JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 16, No. 6, November–December 1993

Identification of Aerodynamic Coefficients Using Computational Neural Networks

Dennis J. Linse* and Robert F. Stengel†
Princeton University, Princeton, New Jersey 08544

Precise, smooth aerodynamic models are required for implementing adaptive, nonlinear control strategies. Accurate representations of aerodynamic coefficients can be generated for the complete flight envelope by combining computational neural network models with an estimation-before-modeling paradigm for on-line training information. A novel method of incorporating first partial derivative information is employed to estimate the weights in individual feedforward neural networks for each aerodynamic coefficient. The method is demonstrated by generating a model of the normal force coefficient of a twin-jet transport aircraft from simulated flight data, and promising results are obtained.

Introduction

MODERN nonlinear control techniques offer exciting potential for aircraft control.[1-3] These techniques require comprehensive aerodynamic models that must be differentiable with respect to the state and control. Given the continuous state and output equations,

$$\dot{x} = f(x) + g(x)u \tag{1}$$

$$y = h(x) \tag{2}$$

a nonlinear-inverse-dynamic (NID) control law[2] is developed by repeated differentiation of Eq. (2) and substitution of Eq. (1) until the control appears in each element of y or its derivatives. The global representation of the aerodynamic model contained in $f(\cdot)$, $g(\cdot)$, and $h(\cdot)$ must be sufficiently smooth to calculate such a control law. Many system identification methods used for aircraft fail to provide the globally smooth models needed by these control laws. For example, the estimation-before-modeling (EBM) technique[4,5] models the aerodynamic coefficients using multiple regression schemes on partitions of the state and control space. Although the partitions span the space, these global models, in general, are not continuous across the partition boundaries, much less differentiable.

Maximum-likelihood estimation (MLE) is the most commonly used technique for extracting aircraft parameters from flight-test data.[6,7] The MLE technique postulates a parametric model (usually linear, more recently nonlinear) of the aircraft and adjusts the parameters to minimize the negative logarithm of the likelihood that the model output fits a given measurement sequence. For flight-test data, individual models are fit for each test point generating estimates of the aircraft stability and control derivatives and uncertainty levels. Since the primary use of the derivatives is verification of predicted performance, little more than fairing is done to generate global models through the test points.[6] Iterative minimization procedures are employed, so on-line estimation is not possible.

Nonsmooth aerodynamic models can be coerced into smooth models with sufficient postprocessing, but this postprocessing may reduce the fidelity of the model. In Ref. 2 extensive tabular aerodynamic data were fitted with cubic splines before applying NID control techniques. Cubic splines ensured continuity and smoothness across subspace boundaries.

An on-line system identification and nonlinear control paradigm that integrates computational neural networks as an aerodynamic modeler within the EBM framework was presented in Refs. 8 and 9. Using the plant measurements z, an extended Kalman-Bucy filter[10] (EKBF) estimates the state of an augmented plant model consisting of the standard aircraft state $\hat{x}(t)$ and the aircraft specific forces and moments $\hat{g}(t)$. The neural network learns an aerodynamic model $\hat{g}(\hat{x}, u)$ from $\hat{x}(t)$, $\hat{g}(t)$, and the control $u(t)$. A nonlinear control law using the estimated state $\hat{x}(t)$, the aerodynamic model $\hat{g}(\hat{x}, u)$, and an external command input v completes the loop and forms an adaptive, nonlinear control system. A block diagram of the complete controller is given in Fig. 1.

Using the state/force EKBF and the network estimation model, excellent matches of aerodynamic coefficient histories were achieved with a simple neural network model for single flight conditions, and the corresponding functional relationship $\hat{g}(\hat{x}, u)$ was well identified.[9] Expanding the number of training flight conditions to span the flight envelope continued to yield network models with excellent matches of the coefficient histories; however, $\hat{g}(\hat{x}, u)$ and the corresponding aircraft stability-and-control derivatives were not equally well estimated over the entire space.

This paper describes a method of improved aerodynamic modeling that augments the EBM-based identifier and neural network modeler with additional information about the first partial derivatives of the aerodynamic coefficients. Including both function and derivative information in the network training algorithm constrains and directs the training by reducing the number of possible functions that match a particular measurement sequence. A simple example shows a 30% improvement in network accuracy with these methods, and representation of aerodynamic derivatives is improved by 50–100%.

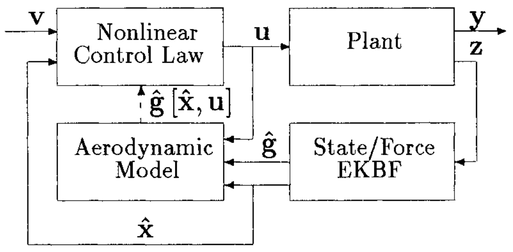

Fig. 1 Integration of system identification and nonlinear control.

Presented as Paper 92-0172 at the AIAA 30th Aerospace Sciences Meeting, Reno, NV, Jan. 6-9, 1992; received March 29, 1992; revision received Jan. 14, 1993; accepted for publication Jan. 29, 1993. Copyright © 1993 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved.

*Graduate Research Assistant, Department of Mechanical and Aerospace Engineering; currently Engineer, Systems Control Technology, Inc., Lexington Park, MD. Member AIAA.

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

Computational Neural Networks

Computational neural networks are biologically inspired, massively parallel computational paradigms. Computational neural networks are used in a variety of applications because they are adaptive, they learn from examples, and they can provide excellent function approximation.

Multilayer feedforward neural networks combine simple nonlinear computational elements into a layered architecture to compute a static nonlinear function. Individual network nodes, with specified nonlinear or activation functions, are grouped together in distinct layers, with multiple layers connected for the full network. The output of the kth layer of the network $r^{(k)}$ is computed as the nonlinear transformation of the weighted sum of the outputs of the $(k-1)$ layer and a bias

$$q^{(k)} = W^{(k-1)}r^{(k-1)} + b^{(k-1)} \tag{3}$$

$$r^{(k)} = \sigma^{(k)}[q^{(k)}] \tag{4}$$

where W and b are the weights and biases of the indicated layers, and $\sigma[q]$ is a vector-valued function composed of individual node activation functions

$$\sigma[q] = [\sigma_1(q_1),\ldots,\sigma_n(q_n)]^T \tag{5}$$

The outputs of layer $k-1$ are connected to the inputs of layer k successively such that the final form of an N_L-layer network is

$$r^{(N_L)} = \sigma^{(N_L)}(W^{(N_L-1)}\sigma^{(N_L-1)}[\cdots\sigma^{(1)}(W^{(0)}r^{(0)} + b^{(0)}) + \cdots] \tag{6}$$

Defining $r = r^{(0)}$ as the input vector and $z = r^{(N_L)}$ as the output vector, the network provides the function approximation,

$$z = h(r; w) \tag{7}$$

where w is a vector combining all of the weights and biases of all of the network layers, and $h(\cdot)$ is the overall function of r parameterized by w. A sample two-input/one-output, single hidden-layer network is given in Fig. 2.

