JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 33, No. 2, March–April 2010

Neural Partial Differential Method for Extracting Aerodynamic Derivatives from Flight Data

S. Das*
Indian Institute of Technology, Kharagpur 721 302, India
R. A. Kuttieri†
Hindustan Aeronautics Limited, Bangalore 560 001, India
M. Sinha‡
Indian Institute of Technology, Kharagpur 721 302 India
and
R. Jategaonkar§
DLR, German Aerospace Center, 38108 Braunschweig, Germany

DOI: 10.2514/1.46053

Although the classical approach based on output or equation error methods is commonly applied to estimate stability and control derivatives from flight data, an alternative approach based on a neural network with universal nonlinear mapping capability is preferred in some applications. The first approach yields explicit models in terms of physically interpretable aerodynamic derivatives, whereas the other leads to implicit models in general. The two approaches being characteristically different, the comparison of the results in terms of aerodynamic derivatives pose difficulties. An approach based on finite differences (termed as the delta method), applied to a trained neural network, has been reported in the literature to overcome the difficulties. In this paper, yet another approach based on the partial differential of neural output is suggested and demonstrated. The performance of the two approaches (namely, the finite difference and the partial differential approach) is evaluated in terms of the estimated aerodynamic derivatives. It is shown that the two methods give comparable results; however, the partial differential approach provides first-order aerodynamic derivatives more accurately. Moreover, the results of the partial differential approach are comparable with the results obtained from the equation error method. Moreover, gradient descent and scaled conjugate gradient methods are evaluated in terms of the mean square error for training the neural network. The proposed partial differential approach does not suffer from numerical instability and provides higher-order derivatives; therefore, it can be extended to the nonlinear aerodynamic regime and to unstable aircraft.

Nomenclature

C_l	=	nondimensional rolling moment coefficient
C_n	=	nondimensional yawing moment coefficient
C_Y	=	nondimensional side force coefficient
C_{Y_0}, C_{l_0}, C_{n_0}	=	zero aerodynamic term in the linear aerodynamic model
d_i	=	normalized observed or measured value of ith output
$E(.)$	=	error function
$E'(.)$	=	gradient of the error function
$f(.)$	=	activation function
$g(.)$	=	activation function
L	=	rolling moment, Nm
N	=	yawing moment, Nm
p	=	roll rate, rad/s
p^*	=	normalized roll rate
q	=	pitch rate, rad/s
r	=	yaw rate, rad/s
r^*	=	normalized yaw rate
$\tilde{U}$	=	weight matrix connecting the input layer and the first hidden layer
$\tilde{V}$	=	weight matrix connecting the first hidden layer and the second hidden layer
$\tilde{W}$	=	weight matrix connecting the second hidden layer and the output layer
$\tilde{w}$	=	weight vector containing all weights of the neural network
$\tilde{x}$	=	output vector of the first hidden layer
Y	=	side force, N
$\tilde{y}$	=	output vector of the second hidden layer
$z_{i,\text{norm}_{\max}}$	=	upper limit of normalization for the ith output
$z_{i,\max}$	=	maximum value of the ith output
$z_{i,\min}$	=	minimum value of the ith output
$z_{i,\text{norm}_{\min}}$	=	lower limit of normalization for the ith output
$\tilde{z}$	=	output vector of the neural network
β	=	sideslip, rad
δa	=	aileron deflection, rad
δr	=	rudder deflection, rad
ν_i	=	ith element of the inputs to the second hidden layer neuron
$\tilde{\chi}$	=	input vector to the neural network

Received 19 June 2009; revision received 10 December 2009; accepted for publication 10 December 2009.

*Research Scholar, Department of Aerospace Engineering.
†Scientist.
‡Assistant Professor, Department of Aerospace Engineering; masinha@aero.iitkgp.ernet.in.
§Senior Scientist, Institute of Flight Systems.

I. Introduction

ACCURATE knowledge of stability and control derivatives helps in making a realistic flight simulator for pilot training, designing control laws, expanding flight envelope, and improving airworthiness. The process of extracting the stability and control derivatives from flight measured data is known as aircraft parameter estimation and can be broadly classified into two categories: namely, direct approach and indirect approach. Within the purview of the direct approach come methods, such as equation error method

(EEM), output error method, and filter error method (FEM) [1–3]. The indirect method consists of a nonlinear filter, for which the states are defined as the parameters of the system to be estimated [2,3]. Furthermore, some methods are recursive in nature and, therefore, can produce estimates in real time, whereas some deal with offline estimation (postflight data processing). However, they require initial estimates of the parameters to start the estimation process. Initial estimates of the parameters may be obtained either from wind-tunnel test data or computational fluid dynamics analysis. Parameter estimation methods may diverge if initialization is poor. Additionally, a priori knowledge of the structure of the aerodynamic model is required. Therefore, dealing with complex aerodynamics phenomena, such as stall hysteresis, high angle of attack flight, and large amplitude time-dependent maneuvers becomes complex in the absence of a thorough understanding about the system dynamics [3], thereby placing constraints on the applicability of these methods.

To overcome these problems, it is imperative to use any technique that obviates the need to know a priori dynamics of the system in the complex aerodynamics domain. The ability of a neural network (NN) to act as a universal approximator [4] provides a potential alternative to address the problem of aircraft parameter estimation, because it can capture highly complex nonlinear phenomena in a global sense without a priori knowledge about the dynamic model. Moreover, it does not require the initial estimates of the parameters, unlike the conventional methods. These two properties of a NN render it a natural tool for aircraft parameter estimation from flight data. An early investigation into the use of a NN for aircraft parameter estimation can be found in [5]. In [5], a NN was used as a mapping function to map the inputs (state and control variables) to the outputs (aerodynamic forces and moments). Back propagation method was used to train the NN. Its extension for the estimation of aerodynamic coefficients has been explored in [6–8]. In most of these applications, the gradient learning algorithm was used to train the NN. This paper investigates the effect of learning algorithms [i.e., the gradient learning and the scaled conjugate gradient algorithm (SCGA)] on the mean square error (MSE), formulated as the error function of the NN. Moreover, it presents a comparative study of the estimated stability and control derivatives obtained from the NN, trained using the SCGA and then applying the delta method [9,10] and an innovative partial differential method (PDM), suggested in this paper, to extract the stability and control derivatives.

The delta and the zero methods, which are based on finite difference approximation, were reported in [9,10] for the estimation of stability and control derivatives of an aircraft using a NN in the posttraining phase. The delta method consists of finding the parameters to be estimated using a small variation in one of the state/control variables (input variables). This is done by keeping all other state/control variables constant at the nominal values and perturbing the state/control variables, with respect to which derivatives of the aerodynamic forces or moments are required. The corresponding variation in the output, once divided by the corresponding variation in the state variable, then provides an estimate of the derivative. The zero method consists of finding the ratio of variation in the value of an aerodynamic coefficient to the incremental variation in one of the motion or control variables, whereas the rest of the motion and control variables remain identically zero. Both of these methods employ central difference approximation for computing the derivatives. The modified delta (MD) method, which is a modification over the delta method, has been recently reported in [11]. The MD method is based on the fact that the ratio of the variation in the value of an aerodynamic coefficient due to the variation in only one of the motion/control variables will yield the corresponding stability/control derivative, whereas the variation in other motion/control variables are identically zero. The MD method uses the variation of measured aircraft motion and control variables as the input to the NN and the variation of aerodynamic force or moment coefficients as the output for training the NN. Thus, all these methods are based on finite difference approximation and, therefore, cannot be used to compute exact partial derivatives.

