use.

The command input can be viewed as some desirable combination of state and control elements, and its dimension e_c is less than or equal to the number of independent controls m:

$$\boldsymbol{y}_c(t) = \boldsymbol{h}_c[\boldsymbol{x}(t), \boldsymbol{u}(t)] \tag{4}$$

The command input could be the result of external trajectory planning, for example, a prescribed path, or it may be due to a loosely defined, subjective process such as the expression of a human operator's intent through command inputs.

The control law can be written as a functional $\boldsymbol{c}[\bullet]$, which may contain functions of its arguments such as integrals or derivatives:

$$\boldsymbol{u}(t) = \boldsymbol{c}[\boldsymbol{x}(t), \boldsymbol{p}(t), \boldsymbol{y}_c(t)] \tag{5}$$

We can always write the control law as the sum of a nominal and a perturbed effect,

$$\begin{aligned}\boldsymbol{u}(t) &= \boldsymbol{c}_0\big[\boldsymbol{x}_0(t), \boldsymbol{p}(t), \boldsymbol{y}_{c_0}(t)\big] + \Delta\boldsymbol{c}[\boldsymbol{x}(t), \boldsymbol{p}(t), \boldsymbol{y}_c(t)] \\ &= \boldsymbol{u}_0(t) + \Delta\boldsymbol{u}(t)\end{aligned} \tag{6}$$

where, for simplicity, we assume that the vector of parameters $\boldsymbol{p}(t)$ is known without error. The anticipated nominal values of the state and command input are $\boldsymbol{x}_0$ and $\boldsymbol{y}_{c_0}$, so that the actual values can be written by adding the respective perturbations, $\Delta\boldsymbol{x}$ and $\Delta\boldsymbol{y}_c$:

$$\boldsymbol{x}(t) = \boldsymbol{x}_0(t) + \Delta\boldsymbol{x}(t), \qquad \boldsymbol{y}_c(t) = \boldsymbol{y}_{c_0}(t) + \Delta\boldsymbol{y}_c(t) \tag{7}$$

Hence, the control law can be expressed conveniently in these terms:

$$\begin{aligned}\boldsymbol{u}(t) &= \boldsymbol{c}_0\big[\boldsymbol{x}_0(t), \boldsymbol{p}(t), \boldsymbol{y}_{c_0}(t)\big] \\ &\quad + \Delta\boldsymbol{c}\big[\boldsymbol{x}_0(t), \boldsymbol{p}(t), \boldsymbol{y}_{c_0}(t), \Delta\boldsymbol{x}(t), \Delta\boldsymbol{y}_c(t)\big]\end{aligned} \tag{8}$$

For sufficiently small state and command perturbations, the perturbed control effect is linear in $\Delta\boldsymbol{x}$ and $\Delta\boldsymbol{y}_c$ and can be written as

$$\begin{aligned}\Delta\boldsymbol{u}(t) = \Delta\boldsymbol{c}[\bullet] &= \frac{\partial\boldsymbol{c}}{\partial\boldsymbol{x}}\big[\boldsymbol{x}_0(t), \boldsymbol{p}(t), \boldsymbol{y}_{c_0}(t)\big]\Delta\boldsymbol{x}(t) \\ &\quad + \frac{\partial\boldsymbol{c}}{\partial\boldsymbol{y}_c}\big[\boldsymbol{x}_0(t), \boldsymbol{p}(t), \boldsymbol{y}_{c_0}(t)\big]\Delta\boldsymbol{y}_c(t) = \boldsymbol{C}_x\Delta\boldsymbol{x}(t) + \boldsymbol{C}_y\Delta\boldsymbol{y}_c(t)\end{aligned} \tag{9}$$

$\boldsymbol{C}_x$ and $\boldsymbol{C}_y$ contain the m gradients, or gains, of the control law evaluated at $[\boldsymbol{x}_0(t), \boldsymbol{p}(t), \boldsymbol{y}_{c_0}(t)]$. Equation (9) can be viewed as a linear, gain-scheduled control law that, when combined with $\boldsymbol{c}_0[\bullet]$, provides a close approximation to the exact nonlinear controller, Eq. (3), for small perturbations $\Delta\boldsymbol{x}$ and $\Delta\boldsymbol{y}_c$:

$$\boldsymbol{u}(t) = \boldsymbol{c}_0\big[\boldsymbol{x}_0(t), \boldsymbol{p}(t), \boldsymbol{y}_{c_0}(t)\big] + \boldsymbol{C}_x\Delta\boldsymbol{x}(t) + \boldsymbol{C}_y\Delta\boldsymbol{y}_c(t) \tag{10}$$

The operating range (OR) denotes the OR of $[\boldsymbol{x}(t), \boldsymbol{p}(t), \boldsymbol{y}_c(t)]$ or some suitably dense set in the space. Then knowledge of $\boldsymbol{c}[\boldsymbol{x}(t), \boldsymbol{p}(t), \boldsymbol{y}_c(t)]$ at a single point in OR and of $\boldsymbol{C}_x$ and $\boldsymbol{C}_y$ everywhere in OR is equivalent to knowledge of $\boldsymbol{c}[\boldsymbol{x}(t), \boldsymbol{p}(t), \boldsymbol{y}_c(t)]$ everywhere in OR.

Gain-scheduled control laws are based on a set of linear time-invariant (LTI) control laws specified everywhere in OR through one or more scheduling variables. In practice, controllers usually are not designed at every value of the scheduling variable but rather at several operating points indexed by selected values of it. Assume that each operating point corresponds to an index $k = 1, 2, \ldots$. Given the dynamic system of Eq. (1), a first-degree expansion can be written

$$\begin{aligned}\dot{\boldsymbol{x}}(t) &= \dot{\boldsymbol{x}}_0(t) + \Delta\dot{\boldsymbol{x}}(t) = \boldsymbol{f}_0[\boldsymbol{x}_0(t), \boldsymbol{p}(t), \boldsymbol{u}_0(t), \boldsymbol{w}_0(t)] \\ &\quad + \Delta\boldsymbol{f}[\boldsymbol{x}_0(t), \boldsymbol{p}(t), \boldsymbol{u}_0(t), \boldsymbol{w}_0(t), \Delta\boldsymbol{x}(t), \Delta\boldsymbol{u}(t), \Delta\boldsymbol{w}(t)] \\ &\approx \boldsymbol{f}_0[\boldsymbol{x}_0(t), \boldsymbol{p}(t), \boldsymbol{u}_0(t), \boldsymbol{w}_0(t)] + \frac{\partial\boldsymbol{f}}{\partial\boldsymbol{x}}\Delta\boldsymbol{x}(t) + \frac{\partial\boldsymbol{f}}{\partial\boldsymbol{u}}\Delta\boldsymbol{u}(t) \\ &\quad + \frac{\partial\boldsymbol{f}}{\partial\boldsymbol{w}}\Delta\boldsymbol{w}(t) = \boldsymbol{f}_0[\bullet] + \boldsymbol{F}\Delta\boldsymbol{x}(t) + \boldsymbol{G}\Delta\boldsymbol{u}(t) + \boldsymbol{L}\Delta\boldsymbol{w}(t)\end{aligned} \tag{11}$$