Smoothness of a network function is important if the network is to be used in the suggested nonlinear control paradigms. The order of differentiability of Eq. (7), by its construction, is the same as that of the node activation functions $\sigma[\cdot]$. Assuming all of the activation functions are at least once differentiable, the derivative of the network with respect to its inputs is

$$\frac{\partial h}{\partial r} = \frac{\partial\sigma^{(N_L)}}{\partial q^{(N_L)}} W^{(N_L-1)}\cdots\frac{\partial\sigma^{(1)}}{\partial q^{(1)}} W^{(0)} \tag{8}$$

The derivatives of the activation function vectors $\partial\sigma^{(i)}/\partial q^{(i)}$ are diagonal matrices with $(\sigma_1',\ldots,\sigma_n')$ along the diagonal. The *logistic sigmoid* $\sigma(q) = (1+e^{-q})^{-1}$ is a commonly used node activation function. It is infinitely differentiable, and its first derivative is quite simply calculated as

$$\frac{\partial\sigma}{\partial q} = \sigma' = \sigma(q)[1-\sigma(q)] \tag{9}$$

A network composed of logistic sigmoid activation functions, weights, and biases is infinitely differentiable.

Layered feedforward neural networks are closely related to basis function approximation and regularization theory.[11] In principle, any continuous function[12] or nth-order differentiable function[13] can be approximated accurately by such networks; however, there is no guarantee that the desired network size can be chosen and weights learned from a set of (possibly noisy) training examples.

Currently, the size of a network is selected by trial and error, with past experience of other network researchers guiding the way. The theorems of Refs. 12 and 13 suggest that one hidden layer is sufficient to represent differentiable functions. The number of nodes in that layer is selected to be large enough to generate good representations but small enough to prevent spurious results. This is an area of much active research in the neural network community.

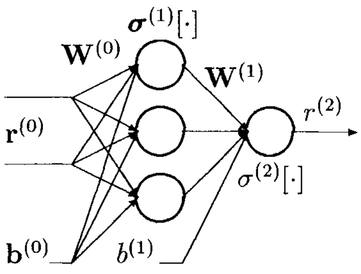

Fig. 2 Simple feedforward network.

The input presented to a network is often scaled, particularly if the elements have magnitudes much greater than or less than 1. An affine scaling is used to bring the elements of the input into the region of significant activity of the network node. For example, an altitude input element that ranges from 0 to 40,000 ft might be scaled to lie in $[-5, +5]$. Since the network performs an affine scaling as its first step [Eq. (6)], such external scaling is unnecessary; the network will learn its own desired scaling. In practice, external input scaling is useful to prevent large initial weight transients that may slow the training procedure or prevent convergence altogether.

Given a set of N_p training examples $\{r_i, d_i\}$, where $d_i = d(r_i)$ is the value of $d(\cdot)$, the function to be approximated, for a given input r_i, a suitable least-squares cost function, is

$$J = \sum_{i=1}^{N_p} \epsilon_i^T\epsilon_i \tag{10}$$

where the network error vector for the ith trial ϵ_i is

$$\epsilon_i = d_i - h(r_i; w) \tag{11}$$

J is minimized with respect to w, giving the minimum mean-squared-error network.

Searching for the weight vector that minimizes Eq. (10) may be a difficult nonlinear optimization problem. The problem is often constrained by the desire to keep the weight adjustment algorithm localized, that is, to update the weights associated with a node using information available only at that node. Localized algorithms simplify hardware implementation, although they may ignore coupling information that is important to training.

Traditional nonlinear optimization algorithms—including steepest descent,[14,15] conjugate gradients,[16] and various Newton-type algorithms such as the Marquardt-Levenberg algorithm[17]—have been applied to feedforward computational neural networks. Back propagation, a steepest-descent algorithm related to the discrete-stage optimal-control solution,[14] is the most common algorithm.[15] Modifications and improvements to the back-propagation algorithm, such as learning momentum, have been developed.[15,16,18]

The extended Kalman filter is a particularly attractive alternative method that transforms the optimization of Eq. (10) and training of Eq. (7) into the estimation of a dynamic system.[19] A localized version of the extended Kalman filter algorithm also has been developed.[20] The version of the extended Kalman filter described later follows the first approach.

Extended Kalman Filter for Network Training

The state and measurement vectors of a nonlinear, discrete-time system are given by

$$x_{k+1} = \phi(x_k) + n_k \tag{12}$$

$$z_k = h(x_k) + v_k \tag{13}$$

where $x, n \in R^n$, $z, v \in R^m$, and $\phi(\cdot)$ and $h(\cdot)$ are general nonlinear functions in R^n and R^m. Symbols $\{n_k\}$ and $\{v_k\}$ are zero-mean, white Gaussian processes with covariances Q_k and R_k. The x_0 is a Gaussian random variable with mean $\bar{x}_0$ and covariance P_0.

Assuming $\phi(\cdot)$ and $h(\cdot)$ are sufficiently smooth, an extended Kalman filter can be defined for the system (12) and (13) (see, e.g., Refs. 10 and 21). The filter equations are

$$\hat{x}_k^{(-)} = \phi(\hat{x}_{k-1}) \tag{14}$$

$$P_k^{(-)} = \Phi_{k-1} P_{k-1} \Phi_{k-1}^T + Q_k \tag{15}$$

$$K_k = P_k^{(-)} H_k^T (H_k P_k^{(-)} H_k^T + R_k)^{-1} \tag{16}$$

$$\hat{x}_k = \hat{x}_k^{(-)} + K_k \{ z_k - h[\hat{x}_k^{(-)}] \} \tag{17}$$

$$P_k = (I - K_k H_k) P_k^{(-)} (I - K_k H_k)^T + K_k R_k K_k^T \tag{18}$$

The filter is initialized with $\hat{x}_0 = \bar{x}_0$ (the mean) and P_0. The superscript $(-)$ indicates an intermediate value after propagation but before updating with new information. Φ_k and H_k are the state and measurement Jacobian matrices:

$$\Phi_k = \left. \frac{\partial \phi}{\partial x} \right|_{\hat{x}_k} \tag{19}$$

$$H_k = \left. \frac{\partial h}{\partial x} \right|_{\hat{x}_k^{(-)}} \tag{20}$$

A symmetric form is used for the covariance update [Eq. (18)]. This ensures that P_k remains symmetric and positive definite as long as $P_k^{(-)}$ is positive definite and R_k is positive semidefinite at the expense of more computation.[10]

The extended Kalman filter is derived from the linear Kalman filter with the assumption that nonlinear effects are modeled adequately by the propagation of the mean [Eq. (14)], whereas disturbances and measurement errors are small enough to allow linear estimates of covariance evolution and measurement update [Eqs. (15-18)]. The extended Kalman filter (EKF) is neither linear nor optimal, but if the system is sufficiently well behaved, good results are obtained.