In this paper, a novel neurocomputing approach is presented to address the problem of aircraft parameter estimation by means of the partial differentiation of the neural outputs [12]. Unlike the delta method and the zero method, the partial differential analysis computes the derivatives without applying the finite difference approximation. It is just a one-shot method, in the sense that it does not require posttraining data processing to obtain the derivatives, whereas in the delta and the zero methods, posttraining data processing is required to get the derivatives at each data point by varying, one-by-one, each of the input variables. The PDM gives an exact first-order derivative at a point, unlike the delta method, which is an approximation of the partial differential using the finite difference approach. The proposed approach is used to model the aerodynamic forces and moments as the outputs and aircraft motion variables and control surface deflections as the inputs to the NN. This paper outlines the methodology and results of flight data analysis and compares the two approaches of deriving the stability and control derivatives from the trained NN. It also evaluates the performance of the delta method against the PDM. Finally, the neural partial differential-based estimated derivatives are used to reconstruct the aerodynamic coefficients. These are compared against the reconstructed aerodynamic coefficients obtained using the EEM. The proposed new approach yields the reconstructed aerodynamic coefficients, which are comparable with those obtained from the EEM. In addition, it is concluded that a NN can be used to derive the explicit model of a system in a polynomial form.

II. Methods of Data Analysis

It was pointed out in the previous section that parameter estimation can be carried out using any one of the classical estimation/filtering approaches. There are various pros and cons related to these methods. Their selection for the estimation purpose depends on the desired accuracy in estimation, the nature of the measurements, and the availability of a priori estimates of the parameters. For details of various methods, [3] may be consulted. In this section, the EEM is outlined, and the proposed PDM is elaborated.

A. Equation Error Method

In this subsection, a brief summary of the EEM is presented, and the formulation behind this method is used to check the consistency of the results obtained using NN. The equation error technique is the simplest method of estimating the aircraft stability and control derivatives and uses linear regression in conjunction with ordinary least-squares technique. The sum of the squared differences between the measured and the predicted aerodynamic forces and the moments coefficients are minimized to estimate the unknown parameters. However, an accurate measurement of all the states and their derivatives are required. The presence of measurement noise makes the estimation biased and inconsistent.

The nondimensional lateral-directional forces and moments coefficients can be written as

$$C_Y = C_{Y_0} + C_{Y_\beta}\beta + C_{Y_{\delta a}}\delta a + C_{Y_{\delta r}}\delta r + C_{Y_p}p^* + C_{Y_r}r^* \quad (1)$$

$$C_l = C_{l_0} + C_{l_\beta}\beta + C_{l_{\delta a}}\delta a + C_{l_{\delta r}}\delta r + C_{l_p}p^* + C_{l_r}r^* \quad (2)$$

$$C_n = C_{n_0} + C_{n_\beta}\beta + C_{n_{\delta a}}\delta a + C_{n_{\delta r}}\delta r + C_{n_p}p^* + C_{n_r}r^* \quad (3)$$

where C_Y, C_l, and C_n are the nondimensional lateral force, the rolling moment, and the yawing moment coefficients, respectively. The side slip, the aileron deflection, the rudder deflection, the roll rate (normalized), and the yaw rate (normalized) are β, δa, δr, p^*, and r^*, respectively. The constant terms, on the right-hand side in Eqs. (1–3), are the stability or control derivatives to be estimated. In the previous equations, it is assumed that the lateral-directional forces and moments do not depend upon the longitudinal variables. However, in data processing, the flight maneuvers with longitudinal excitation resulting from the elevator deflection have been included. The coupling effects in the lateral-directional motion due to the longitudinal motion are usually very small, and the first- and higher-order

terms are neglected. In the previous postulated model, the zero aerodynamic terms C_{Y_0}, C_{l_0}, and C_{n_0} account for any asymmetry in the aircraft structure, as well as weak aerodynamic coupling effects as an approximation. It is assumed here that the higher-order terms are negligible. In fact, the least-squares estimation under such assumption absorbs the derivatives of higher order in the first-order ones.

B. Neural Network

As mentioned earlier, the zero method, the delta method [9,10], and the MD method [11] were proposed to estimate the stability and control derivatives of an aircraft using a feedforward NN. These methods indeed provide hope for estimating the parameters in an alternative way. Especially, the estimation of the parameters of an unstable aircraft becomes possible using these methods. The greatest advantage of using a NN is that it is not susceptible to the problem of numeric overflows encountered in integrating the equations of motion for an unstable aircraft. This also helps in eliminating the need to solve the 6 degree-of-freedom differential equation coupled with the elastic degrees of freedom, thereby presenting an easy method to deal with an aeroelastic aircraft. However, the parameters estimated using these methods show large scatter, which is attributed to training being less than perfect [10]. Moreover, the number of estimated values for each of the parameters is equal to the number of data points used for training the network, which is again attributed to the imperfect training [10]. It is an accepted fact that the quality of training will affect the scatter in the derivatives. However, the different values of derivatives at different time instants that arise due to their dependence on the motion and control variables also contribute to the observed scatter.

In this paper, these issues are resolved, and it is clearly shown that scatter in the derivatives (standard deviation for the complete data set) is a combined result of both the quality of training and the polynomial model, which a NN represents. In the following paragraphs, the method proposed in this paper is discussed, and many apprehensions about the NN for aircraft parameters estimation are addressed.

The proposed method is based on the partial differential of the NN output, carried out analytically at the end of training, using all the terms calculated during training. The input and the output data used for training the NN are normalized. This not only makes various observations equally important but also helps reduce the training error and the training time [13,14]. A general three-layered NN architecture is shown in Fig. 1. It consists of two hidden layers of neurons and one output layer of neurons. The output layer consists of the summation function only. Let the input and the output vectors of the NN be characterized by $\tilde{\chi} \in \Re^{n+1}$ and $\tilde{z} \in \Re^{k}$, respectively. Let the second hidden layer output vector, along with the bias, be represented by $\tilde{y} \in \Re^{l+1}$ and the first hidden layer output vector, along with the bias, be represented by $\tilde{x} \in \Re^{m+1}$. The output of the NN can be written as

$$\tilde{z} = \tilde{W}^T \tilde{y} \tag{4}$$

where $\tilde{W}$ is the weight matrix connecting the second hidden layer and the output layer and is defined as

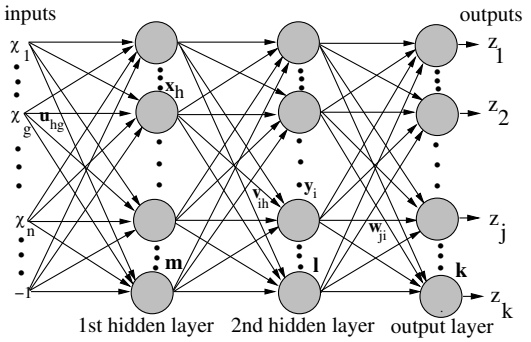

Fig. 1 Schematic of feedforward NN.

$$\tilde{W} = \begin{bmatrix} b_{w1} & \cdots & b_{wk} \\ w_{11} & \cdots & w_{1k} \\ \vdots & \ddots & \vdots \\ w_{l1} & \cdots & w_{lk} \end{bmatrix} \tag{5}$$

The vector $\tilde{y}$ can be written as

$$\tilde{y} = \tilde{f}(\tilde{V}^T \tilde{x}) = \tilde{f}(\tilde{v}) \tag{6}$$

where $\tilde{f}(.)$ denotes the activation function vector defined as

$$\begin{aligned} \tilde{f} &= [-1 \quad f(v_1) \quad f(v_2) \quad \cdots \quad f(v_i) \quad \cdots \quad f(v_k)]^T \\ &= [y_0 \quad y_1 \quad \ldots \quad y_i \quad \ldots \quad y_l]^T \in \Re^{l+1} \end{aligned} \tag{7}$$

and v_i denotes the ith element of the inputs to the neuron; f is the activation function and is chosen to be the tangent hyperbolic function [Eq. (8)], as it provides better mapping [15]:

$$f(\alpha) = \frac{1 - e^{-\lambda\alpha}}{1 + e^{-\lambda\alpha}} \tag{8}$$

The bias term b_{wi} is absorbed into the matrix $\tilde{W}$ for notational convenience and compactness. The weight matrix $\tilde{V}$ can be written as

$$\tilde{V} = \begin{bmatrix} b_{v1} & \cdots & b_{vl} \\ v_{11} & \cdots & v_{1l} \\ \vdots & \ddots & \vdots \\ v_{m1} & \cdots & v_{ml} \end{bmatrix} \tag{9}$$

where b_{vl} denotes the lth element of the bias term. Similarly, the output of the first hidden layer neurons can be written as

$$\tilde{x} = \tilde{g}(\tilde{U}^T \tilde{\chi}) = \tilde{g}(\tilde{\beta}) \tag{10}$$

$$\begin{aligned} \tilde{g} &= [-1,\ g(\beta_1),\ g(\beta_2),\ \ldots,\ g(\beta_i),\ \ldots,\ g(\beta_k)]^T \\ &= [x_0,\ x_1,\ \ldots,\ x_i,\ \ldots,\ x_m]^T \in \Re^{m+1} \end{aligned} \tag{11}$$

