MASTER COPY KEEP THIS COPY FOR REPRODUCTION PURPOSES

AD-A244 711

ATION PAGE

Form Approved
OMB No. 0704-0188

DATE | 3. REPORT TYPE AND DATES COVERED: Reprint

2

4. TITLE AND SUBTITLE

Title shown on Reprint

5. FUNDING NUMBERS

DAAL03-89-K-0092

6. AUTHOR(S)

Author(s) listed on Reprint

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Princeton Univ.
Princeton, NJ 08544

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

U. S. Army Research Office
P. O. Box 12211
Research Triangle Park, NC 27709-2211

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

ARO 25264.13-MA

11. SUPPLEMENTARY NOTES

The view, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

ABSTRACT ON REPRINT

14. SUBJECT TERMS

15. NUMBER OF PAGES

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

AIAA

AIAA 92-0172
Identification of Aerodynamic Coefficients Using Computational Neural Networks

D.J. Linse and R.F. Stengel
Department of Mechanical
and Aerospace Engineering
Princeton University
Princeton, NJ

30th Aerospace Sciences
Meeting & Exhibit
January 6-9, 1992 / Reno, NV

IDENTIFICATION OF AERODYNAMIC COEFFICIENTS USING COMPUTATIONAL NEURAL NETWORKS

Dennis J. Linse* and Robert F. Stengel†
Department of Mechanical and Aerospace Engineering
Princeton University, Princeton, NJ 08544

Abstract

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,

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

$$\mathbf{y} = \mathbf{h}(\mathbf{x}) \quad (2)$$

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 $\mathbf{y}$ or its derivatives. The global representation of the aerodynamic model contained in $\mathbf{f}(\cdot)$, $\mathbf{g}(\cdot)$, and $\mathbf{h}(\cdot)$ 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. While the partitions span the space, these global models are, in general, 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.

Non-smooth aerodynamic models can be coerced into smooth models with sufficient post-processing, but this post-processing may reduce the fidelity of the model. In [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 [8,9]. Using the plant measurements, $\mathbf{z}$, an extended Kalman-Bucy filter (EKBF) [10] estimates the state of an augmented plant model consisting of the standard aircraft state, $\hat{\mathbf{x}}(t)$, and the aircraft specific forces and moments, $\hat{\mathbf{g}}(t)$. The neural network learns an aerodynamic model, $\hat{\mathbf{g}}(\hat{\mathbf{x}}, \mathbf{u})$, from $\hat{\mathbf{x}}(t)$, $\hat{\mathbf{g}}(t)$, and the control, $\mathbf{u}(t)$. A nonlinear control law using the estimated state, $\hat{\mathbf{x}}(t)$, and the aerodynamic model, $\hat{\mathbf{g}}(\hat{\mathbf{x}}, \mathbf{u})$, 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 net-

*Graduate Research Assistant, Member AIAA.
†Professor, Associate Fellow AIAA.

Presented at the 1992 Aerospace Sciences Meeting, Reno, NV, January 1992.

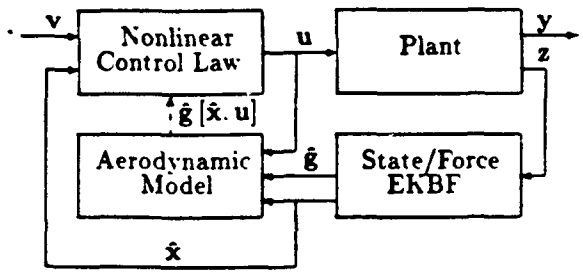

Figure 1: Integration of System Identification and Nonlinear Control.

work model for single flight conditions, and the corresponding functional relationship $\hat{\mathbf{g}}(\hat{\mathbf{x}}, \mathbf{u})$ 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, $\hat{\mathbf{g}}(\hat{\mathbf{x}}, \mathbf{u})$ 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. Considerable gain is achieved in network accuracy with these methods, and representation of aerodynamic derivatives is improved substantially.

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. $\mathbf{r}^{(k)}$, the output of the k^{th} layer of the network, is computed as the nonlinear transformation of the weighted sum of the outputs of the $(k-1)^{st}$ layer and a bias:

$$\mathbf{r}^{(k)} = \sigma^{(k)}\left[\mathbf{W}^{(k-1)}\mathbf{r}^{(k-1)} + \mathbf{b}^{(k-1)}\right] \qquad (3)$$

$\mathbf{W}$ and $\mathbf{b}$ are the weights and biases of the indicated layers, and $\sigma[\cdot]$ is a vector-valued function composed of individual node activation functions:

$$\sigma[\mathbf{q}] = [\sigma_1(q_1)\cdots\sigma_n(q_n)]^T \qquad (4)$$

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

$$\begin{aligned}\mathbf{r}^{(N_L)} &= \sigma^{(N_L)}(\mathbf{W}^{(N_L-1)}\sigma^{(N_L-1)}(\cdots \\ &\quad \sigma^{(1)}(\mathbf{W}^{(0)}\mathbf{r}^{(0)} + \mathbf{b}^{(0)}) + \cdots)) \qquad (5)\end{aligned}$$

Defining $\mathbf{r} = \mathbf{r}^{(0)}$ as the input vector and $\mathbf{z} = \mathbf{r}^{(N_L)}$ as the output vector,

$$\mathbf{z} = \mathbf{h}(\mathbf{r}; \mathbf{w}) \qquad (6)$$

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

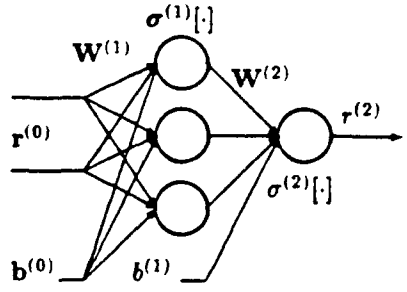

Figure 2: A simple feedforward network.