The Jacobian matrices $\boldsymbol{F}$, $\boldsymbol{G}$, and $\boldsymbol{L}$, are evaluated at selected operating points in OR designated by the index k. The perturbation model is

$$\begin{aligned}\Delta\dot{\boldsymbol{x}}(t) &= \boldsymbol{F}[\boldsymbol{x}_0(t), \boldsymbol{p}(t), \boldsymbol{u}_0(t), \boldsymbol{w}_0(t)]\Delta\boldsymbol{x}(t) \\ &\quad + \boldsymbol{G}[\boldsymbol{x}_0(t), \boldsymbol{p}(t), \boldsymbol{u}_0(t), \boldsymbol{w}_0(t)]\Delta\boldsymbol{u}(t) \\ &\quad + \boldsymbol{L}[\boldsymbol{x}_0(t), \boldsymbol{p}(t), \boldsymbol{u}_0(t), \boldsymbol{w}_0(t)]\Delta\boldsymbol{w}(t)\end{aligned} \tag{12}$$

This model is almost a linear parameter-varying plant, "almost" because the system matrices depend on $\boldsymbol{x}_0(t)$ as well as the remaining variables. In most applications, effects of parameter variation are ignored because time-varying dynamic effects are small, and $[\boldsymbol{F}, \boldsymbol{G}, \boldsymbol{L}]_k$ is treated as a set of LTI plant models. Linear control gains, such as $\boldsymbol{C}_x$ and $\boldsymbol{C}_y$, are computed for each LTI model at each operating point, and the control law is implemented with interpolation of gains to intermediate operating regions.

In past applications, the number of interpolating variables has been kept small. The approach taken here achieves interpolation without additional processing and allows the number of independent variables to be expanded, affording an improvement in comparison to earlier gain-scheduled controllers. More important, it provides an excellent initialization point for the neural network controller whose purpose is to generate the nonlinear control law, Eq. (3), given the gains and the corresponding equilibria at any set of operating points (OP) $\{(k = 1, 2, \ldots, p)\colon k \in \mathrm{OP},\ \mathrm{OP} \subset \mathrm{OR}\}$. OP is defined as the suitably chosen subset of operating points for which the linear control gains and equilibria are to be computed. It is assumed that the LTI control laws satisfy engineering design criteria, based on design principles such as those described in earlier work.[7–11] For example, Monte Carlo evaluation and genetic algorithms could be used to design robust, linear quadratic Gaussian controllers that satisfy classical criteria and to provide the desired values of $\boldsymbol{C}_x$ and $\boldsymbol{C}_y$, along with the corresponding equilibria. However, the approach presented here will work in conjunction with any process that generates the gains of a control law in the form of Eq. (10).

Specifying the Neural Network Controller Structure

The neural network controller structure is specified by the definition of the appropriate system architecture and by the identification of appropriate performance targets for the nonlinear controller. The proposed approach defines a network of networks that is motivated by classical/modern feedback, inner/outer loop, proportional–integral–derivative (PID) control formulations. Desired performance baselines are estimated by considering the performance of an equivalent linear model and are imposed on the neural network structure.

PI control is considered for illustration. A multivariable PI controller, shown in Fig. 1, modifies the stability and transient response

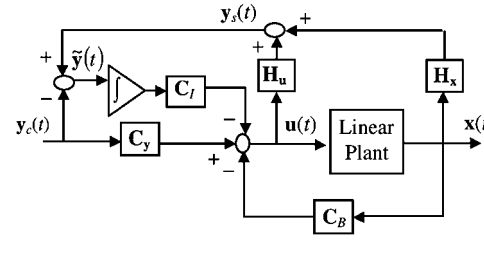

Fig. 1 Example of linear PI feedback control system.

of the system through the feedback gain matrix $\boldsymbol{C}_B$, and it provides type-1 response to command inputs through the proportional gain matrix $\boldsymbol{C}_y$ and the command-integral gain matrix $\boldsymbol{C}_I$. These gains can be computed by minimizing a quadratic cost function of the form

$$J = \lim_{t_f \to \infty} \frac{1}{2} \int_0^{t_f} L[\Delta \boldsymbol{x}_a(\tau), \Delta \boldsymbol{u}(\tau)]\, \mathrm{d}\tau$$

$$= \lim_{t_f \to \infty} \frac{1}{2} \int_0^{t_f} \left[\Delta \boldsymbol{x}_a^T(\tau) \boldsymbol{Q} \Delta \boldsymbol{x}_a(\tau) + 2\Delta \boldsymbol{x}_a^T(\tau) \boldsymbol{M} \Delta \boldsymbol{u}(\tau) + \Delta \boldsymbol{u}^T(\tau) \boldsymbol{R} \Delta \boldsymbol{u}(\tau) \right] \mathrm{d}\tau \tag{13}$$

$\Delta \boldsymbol{x}_a$ represents an augmented state that includes both the deviation $\tilde{\boldsymbol{x}}$, from the state defined by the operating set point $(\boldsymbol{x}_c, \boldsymbol{u}_c)$,

$$\tilde{\boldsymbol{x}} \equiv (\boldsymbol{x} - \boldsymbol{x}_0) - (\boldsymbol{x}_c - \boldsymbol{x}_0) = \boldsymbol{x} - \boldsymbol{x}_c \tag{14}$$

and the time integral of the output error ξ, that is, $\Delta \boldsymbol{x}_a \equiv [\tilde{\boldsymbol{x}}^T \quad \xi^T]^T$.

The minimizing control law takes the form of Eq. (9), in terms of the augmented state, $\Delta \boldsymbol{x}_a$:

$$\Delta \boldsymbol{u}(t) = \boldsymbol{C}_y \Delta \boldsymbol{y}_c(t) - \boldsymbol{C}_B \Delta \boldsymbol{x}(t) - \boldsymbol{C}_I \xi(t)$$

$$= \Delta \boldsymbol{u}_c(t) - \boldsymbol{C}_B \tilde{\boldsymbol{x}}(t) - \boldsymbol{C}_I \xi(t) \tag{15}$$

Therefore, the linear gains $\boldsymbol{C}_B$, $\boldsymbol{C}_y$, and $\boldsymbol{C}_I$ computed for OP can be used to generate the nonlinear control law, Eq. (3). For the PI controller, the gain matrices are obtained by solving a matrix Riccati equation formulated in terms of the augmented state $\Delta \boldsymbol{x}_a$. The weighting matrices $\boldsymbol{Q}$, $\boldsymbol{M}$, and $\boldsymbol{R}$, are designed using implicit model following to induce the closed-loop system to follow the response of an ideal model that satisfies established design criteria.[12] A first-degree expansion of Eq. (2) provides the output perturbation $\Delta \boldsymbol{y}_s$ in terms of the Jacobian matrices $\boldsymbol{H}_u$ and $\boldsymbol{H}_x$.