Modeling the Network for Kalman Filter Training

Identifying the network weight vector w as the state vector x in Eq. (12), a neural network can be expressed as a nonlinear dynamic system

$$w_k = w_{k-1} + n_{k-1} \tag{21}$$

$$z_k = h(w_k, r_k) + v_k \tag{22}$$

With the process and measurement noise sequences $\{n_k\}$ and $\{v_k\}$ as defined earlier, Eqs. (21) and (22) have the form of Eqs. (12) and (13), where $\phi(x) = x$, and h is defined by the network forward propagation [Eq. (7)]. For simplicity, the network is assumed to have one output.

Given a sequence of noise corrupted measurements $\{z_k\}$ and the corresponding network input sequence $\{r_k\}$, the weight vector w is estimated by an extended Kalman filter. For Eq. (21), the state Jacobian matrix Φ_k is an identity matrix for all k. This makes the covariance propagation step, Eq. (15), very simple. Computation of the measurement Jacobian matrix,

$$H_k = \left. \frac{\partial h}{\partial w} \right|_{(w_k, r_k)} \tag{23}$$

is a straightforward application of the chain rule, given the differentiability of the network nonlinearities and the form of Eq. (7). H_k has dimension $1 \times N_w$, where N_w is the number of weights in the network.

The propagation of w_k [Eq. (21)] is modeled as a dynamic system driven by a white Gaussian process although w is an internal variable that is not subject to noise. The covariance Q of the pseudonoise sequence $\{n_k\}$ controls the convergence of the state covariance P_k. In an undriven system ($Q = 0$), the covariance matrix converges to zero.[22] This prevents further state updating since, by Eq. (16), K_k is zero if $P_k = 0$.

The order of an extended Kalman filter used as a network training algorithm grows as the length of the network weight vector. The largest network used in this paper is a single hidden-layer network with 8 inputs, 5 hidden nodes, and 1 output, resulting in 51 weights and biases. Ordinarily, a Kalman filter of this size would be difficult to compute; however, the matrix to be inverted in Eq. (16) has dimension $m \times m$, where m is the output dimension. As $m = 1$, the matrix inversion is simply a scalar division.

Incorporating Partial Derivatives in Network Training

For a scalar function $d(\cdot)$, the network training error ϵ_i [Eq. (11)] is a scalar, and the cost [Eq. (10)] is based only on errors in the function value; however, useful training information also is contained in the error between desired and actual slopes. If, in addition to the training data $\{r_i, d_i\}$, measurements of the first partial derivatives $\partial d / \partial r$, evaluated at r_i, are available, a new weighted least-squares cost function can be defined

$$J = \sum_{i=1}^{N_p} \epsilon_i^T R^{-1} \epsilon_i \tag{24}$$

where R^{-1} is an appropriately dimensioned weighting matrix. The ϵ_i is formed by augmenting the original scalar error ϵ_i such that

$$\epsilon_i = \begin{bmatrix} d_i - h(w, r_i) \\ \dfrac{\partial d}{\partial r}(r_i) - \dfrac{\partial h}{\partial r}(w, r_i) \end{bmatrix} \tag{25}$$

In Eq. (24), R^{-1} weights the relative importance of each element of ϵ_i, that is, the importance of errors in the fit of the function h compared to errors in the fit of each of the derivatives $\partial h / \partial r_j$.

Although any of the network training algorithms just mentioned could be reformulated based on the function-derivative cost, the EKF training method is extended here. The propagation of w_k [Eq. (21)] remains the same, whereas the measurement equation (22) is expanded to include the partial derivatives

$$z_k = \begin{bmatrix} h(w_k, r_k) \\ \dfrac{\partial h}{\partial r}(w_k, r_k) \end{bmatrix} + v_k \tag{26}$$

and $\partial h / \partial r$ is calculated using Eq. (8). With this measurement equation, the measurement Jacobian matrix becomes

$$H_k = \left. \begin{bmatrix} \dfrac{\partial h}{\partial w} \\ \dfrac{\partial^2 h}{\partial w \partial r} \end{bmatrix} \right|_{(w_k, r_k)} \tag{27}$$

This expanded matrix H_k has dimension $(N_i + 1) \times N_w$, where N_i is the number of network inputs.

In the EKF formulation, the covariance R of the measurement-noise sequence v_k is the same as R in the weighted least-squares cost function [Eq. (24)]. It adjusts the relative importance of each of the measurements during network training. Equivalently, it weights the relative accuracy of each of the measurements. Although containing more information about the function, the partial derivatives may also contain more errors. Thus, a suitable $\boldsymbol{R}$ is important for accurate network learning.

Training Example

A simple example demonstrates the added benefits of the function-derivative-training (FD-training) method over a standard function-training (F-training) method. This example exercises only the aerodynamic model block in Fig. 1 using random data rather than the dynamic data that would be produced by simulated or actual flight tests. The demonstration function is a lift coefficient curve assumed to be

$$C_L(\alpha,q,\delta_E) = C_{L_{ST}}(\alpha) + C_{L_q}\hat{q} + C_{L_{\delta_E}}\delta_E \tag{28}$$

where α is the angle of attack, $\hat{q}$ ($= q\bar{c}/2V$) the nondimensionalized pitch rate, and δ_E the elevator deflection angle. Realistic numerical values are chosen for the aerodynamic derivatives based on a twin-jet transport aircraft.[23] The pitch-rate derivative C_{L_q} ($= \partial C_L/\partial\hat{q}$) is 7.0, and the elevator-angle derivative $C_{L_{\delta_E}}$ ($= \partial C_L/\partial\delta_E$) is 0.006 deg^{-1}. The angle-of-attack contribution $C_{L_{ST}}(\alpha)$ is a linear function for low angles of attack and a quadratic function for higher angles of attack. It is modeled as

$$C_{L_{ST}}(\alpha) = 0.0952\alpha + 0.1048 \qquad \alpha < 9 \text{ deg}$$

$$C_{L_{ST}}(\alpha) = -0.0095\alpha^2 + 0.2667\alpha - 0.6667 \qquad \alpha \geq 9 \text{ deg}$$