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. 6, by its construction, is the same as that of the node activation functions, $\sigma[\cdot]$. Assuming all activation functions are at least once differentiable, the derivative of the network with respect to its inputs is

$$\frac{\partial \mathbf{h}}{\partial \mathbf{r}} = \frac{\partial \sigma^{(N_L)}}{\partial \mathbf{q}^{(N_L)}}\mathbf{W}^{(N_L-1)}\cdots\frac{\partial \sigma^{(1)}}{\partial \mathbf{q}^{(1)}}\mathbf{W}^{(0)} \qquad (7)$$

The derivatives of the activation function vectors are diagonal matrices:

$$\frac{\partial \sigma^{(i)}}{\partial \mathbf{q}^{(i)}} = \begin{bmatrix} \sigma_1' & & 0 \\ & \ddots & \\ 0 & & \sigma_n' \end{bmatrix} \qquad (8)$$

The *logistic sigmoid* is a commonly used node activation function:

$$\sigma(q) = \frac{1}{1+e^{-q}} \qquad (9)$$

It is infinitely differentiable, and its first derivative is quite simply calculated as

$$\frac{\partial \sigma}{\partial q} = \sigma' = \sigma(q)[1 - \sigma(q)] \tag{10}$$

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 n^{th}-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.

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

$$J = \sum_{i=1}^{N_p} \boldsymbol{\epsilon}_i{}^{\mathrm{T}} \boldsymbol{\epsilon}_i \tag{11}$$

where the network error vector for the i^{th} trial, $\boldsymbol{\epsilon}_i$, is

$$\boldsymbol{\epsilon}_i = \mathbf{d}_i - \mathbf{h}(\mathbf{r}_i, \mathbf{w}) \tag{12}$$

After selecting the number of layers and the number of nodes in each layer, J is minimized with respect to $\mathbf{w}$, giving the minimum mean-squared-error network.

Searching for the weight vector that minimizes eq. 11 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 only using information available 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, conjugate gradients, and various Newton-type algorithms, have been applied to feedforward computational neural networks [14–17]. Back-propagation, a steepest-descent algorithm, is the most common algorithm [14]. Modifications and improvements to the back-propagation algorithm, such as learning momentum, have been developed [14–16].

Conjugate gradient methods have been successfully applied to neural networks (see, e.g., [15] for a description of the method). A localized version of the Marquardt-Levenberg least-squares optimization technique is developed in [17]. The Marquardt-Levenberg algorithm progresses from a steepest-descent algorithm to a quasi-Newton algorithm as optimization proceeds. Limited testing of the basic Marquardt-Levenberg algorithm for a simple network showed no significant benefit in the present application.

The extended Kalman filter is a particularly attractive alternative method that transforms the optimization of eq. 11 and training of eq. 6 into the estimation of a dynamic system [18]. A localized version of the extended Kalman filter algorithm also has been developed [19]. Our version of the extended Kalman filter follows the first approach, as described below.

Extended Kalman Filter for Network Training

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

$$\mathbf{x}_{k+1} = \boldsymbol{\phi}(\mathbf{x}_k) + \mathbf{n}_k \tag{13}$$

$$\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k) + \mathbf{v}_k \tag{14}$$

where $\mathbf{x}, \mathbf{n} \in R^n$, $\mathbf{z}, \mathbf{v} \in R^m$, and $\boldsymbol{\phi}(\cdot)$ and $\mathbf{h}(\cdot)$ are general nonlinear functions in R^n and R^m. $\{\mathbf{n}_k\}$ and $\{\mathbf{v}_k\}$ are zero mean, white Gaussian processes with covariances $\mathbf{Q}_k$ and $\mathbf{R}_k$. $\mathbf{x}_0$ is a Gaussian random variable with mean, $\bar{\mathbf{x}}_0$, and covariance, $\mathbf{P}_0$.

Assuming $\boldsymbol{\phi}(\cdot)$ and $\mathbf{h}(\cdot)$ are sufficiently smooth, an extended Kalman filter can be defined for the system, eq. 13 and 14 (see, e.g., [10,20]). The filter equations are

$$\hat{\mathbf{x}}_k^{(-)} = \mathbf{f}(\hat{\mathbf{x}}_{k-1}) \tag{15}$$

$$\mathbf{P}_k^{(-)} = \boldsymbol{\Phi}_{k-1}\mathbf{P}_{k-1}\boldsymbol{\Phi}_{k-1}^{\mathrm{T}} + \mathbf{Q}_k \tag{16}$$

$$\mathbf{K}_k = \mathbf{P}_k^{(-)}\mathbf{H}_k^{\mathrm{T}}\left(\mathbf{H}_k\mathbf{P}_k^{(-)}\mathbf{H}_k^{\mathrm{T}} + \mathbf{R}_k\right)^{-1} \tag{17}$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^{(-)} + \mathbf{K}_k\left\{\mathbf{z}_k - \mathbf{h}[\hat{\mathbf{x}}_k^{(-)}]\right\} \tag{18}$$