The equivalent neural network structure is shown in Fig. 2. Each linear gain matrix has been replaced by a nonlinear neural network (NN), NN_B for $\boldsymbol{C}_B$, NN_F for $\boldsymbol{C}_y$, and NN_I for $\boldsymbol{C}_I$. The input–output structure is unchanged, and the command error is integrated, as in the linear system, to produce type-1 response. Therefore, the performance targets for these networks are established locally by the respective linear gains that the networks replace. The scheduling variable generator contains algebraic equations that produce auxiliary inputs to the neural networks based on the command input and an exogenous vector $\boldsymbol{e}$ of measured variables. The command state generator uses the aircraft's kinematic equations to provide secondary elements of the state that are compatible with $\boldsymbol{y}_c$.

All auxiliary inputs are included in a scheduling vector $\boldsymbol{a}$ that informs the neural networks of the dynamically significant variables in the system. In addition, the networks NN_B, NN_F, and NN_I are provided with the state deviation $\tilde{\boldsymbol{x}}(t)$, the command $\boldsymbol{y}_c(t)$, and the command error integral $\xi(t)$, respectively. Each of these vector-output networks contributes to the total control, $\boldsymbol{u}(t) = \boldsymbol{u}_0(t) + \Delta \boldsymbol{u}(t)$,

$$\boldsymbol{u}(t) = \boldsymbol{u}_c(t) + \Delta \boldsymbol{u}_B(t) + \Delta \boldsymbol{u}_I(t) \tag{16}$$

where

$$\Delta \boldsymbol{u}_B(t) = \mathrm{NN}_B[\tilde{\boldsymbol{x}}(t), \boldsymbol{a}] \equiv \boldsymbol{z}_B(t)$$

$$\Delta \boldsymbol{u}_I(t) = \mathrm{NN}_I[\xi(t), \boldsymbol{a}] \equiv \boldsymbol{z}_I(t) \tag{17}$$

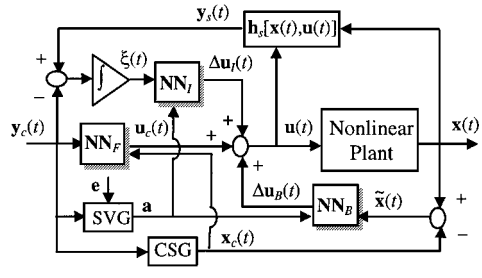

Fig. 2 Nonlinear PI NN control system.

Here, $\boldsymbol{z}$ is an $m \times 1$ vector denoting the output of each network, as shown. To meet the desired performance targets, the neural networks each must satisfy a set of known requirements that are imposed on their output and gradient equations.

Given the state deviation $\tilde{\boldsymbol{x}}(t)$, the feedback neural network must provide for regulation in the control system and produce zero output when the actual state $\boldsymbol{x}(t)$ approaches $\boldsymbol{x}_c(t)$, for example, $\tilde{\boldsymbol{x}}(t) \to \boldsymbol{0}$. Therefore, at each operating point k considered in the linear design process, the following must hold:

$$\boldsymbol{z}_B\left(\boldsymbol{0}_{n \times 1}, \boldsymbol{a}^k\right) = 0 \tag{18}$$

The feedback contribution $\Delta \boldsymbol{u}_B(t)$ to the total control depends on neural network gradients that equal the feedback matrix $\boldsymbol{C}_B$ at the training points:

$$\left.\frac{\partial[\boldsymbol{z}_B(t)]}{\partial[\tilde{\boldsymbol{x}}(t)]}\right|_{\tilde{x}=0, a=a^k} = \left.\frac{\partial[\Delta \boldsymbol{u}(t)]}{\partial[\tilde{\boldsymbol{x}}(t)]}\right|_{\tilde{x}=0, a=a^k} = -\boldsymbol{C}_B^k \tag{19}$$

where $\Delta \boldsymbol{u}(t)$ is given by Eq. (15).

A similar argument holds for the command-integral neural network. As the command error $\xi(t)$ goes to zero, the NN_I output also must vanish. The neural network gradients are known through $\boldsymbol{C}_I$ at all points considered in the linear design. Therefore, the requirements to be imposed on NN_I are equivalent to those of NN_B,

$$\boldsymbol{z}_I\left(\boldsymbol{0}_{e \times 1}, \boldsymbol{a}^k\right) = 0 \tag{20}$$

$$\left.\frac{\partial[\boldsymbol{z}_I(t)]}{\partial[\xi(t)]}\right|_{\xi=0, a=a^k} = \left.\frac{\partial[\Delta \boldsymbol{u}(t)]}{\partial[\xi(t)]}\right|_{\xi=0, a=a^k} = -\boldsymbol{C}_I^k \tag{21}$$

The adjustable parameters, or weights, of the feedback and command-integral neural networks that satisfy these requirements are obtained as explained in the following sections. The forward neural network's solution is based on the same principles and is addressed in a separate paper.[13]

NN Nonlinear Weight Equations

Computational feedforward neural networks with one hidden layer of sigmoidal functions are employed for this controller. Each vector-output neural network is composed of m scalar-output networks that can be solved for independently, and each provide an element of the final $m \times 1$ output vector $\boldsymbol{z}$. A sample scalar-output network with a generic number of inputs q and a generic number of nodes s is shown in Fig. 3. The output of the network is computed as the nonlinear transformation of the weighted sum of the input $\boldsymbol{p}$ and a bias $\boldsymbol{d}$:

$$z = \boldsymbol{v}^T \boldsymbol{\sigma}[\boldsymbol{W}\boldsymbol{p} + \boldsymbol{d}] + b \tag{22}$$

Here, $\boldsymbol{\sigma}[\bullet]$ is a vector-valued function composed of individual sigmoidal functions evaluated at all input-to-node variables n_i with $i = 1, \ldots, s$,

$$\boldsymbol{\sigma}[\boldsymbol{n}] \equiv [\sigma(n_1) \quad \cdots \quad \sigma(n_s)]^T \tag{23}$$

where

$$\boldsymbol{n} = \boldsymbol{W}\boldsymbol{p} + \boldsymbol{d} \tag{24}$$

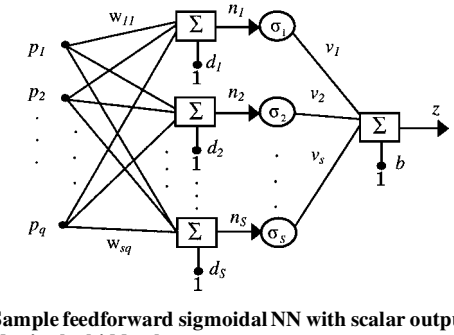

Fig. 3 Sample feedforward sigmoidal NN with scalar output, q inputs, and s nodes in the hidden layer.