Using Eq. (28) as the desired function d, two identical feedforward networks with three inputs, one hidden layer of three sigmoidal nodes, and one output are initialized with the same random weights chosen from a zero-mean Gaussian distribution with variance of 0.1. The training data are selected by randomly choosing 2000 points from a uniform distribution bounded by

$$\alpha \in [-5 \text{ deg}, 19 \text{ deg}] \tag{29}$$

$$\hat{q} \in [-0.005, 0.005] \tag{30}$$

$$\delta_E \in [-20 \text{ deg}, 20 \text{ deg}] \tag{31}$$

Equation (30) represents q variations of ± 20 deg/s at the given velocity V. All network inputs are scaled to lie in $[-1, 1]$ so that the primary region of variation of the node activation functions is initially active.The lift coefficient C_L is calculated from Eq. (28) and is corrupted with zero-mean Gaussian noise with a variance of 0.1. The first network is trained using only function information (F training). The second network is trained with the function-derivative method using additional measurements of $\partial C_L/\partial\alpha$, $\partial C_L/\partial\hat{q}$, and $\partial C_L/\partial\delta_E$ (FD training). These measurements are also corrupted with zero-mean Gaussian noise with variances of 0.5, 6.0×10^{-4}, and 0.7. The initial state covariance $\boldsymbol{P}_0$ is $10^4\boldsymbol{I}$, and the constant process-noise covariance $\boldsymbol{Q}$ is $10^{-3}\boldsymbol{I}$, where $\boldsymbol{I}$ is the appropriately sized identity matrix. The measurement-noise covariance matrices are identical to the simulated noise used in the example. Thus, for F training, $\boldsymbol{R} = 0.1$, and for FD training $\boldsymbol{R}$ is a diagonal matrix with values (0.1, 0.5, 0.7, 6.0×10^{-4}) along the diagonal.

A comparison of the variation of Eq. (28) and two of its three partial derivatives to the two network approximations of this function is given in Figs. 3-5 for angle-of-attack variations. These plots represent one-dimensional slices through the fully trained three-dimensional input space. The FD-trained network performs better than the F-trained network for C_L and $\partial C_L/\partial\alpha$. There is a substantial improvement in the modeling of $\partial C_L/\partial q$. The remaining partial derivative shows a similar improvement. The rms errors are given in Table 1 for both networks and also for two other networks trained with the same data. The errors for the function and derivatives for the FD-trained function are all considerably smaller than the F-trained function. As expected, including partial derivatives in the network training process improves the ability of the training process to find an accurate model of the desired function and its derivatives.

The results also indicate the possibility of faster learning as measured by the number of training points necessary for accurate modeling. The FD-trained network quickly learned a representation of the desired function. The total rms estimate error for the function and its three derivatives approached its

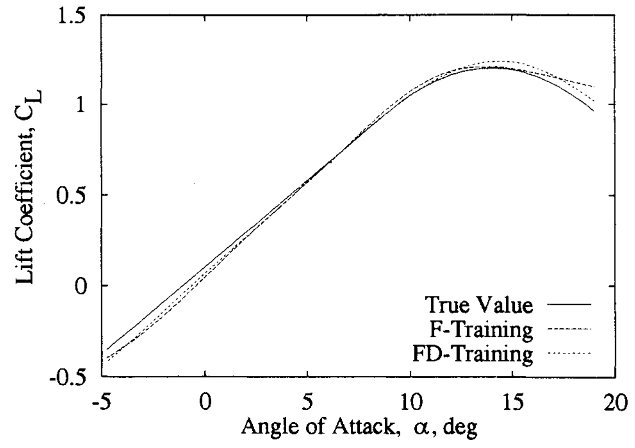

Fig. 3 Example network function comparison at $\hat{q} = 0$, $\delta_E = 0$.

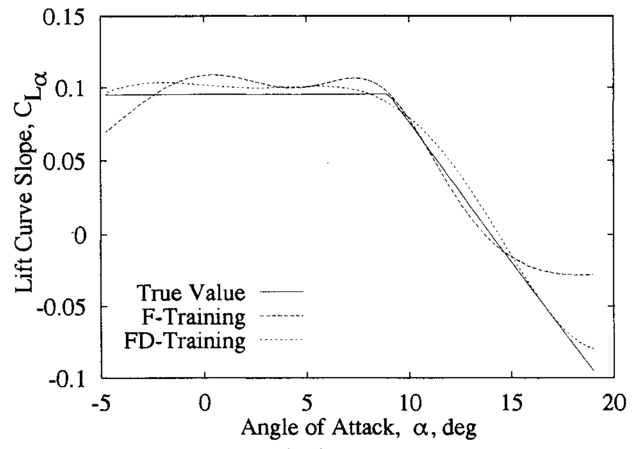

Fig. 4 Example network C_{L_α} derivative comparison at $\hat{q} = 0$, $\delta_E = 0$.

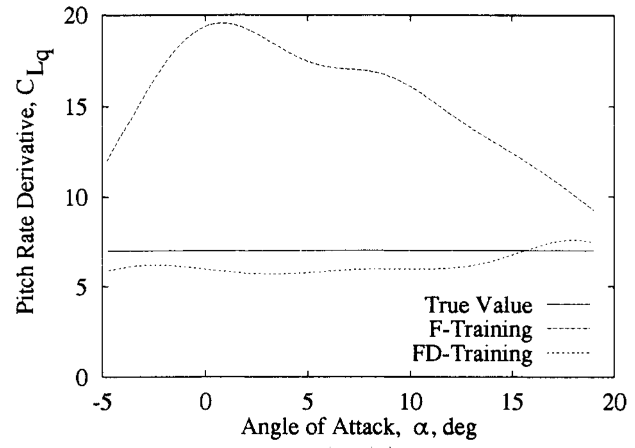

Fig. 5 Example network C_{L_q} derivative comparison at $\hat{q} = 0$, $\delta_E = 0$.

Table 1 Overall network error for simple training example

Training method	Hidden nodes	Total weights	rms error			
			C_L	C_{L_α}	C_{L_q}	$C_{L_{q_E}}$
F	3	16	0.0600	0.1955	0.0440	0.0221
F	5	26	0.0572	0.1532	0.0487	0.0282
F	10	51	0.0521	0.1540	0.0444	0.0291
FD	3	16	0.0360	0.1062	0.0054	0.0128
FD	5	26	0.0310	0.1021	0.0044	0.0106
FD	10	51	0.0366	0.0925	0.0050	0.0105

minimum value after the presentation of about 500 data points, and it remained relatively unchanged for the remaining points. The F-trained network required the full 2000 points to reach a level of accuracy less than that of the FD-trained network.