The activation function defined by Eq. (8) is $g(.)$, and the weight matrix $\tilde{U}$ is defined as

$$\tilde{U} = \begin{bmatrix} b_{u1} & \cdots & b_{um} \\ u_{11} & \cdots & u_{1m} \\ \vdots & \ddots & \vdots \\ u_{n1} & \cdots & u_{nm} \end{bmatrix} \tag{12}$$

where b_{um} is the mth element of the bias term. The input to the NN denoted by the augmented input vector $\tilde{\chi}$ can be defined as $\tilde{\chi} = [-1 \quad \tilde{a}^T]^T = [\chi_0 \quad \chi_1,\ \ldots,\ \chi_n]^T$, where $\tilde{a} \in \Re^n$ is the input vector excluding the bias term. The normalization of the inputs and the outputs is carried out using Eq. (13):

$$z_{i,\text{norm}} = z_{i,\text{norm}_{\min}} + (z_{i,\text{norm}_{\max}} - z_{i,\text{norm}}) \times \frac{z_i - z_{\min}}{z_{\max} - z_{\min}} \tag{13}$$

The upper and lower limits are $z_{i,\text{norm}_{\max}}$ and $z_{i,\text{norm}_{\min}}$, respectively, of the normalization range for the outputs. The maximum and the minimum values of the ith output variable are $z_{i,\max}$ and $z_{i,\min}$. In this paper, $z_{i,\text{norm}_{\max}} = 0.9$ and $z_{i,\text{norm}_{\min}} = -0.9$, thus Eq. (13) maps the outputs in the range $[-0.9, 0.9]$, which is the active range of the tangent hyperbolic function; therefore, normalization is not done in the range $[-1.0, 1.0]$. The values -1.0 or $+1.0$ are achieved when the argument of the tangent hyperbolic function approaches infinity, which in reality never happens. Moreover, the derivative of the tangent hyperbolic function approaches zero as the argument tends to infinity. Therefore, the data points mapped to these values never contribute to the learning process through weights change, using back propagation in conjunction with the gradient descent or the SCGAs.

The output vector of the NN can be written as

$$\tilde{z} = \tilde{W}^T \tilde{f}[\tilde{V}^T \tilde{g}(\tilde{U}^T \tilde{\chi})] \tag{14}$$

For training of the NN, the optimization function used is the MSE and is given by Eq. (15):

$$\text{MSE} = E(\tilde{w}) = \frac{1}{2P} \sum_{j=1}^{P} \sum_{i=1}^{k} (d_i - z_i)_j^2 \tag{15}$$

where $\tilde{w}$ is a vector consisting of all the weights of the NN, E is the error function, P is the total data points (number of observations), d_i is the ith normalized measured value of the output, z_i is the normalized computed value of the output [for which the normalization is done using Eq. (13)], and the index j stands for the jth data point. Because the NN training is done in batch mode, MSE is computed at the end of every step of the iteration, using updated values of the weights.

C. Scaled Conjugate Gradient Algorithm

To train the NN (described in the previous section), it is necessary to suitably estimate the weighting matrices $\tilde{U}$, $\tilde{V}$, and $\tilde{W}$. We consider here only the classical gradient-based algorithm and its variant, called the SCGA [16–18]. Detailed algorithmic descriptions of these and other algorithms can be found in any standard literature on NN. Nevertheless, for the sake of completeness, we provide a brief description of the SCGA proposed by Moller [16], which is a second-order algorithm based on a Levenberg–Marquardt approach, requiring first-order gradients that are used to formulate the Hessian, thereby saving computational efforts.

1. Choose weight vector $\tilde{w}_1$ and scalars $0 < \sigma \leq 10^{-4}$, $0 < \lambda_1 \leq 10^{-6}$, $\bar{\lambda}_1 = 0$. Set $\tilde{p}_1 = \tilde{r}_1 = -E'(\tilde{w}_1)$, $k = 1$ and success = true. Here, $E'(\tilde{w}) = -\tilde{p}$ is the gradient of the error function [Eq. (15)] with respect to the weight vector $\tilde{w}$, and $\tilde{p}$ is the gradient vector. The parameters are σ, λ, and $\bar{\lambda}$.

2. If success = true, then calculate second-order information:

$$\sigma_k = \sigma/|\tilde{p}_k|; \qquad \tilde{s}_k = [E'(\tilde{w}_k + \sigma_k \tilde{p}_k) - E'(\tilde{w}_k)]/\sigma_k; \qquad \delta_k = \tilde{p}_k^T \tilde{s}_k$$

3. Scale

$$\delta_k\text{: } \delta_k = \delta_k + (\lambda_k - \bar{\lambda}_k)|\tilde{p}_k|^2$$

4. If $\delta_k \leq 0$, then make the Hessian matrix positive definite:

$$\bar{\lambda}_k = 2(\lambda_k - \delta_k/|\tilde{p}_k|^2); \qquad \delta_k = -\delta_k + \lambda_k |\tilde{p}_k|^2; \qquad \lambda_k = \bar{\lambda}_k$$

5. Calculate the step size: $\mu_k = \tilde{p}_k^T \tilde{r}_k$ and $\alpha_k = \mu_k/\delta_k$.

6. Calculate the comparison parameter:

$$\Delta_k = 2\delta_k [E(\tilde{w}_k) - E(\tilde{w}_k + \alpha_k \tilde{p}_k)]/\mu_k^2$$

.

7. If $\Delta_k \geq 0$, then a successful reduction in error can be made:

$$\tilde{w}_{k+1} = \tilde{w}_k + \alpha_k \tilde{p}_k; \quad \tilde{r}_{k+1} = -E'(\tilde{w}_{k+1}); \quad \bar{\lambda}_k = 0; \quad \text{success} = \text{true}$$

If $k \text{mod} N = 0$, then restart algorithm:

$$\tilde{p}_{k+1} = \tilde{r}_{k+1}$$

else:

$$\beta_k = (|\tilde{r}_{k+1}|^2 - \tilde{r}_{k+1}^T \tilde{r}_k)/\mu_k; \qquad \tilde{p}_k = \tilde{r}_{k+1} + \beta_k \tilde{p}_k$$

If $\Delta_k \geq 0.75$, then reduce the scale parameter:

$$\lambda_k = 0.25 \lambda_k$$

else:

$$\bar{\lambda}_k = \lambda_k; \qquad \text{success} = \text{false}$$

8. If $\Delta_k < 0.25$, then increase the scale parameter:

$$\lambda_k = \lambda_k + [\delta_k (1 - \Delta_k)/|\tilde{p}_k|^2]$$

.

9. If the steepest descent direction $\tilde{r}_k \neq \tilde{0}$, then set $k = k + 1$ and go to step 2, else terminate and return $\tilde{w}_{k+1}$ as the desired minimum.

It should be noted that the SCGA handles the weights as a vector and not as a matrix; therefore, the weights are converted from vector to matrix and matrix to vector during the training period. Moreover, the SCGA also computes the error gradients with respect to the weights. These terms are used in the next section by the PDM to compute the stability and control derivatives.

