UNSTABLE AIRCRAFT PARAMETER ESTIMATION USING NEURAL PARTIAL DIFFERENTIATION

R.A. Kuttieri
Senior Engineer
ARDC, Hindustan Aeronautics Ltd
Bangalore-560 037, India
Currently Research Scholar
Department of Aerospace Engineering
Indian Institute of Technology Khargpur
Kharagpur-721 302, West Bengal, India
Email : akrnambiar@gmail.com

M. Sinha
Assistant Professor
Department of Aerospace Engineering
Indian Institute of Technology Kharagpur
Kharagpur-721 302, West Bengal, India
Email : ms.aissq@gmail.com

Abstract

An approach based on neural partial differentiation is suggested, to overcome the numerical problems faced by classical methods, for the parameter estimation of an aerodynamically unstable aircraft. Theoretical analysis of the neural modeling, the parameter estimation process, and the nature of the estimates pertaining to unstable aircraft dynamics using the neural partial differential method, are discussed. Equation for the relative standard deviation, which is equivalent to the Cramer-Rao bound in the method like output error approach, is derived using the neural partial differential method and verified through numerical simulation. The aerodynamic derivatives are derived for the simulated and real longitudinal flight data of an unstable aircraft, and the estimates obtained using the neural partial differentiation are compared with the classical methods such as the equation error and the output error methods. The parameter estimates from the simulated noisy data are also presented to assess and support the theoretical developments presented in this paper. The theoretical analysis and the results presented in this paper make the neural partial differential approach more reliable and widely applicable.

Key Words*: neural network, neural partial differentiation, unstable aircraft*

Introduction

The process of extracting the stability and control derivatives from flight measured data is known as aircraft parameter estimation. Accurate knowledge of stability and control derivatives plays a vital role in the development life cycle of a flight vehicle. The parameter estimation is a necessary effort that leads to accurate and validated mathematical models. The aircraft parameter estimation has important roles in the areas such as: verification of wind tunnel and analytical predictions, development of high fidelity aerodynamic databases for flight simulators and off-line digital simulations, design of flight control laws and stability augmentation systems, analysis of handling quality specifications, support for flight envelope expansion, and design of adaptive flight control systems [1]. Therefore, in spite of the availability of the parameters of an aircraft from wind tunnel measurements, estimation using flight data becomes inevitable. Over the last four decades, various estimation techniques were developed and successfully applied to the aircraft parameter estimation problem. These classical estimation methods can be broadly categorized as the equation error, the output error, and the filter error methods [2, 3]. The classical methods require a priori knowledge of the dynamic model and the initial estimate of the parameters.

The classical methods are quite efficient to meet the requirements of stable aircraft aerodynamic charac-

Paper Code : V64 N3/770-2012. Manuscript received on 27 Jan 2012. Reviewed and accepted as a Full Length Contributed Paper on 12 Jul 2012

teristics. However, application of the output error method poses divergence problem in the numerical integration of the unstable aircraft dynamics [4]. The filter error methods which are based on the principle of predictor-corrector approach have inherent stability in numerical propagation of the states, as they utilize measurements to update the states after the prediction step. Therefore, the filter error method can be applied successfully for the parameter estimation of the unstable aircraft. However, its implementation is complicated and it requires setting up of the state equations for various parameters to be estimated, in addition to the state equations for the system dynamics.

In order to overcome the problems with the classical approaches, neural networks were used for the aircraft parameter estimation. A neural network can learn from the input-output pairs and can provide reliable function approximation [5]. This makes it suitable for the aircraft parameter estimation, where the forces and moments are directly represented in terms of a linear or nonlinear function of the motion and control variables. An early investigation into the use of neural network for aircraft parameter estimation was reported in [6]. In [6], feed forward neural network was used as a mapping function to map the inputs (state and control variables) to the outputs (aerodynamic forces and moments). Similar works were reported in [7, 8, and 9]. In most of these cases, gradient learning algorithm was used to train the neural network. In [8], aerodynamic coefficient models are derived from simulated flight data using system identification model composed of extended Kalman Bucy filter for state and force estimation and aerodynamic modeling has been done by computational neural network with first order partial derivatives for weight estimation. Feteih [10] applied a neural network to find the aerodynamic coefficients of an unstable aircraft reported by Hess [6]. The use of a recurrent neural network which has self-feedback to approximate the dynamics of a system was reported for the parameter estimation [11], but the accuracy achieved was not good.

More noted approaches to obtain the aircraft stability and control derivatives from flight data, using neural network based delta and zero methods, were reported in [12, 13]. Both these methods apply finite differencing to obtain the stability and control derivatives in the post training phase. As an extension of the work, the modified delta method [14] was reported. A combination of the neural network and the Gauss Newton method [15] can also be applied to estimate the stability and control derivatives. In this method, the neural network is used to propagate the state of the aircraft, which is followed by application of Gauss Newton method to estimate the stability and control derivatives by minimizing a suitable optimization function. However, all these methods employ finite differencing to obtain the stability and control derivatives in post neural-training phase. These methods require additional data processing in the post training phase in order to extract the aerodynamic derivatives. Use of radial basis function neural network for helicopter parameter estimation from flight data was reported in [16] using post training finite differencing to estimate the parameters. However, radial basis function method requires more number of hidden neurons and it is slower than sigmoid nonlinearity based neural network [17].