Estimating Aerodynamic Coefficient and Derivative Histories

The benefits of incorporating gradient information from the training function into neural network learning are well demonstrated in the preceding section. After a brief summary of the extended Kalman-Bucy filter equations, the EKBF that estimates the aircraft aerodynamic coefficient histories in addition to state histories is described. This filter then provides the data needed for on-line training of a feedforward neural network.

Extended Kalman-Bucy Filter for State and Force Estimation

The state and measurement equations for a combined continuous state/sampled-data measurement system are

$$\dot{x}(t) = f[x(t), n(t), t] \tag{32}$$

$$z_k = h[x(t_k), t_k] + v_k \tag{33}$$

where $x \in R^n$, $z, v \in R^m$, $n \in R^s$, and $f(\cdot)$ and $h(\cdot)$ are general nonlinear functions in R^n and R^m. The process noise $n(t)$ is a white, zero-mean Gaussian random process with spectral density matrix $Q_C(t)$. The measurement-noise sequence $\{v_k\}$ is a zero-mean, Gaussian random process with covariance R_k. The x_0 is a Gaussian random variable with mean $\bar{x}_0$ and covariance P_0.

For sufficiently smooth $f(\cdot)$ and $h(\cdot)$, a hybrid extended Kalman-Bucy filter can be defined.[10] The hybrid EKBF uses the continuous equations for the state and covariance propagation and the discrete equations for the gain and measurement update calculations. The continuous equations are

$$\hat{x}^{(-)}[t_k] = \hat{x}[t_{k-1}] + \int_{t_{k-1}}^{t_k} f[\hat{x}(\tau), 0, \tau]\, d\tau \tag{34}$$

$$P^{(-)}[t_k] = P[t_{k-1}] + \int_{t_{k-1}}^{t_k} \dot{P}(\tau)\, d\tau \tag{35}$$

where

$$\dot{P}(\tau) = F(\tau)P(\tau) + P(\tau)F^T(\tau) + L(\tau)Q_C(\tau)L^T(\tau) \tag{36}$$

The gain and update equations (16–18) are the same as the EKF with the definitions $P_k^{(-)} = P^{(-)}[t_k]$ and $\hat{x}_k^{(-)} = \hat{x}^{(-)}[t_k]$. The linearized matrices F, L, and H are

$$F(\tau) = \frac{\partial f(\cdot)}{\partial x} \tag{37}$$

$$L(\tau) = \frac{\partial f(\cdot)}{\partial n} \tag{38}$$

$$H_k = \frac{\partial h(\cdot)}{\partial x} \tag{39}$$

and each is evaluated at the current value of $\hat{x}$.

Estimating State and Force Histories

Following a procedure analogous to estimation before modeling,[4,5] the system is represented by

$$\dot{x}(t) = f[x(t)] + g[x(t), u(t)] + n(t) \tag{40}$$

$$z_k = h[x(t_k)] + v_k \tag{41}$$

in which $f(x)$ is known without error, and $g[x, u]$ is subject to uncertainty or change. Let $g(t) = g[x(t), u(t)]$ for some period of system operation. Subject to necessary observability requirements, an augmented state vector $x_a(t)$ consisting of

$$x_a(t) = \begin{bmatrix} x(t) \\ g(t) \end{bmatrix} \tag{42}$$

is estimated with an extended Kalman-Bucy filter. If Eq. (40) represents the six-degree-of-freedom equations of motion of a rigid aircraft (see Ref. 24), six elements of $g(t)$ are the specific forces and moments due to aerodynamic, thrust, and control inputs. The remaining elements of $g(t)$ are zero, as the kinematic relations are independent of the aerodynamic, thrust, and control effects.

In Refs. 4 and 5, the specific forces and moments $g(t)$ are modeled as second-order, integrated, random-walk processes, ensuring continuity and smoothness of the estimated histories. The equations for each $g_i(t)$ are

$$\begin{bmatrix} \dot{g}_i \\ \dot{g}_{i,1} \\ \dot{g}_{i,2} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} g_i \\ g_{i,1} \\ g_{i,2} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ n_i \end{bmatrix} \tag{43}$$

where n_i is a zero-mean, white Gaussian random process, and $g_{i,j}$ are internal state elements. Using this model of $g(t)$, the aircraft state dynamics are augmented with 18 (= six forces/moments × three equations) specific force and moment equations such that

$$\dot{x}_a(t) = f_a[x_a(t)] + n_a(t) \tag{44}$$

After estimating $\hat{x}_a$, the six forces and moments can be normalized using air density, airspeed, reference area, and reference length to the nondimensional coefficients: C_X, C_Y, C_Z, C_L, C_M, and C_N.

Aerodynamic Model Identification from Simulated Flight-Test Data

The aerodynamic coefficient identification scheme is demonstrated by generating a model of the normal force coefficient $C_Z(x, u)$ of a twin-jet transport aircraft based on simulated flight-test data. This demonstration exercises the plant, the state/force EKBF, and the aerodynamic model (in the form of a neural network) of Fig. 1. The results of training two identically initialized networks with the EKF F-training and FD-training methods are presented and compared. Three estimated derivative histories, $\hat{C}_{Z_\alpha}(t)$, $\hat{C}_{Z_q}(t)$, and $\hat{C}_{Z_{\delta_E}}(t)$, are included during FD training.

Simulated Flight-Test and Training Data Generation

Simulated flight-test data are generated from a full, nonlinear, six-degree-of-freedom model of a twin-jet transport. Standard rigid-body aircraft equations of motion are used.[24] The simulation contains an internal aerodynamic model based on extensive tabular data and algebraic constraints.[23] Five control inputs drive the system: throttle position δ_T, elevator deflection δ_E, aileron deflection δ_A, rudder deflection δ_R, and flap position δ_F.

Static training data are generated at 60 trim points that span the aircraft operational flight envelope (Fig. 6). These data consist of the aircraft state and all of the required aerodynamic coefficients and derivatives for the trimmed condition. Static data points are calculated exactly, as no estimation is required.

Dynamic training data are generated around nine additional trim points. These points are numbered in Fig. 6. The first six points outline the basic operating envelope, and the remaining three points are distributed through the interior. The dynamic data are produced using the nonlinear aircraft simulation and the state/force EKBF. At each point, the aircraft is excited by a specific input, measurements are recorded, and an EKBF is used to estimate the aircraft states and forces needed for network training.