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_k^{(-)}(\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)^{\mathrm{T}} + \mathbf{K}_k\mathbf{R}_k\mathbf{K}_k^{\mathrm{T}} \tag{19}$$

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

$$\boldsymbol{\Phi}_k = \left.\frac{\partial \boldsymbol{\phi}}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_k} \tag{20}$$

$$\mathbf{H}_k = \left.\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_k^{(-)}} \tag{21}$$

A symmetric form is used for the covariance update (eq. 19) ensuring that $\mathbf{P}_k$ remains symmetric and positive definite as long as $\mathbf{P}_k^{(-)}$ is positive-definite and $\mathbf{R}_k$ 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. 15), while disturbances and measurement errors are small enough to allow linear estimates of covariance evolution and measurement up-

date (eq. 16-19). The 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 $\mathbf{w}$ as the state vector $\mathbf{x}$ in eq. 13, a neural network can be expressed as a nonlinear dynamic system:

$$\mathbf{w}_k = \mathbf{w}_{k-1} + \mathbf{n}_{k-1} \quad (22)$$

$$z_k = h(\mathbf{w}_k, \mathbf{r}_k) + v_k \quad (23)$$

With the process and measurement noise sequences, $\{\mathbf{n}_k\}$ and $\{v_k\}$, as defined above, eq. 22 and 23 have the form of eq. 13 and 14, where $\phi(\mathbf{x}) = \mathbf{x}$ and h is defined by the network forward propagation (eq. 6). 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, $\{\mathbf{r}_k\}$, the weight vector, $\mathbf{w}$, is estimated by an extended Kalman filter. For eq. 22, the state Jacobian matrix, $\mathbf{F}_k$, is an identity matrix for all k. This makes the covariance propagation step, eq. 16, very simple. Computation of the measurement Jacobian matrix,

$$\mathbf{H}_k = \left.\frac{\partial h}{\partial \mathbf{w}}\right|_{(\mathbf{w}_k, \mathbf{r}_k)} \quad (24)$$

is a straightforward application of the chain rule, given the differentiability of the network nonlinearities and the form of eq. 6. $\mathbf{H}_k$ has dimension $1 \times N_w$, where N_w is the number of weights in the network.

The propagation of $\mathbf{w}_k$ (eq. 22) is modeled as a dynamic system driven by a white Gaussian process although $\mathbf{w}$ is an internal variable that is not subject to noise. The covariance, $\mathbf{Q}$, of the pseudo-noise sequence, $\{\mathbf{n}_k\}$, controls the convergence of the state covariance, $\mathbf{P}_k$. In an undriven system ($\mathbf{Q} = 0$), the covariance matrix converges to zero [21]. This prevents further state updating since, by eq. 17, $\mathbf{K}_k$ is zero if $\mathbf{P}_k = 0$.

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 9 inputs, 20 hidden nodes, and one output, resulting in 221 weights and biases. Ordinarily, a Kalman filter of this size would be difficult to compute; however, the matrix to be inverted in eq. 17 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 (eq. 12) is scalar and the cost (eq. 11) 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 $\{\mathbf{r}_i, d_i\}$, measurements of the first partial derivatives of $d(\cdot)$ evaluated at $\mathbf{r}_i$,

$$\left.\frac{\partial d}{\partial \mathbf{r}}\right|_{\mathbf{r}_i} \quad (25)$$

are available, a new weighted least-squares cost function can be defined:

$$J = \sum_{i=1}^{N_p} \boldsymbol{\epsilon}_i^T \mathbf{R}^{-1} \boldsymbol{\epsilon}_i \quad (26)$$

where $\mathbf{R}^{-1}$ is an appropriately dimensioned weighting matrix. $\boldsymbol{\epsilon}_i$ is formed by augmenting the original scalar error, ϵ_i, such that

$$\boldsymbol{\epsilon}_i = \begin{bmatrix} d_i - h(\mathbf{w}, \mathbf{r}_i) \\ \frac{\partial d}{\partial \mathbf{r}}(\mathbf{r}_i) - \frac{\partial h}{\partial \mathbf{r}}(\mathbf{w}, \mathbf{r}_i) \end{bmatrix} \quad (27)$$

In eq. 26, $\mathbf{R}^{-1}$ weights the relative importance of each element of $\boldsymbol{\epsilon}_i$, 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 previously mentioned network training algorithms could be reformulated based on the function-derivative cost, the EKF training method is extended here. The propagation of $\mathbf{w}_k$ (eq. 22) remains the same, while the measurement equation (eq. 23) is expanded to include the partial derivatives:

$$\mathbf{z}_k = \begin{bmatrix} h(\mathbf{w}_k, \mathbf{r}_k) \\ \frac{\partial h}{\partial \mathbf{r}}(\mathbf{w}_k, \mathbf{r}_k) \end{bmatrix} + \mathbf{v}_k \quad (28)$$

$\partial h / \partial \mathbf{r}$ is calculated using eq. 7. With this measurement equation, the measurement Jacobian matrix becomes

$$\mathbf{H}_k = \left.\begin{bmatrix} \frac{\partial h}{\partial \mathbf{w}} \\ \frac{\partial^2 h}{\partial \mathbf{w} \partial \mathbf{r}} \end{bmatrix}\right|_{(\mathbf{w}_k, \mathbf{r}_k)} \quad (29)$$

This expanded matrix $\mathbf{H}_k$ has dimension $(N_i+1) \times N_w$, where N_i is the number of network inputs.