In order to avoid the need of post training data processing and to improve the accuracy of estimation, neural partial differential method [18] was reported for the estimation of the lateral-directional stability and control derivatives from real flight data of an aerodynamically stable aircraft. The results obtained using the neural partial differential method showed superior performance as compared to those obtained using the zero, the delta, and the classical equation error methods on real flight data. However, the work presented [18] did not show theoretical developments on the accuracy and nature of the estimates obtained using the neural partial differential approach.

Unlike the other neural methods reported in literature, the neural partial differential approach can give theoretical insight into the statistical information such as the relative standard deviation, which is equivalent to the Cramer-Rao bound in the output error method. In this work, the nature of the relative standard deviation of the estimates, obtained from the neural partial differential approach, is theoretically developed and later proved through numerical studies on the real and simulated flight data of unstable aircraft. The parameter estimation, pertaining to the longitudinal motion, of the unstable X-31A aircraft (open loop) from the real flight data is presented. The results of estimation on the real and simulated data obtained using the neural partial differential approach are compared with those obtained using the equation error method and the artificially stabilized output error method [4]. The results of estimation for an unstable aircraft, from two simulated noisy data set, are also presented to show the effect of noise on the accuracy of the estimates and to support the theoretical developments elaborated in the next section.

Methodology

Mathematical formulation of a multilayer neural network, for estimating the stability and control derivatives of an unstable aircraft, using the neural partial differential method is discussed in this section. Mathematical formulation of the standard deviation and relative standard deviation is presented for better understanding of the nature of the estimates. The effect of noise on the derivatives is theoretically investigated in this section and later simulated in the results section. The architecture of the neural network to be used for various problems is also discussed.

Neural Network

The schematic of a three layer feedforward neural network, henceforth to be referred as a neural network, is shown in Fig. (1). It consists of two hidden layers and one output layer of neurons wherein the output layer neurons contain summation function only, i.e. these are free from activation function. The input and the output vectors of the neural network are represented as $\tilde{\chi} \in \Re^{n+1}$ and $\tilde{Z} \in \Re^{k}$ respectively. Let the first and the second hidden layers augmented output vectors (containing bias) be represented as $\tilde{X} \in \Re^{m+1}$ and $\tilde{Y} \in \Re^{l+1}$ respectively. The output of the neural network can be written as $\tilde{Z} = \tilde{W}^T \ \tilde{Y}$, where $\tilde{W}$ is the augmented weight matrix connecting the second hidden layer to the output layer and is defined as

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

The vector $\tilde{Y}$ can be expressed as $\tilde{Y} = \tilde{f}(\tilde{V}^T \ \tilde{x}) = \tilde{f}(\tilde{v})$,

where $f(.)$ is tangent hyperbolic function which serves as a nonlinear activation function as it provides better result [19]; and it is represented as

$$f(v_i) = \frac{1 - e^{-\lambda v_i}}{1 + e^{-\lambda v_i}}$$

The weight matrix $\tilde{V}$ can be represented as

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

Similarly, the output of the 1st hidden layer neurons can be written as $\tilde{X} = \tilde{g}(\tilde{U}^T \ \tilde{\chi}) = \tilde{g}(\tilde{\beta})$, where $g(.)$ is the nonlinear activation function (tangent hyperbolic). The weight matrix $\tilde{U}$ is defined as

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

where b_{um} is the m[th] element of the weights related to the bias of the input layer. The input to the neural network denoted by the augmented input vector $\tilde{\chi}$ can be defined as $\tilde{\chi} = \begin{bmatrix} -1 & \tilde{a}^T \end{bmatrix}^T = \begin{bmatrix} \chi_0 & \chi_1 & \cdots & \chi_n \end{bmatrix}^T$ 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 the following equation,

$$Z_{i,norm} = Z_{i,norm_{\min}} + \left(Z_{i,norm_{\max}} - Z_{i,norm_{\min}} \right) * \frac{Z_i - Z_{i,\min}}{Z_{i,\max} - Z_{i,\min}}$$

where $Z_{i,norm_{\max}}$ and $Z_{i,norm_{\min}}$ denote the upper and the lower limits of normalization range for the Z_i respectively; while $Z_{i,\max}$ and $Z_{i,\min}$ represent the maximum and the minimum values of Z_i respectively. $Z_{i,norm}$ is the normalized value of the output variable Z_i. In this paper, $Z_{i,norm_{\max}} = 0.9$ and $Z_{i,norm_{\min}} = -0.9$, and therefore, the normalization equation maps the outputs in the range [-0.9, 0.9]. The output vector of the neural network can be expressed as

$$\tilde{Z} = \tilde{W}^T \ \tilde{f}(\tilde{V}^T \ \tilde{g}(\tilde{U}^T (\tilde{\chi}))).$$

The cost function, to optimize the weights of the neural network during the learning process, is taken as the mean square error (MSE) function as given below:

$$MSE = E(\tilde{\omega}) = \frac{1}{2P}\sum_{j=1}^{p}\sum_{i=1}^{k}(d_i - Z_i)_j^2$$

Here, $\tilde{\omega}$ is a vector consisting of all the weights of the neural network; E is the error function; P is the total data points (number of observations/patterns); d_i is the i^{th} normalized measured value of the output; Z_i is the normalized computed value of the output, and the index j stands for the j^{th} data point. Since the neural network training is done in batch mode, therefore, MSE is computed at the end of every step of iteration using updated values of the weights. The neural network training is carried out using the back propagation method and the scaled conjugate gradient algorithm (SCGA) [20] to minimize the MSE.