To ensure the observability required by the Kalman-Bucy filter, an extensive, but feasible,[5] measurement vector is defined. The 13 measured variables are linear accelerations a_x, a_y, a_z; angular accelerations a_l, a_m, a_n; angular rates p, q, r; total velocity V; angle of attack α; angle of sideslip β; and altitude h. Process and measurement noise are modeled as zero-mean, Gaussian random sequences with covariances determined from Ref. 5.

Starting from a specified trim condition, the aircraft is excited using a $\pm$5-deg "3211" elevator input. The 3211 inputs consist of alternating steps of 3, 2, 1, and 1 time-unit widths. With appropriately chosen widths, this input history provides a sufficiently rich input for good estimation of aircraft parameters.[25] Measurement histories, 20 s long, capturing all of the relevant motion are stored for each dynamic training point.

Using the state/force EKBF, the aircraft state and force histories are estimated from the given measurement histories. Initial state, process-noise, and measurement-noise covariance matrices are based on Ref. 5. From the estimated force histories, the estimated normal force coefficient history $\hat{C}_Z(t)$ is calculated from the estimated density, airspeed, and the aircraft wing area. To demonstrate the anticipated benefits of the function-derivative training algorithm, $\hat{C}_{Z_\alpha}(t)$, $\hat{C}_{Z_q}(t)$, and $\hat{C}_{Z_{\delta_E}}(t)$ are calculated from the tabulated aerodynamic data using a finite-difference scheme and the currently estimated state $\hat{x}(t)$. A method to extract these derivative histories based on an extension of the state/force EKBF currently is under development.

Neural Network Model and Training Procedure

An eight-input, single-hidden-layer network structure with five logistic sigmoid nodes in the hidden layer and a single linear node in the output layer is chosen. The eight elements of the input vector are Mach number M, angle of attack α, pitch rate q, air density ρ, angle of attack rate $\dot{\alpha}$, throttle position δ_T, elevator deflection δ_E, and flap position δ_F; thus, $C_Z = C_Z(M, \alpha, q, \rho, \dot{\alpha}, \delta_T, \delta_E, \delta_F)$. Each input to the network is initially scaled to lie within $[-1, 1]$. The 8-5-1 network has 51 adjustable weights and biases. The network is initialized with random weights selected from a zero-mean, Gaussian distribution with a variance of 0.1.

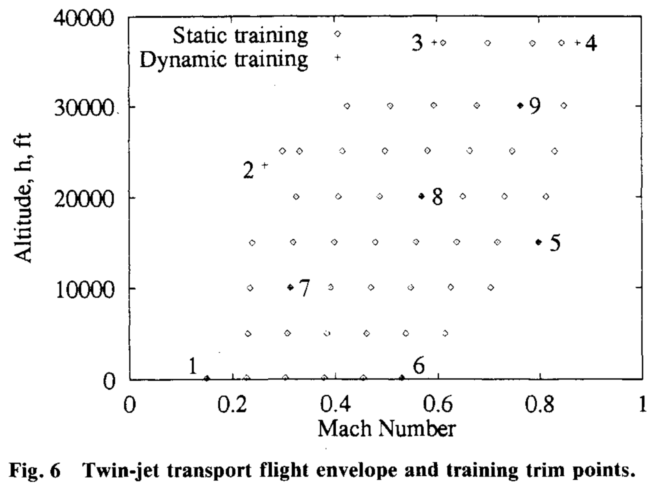

Fig. 6 Twin-jet transport flight envelope and training trim points.

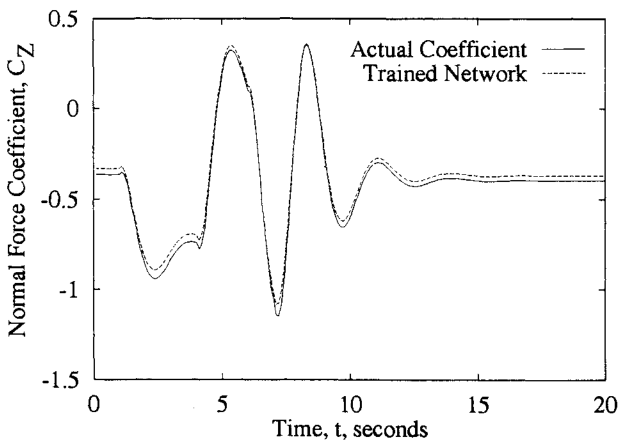

Fig. 7 Normal force coefficient history using function training, Mach 0.76, 30,000 ft.

The initial state and process-noise covariance matrices for the two training algorithms are identical. The initial state covariance P_0 is I, and the process-noise covariance matrix Q is $10^{-2}I$. The initial state covariance indicates that the initial weights are unknown. The small process-noise covariance allows the weights to converge but prevents the state covariance from converging to zero.

Training of the networks proceeds in an identical manner for each of the algorithms.

1) Iterate 50 times through each of the 60 static training points to initialize the network in the neighborhood of the desired function.

2) Present the first dynamic training history.

3) Present the 60 static training points.

4) Repeat steps 2 and 3 for the next dynamic training point.

5) Repeat steps 2–4 until the desired convergence is achieved.

For the following results, 12 iterations through the 9 dynamic training points were conducted. Since each history is 20 s long, a total of 2160 s (36 min) of simulated flight data (sampled at 0.1-s intervals) plus 9480 static training points were presented to each network.

Coefficient Identification

Using the procedure just described, a neural network model of $C_Z(x, u)$ is developed with the F-training method. The process-noise matrix R is 0.1 for this scalar example. After training, the network model matches the coefficient histories at each of the dynamic training points. The actual and network-estimated C_Z histories are given in Fig. 7 for dynamic training point 9. The histories are well matched with only a small bias. The other eight histories also are well matched. It must be emphasized that these promising results are obtained for large maneuvers (angles of attack ranging from -5 deg to well over the stall angle of attack of ≈ 16 deg) at nine training points covering the entire flight envelope from stall speeds at sea level (point 1) to Mach 0.85 at 37,000 ft (point 4).

For use in nonlinear control laws, the network estimate of $\hat{C}_Z(x,u)$ is more important than the estimate of $\hat{C}_Z(t)$. The network value of $C_Z(\alpha)$ is plotted in Fig. 8 based on dynamic training point 9. The intersection of the two curves occurs at the trim condition, but no where else does the model fit well. Equally poor results are found at all of the other training points. The network has apparently assigned false dependences of C_Z on other network inputs that are closely correlated with $\alpha(t)$ for the training maneuvers. Although generating excellent histories, the network has converged to an inadequate representation of the underlying aerodynamic coefficient model.