In the EKF formulation, the covariance, $\mathbf{R}$, of the measurement noise sequence, $\mathbf{v}_k$, is the same as $\mathbf{R}$ in the weighted least-squares cost function (eq. 26). It adjusts the relative importance of each of the measurements during network training. Equivalently, it weights the relative accuracy of each of the measurements. While containing more information about the function, the partial derivatives may also contain more errors. Thus, a suitable $\mathbf{R}$ is important for accurate network learning.

Training Example

A simple example demonstrates the added benefits of function-derivative training (FD-training) 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

$$C_L(\alpha, q, \delta_E) = C_{L_{ST}}(\alpha) + C_{L_q}\hat{q} + C_{L_{\delta_E}}\delta_E \qquad (30)$$

where α is the angle of attack, $\hat{q} = q\bar{c}/2V$ is the non-dimensionalized pitch rate, and δ_E is the elevator deflection angle. Realistic numerical values are chosen for the aerodynamic derivatives based on a twin-jet transport aircraft [22]. 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

$$C_{L_{ST}}(\alpha) = \begin{cases} 0.0952\alpha + 0.1048 & \alpha < 9^\circ \\ -0.0095\alpha^2 + 0.2667\alpha - 0.6667 & \alpha \geq 9^\circ \end{cases}$$

The restriction of C_L to α and δ_E inputs is plotted in Fig. 3.

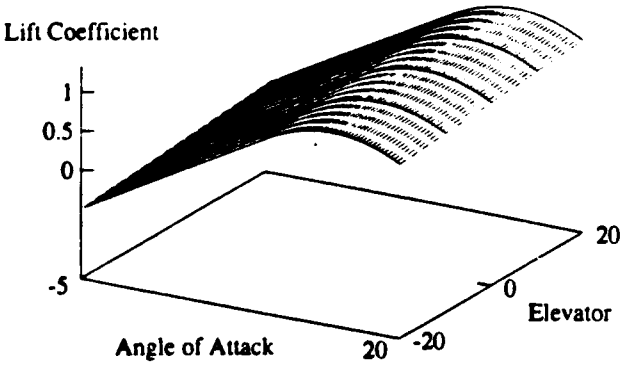

Figure 3: Example Lift Coefficient, C_L, for Fixed $\hat{q}$.

Using eq. 30 as the desired function, d, two identical feedforward networks with three inputs, one hidden layer of 10 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 1500 points from a uniform distribution bounded by

$$\alpha \in [-5^\circ, 19^\circ] \qquad (31)$$

$$\hat{q} \in [-0.005, 0.005] \qquad (32)$$

$$\delta_E \in [-20^\circ, 20^\circ] \qquad (33)$$

Equation 32 represents q variations of $\pm 20^\circ$/sec at the given velocity, V. The lift coefficient, C_L, is calculated from eq. 30 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 covariances, 0.5, 6.0×10^{-4}, and 0.7. The initial state covariance, $\mathbf{P}_0$, is $10^4\mathbf{I}$ and the constant process noise covariance, $\mathbf{Q}$, is $10^{-3}\mathbf{I}$, where $\mathbf{I}$ 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, $\mathbf{R} = 0.1$, while for FD-training

$$\mathbf{R} = \begin{bmatrix} 0.1 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 \\ 0 & 0 & 6.0 \times 10^{-4} & 0 \\ 0 & 0 & 0 & 0.7 \end{bmatrix} \qquad (34)$$

A comparison of the variation of eq. 30 and its three partial derivatives to the two network approximations of this function is given in Fig. 4 for angle-of-attack variations. These plots represent one-dimensional slices through the fully trained three-dimensional input space. The FD-training is better for the function, and it is much better for the derivatives. The rms errors of the function and derivatives for the FD-trained function are all considerably smaller than the F-trained function.

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 error between the function and its three derivatives and the network model approached its minimum value after the presentation of about 500 data points and remained relatively unchanged for the remaining points. The F-trained network required the full 1500 points to reach a level of accuracy less than that of the FD-trained network.

Including partial derivatives in the network training process substantially improves the ability of the training process to find an accurate model of the desired function and its derivatives.

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 previous section. After a brief summary of the extended Kalman-Bucy filter equations, the EKBF that estimates the aircraft