The Neural Partial Differentiation

This methodology is based on the partial differential of the neural output and is performed analytically at the end of training using the terms computed during the training. Therefore, this does not require post training neural processing unlike the Delta method [12] and its variants. The basic assumption is that, a trained neural network represents the mapping function between the inputs and outputs explicitly. Hence, the outputs can be differentiated analytically to yield the first order derivatives. If the training is proper then it is possible to find the higher order derivatives. However, the derivatives thus computed will deteriorate in accuracy due to the imperfect training as the order is increased. This is due to the fact that the neural network is only an approximation of the function which it tries to learn from the input-output pairs. Therefore, due to the errors in the input-output data, the derivatives will fall away from the actual values as the order increases.

For the current application, aircraft motion and control variables are the inputs, and the aerodynamic forces and moments are the outputs for the neural network. The partial differential of the neural output $\tilde{Z}$ with respect to the augmented input vector $\tilde{\chi}$ can be written using the chain rule of partial differentiation as

$$\left[\frac{\partial \tilde{Z}}{\partial \tilde{\chi}}\right] = \left[\frac{\partial \tilde{Z}}{\partial \tilde{Z}_{norm}}\right]\left[\frac{\partial \tilde{Z}_{norm}}{\partial \tilde{Y}}\right]\left[\frac{\partial \tilde{Y}}{\partial \tilde{X}}\right]\left[\frac{\partial \tilde{X}}{\partial \tilde{\chi}_{norm}}\right]\left[\frac{\partial \tilde{\chi}_{norm}}{\partial \tilde{\chi}}\right]$$

where

$$\left[\frac{\partial \tilde{Z}}{\partial \tilde{\chi}}\right] = \begin{bmatrix} \frac{\partial Z_1}{\partial \chi_o} & \cdots & \frac{\partial Z_1}{\partial \chi_n} \\ \cdots & \cdots & \cdots \\ \frac{\partial Z_k}{\partial \chi_o} & \cdots & \frac{\partial Z_k}{\partial \chi_n} \end{bmatrix}$$

and

$$\left[\frac{\partial \tilde{Z}_{norm}}{\partial \tilde{Y}}\right] = \begin{bmatrix} \frac{\partial Z_1}{\partial Z_{1,norm}} & 0 & \cdots & 0 \\ 0 & \frac{\partial Z_2}{\partial Z_{2,norm}} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \frac{\partial Z_k}{\partial Z_{k,norm}} \end{bmatrix}$$

is a diagonal matrix of size (k x k).

Similarly,

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

$$\left[\frac{\partial \tilde{Z}_{norm}}{\partial \tilde{Y}}\right] = \begin{bmatrix} \frac{\partial Z_{1,norm}}{\partial Y_0} & \frac{\partial Z_{1,norm}}{\partial Y_1} & \cdots & \frac{\partial Z_{1,norm}}{\partial Y_l} \\ \frac{\partial Z_{2,norm}}{\partial Y_0} & \frac{\partial Z_{2,norm}}{\partial Y_1} & \cdots & \frac{\partial Z_{2,norm}}{\partial Y_l} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial Z_{k,norm}}{\partial Y_0} & \frac{\partial Z_{2,norm}}{\partial Y_1} & \cdots & \frac{\partial Z_{n,norm}}{\partial Y_l} \end{bmatrix} = \tilde{W}^T$$

$$\left[\frac{\partial \tilde{Y}}{\partial \tilde{X}}\right] = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & f_1' & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & f_1' \end{bmatrix} \begin{bmatrix} b_{v1} & \cdots & b_{vl} \\ v_{11} & \cdots & v_{1l} \\ \cdots & \cdots & \cdots \\ v_{m1} & \cdots & v_{ml} \end{bmatrix}^T = diag\left[0\ f_1'\ f_2'\ \cdots f_l'\right]\tilde{V}^T,$$

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

The matrices listed above are available during training of the neural network; therefore, do not put extra computational burden unlike the hitherto neural delta method and its variants. The stability and control derivatives can be computed at the end of training without adding the burden of post training data processing or can be computed after each iteration. This saves time and effort unlike that required for the finite difference methods in the post training phase. The derivatives thus obtained at different data points are averaged and their standard deviations are calculated which can be used to compute the relative standard deviations which also corresponds to the Cramer-Rao bound for the non-statistical approaches like the equation error method using equation

$$RSTD = \frac{STD}{Average} \times 100\%$$

where *RSTD* and *STD* are relative standard deviation and standard deviation respectively.