$\boldsymbol{W}$ and $\boldsymbol{v}$ contain the input and output weights, respectively. Together with the input and output biases, $\boldsymbol{d}$ and b, they represent the adjustable parameters of the network.

The order of differentiability of Eq. (22) is the same as that of the activation function $\sigma(\bullet)$. Given that the sigmoid functions are infinitely differentiable, the derivative of the network output with respect to its inputs is

$$\frac{\partial z}{\partial p_j} = \sum_{i=1}^{s} \frac{\partial z}{\partial n_i}\frac{\partial n_i}{\partial p_j} = \sum_{i=1}^{s} v_i \sigma'(n_i) w_{ij}, \qquad j = 1, \ldots, q \quad (25)$$

where $\sigma'(\bullet)$ denotes the derivative of the sigmoidal function with respect to its scalar input. The (i, j)th element w_{ij} of the input weight matrix $\boldsymbol{W}$ represents the strength of the connection from the jth input to the ith node. The input $\boldsymbol{p}$ to the feedback networks consists of the state deviation $\tilde{\boldsymbol{x}}(t)$ and the scheduling vector $\boldsymbol{a}$.

Feedback Networks

$\boldsymbol{W}$ is partitioned into two matrices, $\boldsymbol{W}_{\tilde{x}}$ and $\boldsymbol{W}_a$, corresponding to the $\tilde{\boldsymbol{x}}$ input weights and to the $\boldsymbol{a}$ input weights, respectively,

$$\boldsymbol{W} = [\boldsymbol{W}_{\tilde{x}} \mid \boldsymbol{W}_a] \quad (26)$$

Feedback neural network contributions should be zero when the state deviations are zero; hence, output equations, obtained from Eqs. (18) and (22), must be satisfied:

$$0 = \boldsymbol{v}^T \boldsymbol{\sigma}\left\lfloor \boldsymbol{W}_a \boldsymbol{a}^k + \boldsymbol{d} \right\rfloor + b \quad (27)$$

The gradients of the feedback neural networks are made to match the corresponding gain matrix elements by imposing the requirement of Eq. (19) on the neural networks' slopes [Eq. (25)]. This leads to gradient equations for the feedback networks,

$$-\left[\boldsymbol{C}_B^k(l, \bullet)\right]^T = \boldsymbol{W}_{\tilde{x}}^T \left\{\boldsymbol{v} \otimes \boldsymbol{\sigma}'\left[\boldsymbol{W}_a \boldsymbol{a}^k + \boldsymbol{d}\right]\right\}, \qquad l = 1, \ldots, m \quad (28)$$

where $\boldsymbol{C}_B^k(l, \bullet)$ is the lth row of the matrix $\boldsymbol{C}_B^k$. The symbol $\otimes$ denotes elementwise multiplication between vectors of equal dimensions, and $\boldsymbol{\sigma}'[\bullet]$ is a vector-valued function whose elements consist of the function $\sigma'(\bullet)$ evaluated componentwise at each element of its vector argument, for example,

$$\boldsymbol{\sigma}'[\boldsymbol{n}] \equiv [\sigma'(n_1) \quad \cdots \quad \sigma'(n_s)]^T \quad (29)$$

The output and gradient relations, Eqs. (27–28), must hold for all k, $k \in$ OP; together, they constitute feedback NN weight equations. They form a system of nonlinear transcendental equations whose order grows linearly with the number of operating points p in OP. The unknowns in the system consist of the neural network weights $\boldsymbol{W}_a$, $\boldsymbol{W}_{\tilde{x}}$, $\boldsymbol{v}$, $\boldsymbol{d}$, and b, and their number grows linearly with the number of nodes in the network s. If the system is not well-determined, that is, if the number of unknowns is not equal to the number of equations, it usually is not possible to obtain an exact solution. Furthermore, in data fitting problems, it often is the case that the systems to be solved are overdetermined, with more equations than there are unknowns. In the latter case, one can seek a least-squares solution,[14] which is obtained by minimizing some well-posed function of the error brought about by the approximate solution. However, because the information provided to the networks is considered to be satisfactory, an exact solution is sought, when possible, to achieve exact fitting of the data.

Command-Integral Networks

The command-integral NNs are expected to minimize integrals of the command-vector error to reduce the long-term effect of uncertain parameters or constant disturbances on the set point $(\boldsymbol{x}_c, \boldsymbol{u}_c)$. The dimension of the network input, comprising the command-error integral and the scheduling vector, is typically smaller than for feedback networks. However, because the set of requirements dictated by Eqs. (20) and (21) is equivalent to those of the feedback NN, the command-integral weight equations are equivalent to Eqs. (27) and (28). Of course, the gradients, in this case, are provided by the command-integral gain matrix $\boldsymbol{C}_I$ computed for OP. The structure of the two nonlinear systems is equivalent because only the input weights associated with the scheduling vector $\boldsymbol{W}_a$ appear implicitly in the equations (as distinct from all of the input weights $\boldsymbol{W}$). Furthermore, the command-integral networks must provide zero output for zero error input. Therefore, the command-integral networks can be determined in the same way as the feedback networks.

For each scalar network, the number of output equations equals the number of operating points p in OP. The number of gradient equations grows linearly with respect to p, with a proportionality constant that equals the number of state elements n. It can be shown that the number of unknowns in each network, that is, $\boldsymbol{W}_a$, $\boldsymbol{W}_{\tilde{x}}$, $\boldsymbol{v}$, and $\boldsymbol{d}$, grows linearly with the number of nodes s. In particular, the number of output weights is equal to s, $\boldsymbol{v} \in \Re^s$, whereas the number of input weights in $\boldsymbol{W}_{\tilde{x}}$ is equal to sn, $\boldsymbol{W}_{\tilde{x}} \in \Re^{s \times n}$. Therefore, when the number of nodes is chosen to equal the number of operating points, the number of output equations equals the number of output weights, and the number of gradient equations equals the number of $\tilde{\boldsymbol{x}}$ input weights. Although the nonlinear weight equations are underdetermined, with more unknowns than equations, the linear output and gradient equations that can be derived from them are fully determined and always can be computed exactly.