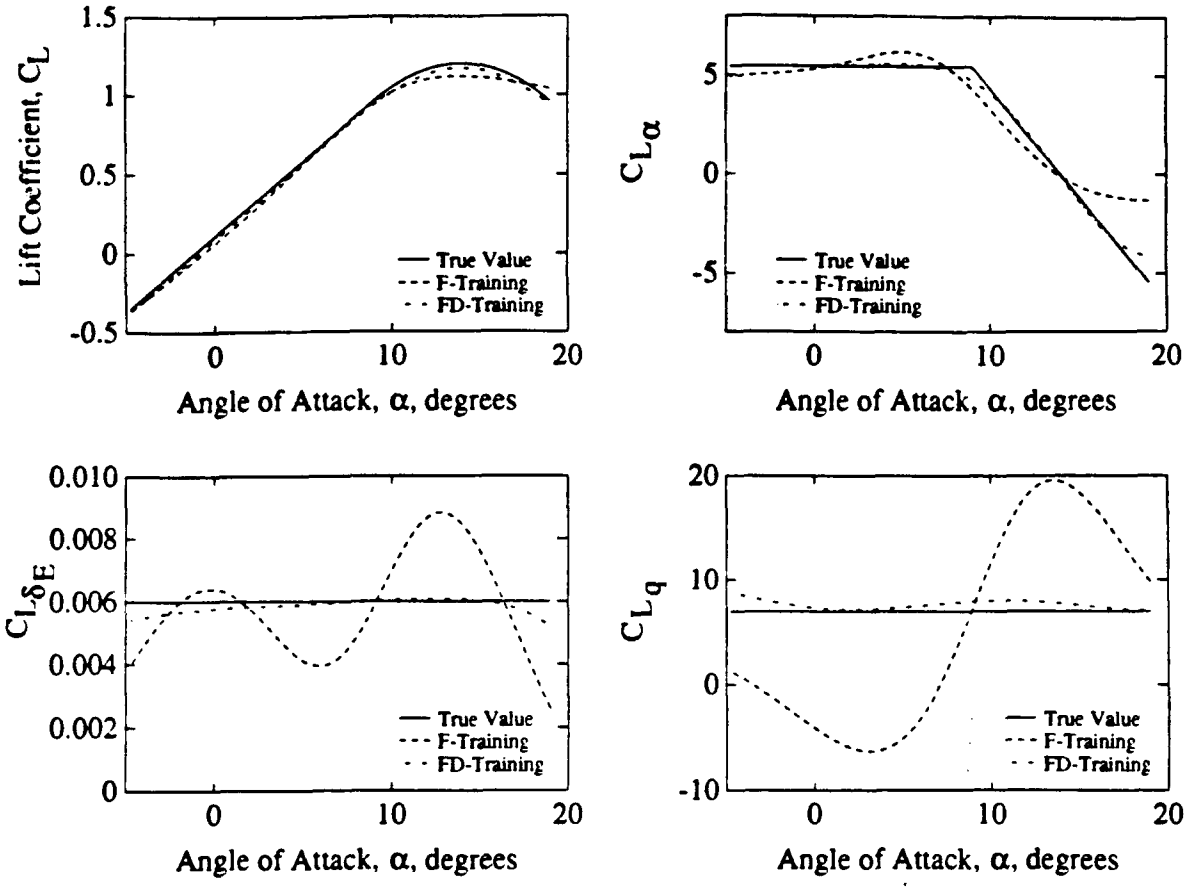

Figure 4: Network Function and Derivative Comparison at $\hat{q} = 0$, $\delta_E = 0$.

aerodynamic coefficient histories in addition to state histories is described. Subsequently, an extension of the state/force EKBF is developed that estimates the desired partial derivative histories in addition to the coefficient histories.

Extended Kalman-Bucy Filter for State and Force Estimation

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

$$\dot{\mathbf{x}}(t) = \mathbf{f}[\mathbf{x}(t), \mathbf{n}(t), t] \quad (35)$$

$$\mathbf{z}_k = \mathbf{h}[\mathbf{x}(t_k), t_k] + \mathbf{v}_k \quad (36)$$

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

For sufficiently smooth $\mathbf{f}(\cdot)$ and $\mathbf{h}(\cdot)$, 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

$$\hat{\mathbf{x}}^{(-)}[t_k] = \hat{\mathbf{x}}[t_{k-1}] + \int_{t_{k-1}}^{t_k} \mathbf{f}[\hat{\mathbf{x}}(\tau), 0, \tau]\, d\tau \quad (37)$$

$$\mathbf{P}^{(-)}[t_k] = \mathbf{P}[t_{k-1}] + \int_{t_{k-1}}^{t_k} \dot{\mathbf{P}}(\tau)\, d\tau \quad (38)$$

where

$$\dot{\mathbf{P}}(\tau) = \mathbf{F}(\tau)\mathbf{P}(\tau) + \mathbf{P}(\tau)\mathbf{F}^{\mathrm{T}}(\tau) + \mathbf{L}(\tau)\mathbf{Q}_C(\tau)\mathbf{L}^{\mathrm{T}}(\tau) \quad (39)$$

The gain and update equations (eq. 17-19) are the same as the EKF with the definitions $\mathbf{P}_k^{(-)} = \mathbf{P}^{(-)}[t_k]$ and $\hat{\mathbf{x}}_k^{(-)} = \hat{\mathbf{x}}^{(-)}[t_k]$. The linearized matrices $\mathbf{F}$, $\mathbf{L}$, and $\mathbf{H}$ are

$$\mathbf{F}(t) = \frac{\partial \mathbf{f}(\cdot)}{\partial \mathbf{x}} \quad (40)$$

$$\mathbf{L}(t) = \frac{\partial \mathbf{f}(\cdot)}{\partial \mathbf{n}} \quad (41)$$

$$\mathbf{H}_k = \frac{\partial \mathbf{h}(\cdot)}{\partial \mathbf{x}} \quad (42)$$

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

Estimating State and Force Histories

Following a procedure analogous to *Estimation Before Modeling* [4,5], the system is represented by

$$\dot{\mathbf{x}}(t) = \mathbf{f}[\mathbf{x}(t)] + \mathbf{g}[\mathbf{x}(t), \mathbf{u}(t)] + \mathbf{n}(t) \quad (43)$$

$$\mathbf{z}_k = \mathbf{h}[\mathbf{x}(t_k)] + \mathbf{v}_k \quad (44)$$

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

$$\mathbf{x}_a(t) = \begin{bmatrix} \mathbf{x}(t) \\ \mathbf{g}(t) \end{bmatrix} \quad (45)$$

is estimated with an extended Kalman-Bucy filter. If eq. 43 represents the 6-degree-of-freedom equations of motion of a rigid aircraft (see [23]), 6 elements of $\mathbf{g}(t)$ are the specific forces and moments due to aerodynamic, thrust, and control inputs. The remaining elements of $\mathbf{g}(t)$ are zero, as the kinematic relations are independent of the aerodynamic, thrust, and control effects.