Analysis of the Estimates

The multi-input single output system considered in this paper can be represented as

$$\Gamma(\alpha, q, \delta_e) = A_0 + A_1\alpha + A_2 q + A_3\delta_e + A_4\alpha^2 + A_5 q^2 + A_6\delta_e^2 + \dots$$

where A_i ($i = 0, 1, 2 \dots$) is a coefficient; α is the angle of attack, q is pitch rate, and δ_e is the elevator deflection, which serve as the inputs to the neural network. The neural output is the pitch acceleration $\dot{q}$. The objective is to estimate the coefficient A_i such that in the trained state the neural network output represents this polynomial. This is achieved by minimizing a mean square cost function as described earlier. The neural partial differential method can be applied to the trained neural network to obtain the first and higher order partial derivatives.

If the inputs to the neural network are highly noisy, the estimated coefficients may get biased due to improper learning. Such cases will be obvious from the mean square error which does not decay to the desired level, but may yield small relative standard deviation indicating a biased estimate. A mean square error of the order of 10^{-4} - 10^{-5} may be a practical value on the real data set to get realistic estimates of the derivatives. In the case of low mean square error, the relative standard deviation is a good indication of the quality of estimates. It can be shown that the relative standard deviation of the partial derivative of a neural output Z_k can be written as

$$STD = \frac{\sum_{p=1}^{P}\left[\sum_{m=1}^{M}\left(\sum_{l=1}^{L} Y'_{l_p} v_{lm} w_{kl} Z'_{k_p}\right) X'_{m_p} u_{m_i} - \frac{\sum_{p=1}^{P}\sum_{m=1}^{M}\left(\sum_{l=1}^{L} Y'_{l_p} v_{lm} w_{kl} Z'_{k_p}\right) X'_{m_p} u_{m_i}}{P}\right]^2}{P}$$

where

$$\frac{\sum_{p=1}^{P}\sum_{m=1}^{M}\left(\sum_{l=1}^{L} Y'_{l_p} v_{lm} w_{kl} Z'_{k_p}\right) X'_{m_p} u_{m_i}}{P} = Avg,$$

for one neuron in the first hidden layer (M = 1),

$$STD = \frac{\sum_{p=1}^{P}\left[\left(\sum_{l=1}^{L} Y'_{l_p} v_{lm} w_{kl} Z'_{k_p}\right) X'_{m_p} - \frac{\sum_{p=1}^{P}\left(\sum_{l=1}^{L} Y'_{l_p} v_{lm} w_{kl} Z'_{k_p}\right) X'_{m_p}}{P}\right]^2}{P} u_{m_i}$$

Hence, the relative standard deviation can be written as

$$RSTD = \frac{\sum_{p=1}^{P}\left[\left(\sum_{l=1}^{L} Y'_{l_p} v_{lm} w_{kl} Z'_{k_p}\right) x'_{m_p} - \frac{\sum_{p=1}^{P}\left(\sum_{l=1}^{L} Y'_{l_p} v_{lm} w_{kl} z'_{k_p}\right) x'_{m_p}}{P}\right]^2}{P} \times 100\%$$

$$\frac{\sum_{p=1}^{P}\left(\sum_{l=1}^{L} Y'_{l_p} v_{lm} w_{kl} z'_{k_p}\right) x'_{m_p}}{P}$$

It can be observed in the above equation that the quantities X'_m, Y'_l, and Z'_k are same for all the input variables because m=1, they vary only with patterns (data points). In addition, the weights are fixed quantities after the training is over. It can be observed that the input layer weights do not appear in the RSTD equation. Thus, the relative standard deviations of the partial derivatives for an output with respect to the inputs will be the same, provided there is only one neuron in the first hidden layer. One neuron in the first hidden layer also ensures low standard deviation by allowing a linear model for the input-output data.

The derivatives are not affected by the existence of the measurement bias either in the inputs or the outputs due to the partial differential approach. The bias in the inputs and the output get eliminated due to partial differentiation. If the input data is highly contaminated with noise then the training will not be good which will be reflected in the large mean square error at the end of iteration. Therefore, even with one neuron in the first hidden layer, the estimated derivatives may be away from the actual value indicating biased estimates. For the cases where the training is not good, the output reconstructed using the estimated derivatives will not match with the measured output implying poor estimation of the derivatives. In addition, the match between the actual and the neural predicted values of the output may be poor. Thus, it can be concluded analytically that biased estimation is due to the poor learning which arises from highly noisy input data. When the mean square error at the end of training is quite low, the higher degree terms will be negligible yielding

$$\sum_{p=1}^{P} \frac{\partial Z}{\partial \alpha} = PA_1$$

which results in

$$A_1 = \frac{1}{P} \sum_{p=1}^{P} \frac{\partial Z}{\partial \alpha}$$

implying almost exact value of the coefficient. The coefficient thus obtained will match with the value obtained by setting all the neural inputs to zero, i.e.

$$A_1 = \frac{\partial Z}{\partial \alpha} \Big|_{\alpha=0,\, \delta_e=0,\, q=0}$$

Results and Discussion

In this section, the neural partial differential method is applied to the real flight data from the X-31A aircraft and the simulated data for the test aircraft de Havilland DHC-2 "BEAVER" to estimate the stability and control derivatives for the longitudinal mode.