Algebraic Linear Weight Equations

Once the linear design specifications are obtained for all points in OP, the neural parameters are computed from the weight equations that specify the desired performance. For both the feedback and the command-integral networks, the two systems of nonlinear transcendental equations (27) and (28) can be written as three sets of linear equations that are readily solved using matrix inversion. The solution of the linear equations constitutes one of the multiple exact solutions of the corresponding nonlinear systems. The approach is based on the idea that if the input n_i^k to each node i is known for all k, $k \in$ OP, the left-hand side of the following linear system is a known vector:

$$\boldsymbol{n}^k = \boldsymbol{W}_a \boldsymbol{a}^k + \boldsymbol{d}, \qquad k = 1, \ldots, p \quad (30)$$

As a consequence, the output equations (27) are linear in the network parameters, taking the form

$$\boldsymbol{b} = -\boldsymbol{S}\boldsymbol{v} \quad (31)$$

where $\boldsymbol{b}$ is an s-dimensional vector composed of the scalar output bias b and $\boldsymbol{S}$ is a matrix of sigmoids evaluated at each input-to-node value n_i^k:

$$\boldsymbol{S} \equiv \begin{bmatrix} \sigma\left(n_1^1\right) & \sigma\left(n_2^1\right) & \cdots & \sigma\left(n_s^1\right) \\ \sigma\left(n_1^2\right) & \sigma\left(n_2^2\right) & \cdots & \sigma\left(n_s^2\right) \\ \vdots & \vdots & \ddots & \vdots \\ \sigma\left(n_1^p\right) & \sigma\left(n_2^p\right) & \cdots & \sigma\left(n_s^p\right) \end{bmatrix} \quad (32)$$

When the number of nodes is chosen to equal the number of operating points, $s = p$, the matrix $\boldsymbol{S}$ is square. Provided it also is nonsingular, the system in Eq. (31) admits a unique solution for $\boldsymbol{v}$, that is, $\boldsymbol{v} = -\boldsymbol{S}^{-1}\boldsymbol{b}$.

Once the input-to-node values are known and $\boldsymbol{v}$ has been determined, the gradient equations (28) become linear in the remaining neural parameters $\boldsymbol{w}_{\tilde{x}}$:

$$\boldsymbol{c} = \boldsymbol{X}\boldsymbol{w}_{\tilde{x}} \quad (33)$$

For convenience, the elements of the input weight matrix $\boldsymbol{W}_{\tilde{x}}$ have been reorganized columnwise into a vector through a vec operation,[15] that is, $\boldsymbol{w}_{\tilde{x}} \equiv \mathrm{vec}(\boldsymbol{W}_{\tilde{x}})$. The vector $\boldsymbol{c}$ is obtained from the linear gains computed at all points in OP. Take, for example, the initialization of the NN providing the first element of the control deviation $\Delta \boldsymbol{u}_B(t)$. If we let $C_{B_{ij}}^k$ symbolize the gain in the ith row and jth column of the feedback gain matrix $\boldsymbol{C}_B^k$, evaluated at the kth OP, then $\boldsymbol{c}$ is known from the linear control data:

$$\boldsymbol{c} \equiv \left[-C_{B_{11}}^1 \quad \cdots \quad -C_{B_{1n}}^1 \,\middle|\, \cdots \,\middle|\, -C_{B_{11}}^p \quad \cdots \quad -C_{B_{1n}}^p\right]^T \quad (34)$$

$\boldsymbol{X}$ denotes a $np \times ns$ sparse matrix composed of p block-diagonal submatrices, each of dimensions $n \times ns$:

$$\boldsymbol{X} \equiv \left[\begin{array}{cccc} \boldsymbol{B}^1 & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{B}^1 & \boldsymbol{0} & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{B}^1 \\ \hline & & \vdots & \\ \hline \boldsymbol{B}^p & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{B}^p & \boldsymbol{0} & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{B}^p \end{array}\right] \begin{array}{l} \left.\begin{array}{l} \\ \\ \\ \\ \end{array}\right\} n\text{-rows} \\ \\ \left.\begin{array}{l} \\ \\ \\ \\ \end{array}\right\} n\text{-rows} \end{array} \tag{35}$$

Every block $\boldsymbol{B}^k$ consists of a row vector whose elements can be obtained from the output weights and from the sigmoid-function derivative evaluated at each input-to-node value n_i^k:

$$\boldsymbol{B}^k \equiv \left\lfloor v_1 \sigma'\left(n_1^k\right) \quad v_2 \sigma'\left(n_2^k\right) \quad \cdots \quad v_s \sigma'\left(n_s^k\right) \right\rfloor \qquad k = 1, \ldots, p \tag{36}$$

When $s = p$, $\boldsymbol{X}$ is a square matrix, and the system in Eq. (33) can be solved uniquely for $\boldsymbol{w}_{\tilde{x}}$, provided Eq. (35) is nonsingular.

Finally, the assumption of known input-to-node values n_i^k is realized in a third linear system obtained from Eq. (30),

$$\boldsymbol{n} = \boldsymbol{A}\boldsymbol{w}_a \tag{37}$$

whose parameters are grouped in the vector $\boldsymbol{w}_a \equiv \text{vec}[\boldsymbol{W}_a \mid \boldsymbol{d}]$. $\boldsymbol{A}$ is a $ps \times (\ell + 1)s$ matrix obtained from all ℓ-dimensional scheduling vectors, $\boldsymbol{a}^k = [a_1^k \ \cdots \ a_\ell^k]^T$, $k = 1, \ldots, p$,

$$\boldsymbol{A} \equiv \left[\begin{array}{c|c|c} a_1^1 \boldsymbol{I}_s & & a_\ell^1 \boldsymbol{I}_s \\ a_1^2 \boldsymbol{I}_s & \cdots & a_\ell^2 \boldsymbol{I}_s \\ \vdots & & \vdots \\ a_1^p \boldsymbol{I}_s & & a_\ell^p \boldsymbol{I}_s \end{array} \middle| \begin{array}{c} \boldsymbol{I}_s \\ \boldsymbol{I}_s \\ \vdots \\ \boldsymbol{I}_s \end{array}\right] \tag{38}$$

The superscripts indicate at which point in OP the scheduling variables are being evaluated. The column vector $\boldsymbol{n}$ contains all input-to-node values:

$$\boldsymbol{n} \equiv \left[n_1^1 \quad n_2^1 \quad \cdots \quad n_s^1 \,\middle|\, \cdots \,\middle|\, n_1^p \quad n_2^p \quad \cdots \quad n_s^p\right]^T \tag{39}$$

When $p > (\ell + 1)$, Eq. (37) is overdetermined and must be consistent to have a solution.