Coefficient and Derivative Identification

Using the procedure outlined earlier and an FD-learning algorithm, a second network is developed to model $C_Z(\boldsymbol{x}, \boldsymbol{u})$. In addition to $\hat{C}_Z(t)$, three partial derivative histories are used during training: $\hat{C}_{Z_\alpha}(t)$, $\hat{C}_{Z_q}(t)$, and $\hat{C}_{Z_{\delta_E}}(t)$. The process-noise covariance matrix $\boldsymbol{R}$ is diagonal with values (0.1, 2.5, 1.4×10^{-3}, 5.0×10^{-9}) along the diagonal. The elements of $\boldsymbol{R}$ are balanced to give approximately equal weight to relative errors in each coefficient and derivative. Alternatively, larger elements could be used in $\boldsymbol{R}$ to indicate that individual derivative histories are assumed to be noisier and less reliable for training. As the elements of $\boldsymbol{R}$ corresponding to the derivatives are increased, the results approach the F-trained network, and the derivative information is ignored.

$C_Z(t)$ and $C_Z(\alpha)$ plots for the fully trained network at dynamic training point 9 are given in Figs. 9 and 10. Both $C_Z(t)$ and $C_Z(\alpha)$ improved using FD training when compared to the plots for the F-trained network. The additional training information forces an excellent match of the slope in the training regions around the initial trim condition of 5-deg angle of attack. The $C_Z(\alpha)$ fit deteriorates in the region above approximately 15 deg, where the training set contained little data. This mismatch gives rise to the mismatched histories at about 7 s in Fig. 9, where the angle of attack is outside of the range (– 5 deg, 15 deg). Additional training data in the higher angle-of-attack regions could make the fit better in those regions, as might the changes to the size or structure of the network.

The performance of both networks for a novel input history is shown in Fig. 11. The trim point is approximately halfway between dynamic test points 8 and 9. The input history is a 2-s elevator doublet with an amplitude of 10 deg. Both networks follow the normal force coefficient history very accurately. Neither network entirely captures the high-angle-of-attack region between 3 and 4 s. The networks have developed reasonably accurate models of the underlying aerodynamic performance rather than just memorizing a particular response.

Discussion

Although present results are promising, the neural network aerodynamic model was trained and tested in a limited setting. Many problems relating to traditionally hard system identification questions remain to be addressed before final judgments can be made. For example, the inputs to the plant must be rich enough for the identifier to extract the best estimate of the true model from the data. In the preceding examples, the 3211 input provides good excitation of the two dominant longitudinal aircraft modes, but it probably is not rich enough to identify a complete aerodynamic model.

It clearly is important not only to span the space over which the aerodynamic model is to be defined but to minimize corre-

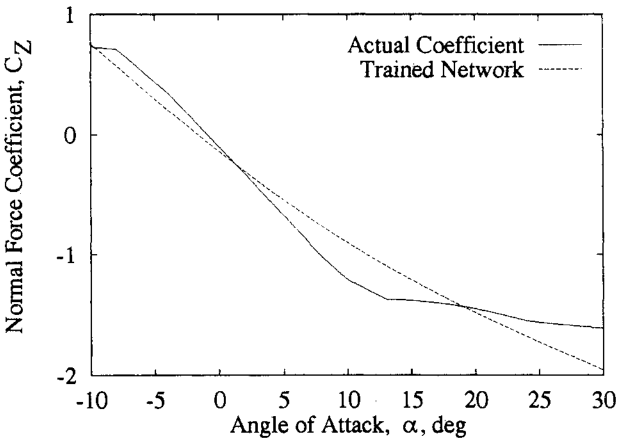

Fig. 8 Normal force curve using function training, Mach 0.76, 30,000 ft.

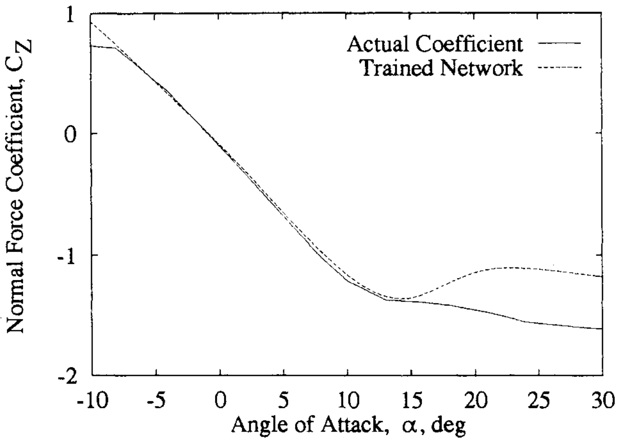

Fig. 10 Normal force curve using function-derivative training, Mach 0.76, 30,000 ft.

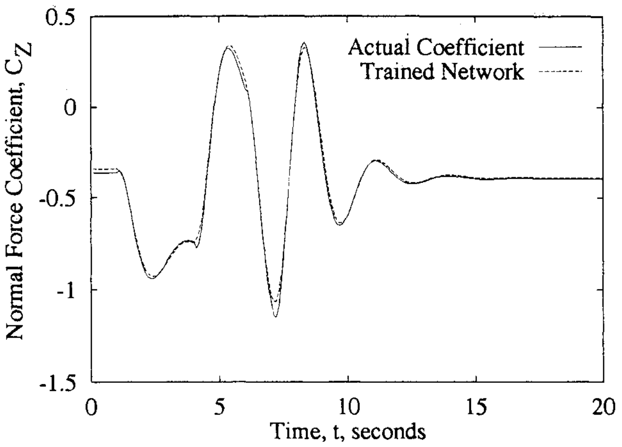

Fig. 9 Normal force coefficient history using function-derivative training, Mach 0.76, 30,000 ft.

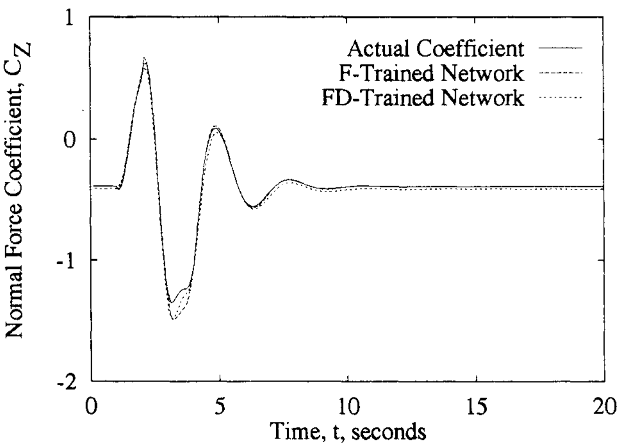

Fig. 11 Network performance for novel input data, Mach 0.66, 25,000 ft.

lations in the network input histories. Operating points and maneuvers must be chosen to promote orthogonality of the network inputs.