As a first example, the flight data is generated in open loop for the simplified model of the test aircraft de Havilland [4, 21] DHC-2 "BEAVER" and processed for the evaluation of the methods developed in the last section. The simulated flight data, spanning over 11.5 seconds, corresponds to an overall unstable system which is achieved by the adjustment of static stability parameter $\dot{q}_w$ [4]. One of the eigen values of the system matrix for the system described by the following state equations is positive and it corresponds to unstable mode.

$$\dot{q} = \dot{q}_w w + \dot{q}_q q + \dot{q}_{\delta_e} \delta_e ,$$

$$\dot{w} = Z_w w + (U_o + Z_q) q + Z\delta_e \delta_e ,$$

The observation equations are

$$N_z = Z_w w + Z_q q + Z\delta_e \delta_e ,$$

$$w = w ,$$

$$q = q ,$$

$$\dot{q} = \dot{q}_w w + \dot{q}_q q + \dot{q}\delta_e \delta_e ,$$

where w is the velocity along the z -body axis, δ_e is the elevator deflection, q is the pitch rate, N_z is the normal acceleration, and U_o is the initial forward speed. The parameters Z_w, Z_q, $Z_{\delta e}$, $\dot{q}_w$, $\dot{q}_q$, and $\dot{q}_{\delta e}$ are to be estimated from the simulated observation data. The nominal values of the coefficients along with the elevator input [1] are taken and the state equations are integrated using the fourth order Runge-Kutta algorithm in FORTRAN to generate the state variables w and q. The stability and control derivatives Z_w, Z_q, $Z_{\delta e}$, $\dot{q}_w$, $\dot{q}_q$, and $\dot{q}_{\delta e}$ are estimated using the neural partial differential method. Moreover, the equation error and the stabilized output error methods are also applied to the simulated data to estimate the stability and control derivatives. The number of neurons used in the first hidden layer is one, while in the second hidden layer and the output layer three and one respectively (1-3-1 architecture). The output layer neuron consists of a summation function only, i.e. no non-linearity.

The values of the stability and control derivatives estimated using the neural partial differential approach from the simulated data (without noise) are compared with the nominal, the equation error method, and the stabilized output error method values. The results are presented in Table-1. The identification of open loop system is carried out by treating the control surface deflection as the neural input. The neural network is trained for 5000 iterations. It can be observed from Table-1 that the parameters estimated using the neural partial differential approach closely match with the nominal values and the equation error method estimates. The relative standard deviations

Table-1 : Stbility and Control Derivatives Estimated Using the Equation Error Method (EEM), Neural Partial Differential (NPD), and Stabilized Output Error (SOEM) Methods

Parameters	NV [4]	EEM	SOEM		NPD		
		EST	EST	RSTD (%)	AVG	STD	RSTD (%)
Z_w	-1.4249	-1.4249	-1.4268	0.0090	-1.4246	0.00080	0.05657
Z_q	-1.4768	-1.4768	-1.4768	fixed*	-1.4782	0.00084	0.05657
$Z_{\delta e}$	-6.2632	-6.2632	-6.1711	0.0169	-6.2665	0.00354	0.05657
$\dot{q}_w$	0.2163	0.2163	0.2170	0.0065	0.2162	0.00012	0.05584
$\dot{q}_q$	-3.7067	-3.7067	-3.7201	0.0071	-3.7055	0.00207	0.05584
$\dot{q}_{\delta e}$	-12.7840	-12.7840	-12.8148	0.0087	-12.7808	0.00714	0.05584

(NV = Nominal Value; EST = Estimated; AVG = Average; STD = Standard Deviation; RSTD = Relative Standard Deviation; SOEM = Stabilized Output Error Method; * Z_q is kept fixed at the nominal value for processing)

of the derivatives obtained using the neural partial differential approach, corresponding to the outputs $\dot{q}$ and N_z are according to the conclusions arrived at in the previous section. This confirms the mathematical validity of the derivations carried out in the last section regarding the relative standard deviation of the estimated parameters. The relative standard deviation for the least square method is not given in the Table as this approach yields the exact value. The relative standard deviations for the stabilized output error method seem to be better than the neural approach, however, the estimates of neural approach are closer to the nominal values. The relative standard deviation in the case of neural method is slightly larger than that from the stabilized output error method as it is derived by averaging the derivatives at various data points. A value without standard deviation can be obtained if all the input variables are set to zero. This is demonstrated later for the simulated noisy data.