NN Weight Solution

Input-to-node values must be obtained from Eq. (37) to solve Eqs. (31) and (33). A unique solution to the linear systems in Eqs. (31) and (33) exists when $\boldsymbol{S}$ and $\boldsymbol{X}$ are nonsingular. It can be shown that the input-to-node values determine the nature of $\boldsymbol{S}$ and $\boldsymbol{X}$ because repetitive values in $\boldsymbol{n}$ will render their determinants zero. It is the task of the following algorithm to use an effective strategy for the choice of $\boldsymbol{n}$ that will make all linear systems, Eqs. (31), (33), and (37), consistent. Because it is always possible to determine an effective distribution for the elements in $\boldsymbol{n}$, the neural parameters can be determined in one step.

The solution order of the preceding linear equations is key. By the use of all of the data available from linear control for OP and letting $s = p$, the matrix $\boldsymbol{A}$ and the vector $\boldsymbol{c}$ are computed from Eqs. (38) and (34), respectively. Next, the vector $\boldsymbol{n}$ is determined such that the matrix $\boldsymbol{S}$ is well-conditioned (i.e., with condition number less than $\varepsilon^{-1/2}$), where ε is the smallest positive number such that $\varepsilon + 1 > 1$ on the computing machine.

A strategy that produces a well-conditioned $\boldsymbol{S}$, with probability one, consists of generating $\boldsymbol{n}$ according to the following rule:

$$n_i^k = \begin{cases} r_i^k, & \text{if} \quad i \neq k \\ 0, & \text{if} \quad i = k \end{cases} \tag{40}$$

where r_i^k is chosen from a normal distribution with mean zero and variance, one obtained using a random number generator with a single seed. When $i = k$, n_i^k is assigned a zero value so that each sigmoid, $\sigma(n) \equiv (e^n - 1)/(e^n + 1)$, can be centered at one of the operating points in OP. Then, Eq. (37) is solved for $\boldsymbol{w}_a$ using the left pseudoinverse of $\boldsymbol{A}$, $\boldsymbol{A}^{\text{PI}}$:

$$\hat{\boldsymbol{w}}_a = \boldsymbol{A}^{\text{PI}}\boldsymbol{n} \tag{41}$$

As a consequence, $\hat{\boldsymbol{w}}_a$ is the best approximation to the solution of the system because the system is not likely to have an exact solution. When this value for $\boldsymbol{w}_a$ is substituted back in Eq. (37), only a relatively close estimate to the values chosen earlier [Eq. (40)] is obtained for $\boldsymbol{n}$:

$$\hat{\boldsymbol{n}} = \boldsymbol{A}\hat{\boldsymbol{w}}_a \tag{42}$$

However, nothing prevents us from using this value for $\boldsymbol{n}$. Although overdetermined, this system has an exact and unique solution because the equations are now consistent.

Because $\hat{\boldsymbol{n}}$ is computed on the basis of Eq. (40), the sigmoids are very nearly centered. Whereas it is desirable that each sigmoid be centered for a given input $\boldsymbol{p}^k$, the same sigmoid should be close to saturation when the input corresponds to any other point in OP; this prevents ill-conditioning of $\boldsymbol{S}$. When it is considered that the sigmoids come close to being saturated for an input whose absolute value is greater than 5, it is found desirable for the input-to-node values in $\boldsymbol{n}$ to have variance of about 10. A factor f can be obtained for this purpose from the element in $\hat{\boldsymbol{n}}$ with largest absolute value $n_{\max}$, as follows:

$$f = 10/n_{\max} \tag{43}$$

The final vector of input-to-node values $\boldsymbol{n}$ is obtained by multiplying Eq. (42) by f:

$$\boldsymbol{n} = f\hat{\boldsymbol{n}}, \qquad \boldsymbol{w}_a = f\hat{\boldsymbol{w}}_a \tag{44}$$

Consequently, Eq. (42) remains unaltered, and $\boldsymbol{n}$ has the desired variance, 10.

The matrix $\boldsymbol{S}$ can be computed from $\boldsymbol{n}$, and $\boldsymbol{v}$ is obtained by inverting the system in Eq. (31). With the knowledge of $\boldsymbol{v}$ and $\boldsymbol{n}$, the matrix $\boldsymbol{X}$ can be formed as stated in Eqs. (35–36), and $\boldsymbol{w}_{\tilde{x}}$ can be computed by inverting the system in Eq. (33). The matrices $\boldsymbol{S}$ and $\boldsymbol{X}$ in Eqs. (32) and (33) are consistently well conditioned, rendering the solution of these linear systems by matrix inversion straightforward as well as highly accurate. Thus, the network's parameters are obtained in a noniterative fashion and satisfy both output and gradient weight equations (27) and (28) exactly.

A simple two-node neural network makes the point. The linear data are provided for two OPs, A and B (Fig. 4), for a one-dimensional state x and a single scheduling variable a. As anticipated, a two hidden node NN fulfills the initialization requirements of Eqs. (27–28). The network output is required to approach zero for a zero state deviation $\tilde{x}$, and the gradient of the output surface is required to match C_B at the two operating points. The method computes the weights of the network producing a minimal-order,

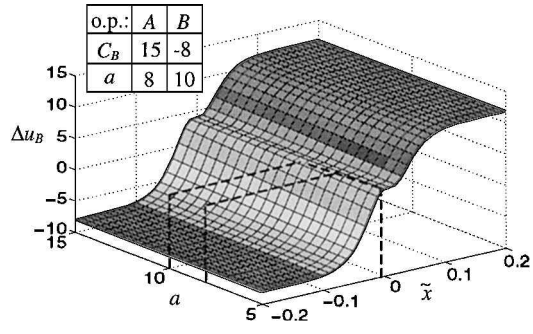

Fig. 4 Output surface of a two-node feedback NN corresponding to the algebraic solution matching the linear data provided in the legend at operating points A and B.

smooth interpolating surface, shown in Fig. 4, defining the network size and preventing data overfitting at their origin.