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

$$\begin{bmatrix} \dot{g}_i \\ \dot{g}_{i,1} \\ \dot{g}_{i,2} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} g_i \\ g_{i,1} \\ g_{i,2} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ n_i \end{bmatrix} \quad (46)$$

where n_i are zero-mean, white Gaussian random processes and $g_{i,j}$ are internal state elements. Using this model of $\mathbf{g}(t)$, the aircraft state dynamics are augmented with 18 (= 6 forces/moments × 3 equations) specific force and moment equations such that

$$\dot{\mathbf{x}}_a(t) = \mathbf{f}_a[\mathbf{x}_a(t)] + \mathbf{n}_a(t) \quad (47)$$

After estimating $\hat{\mathbf{x}}_a$, the 6 forces and moments can be normalized using air density, airspeed, reference area, and reference length to the non-dimensional coefficients: C_X, C_Y, C_Z, C_L, C_M, and C_N.

Estimating State, Force, and Force-Gradient Histories

Using the state/force estimator and a feedforward neural network, excellent representations of normal force coefficient (C_Z) histories were demonstrated in [9]. Deficiencies in the representation of $C_Z(\mathbf{x}, \mathbf{u})$ and the corresponding aircraft stability and control derivatives prompted further research into the network training process and resulted in the FD-training methodology. To provide the needed gradient histories, the state/force estimator must be augmented.

Assuming the same state and measurement equations, eq. 43 and 44, first partial derivatives of the forces and moments are estimated by retaining some state explicit dependence in $\mathbf{g}(\cdot)$. As an example, $\partial \mathbf{g}/\partial x_1$ is estimated in the following manner. Let $\mathbf{g}[x_1(t), t)] = \mathbf{g}[\mathbf{x}(t), \mathbf{u}(t)]$ for some period of system operation. With the dependence on $x_1(t)$ remaining, the derivative of each g_i is

$$\dot{g}_i = \frac{\partial g_i}{\partial t} + \frac{\partial g_i}{\partial x_1}\frac{\partial x_1}{\partial t} = g_{i,01} + g_{i,11}\dot{x}_1 \quad (48)$$

where $\dot{x}_1$ is obtained from eq. 43, and $g_{i,01}$ and $g_{i,11}$ are the unknown partial derivatives. Following the previous development, each of the unknown partial derivatives is modeled as first-order, integrated, random-walk processes. For each element of $\mathbf{g}(t)$,

$$\begin{bmatrix} \dot{g}_i \\ \dot{g}_{i,01} \\ \dot{g}_{i,11} \\ \dot{g}_{i,02} \\ \dot{g}_{i,12} \end{bmatrix} = \begin{bmatrix} 0 & 1 & \dot{x}_1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} g_i \\ g_{i,01} \\ g_{i,11} \\ g_{i,02} \\ g_{i,12} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ n_{i,0} \\ n_{i,1} \end{bmatrix} \quad (49)$$

where $n_{i,0}$ and $n_{i,1}$ are zero-mean, white Gaussian random processes and $g_{i,02}$ and $g_{i,12}$ are internal state elements.

A new augmented state equation (eq. 47) is formed using the original aircraft equations of motion and eq. 49. Subject to the necessary (possibly more stringent) observability conditions, the augmented state is estimated with an extended Kalman-Bucy filter. Included in the estimated state, $\hat{\mathbf{x}}_a$, are estimated histories of the original state, $\hat{\mathbf{x}}(t)$, of the forces and moments, $\hat{\mathbf{g}}(t)$, and of the partial derivatives of the forces and moments with respect to the state x_1,

$$\widehat{\frac{\partial \mathbf{g}}{\partial x_1}}(t)$$

The partial derivative of $\mathbf{g}$ with respect to any other state elements can also be estimated at the expense of additional elements in the augmented state vector, $\hat{\mathbf{x}}_a$. Each force/force-gradient history requires the addition of $3+2l$ state elements, where l is the number of partial derivatives estimated.

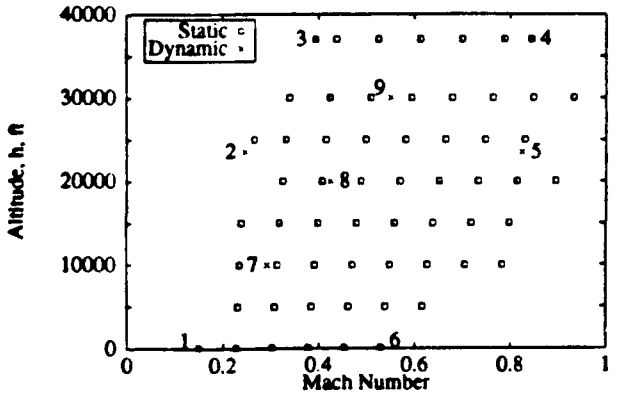

Figure 5: Twin-Jet Transport Flight Envelope and Training Trim Points.