In Fig.2 the actual and the neural network predicted N_z are plotted showing a good match. In Fig.3 the RMS values of the estimated parameters are plotted against the iteration number. It is clear that the convergence takes place within 200 iterations. However, the fine adjustment in the weights continues as the iteration number increases, which leads to a better estimate of the derivatives that closely match with the nominal and the equation error method values. A plot of the actual values of the stability and the control derivatives, predicted using the neural partial differential approach against the data points, is shown in Fig.4. It can be observed from Fig.4 that the values of the normal acceleration derivatives show very small oscillations. The oscillations occur wherever the input variables are appreciable in magnitude. This is due to the fact that neural network effectively models a polynomial as explained in the last section. The coefficients of higher degree terms in such polynomial are negligible, but add to the derivatives as very small perturbations, at the large values of the input variables, leading to the small oscillations in the estimated values of the derivatives. For the real data, this effect is more prominent as shown later for the flight data of X-31A aircraft. In Figs.5 to 7, results for the pitch acceleration and its derivative with respect to the motion and control variables are shown. It can be observed from Fig.5 that the actual and the neural predicted values of the output $\dot{q}$ match well. Moreover, RMS values of the $\dot{q}$ derivatives converge within 200 iterations as shown in Fig.6. However, further iterations are necessary for the fine adjustments in the weights leading to the correct values of the derivatives. The estimated values of the derivatives ($\dot{q}_{\delta e}$, $\dot{q}_q$ and $\dot{q}_w$) are quite close to the nominal/actual and the equation error method estimates as shown in Table-1. The plots for the estimated derivatives are shown in Fig.7. The results obtained above confirm the theoretical analysis of the previous section proving efficacy and accuracy of the proposed neural approach. Adding more hidden layers in the neural network improves the processing power and system flexibility at the cost of complexity in training algorithm. Moreover, the network with too many hidden neurons becomes over specified and lacks the generalization capability; and the network with too few hidden neurons prevents the system from proper mapping and reduces the robustness.

Evaluation of the proposed neural partial differential approach on the real flight data of the experimental unstable aircraft X-31A is considered as a second example. The

X-31A aircraft was designed for enhanced maneuverability. It is a single-seater fighter configuration with a takeoff gross weight of approximately 16000 lb. The open loop configuration of the basic X-31A aircraft is aerodynamically unstable in the longitudinal axis at low angle of attack. The numerical flight data for this aircraft is unavailable, hence, digitized the real flight data plots in [4] to obtain the numerical data for the estimation purpose. The plots of the state variables and control input (q , α , and δe), and the derived pitch acceleration $\dot{q}$ corresponding to the two consecutive pitch doublets [4] are digitized. Here, q, α, and δe serve as the inputs to the neural network while $\dot{q}$ serves as the output. The direct estimation of the stability and control derivatives $C_{m\alpha}$, C_{mq} , and $C_{m\delta e}$ is not possible as many of the information required to process the data are not available (for example, the altitude which decides the density which in turn decides the dynamic pressure). However, it is possible to estimate the derivatives $\dot{q}_\alpha$, $\dot{q}_q$, and $\dot{q}_{\delta e}$ from the digitized data. In fact, $\dot{q}$ can be written as

$$\dot{q} = \frac{\bar{q}\, S\, \bar{c}}{I_y}\; C_m^{CG} + \frac{1}{I_y} M_{eng}^{CG},$$

where $\bar{q}$, S , and $\bar{c}$ are the dynamic pressure, reference area, and mean aerodynamic chord respectively. I_y is the pitching moment of inertia, C_m^{CG} is the pitching moment coefficient about the center of gravity (C.G.), and M_{eng}^{CG} is the moment due to the thrust of the engine at the center of gravity. If the effect of the engine is neglected then it can be written as

$$\dot{q} = \frac{\bar{q}\, S\, \bar{c}}{I_y}\; C_m^{CG}$$

A simplified model for the moment about the center of gravity under the assumption that the thrust line coincides with the drag line can be written as

$$C_m^{CG} = C_m^{AC} + C_L\, h_{CG},$$

where C_m^{AC} is the pitching moment coefficient about the aerodynamic center (A.C.) [4], h_{CG} is a non-dimensional distance measured from the aerodynamic center (A.C.) to the center of gravity, and C_L is the lift coefficient. A lift L and a moment M^{AC} can be assumed to act at the aerodynamic center [4]. The above equations for pitch rate and pitching moment can be combined to write as

$$\dot{q} = \frac{\bar{q}\, S\, \bar{c}}{I_y} \left(C_m^{AC} + C_L\, h_{CG} \right)$$

The expression for C_m^{AC} [4] can be written as

$$C_m^{AC} = C_m^{*} + C_{m_\alpha}^{AC} (\alpha - \alpha^{*}) + C_{m_q}^{AC} (q - q^{*}) \left(\frac{\bar{c}}{2V} \right) + C_{m\delta_e}^{AC} (\delta_e - \delta_e^{*})$$

where the symbols have their usual meaning and the terms such as α^{*} , δ_e^{*} , and q^{*} correspond to the trim state. For training of the neural network, the trim values need not be computed separately because these are implicitly modeled by the neural network. The canard effectiveness parameter is kept constant [4] as it is highly correlated with the angle of attack and its estimation is not possible. Therefore, it is eliminated from the above equation and its effect can be assumed to be absorbed in the constant C_m^{*}. Nominal values of the derivatives of the parameters C_L and C_m^{AC} are available [4] as given in Table-2. The observation model for estimating the derivatives $\dot{q}_a$, $\dot{q}_q$, and $\dot{q}_{\delta e}$ using the neural partial differential approach can be written as