D. Partial Differential Method

After the training is over, the NN represents the mapping function between the inputs and the outputs. For this research, the forces and moments represent the outputs, whereas the control and motion variables represent the inputs. The partial differential of Eq. (14), with respect to the augmented input vector $\tilde{\chi}$, then represents the stability and control derivatives terms in the equation of forces and moments. The partial derivatives of the outputs, with respect to the inputs {i.e., the desired stability and control derivatives [Eq. (16)]}, can be worked out according to the chain differentiation rule:

$$\left[\frac{\partial \tilde{z}}{\partial \tilde{\chi}}\right] = \left[\frac{\partial \tilde{z}}{\partial \tilde{z}_{\text{norm}}}\right]\left[\frac{\partial \tilde{z}_{\text{norm}}}{\partial \tilde{y}}\right]\left[\frac{\partial \tilde{y}}{\partial \tilde{x}}\right]\left[\frac{\partial \tilde{x}}{\partial \tilde{\chi}_{\text{norm}}}\right]\left[\frac{\partial \tilde{\chi}_{\text{norm}}}{\partial \tilde{\chi}}\right] \tag{16}$$

where

$$\left[\frac{\partial \tilde{z}}{\partial \tilde{\chi}}\right] = \begin{bmatrix} \frac{\partial z_1}{\partial \chi_0} & \cdots & \frac{\partial z_1}{\partial \chi_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial z_k}{\partial \chi_1} & \cdots & \frac{\partial z_k}{\partial \chi_n} \end{bmatrix} \tag{17}$$

and

$$\left[\frac{\partial \tilde{z}}{\partial \tilde{z}_{\text{norm}}}\right] = \begin{bmatrix} \frac{\partial z_1}{\partial z_{1,\text{norm}}} & 0 & \cdots & 0 \\ 0 & \frac{\partial z_2}{\partial z_{2,\text{norm}}} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{\partial z_k}{\partial z_{k,\text{norm}}} \end{bmatrix} \tag{18}$$

is a diagonal matrix of size $(k \times k)$. The second, third, and fourth terms on the right-hand side in Eq. (16) are matrices of size $[k \times (l+1)]$, $[(l+1) \times (m+1)]$, and $[(m+1) \times (n+1)]$, and the fifth term is a diagonal matrix of size $[(n+1) \times (n+1)]$, similar to that in Eq. (18). The ith diagonal term in the matrix on the right-hand side in Eq. (18) can be written as

$$\frac{\partial z_i}{\partial z_{i,\text{norm}}} = \frac{z_{i,\max} - z_{i,\min}}{z_{i,\text{norm}_{\max}} - z_{i,\text{norm}_{\min}}} \tag{19}$$

in which the term

$$(z_{i,\text{norm}_{\max}} - z_{i,\text{norm}_{\min}}) = 1.8$$

in the present case. Similarly, $[(\partial \tilde{\chi}_{\text{norm}})/\partial \tilde{\chi}]$ can be written as

$$\left[\frac{\partial \tilde{\chi}_{\text{norm}}}{\partial \tilde{\chi}}\right] = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & \frac{\partial \chi_{1,\text{norm}}}{\partial \chi_1} & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & \frac{\partial \chi_{n,\text{norm}}}{\partial \chi_n} \end{bmatrix} \tag{20}$$

In Eq. (20), the ith ($i \neq 0$) diagonal term of the matrix $[(\partial \tilde{\chi}_{\text{norm}})/\partial \tilde{\chi}]$ can be written as

$$\frac{\partial \chi_{i,\text{norm}}}{\partial \chi_i} = \frac{\chi_{i,\text{norm}_{\max}} - \chi_{i,\text{norm}_{\min}}}{\chi_{i,\max} - \chi_{i,\min}} \tag{21}$$

here, $\chi_{i,\min}$ and $\chi_{i,\max}$ stand for the minimum and the maximum values of the ith input variable. Because the output layer does not have any nonlinearity, $[(\partial \tilde{z}_{\text{norm}})/\partial \tilde{y}]$ can be written as

$$\left[\frac{\partial \tilde{z}_{\text{norm}}}{\partial \tilde{y}}\right] = \begin{bmatrix} \frac{\partial z_{1,\text{norm}}}{\partial y_0} & \frac{\partial z_{1,\text{norm}}}{\partial y_1} & \cdots & \frac{\partial z_{1,\text{norm}}}{\partial y_l} \\ \frac{\partial z_{2,\text{norm}}}{\partial y_0} & \frac{\partial z_{2,\text{norm}}}{\partial y_1} & \cdots & \frac{\partial z_{2,\text{norm}}}{\partial y_l} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial z_{k,\text{norm}}}{\partial y_0} & \frac{\partial z_{2,\text{norm}}}{\partial y_1} & \cdots & \frac{\partial z_{n,\text{norm}}}{\partial y_l} \end{bmatrix} = \tilde{W}^T \tag{22}$$

As the outputs of the neurons are obtained in normalized form during training, Eq. (22) is valid. Similarly, $[\partial \tilde{y}/\partial \tilde{x}]$ can be written as

$$\left[\frac{\partial \tilde{y}}{\partial \tilde{x}}\right] = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & f_1' & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & f_l' \end{bmatrix} \begin{bmatrix} b_{v1} & \cdots & b_{vl} \\ v_{11} & \cdots & v_{1l} \\ \vdots & \ddots & \vdots \\ v_{m1} & \cdots & v_{ml} \end{bmatrix}^T$$
$$= \text{diag}[\,0 \quad f_1' \quad f_2' \quad \cdots \quad f_l'\,]\tilde{V}^T \tag{23}$$

and $[\partial \tilde{x}/(\partial \tilde{a}_{\text{norm}})]$ is written as

$$\left[\frac{\partial \tilde{x}}{\partial \tilde{\chi}_{\text{norm}}}\right] = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & g_1' & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & g_m' \end{bmatrix} \begin{bmatrix} b_{u1} & \cdots & b_{um} \\ u_{11} & \cdots & u_{1m} \\ \vdots & \ddots & \vdots \\ u_{n1} & \cdots & u_{nm} \end{bmatrix}^T$$
$$= \text{diag}[\,0 \quad g_1' \quad g_2' \quad \cdots \quad g_i' \quad \cdots \quad g_l'\,]\tilde{U}^T \tag{24}$$

It should be noted that during the training of the NN, the previous matrices are available; therefore, they do not require any extra computation. The stability and control derivatives are computed at the end of training or, if required, can be computed after each iteration. For example, in Eq. (20), all the diagonal terms are actually calculated at the start of training, which is obvious from Eq. (21) [which represents the ith term of the diagonal elements of the matrix in Eq. (20)]. In fact, each diagonal element is a constant, which is fixed by the minimum and maximum of the particular input variable and the limits of the normalization range [see Eq. (19)]. Similarly, the diagonal terms in Eq. (18) are calculated at the end of training. These terms may also be computed for the performance evaluation at an intermediate stage of training, which requires denormalized outputs. Denormalization of the outputs is necessary, because at the start of training, they are normalized using Eq. (13). Similarly, in Eqs. (22–24), the weight matrices $\tilde{W}$, $\tilde{V}$, and $\tilde{U}$, respectively, are available during training from the seventh or ninth step of the SCGA algorithm for the weights update (i.e., $\tilde{w}_{k+1}$). The partial differential vector terms appearing in Eqs. (22–24) are also available, because these terms are computed in the SCGA while computing the partial differential of the performance index (error function) with respect to the weight vector [i.e., $E'(\tilde{w})$ (second step of the SCGA algorithm)]. Thus, all the terms required for computing the partial differential are available. This saves time and effort, unlike that required in the delta method (explained in the next subsection) in the posttraining phase. The derivatives, thus computed for different sets of initial weights at different data points, are averaged, and their standard deviations are calculated. The relative standard deviation (RSTD), which also corresponds to the Cramer–Rao bound for nonstatistical approaches (such as the EEM), is defined as

$$\text{RSTD} = \frac{\text{Standard Deviation}}{\text{Average Value}} \times 100\% \tag{25}$$

The derivatives obtained using the PDM, being of the first-order, may differ slightly from that used in the linear models in Eqs. (1–3), because unlike the least-squares estimation, the coefficients of higher-order terms are not absorbed. In addition to these derivatives, constant terms having small magnitude may be present in the model. Such terms can be extracted, once the derivatives have been estimated, by summing the motion dependent terms in Eqs. (1–3) and then subtracting from the measured value. This is followed by averaging the results obtained at different data points.