Aerodynamic Model Identification from Simulated Flight-Test Data

The aerodynamic coefficient identification scheme is demonstrated by generating a model of the normal force coefficient, $C_Z(\mathbf{x}, \mathbf{u})$, 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. A single estimated derivative history, $\hat{C}_{Z_\alpha}(t)$, is included in the FD-training method.

Simulated Flight-Test and Training Data Generation

Simulated flight-test data are generated from a full, nonlinear, 6-degree-of-freedom model of a twin-jet transport. Standard rigid-body aircraft equations of motion are used [23]. The simulation contains an internal aerodynamic model based on extensive tabular data and algebraic constraints [22]. 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).

The static training data are generated at 60 trim points that span the aircraft operational flight envelope (Fig. 5). Static training data consist of the aircraft state and all of the require aerodynamic coefficients and derivatives for the trimmed condition. Static data points were calculated exactly, as no estimation was required.

Dynamic training data are generated around 9 additional trim points. These points are numbered in Fig. 5. 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 [5].

Starting from a specified trim condition, the aircraft is excited using a $\pm 10°$ "3211" elevator input. "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 [24]. Twenty-second-long measurement histories 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 [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. At the time of writing, the force-gradient estimation algorithm had not been implemented. To demonstrate the anticipated benefits of the function-derivative training algorithm, $\hat{C}_{Z_\alpha}(t)$ is calculated from the tabulated aerodynamic data using a finite-difference scheme and the currently estimated state, $\hat{\mathbf{x}}(t)$.

Neural Network Model and Training Procedure

A 9-input, single-hidden-layer network structure with 20 logistic sigmoid nodes in the hidden layer and a single linear node in the output layer is chosen. The 9 elements of the input vector are Mach number, angle of attack, pitch rate, density, angle of attack rate, altitude, throttle position, elevator deflection, and flap position:

$$C_Z = C_Z(M, \alpha, q, \rho, \dot{\alpha}, h, \delta_T, \delta_E, \delta_F) \qquad (50)$$

This 9-20-1 network has 221 adjustable weights and biases. The network is initialized with random weights selected from a zero-mean, Gaussian distribution with a variance of 0.1.

The initial state and process noise covariance matrices for the two training algorithms are identical. The

initial state covariance, $\mathbf{P}_0$, is $10^4\mathbf{I}$, and the process noise covariance matrix, $\mathbf{Q}$, is $10^{-2}\mathbf{I}$. The large 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 to 4 until the desired convergence is achieved.

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

Coefficient Identification

Using the procedure described above, a neural network model of $C_Z(\mathbf{x},\mathbf{u})$ is developed with the F-training method. The process noise matrix, $\mathbf{R}$, is 0.1 for this scalar example. After training, the network model precisely matches the coefficient histories at each of the dynamic training points. The actual and network estimated C_Z histories are given in Fig. 6 for dynamic training point #9. The histories are indistinguishable. The other 8 histories are equally well matched. It must be emphasized that these excellent results are obtained for large maneuvers (angles of attack ranging from $-5°$ to well over the stall angle of attack of $\approx 16°$) at 9 training points covering the entire flight envelope from stall speeds at sea level (Point #1) to Mach 0.85 at 37,000 feet (Point #4).

For use in nonlinear control laws, the network estimate of $\hat{C}_Z(\mathbf{x},\mathbf{u})$ is more important than the estimate of $\hat{C}_Z(t)$. The network value of $C_Z(\alpha)$ is plotted in Fig. 7 based on dynamic training point #9. The intersection of the two curves occurs at the trim condition, but nowhere else does the model fit well. Although generating excellent histories, the network has converged to an inadequate representation of the underlying aerodynamic coefficient model. 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.

Coefficient and Derivative Identification

Using the procedure outlined above and an FD-learning algorithm, a second network is developed to model $C_Z(\mathbf{x},\mathbf{u})$. In addition to $\hat{C}_Z(t)$, a single partial derivative history, $\hat{C}_{Z_\alpha}(t)$, is used during training. The process noise covariance matrix is

$$\mathbf{R} = \begin{bmatrix} 0.1 & 0 \\ 0 & 1.0 \end{bmatrix} \tag{51}$$

The larger $\mathbf{R}(2,2)$ element means that the derivative history is assumed to be noisier and less reliable for training. As this element gets larger, the results approach that of 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. 8 and 9. The $C_Z(t)$ history is not as well matched for the FD-trained network as it was for the F-trained network. The $C_Z(\alpha)$ match is much better. The additional training information forces an excellent match of the slope in the training regions around the initial trim condition of 5° angle of attack. The $C_Z(\alpha)$ fit deteriorates in the region above approximately 10°, where the training set contained little data. This mismatch gives rise to the mismatched histories around 5 and 8 seconds in Fig. 8, where the angle of attack is outside of the range $(-5°, 10°)$. Additional training data in the higher angle-of-attack regions should make the fit better in those regions.

Discussion

While 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 any 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 examples above, 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 correlations in the network input histories. Operating points and maneuvers must be chosen to promote orthogonality of the network inputs.

The state/force/force-derivative EKBF is not yet implemented. Questions relating to observability of the various derivatives are likely to arise. It should be possible to extract the dominant derivatives, such as C_{Z_α}, with sufficient accuracy to aid the neural network training process.