NN Flight Control Implementation and Results

The nonlinear PI NN controller structure specified in earlier sections and illustrated in Fig. 2 is implemented for the steady-state response control of a transport aircraft in its operational domain, that is, the flight envelope shown in Fig. 5. The set point $(\boldsymbol{x}_c, \boldsymbol{u}_c)$ is determined from the aircraft's dynamic and output equations linearized at the existing flight conditions.[12] With the procedure described earlier, the linear gains of the PI controller, $\boldsymbol{C}_B$ and $\boldsymbol{C}_I$, are designed at 14 OPs that are referred to as design points and constitute OP (Fig. 5). For simplicity, a reduced-order lateral-axis model is considered in the linear design. The state vector consists of yaw rate r (degrees per second), sideslip angle β (degrees), roll rate p (degrees per second) and roll angle ϕ (degrees), and $\boldsymbol{x} = [r \;\; \beta \;\; p \;\; \phi]^T$. The controls are aileron δA (degrees) and rudder δR (degrees), and $\boldsymbol{u} = [\delta A \;\; \delta R]^T$. Both the output $\boldsymbol{y}_s$ and the command input $\boldsymbol{y}_c$ consist of two state elements: the sideslip angle and the roll angle; thus, $e = e_c = 2$.

The nonlinear controller includes four feedforward NNs of the type shown in Fig. 3, each with 14 nodes in the hidden layer. The feedback NN NN_B is composed of two scalar networks with the same input, one for each control element:

$$\begin{bmatrix} \Delta\delta A(t) \\ \Delta\delta R(t) \end{bmatrix}_B = \begin{bmatrix} \mathrm{NN}_{B_1}(\tilde{\boldsymbol{x}}(t), \boldsymbol{a}) \\ \mathrm{NN}_{B_2}(\tilde{\boldsymbol{x}}(t), \boldsymbol{a}) \end{bmatrix} \tag{45}$$

Similarly, the command-integral NN NN_I consists of two scalar networks:

$$\begin{bmatrix} \Delta\delta A(t) \\ \Delta\delta R(t) \end{bmatrix}_I = \begin{bmatrix} \mathrm{NN}_{I_1}(\xi(t), \boldsymbol{a}) \\ \mathrm{NN}_{I_2}(\xi(t), \boldsymbol{a}) \end{bmatrix} \tag{46}$$

where $\xi(t)$ is the time integral of the command error, $\boldsymbol{y}(t) - \boldsymbol{y}_c(t)$. The scheduling vector $\boldsymbol{a}$ is composed of nominal true airspeed V_0 and altitude H_0 with both variables scaled so that they vary between zero and one.

The final architecture used for each NN_B, shown in Fig. 6, is fully motivated by control specifications; a similar one also is used for each NN_I with four inputs instead of six. The networks are guaranteed to provide zero output for zero state deviations (NN_B) or for zero integral command errors (NN_I). The direct contribution of the scheduling variables to the output is subtracted using a mirror image of the initialized network with zero state perturbations because they would otherwise produce small biases between interpolating points. However, the contribution of the scheduling variables to the network gradients remains unaltered and, as expected, $\boldsymbol{a}$ schedules the gain interpolation throughout OR. The weights of NN_B and NN_I are determined by solving algebraic equations in the form of Eqs. (37), (31), and (33), such that the requirements in Eqs. (18–19) and (20–21) are fulfilled exactly at all design points. Any other operating point in OR is an interpolation point for which the linear gains have not been designed; therefore, they must be produced by the neural networks.

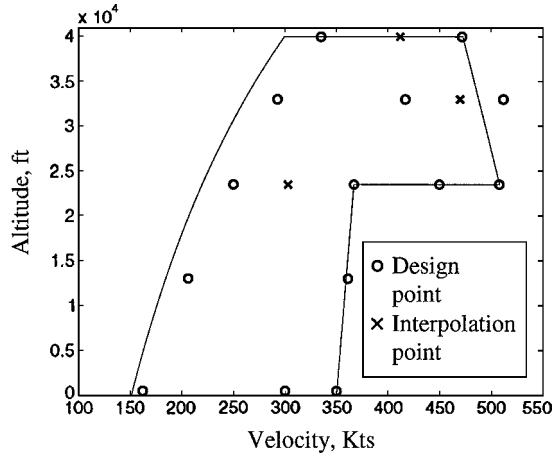

Fig. 5 Flight envelope and significant operating points used for design and evaluation of the NN controller.

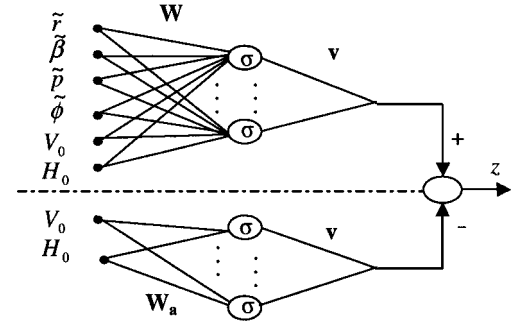

Fig. 6 Final NN_B architecture (input–output biases are omitted for simplicity).

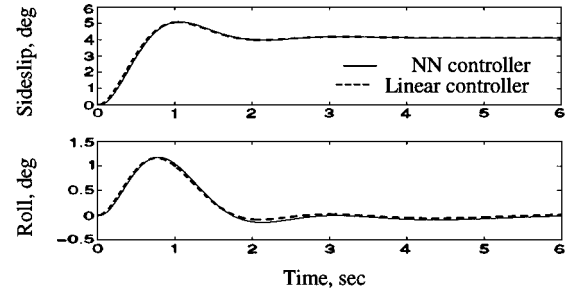

Fig. 7 Comparison between the response of the NN controller and the response of the linear controller associated to the present design point, (V_0, H_0) = (367 kn, 23,456 ft), subject to a 4-deg sideslip angle step command (case 1).

The state histories, in particular sideslip and roll angle, of a full, six-degree-of-freedom, nonlinear transport aircraft simulation subject to a command input $\boldsymbol{y}_c$ are used as performance evaluations. Three command inputs are used; in each case the aircraft is flying at operating conditions corresponding to a design point or to an interpolation point. The state response is judged against that of a linear PI controller that is specifically designed for the OP under consideration.

Case 1: Response at a Design Point

The neural controller response to a 4-deg sideslip angle command is plotted with a solid line in Fig. 7, for a design point with a nominal airspeed V_0 of 367 kn and an altitude H_0 of 23,456 ft. The response of the linear controller designed for these flight conditions is also shown (represented by a dashed line) for comparison. As expected, the response obtained with the neural controller is virtually identical to that obtained with the linear controller because all linear gains designed are matched exactly by the algebraic neural network solution. In fact, this level of performance is achieved at all flight conditions in OP.

Case 2: Response at an Interpolation Point

The response of the aircraft flying at an interpolation point, $(V_0, H_0) = (470 \text{ kn}, 33{,}000 \text{ ft})$, and subject to a 6-deg roll-angle command input, is shown in Fig. 8 for the neural controller and for a linear controller specifically designed for these flight conditions. Because this OP was not considered in the linear design, the neural controller must interpolate between the available linear controllers in producing the control history. In this region of high dynamic pressure, controller gains change rapidly with varying flight conditions. Still, in comparison to linear gains that are purposely generated for evaluation, the neural controller response shows a small improvement in comparison to its linear counterpart; the sideslip angle has a lower transient peak, though it takes a little longer to decay. If the match at any given test point is not satisfactory, the neural network control response can be made satisfactory by adding that point to the design set.