Dependence of the stability and control aerodynamic derivatives on the motion and control variables can be accommodated for assuming the fact that a trained NN represents a polynomial of a higher degree in motion and control variables. Therefore, the derivatives obtained through the partial differential of this polynomial are also dependent on the motion and control variables, unlike those concluded in [9,10]. However, these coefficients can be averaged and can be used in Eqs. (1–3). This fact was not highlighted in earlier papers [9,10], nor were simulations carried out to verify this fact.

E. Delta Method

The training part for both the PDM and the delta method is similar. The derivatives for the PDM become available at the end of training. However, the delta method requires posttraining processing of data, as explained in the following paragraphs.

The derivatives estimation using the delta method requires two sets of perturbed input variables (say, β for computing C_{Y_β}), keeping the other inputs as used in the training unchanged. Let the perturbation given to β be denoted by $\Delta\beta$. The same perturbation is given to all the data points used for training the NN. Therefore, the perturbed input variable in the two sets can be denoted by $\beta + \Delta\beta$ and $\beta - \Delta\beta$, and the corresponding output C_Y can be denoted by C_Y^+ and C_Y^-, respectively. Then, the derivative C_{Y_β} can be approximated as

$$C_{Y_\beta} = \frac{C_Y^+ - C_Y^-}{2\Delta\beta} \tag{26}$$

Equation (26) is applied to the complete data set in the posttraining phase to extract the derivative C_{Y_β}. Derivatives at different data points, thus obtained, are then averaged, and the standard deviation is calculated. RSTD can be found using Eq. (25). Following the same procedure, other derivatives can be computed.

III. Results and Discussion

As a typical case, we consider here the modeling of aerodynamic coefficients and the extraction of the stability and control derivatives pertaining to the lateral-directional motion of an aircraft. The flight data analyzed here were generated with the Advanced Technology Testing Aircraft System (ATTAS) [19] of DLR, German Aerospace Center in Germany. This aircraft was developed by modifying, in collaboration with the aircraft manufacturer Messerschmitt–Bölkow–Blohm, a twin engine, short haul, 44-passenger aircraft (of the type VFW-614) instrumented with a wide range of modern test and recording equipment [19]. The data are 85 s in duration, with a sampling time of 0.04 s. Three distinct maneuvers performed by the pilot are considered: short period mode (for the first 25 s, using multistep elevator input), bank-to-bank maneuver (for the next 30 s, using aileron input), and Dutch roll (for last 30 s, using rudder doublet input). The lateral-directional measured variables (implicit) consist of the side force Y, the rolling moment L, and the yawing moment N, which can be written in a nondimensional format as C_Y, C_l, and C_n, respectively. Therefore, the output vector for the NN can be written as

$$\tilde{z} = [\,C_Y \quad C_l \quad C_n\,]^T \tag{27}$$

Similarly, the inputs consist of the motion variables β, p^*, and r^* and the control inputs δa and δr, as described earlier in Sec. II.A. Hence, the input vector to the NN can be written as

$$\tilde{\chi} = [\,-1 \quad \beta \quad \delta a \quad \delta r \quad p^* \quad r^*\,]^T \tag{28}$$

Here, the -1 term in the vector $\tilde{\chi}$ is a bias. A linear modeling of C_Y, C_l, and C_n is given by Eqs. (1–3). The input variables are directly measured, whereas the output variables are derived from the measured linear accelerations and angular rates through a

preprocessing step. The body-fixed accelerations at the center of gravity are computed from the measured translational accelerations using an accelerometer sensor positioned somewhere away from the center of gravity. The relevant expressions for computing the body-fixed acceleration at the center of gravity are given in [19,20]. The angular accelerations ($\dot{p}$, $\dot{q}$, $\dot{r}$) are computed by numerical differentiation of the measured angular rates (p, q, r). The three body axes aerodynamic force coefficients are computed using appropriate equations, as described in [19,20]. Similarly, the body-axis rolling, pitching, and yawing moment coefficients (C_l, C_m, C_n) referenced to the center of gravity are computed using appropriate equations, as given in [19,20]. Computation of body-axis rolling, pitching, and yawing moment coefficients requires moments of inertia and the center of gravity position at different time instants, both of which change due to the consumption of fuel and are computed using the recorded data. Once trained, the NN was checked for proper mapping by comparing the actual outputs and the measured outputs. Simulations were performed for a varying number of iterations to assess the accuracy of mapping.

To make a direct comparison, the delta method was applied in the posttraining phase for the estimation of derivatives. It should be noted that the PDM results were available at the end of the training itself. The NN architecture used contains one neuron in the input layer, which makes the derived derivatives insensitive to the initial weights used in the NN. The hidden layer consists of 13 neurons. Table 1 clearly shows that smaller MSE is achieved in fewer iterations by the SCGA when compared with the gradient algorithm.

Because the performance of the gradient algorithm was inferior to the SCGA, it was decided to use the SCGA for both the delta and the partial differential methods, so that the derivatives estimated can be compared against the same benchmark. As a typical set of results, in Table 2, the average values and the RSTD of the parameters using the delta method and the PDM, along with those obtained using the EEM, are provided.

Table 1 MSE at the end of various number of iterations for gradient method and scaled conjugate gradient method

	MSE	
Number of iterations	Gradient learning [3]	Scaled conjugate learning [16]
1	5.27×10^{-3}	6.71×10^{-5}
500	6.29×10^{-4}	4.49×10^{-6}
5000	8.50×10^{-5}	3.40×10^{-7}
15,000	4.50×10^{-5}	——

The delta method and the PDM give comparable results. In Table 2, rounded average and RSTD values are given. The RSTD for the PDM and the delta method differ after four decimal places. This small difference may be attributed to the fact that the delta method merges the effect of higher-order derivatives in the first-order derivative, like the EEM; therefore, it may produce slightly less standard deviation as compared with the PDM. However, theoretically, PDM gives the first-order derivatives accurately. Unlike the classical methods, the NN-based methods produce derivatives that vary at different data points due to their dependence on the motion and control variables, as explained in the previous section. The higher-order derivatives can be analytically determined and then programmed.

It is obvious from Table 2 that performance of the neural methods is as good as the EEM. In Figs. 2 and 3, the measured and the NN predicted rolling moment and yawing moment coefficients are plotted against time. Similar results were obtained for the side force coefficient also. However, for the sake of brevity, the plots related to the side force coefficients and the derivatives are not included. It can be observed from Figs. 2 and 3 that the effect of elevator deflection on the rolling and yawing moments is quite weak, as compared with the aileron and rudder deflections. This implies that there is not much coupling between the longitudinal and the lateral-directional modes. This is also evident from the fact that the constant coefficients C_{Y_0}, C_{l_0}, and C_{n_0} (Table 2) obtained from the EEM or the PDM are quite small, which in fact absorb the very weak coupling between the longitudinal and the lateral-directional modes. Therefore, in this paper, results for only lateral-directional modeling is presented. However, NN will equally work if the longitudinal motion variables are included, but the results will show only negligible coupling. Figures 4 and 5 show the rms values of the stability and control derivatives against the number of iterations, truncated to 2500. It is obvious from these figures that the rms values of these derivatives converge as the training proceeds. In fact, the convergence is very fast and stabilizes within 500 iterations.