$$\dot{q} = \dot{q}_o + \dot{q}_\alpha\, \alpha + \dot{q}_q\, q + \dot{q}_{\delta e}\, \delta_e,$$

where $\dot{q}$ is available from the differentiation of the measured pitch rate which serves as the output for the neural network. Here, it is obtained by the digitization of the $\dot{q}$ plot [4] as mentioned earlier. The variables α , q , and δ_e are available from the measurements and serve as the inputs to the neural network. It is possible to correlate these coefficients ($\dot{q}_\alpha$ etc.) with the derivatives $C_{m\alpha}$, C_{mq} , and $C_{m\delta e}$, this allows verification of the estimated coefficients to some extent. The results obtained using the neural partial differential method are analyzed below and supported by various data to show the efficacy of the neural partial differential approach.

A neural network consisting of one neuron in the first hidden layer, three neurons in the second hidden layer, and one neuron in the output layer is used to map the inputs to the output (1-3-1). The output layer neuron is merely a summation function. A total of 180 data points are used to train the neural network, while 100 data points are kept for verification. In fact, such a division of the data set is not

Table-2 : Stability and Control Derivatives of the X-31A Aircraft, Estimated Using the Equation Error Method (EEM), Neural Partial Differential (NPD), and Stabilized Output Error (SOEM) Methods, from the Digitized Flight Data

Parameters	NV [4]	EEM		SOEM		NPD		
		EST	RSTD (%)	EST	RSTD (%)	AVG	STD	RSTD (%)
$\dot{q}_o$	-	-0.350	8.518	-0.326	1.5765	-0.351	-	-
$\dot{q}_\alpha$	-	0.251	57.094	0.111	19.2785	0.252	0.002	0.620
$\dot{q}_q$	-	-0.473	9.296	-0.471	6.4214	-0.475	0.003	0.620
$\dot{q}_{\delta e}$	-	-7.504	1.8444	-7.170	1.0799	-7.526	0.047	0.620
$C_{L\alpha}$	3.057*	2.661*	-	-	-	-	-	-
$C_{L_{\delta e}}$	1.354*	0.863*	-	-	-	-	-	-
$C_{m\alpha}$	0.119*	0.217*	-	0.238	-	-	-	-
C_{mq}	-1.650*	0.137	-	-0.871	-	-	-	-
$C_{m_{\delta e}}$	-0.571*	-0.467*	-	-0.5-4	-	-	-	-

(NV = Nominal Value; AVG = Average; STD = Standard Deviation; RSTD = Relative Standard Deviation; EST = Estimated; * = from Reference [4])

required if only one neuron is used in the first hidden layer. A single neuron in first hidden layer ensures that only a linear model is fitted to the input-output data, provided the mean square error is small. In addition, if the derivatives obtained from the neural partial differential operation are correct then these can be used to reconstruct the output values which should match with the given output values. Therefore, reconstruction of the output can also be used to ensure the proper training of the neural network. Merely, checking on the basis of data division is not the only way of ensuring correct training. In fact, the reconstruction technique is much more reliable than the data division approach. Hence, no data division was applied for the synthetic data case in the first example. However, in the present case the data is highly noisy and the nominal values of the $\dot{q}$ derivatives are not available. Therefore, in order to provide a double check, the data division is applied. As mentioned earlier, α , q , and δ_e are used as the inputs to the neural network while $\dot{q}$ serves as the output. The weights of the neural network are generated using a uniform random number generator in the range -0.1 to +0.1. It is to be noted that the initialization with the different set of weights does not affect the derivatives obtained from the neural partial differential method using the (1-3-1) architecture. This remains valid as long as only one neuron is used in the first hidden layer. Therefore, neither study on the effect of weights initialization is provided nor any averaging of the resulting derivatives is carried out from different runs.

In Table-2, the values of the derivatives $\dot{q}_\alpha$, $\dot{q}_q$, and $\dot{q}_{\delta e}$ obtained using the neural partial differential approach are presented. For the comparison purpose, the least square estimates in combination with the equations error method and the results for the stabilized output error method are also given. It can be observed from the Table-2 that the derivatives from the stabilized output error method register high relative standard deviation. The estimates using the neural partial differential approach seem to be reliable. This will be evident from the graphs, for the reconstructed $\dot{q}$ using the derivatives obtained from the neural partial differential and stabilized output error methods, to be explained in the following paragraph.

The actual and the neural predicted values of $\dot{q}$ for the training data are plotted in Fig.8. In addition, the neural predicted values of $\dot{q}$ on the unseen data (100 data points after 180) are also plotted in the same figure. The root mean square values of the estimated derivatives on the training data, as the training proceeds, are shown in Fig.9 with respect to the iteration number. It can be observed from Fig.9 that around 5500 iterations are required for the convergence. The estimated values of the derivatives are plotted in Fig.10. The neural predicted, reconstructed, and actual values of $\dot{q}$ are plotted in Fig.11. The data points, plotted after 180 in Fig.11 correspond to the reconstructed values of the output on the unseen data (untrained) along with the neural predicted and actual ones. Moreover, in

Fig.12, the reconstructed values of $\dot{q}$ obtained using the derivatives from the neural partial differential approach and the stabilized output error approach are plotted. The curve for the stabilized output error approach does not match at the peak values. Also, its match on the unseen data is also poor as compared to the neural derivatives based reconstructed values. This gives more confidence in the neural approach. It can be observed in Fig.10 that small oscillations in the values of the derivatives are present. This is due to the effect of the coefficients in the polynomial corresponding to the higher degree terms which are quite small; therefore, these coefficients do not affect the estimates except increasing the standard deviation by small amount. Thus, results from the real flight data also confirm the theoretical developments of the last section.