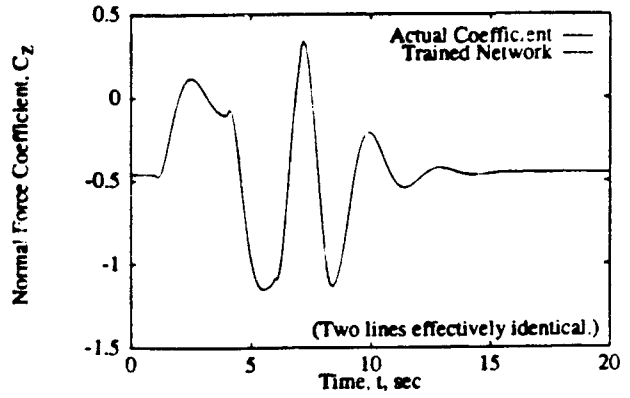

Figure 6: Normal Force Coefficient History using Function Training (Mach 0.53, 30,000 ft.).

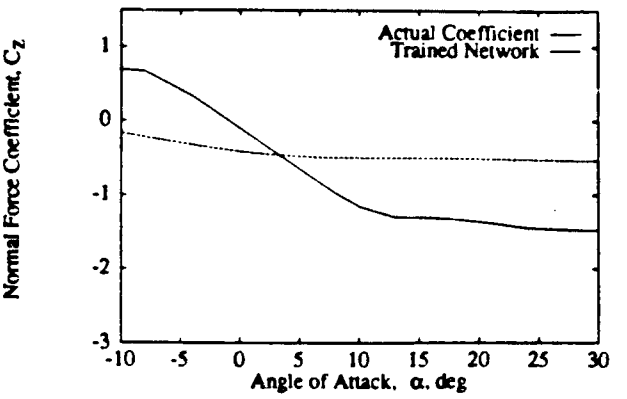

Figure 7: Normal Force Curve using Function Training (Mach 0.53, 30,000 ft.).

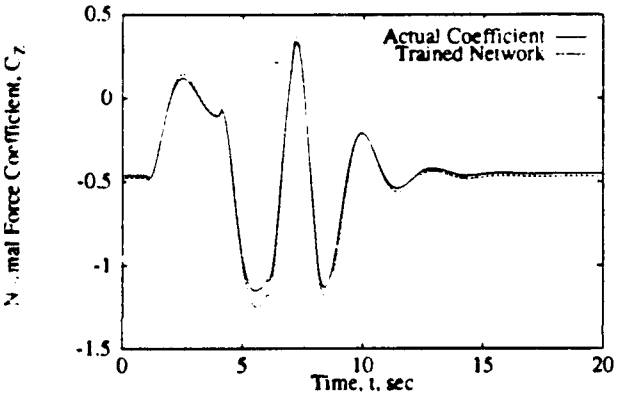

Figure 8: Normal Force Coefficient History using Function-Derivative Training (Mach 0.53, 30,0(ft.).

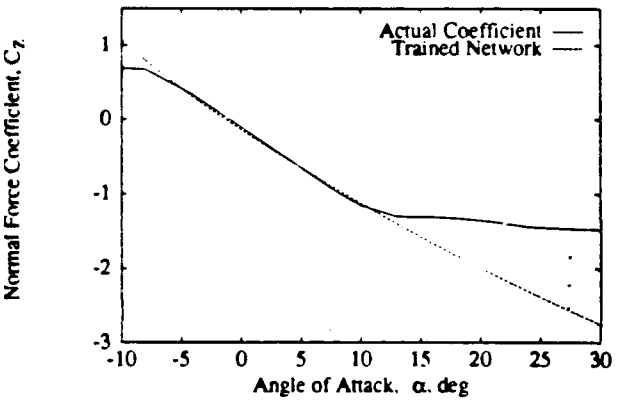

Figure 9: Normal Force Curve using Function-Derivative Training (Mach 0.53, 30,000 ft.).

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, though the functional dependence of force on specific inputs is not well identified. Including information about derivatives of the function with respect to its inputs greatly improves the functional fit. Networks are well-trained using static input data uniformly distributed through the input space and using dynamic input data generated from simulated flight tests. Good functional fits were obtained with both static and dynamic inputs in the regions of available training data. 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 FAA and NASA Langley Research Center under Grant No. NGL 31-001-252 and by the Army Research Office under Grant No. DAAL03-89-K-0092.

References

[1] A. Isidori, *Nonlinear Control Systems: An Intro-*

duction, Springer-Verlag, Berlin, 1989.

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

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

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

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

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

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

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

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

[10] R.F. Stengel, *Stochastic Optimal Control: Theory and Application*, John Wiley & Sons, Inc., New York, 1986.

[11] T. Poggio and F. Girosi, "Regularization Algorithms for Learning That Are Equivalent to Multilayer Networks," *Science*, Vol. 247, No. 4945, Feb. 23, 1990, pp. 978–982.

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

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

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

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

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

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

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

[19] S. Shah and F. Palmieri, "MEKA – A Fast, Local Algorithm for Training Feedforward Neural Networks," *1990 International Joint Conference on Neural Networks*, Vol. 3, San Diego, CA, 1990, pp. 41–46.

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

[21] F.C. Schweppe, *Uncertain Dynamic Systems*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1973.

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

[23] D. McRuer, I. Ashkenas, and D. Graham, *Aircraft Dynamics and Automatic Control*, Princeton University Press, Princeton, NJ, 1973.

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