The most important observation that can be extracted from these figures is the consistency of the results obtained. Strong dependence of the rolling moment on the roll rate is obvious from the curve for C_{l_p} in Fig. 4. Similarly, other derivative appearances in Figs. 4 and 5 are consistent with their weak or strong dependence on the motion or the control variables. In Figs. 6 and 7, the estimated values of the derivatives at different data points are plotted. These figures clearly show minor fluctuations in the estimated derivatives when the aircraft motion is excited using different control inputs, indicating

Table 2 Average (AVG) and RSTD of estimated derivatives of side force coefficient, rolling moment coefficient, and yawing moment coefficient, using Partial differential, delta, and equation error methods

	PDM		Delta method		EEM	
Parameters	AVG	RSTD, %	AVG	RSTD, %	AVG	RSTD, %
C_{Y_0}	−0.0071	5.95	−0.0071	20.78	−0.0071	0.70
C_{Y_β}	−1.0557	1.68	−1.0552	1.68	−1.0483	0.20
C_{Y_p}	0.2027	1.68	0.2027	1.68	0.2058	2.81
C_{Y_r}	0.6116	1.68	0.6117	1.68	0.6157	2.57
$C_{Y_{\delta a}}$	0.0080	1.68	0.0080	1.68	0.0083	12.93
$C_{Y_{\delta r}}$	0.1902	1.68	0.1902	1.68	0.1909	1.79
C_{l_0}	−0.0002	40.97	0.0002	211.21	−0.0002	7.18
C_{l_β}	−0.1134	0.49	−0.1134	0.49	−0.1127	0.74
C_{l_p}	−0.7584	0.49	−0.7581	0.49	−0.7557	0.30
C_{l_r}	0.2809	0.49	0.2809	0.49	0.2866	2.18
$C_{l_{\delta a}}$	−0.1934	0.49	−0.1934	0.49	−0.1928	0.22
$C_{l_{\delta r}}$	0.0442	0.49	0.0442	0.49	0.0439	3.06
C_{n_0}	0.0029	4.75	0.0029	21.01	0.0029	0.71
C_{n_β}	0.2587	2.39	0.2585	2.40	0.2572	0.34
C_{n_p}	−0.0912	2.39	−0.0912	2.39	−0.0921	2.57
C_{n_r}	−0.1252	2.39	−0.1253	2.39	−0.1265	5.13
$C_{n_{\delta a}}$	−0.0118	2.39	−0.0118	2.39	−0.0119	3.74
$C_{n_{\delta r}}$	−0.1419	2.39	−0.1420	2.39	−0.1430	0.98

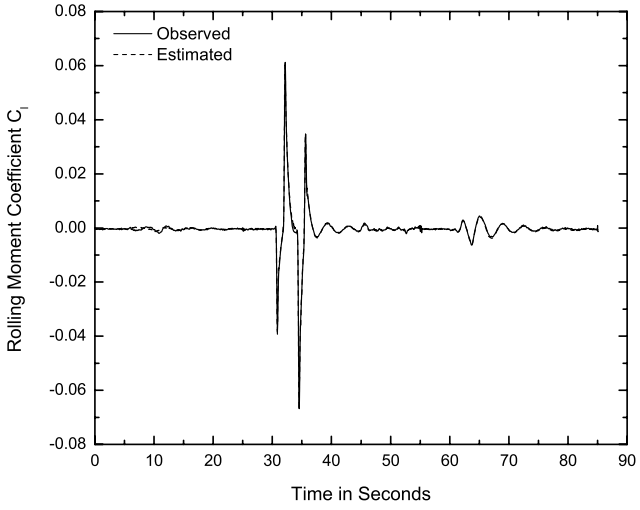

Fig. 2 Rolling moment coefficient C_l, measured and predicted using NN.

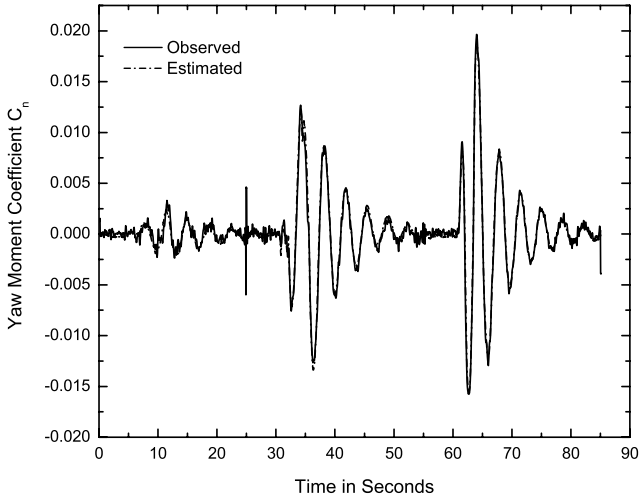

Fig. 3 Yawing moment coefficient C_n, observed and predicted using NN.

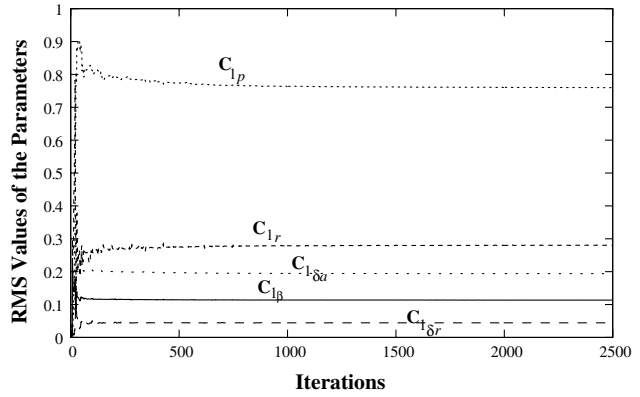

Fig. 4 RMS values of rolling moment derivatives truncated up to 2500 iterations.

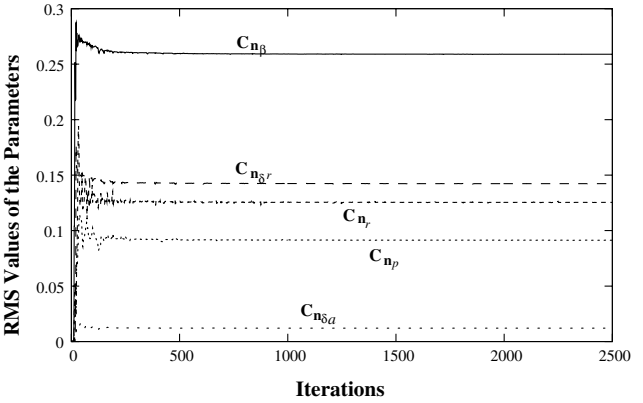

Fig. 5 RMS values of yawing moment derivatives truncated up to 2500 iterations.

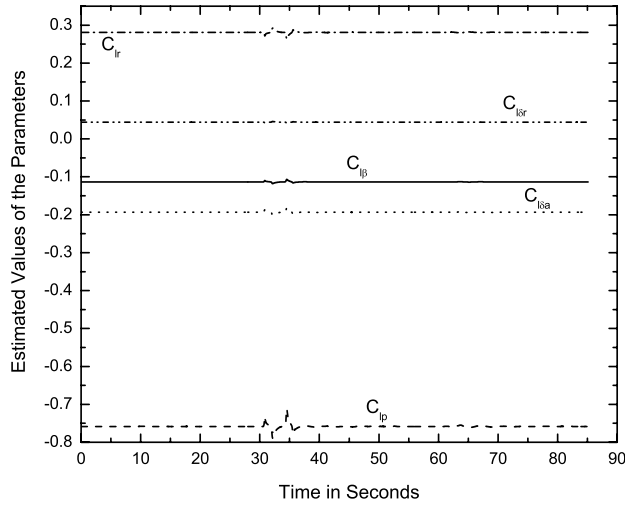

Fig. 6 Estimated values of the rolling moment derivatives over the entire training set, using PDM.

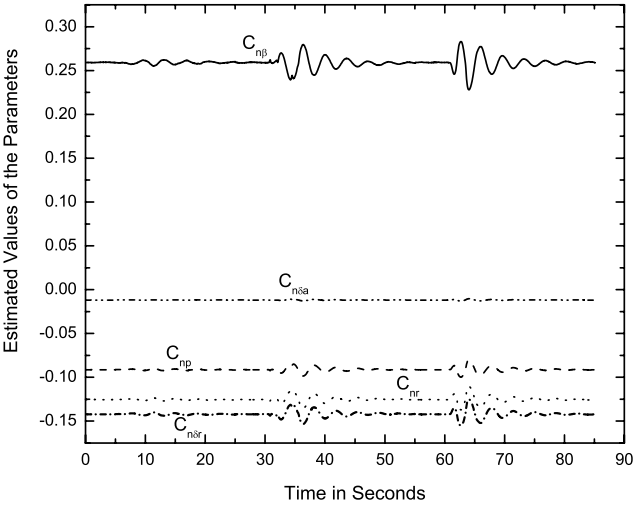

Fig. 7 Estimated values of the yawing moment derivatives over the entire training set, using PDM.

dependence of the derivatives on the motion variables. This is not reflected in the EEM, because of assumption that a linear relation between the motion variables and the forces or moments exists.