The FD-training algorithm requires histories of the force gradients to properly train the network. A state/force/force-gradient EKBF is being developed by the authors. At the very least, it should be possible to extract the dominant derivatives, such as C_{Z_α}, with sufficient accuracy to aid the neural network training process.

Conclusion

Accurate aerodynamic coefficient models are derived from simulated flight-test data using a system identification model composed of an extended Kalman-Bucy filter for state and force estimation and a computational neural network for aerodynamic modeling. An extended-Kalman-filter network-training algorithm based on function error alone is shown to produce an excellent force-history match, although the functional dependence of force on specific inputs is not well identified. Augmenting the network training with information about derivatives of the function with respect to its inputs improves the force-history fit and especially the first partial derivative functional fit. Networks are well trained using dynamic input data generated from simulated flight tests. Dynamic maneuvers that span the input space and minimize correlations of the inputs must be defined for effective aerodynamic modeling using computational neural networks.

Acknowledgments

This research has been sponsored by the Federal Aviation Administration and NASA Langley Research Center under Grant NGL 31-001-252 and by the Army Research Office under Grant DAAL03-89-K-0092.

References

[1]Isidori, A., *Nonlinear Control Systems: An Introduction,* Springer-Verlag, Berlin, 1989, Chap. 7.

[2]Lane, S. H., and Stengel, R. F., "Flight Control Design Using Nonlinear Inverse Dynamics," *Automatica,* Vol. 24, No. 4, 1988, pp. 471–484.

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

[4]Stalford, H. L., "High-Alpha Aerodynamic Model Identification of T-2C Aircraft Using the EBM Method," *Journal of Aircraft,* Vol. 18, No. 10, 1981, pp. 801–809.

[5]Sri-Jayantha, M., and Stengel, R. F., "Determination of Nonlinear Aerodynamic Coefficients Using the Estimation-Before-Modeling Method," *Journal of Aircraft,* Vol. 25, No. 9, 1988, pp.796–804.

[6]Iliff, K. W., "Parameter Estimation for Flight Vehicles," *Journal of Guidance, Control, and Dynamics,* Vol. 12, No. 5, 1989, pp. 609–622.

[7]Jategaonkar, R. V., and Plaetschke, E., "Identification of Moderately Nonlinear Flight Mechanics Systems with Additive Process and Measurement Noise," *Journal of Guidance, Control, and Dynamics,* Vol. 13, No. 2, 1990, pp. 277–285.

[8]Stengel, R. F., and Linse, D. J., "System Identification for Nonlinear Control Using Neural Networks," *Proceedings of the 1990 Conference on Information Sciences and Systems,* Vol. 2, Princeton Univ., Princeton, NJ, March 1990, pp. 747–752.

[9]Linse, D. J., and Stengel, R. F., "A System Identification Model for Adaptive Nonlinear Control," *Proceedings of the 1991 American Control Conference* (Boston, MA), American Automatic Control Council, Evanston, IL, June 1991, pp. 1752–1757.

[10]Stengel, R. F., *Stochastic Optimal Control: Theory and Application,* Wiley, New York, 1986, Chap. 4.

[11]Poggio, T., and Girosi, F., "Regularization Algorithms for Learning That Are Equivalent to Multilayer Networks," *Science,* Vol. 247, No. 4945, 1990, pp. 978–982.

[12]Cybenko, G., "Approximation by Superpositions of a Sigmoidal Function," *Mathematics of Control, Signals, and Systems,* Vol. 2, No. 4, 1989, pp. 303–314.

[13]Hornik, K., Stinchcombe, M., and White, H., "Universal Approximation of an Unknown Mapping and Its Derivatives Using Multilayer Feedforward Networks," *Neural Networks,* Vol. 3, No. 5, 1990, pp. 551–560.

[14]Dreyfus, S. E., "Artificial Neural Networks, Back Propagation, and the Kelley-Bryson Gradient Procedure," *Journal of Guidance, Control, and Dynamics,* Vol. 13, No. 5, 1990, pp. 926–928.

[15]Rumelhart, D. E., Hinton, G. E., and Williams, R. J.,"Learning Internal Representations by Error Propagation," *Parallel Distributed Processing: Explorations in the Microstructure of Cognition,* edited by D. E. Rumelhart and J. L. McClelland, MIT Press, Cambridge, MA, 1986, Chap. 8.

[16]Battiti, R., "Accelerated Backpropagation Learning: Two Optimization Methods," *Complex Systems,* Vol. 3, No. 4, 1989, pp. 331–342.

[17]Kollias, S., and Anastassiou, D., "Adaptive Training of Multilayer Neural Networks Using a Least Squares Estimation Technique," *IEEE Transactions on Circuits and Systems,* Vol. 36, No. 8, 1989, pp. 1092–1101.

[18]Leonard, J., and Kramer, M. A., "Improvement of the Backpropagation Algorithm for Training Neural Networks," *Computers and Chemical Engineering,* Vol. 14, No. 3, 1990, pp. 337–341.

[19]Singhal, S., and Wu, L., "Training Feed-forward Networks with the Extended Kalman Algorithm," *Proceedings of the 1989 International Conference on ASSP* (Glasgow, Scotland), IEEE, New York, May 1989, pp. 1187–1190.

[20]Shah, S., and Palmieri, F., "MEKA—A Fast, Local Algorithm for Training Feedforward Neural Networks," *1990 International Joint Conference on Neural Networks* (San Diego, CA), Vol. 3, IEEE Neural Network Council, New York, 1990, pp. 41–46.

[21]Anderson, B. D. O., and Moore, J. B., *Optimal Filtering,* Prentice-Hall, Englewood Cliffs, NJ, 1979, Chap. 5.

[22]Schweppe, F. C., *Uncertain Dynamic Systems,* Prentice-Hall, Englewood Cliffs, NJ, 1973, Chap. 4.

[23]Anon., "TCV/User Oriented FORTRAN Program for the B737 Six DOF Dynamic Model," Sperry Systems Management, Langley Operations, SP-710-021, Hampton, VA, March 1981.

[24]McRuer, D., Ashkenas, I., and Graham, D., *Aircraft Dynamics and Automatic Control,* Princeton Univ. Press, Princeton, NJ, 1973, Chap. 4.

[25]Plaetschke, E., Mulder, J. A., and Breeman, J. H., "Flight Test Results of Five Input Signals for Aircraft Parameter Identification," *Proceedings of the Sixth IFAC Symposium on Identification and System Parameter Estimation* (Washington, DC), Vol. 2, Pergamon, New York, June 1982, pp. 1149–1154.