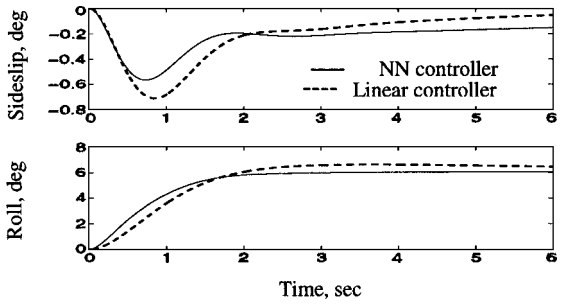

Fig. 8 Comparison between the response of the NN controller and the response of a linear controller specifically generated for the present interpolation point, $(V_0, H_0) = (470$ kn, 33,000 ft), subject to a 6-deg roll angle step command (case 2).

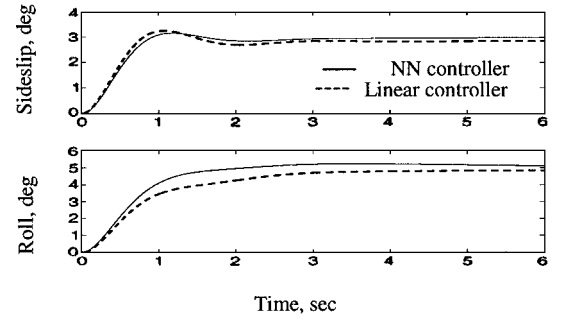

Fig. 9 Comparison between the response of the NN controller and the response of a linear controller specifically generated for the present interpolation point, $(V_0, H_0) = (412$ kn, 40,000 ft), subject to 3-deg sideslip angle and 5-deg roll angle step command (case 3).

Case 3: Response at an Interpolation Point

The aircraft response is evaluated for a second interpolation point, $(V_0, H_0) = (412$ kn, 40,000 ft), and a command input of 3-deg sideslip and 5-deg roll angle, to demonstrate the consistency of the results for different flight conditions. Figure 9 shows the comparison between the response of the same NN controller and that of a linear controller computed for the OP. The network's generalized response is again seen to be close to the linear one. Although it only is feasible to design linear controllers for a finite set OP $\subset$ OR, a global neural controller handles the entire envelope OR with a performance analogous to that of a linear controller specifically generated for the given flight condition.

Conclusions

The foundations have been laid for a novel approach that combines the expertise gained from linear control design with adaptive, nonlinear control concepts and computational NNs. The principles introduced can be applied to any nonlinear control law that affords the established gain-scheduled law formulation and for which the gains can be specified at a set of points in the plant's operating envelope.

The nonlinear control system is obtained by replacing the linear gains of a well-known linear control structure, here exemplified by a PI controller, with NNs. The networks' parameters, size, and architecture, apt to meet the linear control specifications available, are determined in one step, by solving linear algebraic equations. This produces a network of minimal NNs that approximate the hypersurfaces of a global, nonlinear control law and that match equivalent, locally linearized controllers exactly.

The effectiveness of this novel technique is demonstrated by implementing feedback and command-integral NN controllers for the control of the command-input response of a transport aircraft over its flight envelope. The simulations show that a relatively small number of OPs and, thus, of hidden neural nodes, is sufficient to obtain satisfactory NN control over a wide operating region.

Acknowledgments

This research has been supported by the Federal Aviation Administration (FAA) and NASA FAA Grant 95-G-0011.

References

[1]Cybenko, G., "Approximation by Superposition of a Sigmoidal Function," *Mathematics of Control, Signals, and Systems*, Vol. 2, 1989, pp. 359–366.

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

[3]Narendra, K. S., "Neural Networks for Control: Theory and Practice," *Proceedings of the IEEE*, Vol. 84, No. 10, 1996, pp. 1385–1406.

[4]Neidhoefer, J., and Krishnakumar, K., "Nonlinear Control Using Neural Approximators with Linear Control Theory," *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, AIAA, Reston, VA, 1997, pp. 364–372.

[5]Iwasa, T., Morizumi, N., and Omatu, S., "Temperature Control in a Batch Process by Neural Networks," *Proceedings of the 1997 IEEE Conference on Neural Networks*, Vol. 4, Inst. of Electrical and Electronics Engineers, New York, 1997, pp. 2430–2433.

[6]Barto, A. G., "Reinforcement Learning and Adaptive Critic Methods," *Handbook of Intelligent Control*, edited by D. A. White and D. A. Sofge, Van Nostrand Reinhold, New York, 1992, pp. 469–481.

[7]Stengel, R. F., and Ray, L. R., "Stochastic Robustness of Linear-Time Invariant Control Systems," *IEEE Transactions on Automatic Control*, Vol. 36, No. 1, 1993, pp. 82–87.

[8]Stengel, R. F., and Ray, L. R., "A Monte Carlo Approach to the Analysis of Control System Robustness," *Automatica*, Vol. 29, No. 1, 1993, pp. 229–236.

[9]Stengel, R. F., and Marrison, C., "Stochastic Robustness Synthesis Applied to a Benchmark Control Problem," *International Journal of Robust and Nonlinear Control*, Vol. 5, No. 1, 1995, pp. 13–31.

[10]Stengel, R. F., and Marrison, C., "Robust Control System Using Random Search and Genetic Algorithms," *IEEE Transactions on Automatic Control*, Vol. 42, No. 6, 1997, pp. 835–839.

[11]Stengel, R. F., and Marrison, C., "Design of Robust Control Systems for Hypersonic Aircraft," *Journal of Guidance, Control, and Dynamics*, Vol. 21, No.1, 1997, pp. 58–63.

[12]Huang, C., and Stengel, R. F., "Restructurable Control Using Proportional–Integral Model Following," *Journal of Guidance, Control, and Dynamics*, Vol. 13, No. 2, 1990, pp. 303–309.

[13]Ferrari, S., and Stengel, R. F., "Algebraic Training of a Neural Network," *Proceedings of the American Control Conference*, Inst. of Electrical and Electronics Engineers, New York, 2001, pp. 1605–1610.

[14]Fletcher, R., *Practical Methods of Optimization*, Wiley, New York, 1980.

[15]Graham, A., *Kronecker Products and Matrix Calculus: with Applications*, Ellis Horwood, Chichester, England, UK, 1981.