It can be shown through simulation, and it can be proved analytically, that any offset of the estimated value with the actual value will only be due to the imperfect learning. The result is neither affected by the presence of bias in the measured lateral-directional forces and moments nor the presence of bias in the measured motion/control variables. Thus, from the results, it is observed that parameters estimated using the PDM shed light on the dependency of

the stability and control derivatives on the motion variables. In addition, this method does not require any initial estimate of the parameters to start the estimation process, as it avoids any integration of the equation of motion.

In Figs. 8 and 9, rolling moment and yawing moment coefficients, reconstructed using derivatives estimated through the PDM, are plotted against time, along with the NN predicted coefficients and the measured coefficients. The reconstructed curve shows a very good match between the neural predicted and the measured values, showing confidence in the NN estimated derivatives. In Figs. 10 and 11, rolling moment and yawing moment coefficients, which are reconstructed using derivatives estimated through the EEM, predicted using the NN, and measured, are plotted against time. From Figs. 8 and 10, it can be observed that the reconstructed curve of C_l, using derivatives extracted through the PDM, retraces the measured and the NN predicted curves for C_l, and the same is true for the reconstructed curve of C_l obtained using the EEM. From Figs. 9 and 11, it is obvious that C_n, computed using the derivatives obtained through the PDM and the EEM, retraces the measured curve exactly.

The present analysis sheds light on the utility of the PDM for the estimation of stability and control derivatives, especially for the case in which the aerodynamics of the aircraft are highly nonlinear, which rules out the use of any presumed structure of aerodynamic model for data processing using the EEM. In such cases, applicability of the output error and the filter error methods also become restricted, unlike the PDM, because they require the initial estimates of the derivatives to start the estimation process. Comparison of the reconstructed curves, using the derivatives obtained through the PDM and the EEM, is given in Figs. 12 and 13. To the visual resolution provided, both the curves overlap in these two figures, indicating consistency of the results obtained using the NN.

Thus, it may be stated that the performance of the PDM for estimation of the stability and control derivatives is comparable with the other classical methods and, therefore, provides an alternative tool that does not even require initial estimates of the derivatives to

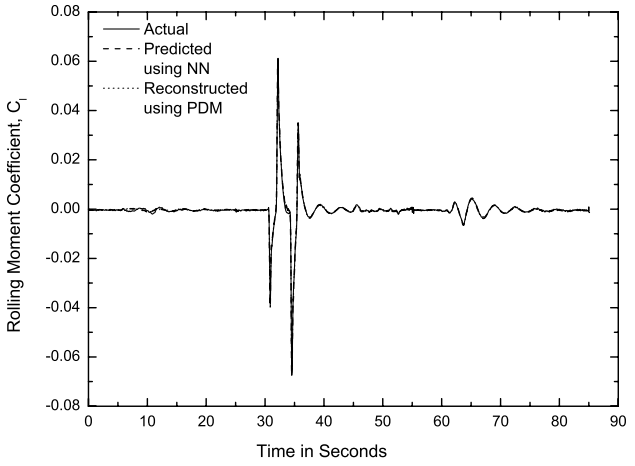

Fig. 8 Reconstructed values of the rolling moment coefficient based on derivatives estimated using PDM.

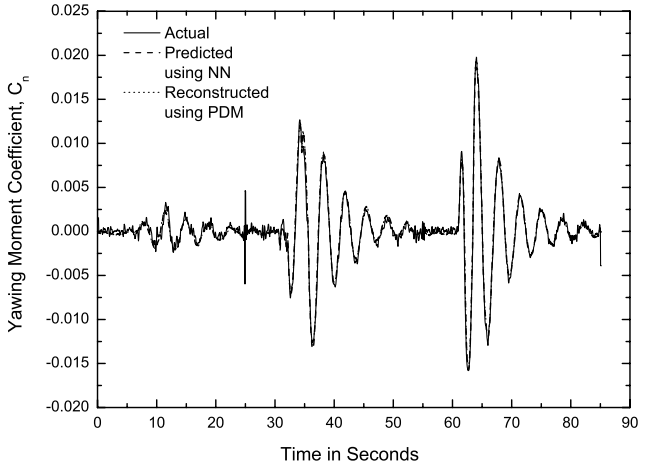

Fig. 9 Reconstructed values of the yawing moment coefficient based on derivatives estimated using PDM.

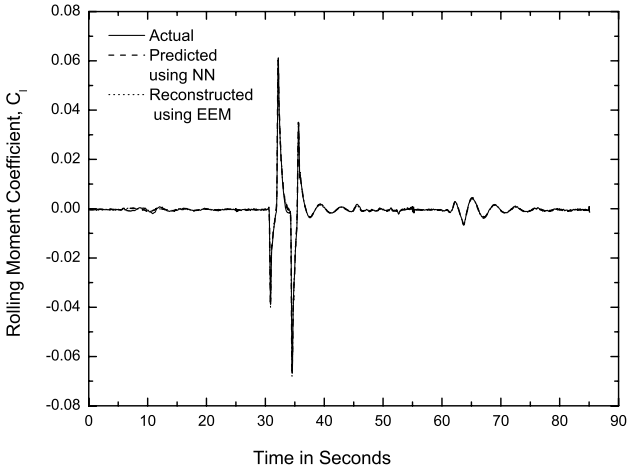

Fig. 10 Reconstructed values of the rolling moment coefficient based on derivatives estimated using EEM.

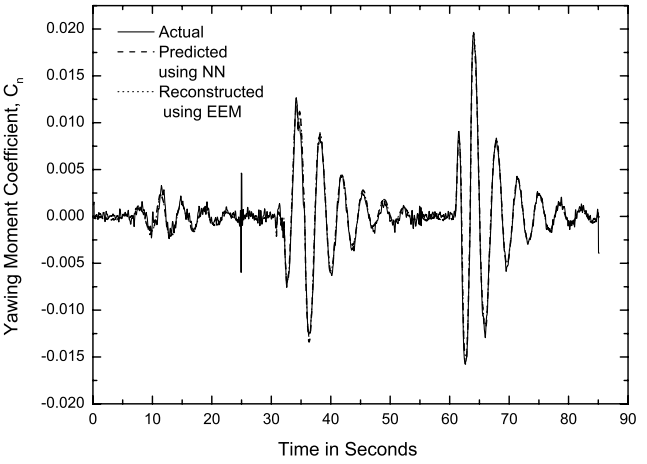

Fig. 11 Reconstructed values of the yawing moment coefficient based on derivatives estimated using EEM.

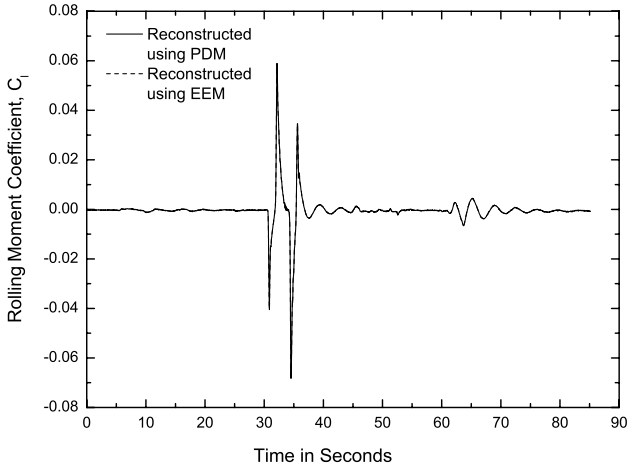

Fig. 12 Comparison of reconstructed values of the rolling moment coefficient based on derivatives estimated using PDM and EEM.

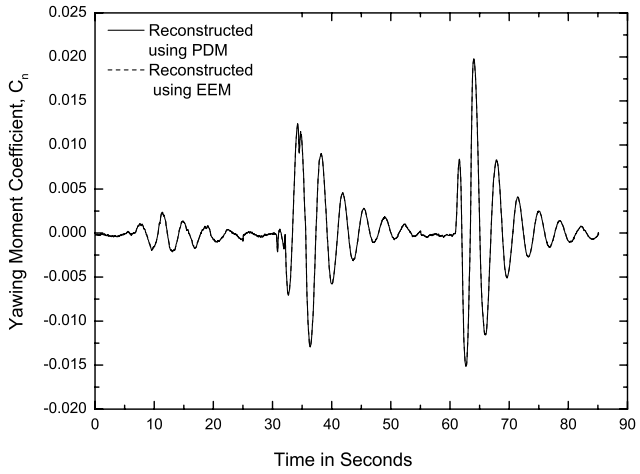

Fig. 13 Comparison of reconstructed values of the yawing moment coefficient based on derivatives estimated using PDM and EEM.

start the estimation process. Although the presented results are comparable for a simple test case, the NNs have capabilities to map highly nonlinear phenomenon; in which case, the linear model identified using the EEM may not be adequate. In such cases, the NN-based partial differential approach yields not only good mapping, but it also provides linearized derivatives. The PDM does not involve any integration of the equation of motion, unlike the output error and the filter error methods; therefore, it can be applied for the parameter estimation of unstable aircraft, especially at a high angle of attack, for which there are highly nonlinear aerodynamics.

IV. Conclusions

A NN approach has been applied to model the aerodynamic force and moment coefficients pertaining to the lateral-directional motion of an aircraft from recorded flight data. The approach yields a good match between the flight measured and the NN predicted outputs. Investigations confirmed the fact that the SCGA performs better when compared with the simple gradient algorithm. A novel method based on neural partial differentiation to estimate aircraft stability and control derivatives from a trained NN was proposed. This proposed approach was compared with the hitherto procedures based on finite differences (i.e., the delta method); the two approaches yield comparable values for the stability and control derivatives. The partial differential approach provides the exact values of the first-order stability and control derivatives, whereas the delta method is just an approximation to the PDM. The amount of data processing time required for the PDM is quite small, as compared with the delta method that requires a lot of posttraining data processing. Therefore, the PDM also saves a lot of posttraining effort, unlike the delta method that involves perturbing the inputs and computation of the derivatives using a finite difference approach. Moreover, the PDM can be used to extract higher-order derivatives, which sheds light on the aerodynamic characteristics of an aircraft, unlike the delta method that absorbs all the higher-order derivatives into the first-order derivatives. The NN estimated derivatives were also validated against reconstructed side force and moment coefficients. The reconstructed coefficient plots showed a very good match between the neural estimated and measured values, showing a high level of confidence in the extracted model. In addition, the accuracy of the neural estimated aerodynamics derivatives was found to be as good as that using the EEM. However, the NN-based PDM promises to be more readily extendable to highly nonlinear aerodynamic phenomenon, whereas the linear models assumed in the EEM may not be suitable.

Acknowledgment

This work is supported by Extramural and Intellectual Property Rights, Defence Research Development Organization, India.

References

[1] Hamel, P. G., and Jategaonkar, R. V., "Evolution of Flight Vehicle System Identification," *Journal of Aircraft*, Vol. 33, No. 1, Jan–Feb 1996, pp. 9–28. doi:10.2514/3.46898

[2] Iliff, K. W., "Parameter Estimation for Flight Vehicle," *Journal of Guidance, Control, and Dynamics*, Vol. 12, No. 5, 1989, pp. 609–622. doi:10.2514/3.20454

[3] Jategaonkar, R. V., "Flight Vehicle System Identification: A Time Domain Methodology," *Progress in Astronautics and Aeronautics*, 1st ed., Vol. 216, AIAA, Reston, VA, 2006.

[4] Hornik, K., Stinchcombe, M., and White, H., "Multilayer Feedforward Networks Are Universal Approximators," *Neural Networks*, Vol. 2, No. 5, 1989, pp. 359–366. doi:10.1016/0893-6080(89)90020-8

[5] Hess, R. A., "On the Use of Back Propagation with Feedforward Neural Networks for Aerodynamic Estimation Problem," AIAA Paper 93-3638, Aug. 1993.

[6] Youseff, H. M., "Estimation of Aerodynamic Coefficients using Neural Networks," AIAA Paper 93-3639, Aug. 1993.

[7] Linse, D. J., and Stengel, R. F., "Identification of Aerodynamic Coefficients using Computational Neural Networks," *Journal of Guidance, Control, and Dynamics*, Vol. 16, No. 6, 1993, pp. 1018–1025. doi:10.2514/3.21122

[8] Basappa, K., and Jategaonkar, R. V., "Aspects of Feedforward Neural Network Modeling and its Application to Lateral-Directional Flight Data," DLR Internal Rept. 111-95/30, Braunschweig, Germany, Sept. 1995.

[9] Raisinghani, S. C., Ghosh, A. K., and Kalra, P. K., "Two New Techniques for Aircraft Parameter Estimation Using Neural Networks," *Aeronautical Journal*, Vol. 102, No. 1011, 1998, pp. 25–29.

[10] Raisinghani, S. C., Ghosh, A. K., and Khubchandani, S., "Estimation of Aircraft Lateral-Directional Parameters Using Neural Networks," *Journal of Aircraft*, Vol. 35, No. 6, Nov.–Dec. 1998, pp. 876–881. doi:10.2514/2.2407

[11] Singh, S., and Ghosh, A. K., "Parameter Estimation from Flight Data Using Maximum Likelihood Method And Neural Network," AIAA Paper 2006-6284, 2006.

[12] Jurada, J. M., Malinowski, A., and Cloete, I., "Sensitivity Analysis for Minimization of Input Data Dimension for Feedforward Neural Network," *IEEE International Symposium on Circuits and Systems*, Vol. 6, IEEE Publications, Piscataway, NJ, 30 May–2 June 1994, pp. 447–450.

[13] Sola, J., and Sevilla, J., "Importance of Data Normalization for the Application of Neural Networks to Complex Industrial Problems," *IEEE Transactions on Nuclear Science*, Vol. 44, No. 3, 1997, pp. 1464–1468. doi:10.1109/23.589532

[14] Leeghim, H., Seo, I.-H., and Bang, H., "Adaptive Nonlinear Control Using Input Normalized Neural Networks," *Journal of Mechanical Science and Technology*, Vol. 22, No. 6, 2008, pp. 1073–1083. doi:10.1007/s12206-007-1119-1

[15] Sinha, M., Kumar, K., and Kalra, P. K., "Some New Neural Network Architectures with Improved Learning Schemes," *Soft Computing*, Vol. 4, No. 4, Dec. 2000, pp. 214–223. doi:10.1007/s005000000057

[16] Moller, M. F., "A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning," *Neural Networks*, Vol. 6, No. 4, 1993, pp. 525–533. doi:10.1016/S0893-6080(05)80056-5

[17] Johansson, E. M., Dowla, F. U., and Goodman, D. M., "Backpropagation Learning for Multilayer Feedforward Neural Networks Using the Conjugate Gradient Method," *International Journal of Neural Systems*, Vol. 2, No. 4, 1991, pp. 291–302. doi:10.1142/S0129065791000261

[18] Watrous, R., and Shastri, L., "Learning Phonetic Features using Connectionist Networks: An Experiment in Speech Recognition," *Proceedings of the 1st IEEE International Conference on Neural Networks*, IEEE Publications, Piscataway, NJ, June 1987, pp. 851–854.

[19] Hamel, P. G., Jategaonkar, R. V., and Doherr, K. F., "Identification of the Aerodynamic Model of the DLR Research Aircraft ATTAS from Flight Test Data," DLR Rept. DLR-FB 90-40, Braunschweig, Germany, 1990.

[20] Jategaonkar, R. V., "Flight Vehicle System Identification: A Time Domain Methodology," *Progress in Astronautics and Aeronautics*, 1st ed., Vol. 216, AIAA, Reston, VA, 2006, pp. 204–207.