Two more cases are considered for assessing the effect of the noisy input and output data on the parameter estimates. The data is simulated by adding Gaussian white noise with zero mean to the data for the linear case in the first example. The standard deviations of the noise for two different cases are shown in Table-3. The noise generated is eliminated whenever the absolute value lies beyond the 3σ range, where σ is the standard deviation of the noise. The results of neural partial differential approach along with the other methods are presented in Tables-4 and 5. It can be observed that as the noise level increases, the performance of the neural network deteriorates, which was theoretically predicted earlier. However, if all the inputs are set to zero then the estimates (AVG(0)) improve slightly, which is due to the elimination of the terms of higher degree and of the same degree from the other variables. The estimates obtained by setting all the input variables to zero in the neural partial differential approach are better than those obtained using the equation error method. However, in this case the relative standard deviation is not available. Thus, it can be observed that the estimates from the neural partial differential approach get slightly biased and this is true for the equation error method also. The stabilized output error approach also produces biased results, as shown in Tables-4 and 5. Moreover, the relative standard deviation for the stabilized output error method is higher than that for the neural method. It can be concluded that although the neural partial differential approach produces biased results for the noisy data set, the estimates are better than those from the stabilized output error approach. In the stabilized output error case, Z_q needs to be fixed to the nominal/actual value for processing rest of the derivatives. The cases,

Table-3 : Standard Deviation of the Noise to be Added to the Different Input and Output Variables (STD (1) = Standard Deviation for the Less Noisy Data Set, STD (2) = Standard Deviation for the More Noisy Data Set)

Variables	STD (1)	STD (2)
w	0.01 m/s	0.05 m/s
q	0.001 rad/s	0.005 rad/s
δ_e	0.001 rad	0.005 rad
$\dot{w}$	0.001 m/s^2	0.005 m/s^2
$\dot{q}$	0.001 rad/s^2	0.005 rad/s^2

Table-4 : Stability and Control Derivatives Estimated Using the Simple Equation Error Method (EEM), Neural Partial Differential (NPD) Approach, and output Error Method with Artificial Stabilization (SOEM) on the Less Noisy Data Set

Parameter	NV	EEM		SOEM		NPD		
		EST	RSTD (%)	EST	RSTD (%)	AVG (0)	AVG	RSTD (%)
Z_w	-1.4249	-1.4252	0.1610	-1.4498	0.4802	-1.4257	-1.4245	0.1139
Z_q	-1.4768	-1.4685	2.0186	-1.4768	fixed*	-1.4747	-1.4733	0.1139
$Z_{\delta e}$	-6.2632	-6.2344	0.9840	-5.6560	3.1991	-6.2507	-6.2449	0.1139
$\dot{q}_w$	0.2163	0.2161	0.9486	0.2565	0.9810	0.2162	0.2160	0.1066
$\dot{q}_q$	-3.7067	-3.7067	0.7148	-4.7024	0.8621	-3.7082	-3.7044	0.1066
$\dot{q}_{\delta e}$	-12.7840	-12.7626	0.4296	-14.3422	0.5695	-12.7695	-12.7563	0.1066

(NV = Nominal Value; AVG = Average; STD = Standard Deviation; RSTD = Relative Standard Deviation;
EST = Estimate; AVG (0) = Estimate when all the neural inputs are set to zero; *Z_q is kept fixed at the nominal value for processing)

Table-5 : Stability and Control Derivatives Estimated Using the Simple Equation Error Method (EEM), Neural Partial Differential (NPD) Approach, and output Error Method with Artificial Stabilization (SOEM) on the Highly Noisy Data Set

Parameter	NV	EEM		SOEM		NPD		
		EST	RSTD (%)	EST	RSTD (%)	AVG (0)	AVG	RSTD (%)
Z_w	-1.4249	-1.4293	0.7763	-1.4750	1.4083	-1.4326	-1.4267	0.5020
Z_q	-1.4768	-1.3850	10.3366	-1.4768	fixed*	-1.4100	-1.4040	0.5020
$Z_{\delta e}$	-6.2632	-5.9959	4.8883	-6.3995	8.0281	-6.0624	-6.0377	0.5020
$\dot{q}_w$	0.2163	0.1781	5.3335	0.2245	3.0946	0.1811	0.1720	5.4034
$\dot{q}_q$	-3.7067	-3.2224	3.8176	-4.4482	2.5611	-3.3054	-3.1391	5.4034
$\dot{q}_{\delta e}$	-12.7840	-11.6367	2.1643	-13.1867	1.6719	-11.9911	-11.3880	5.4034

(NV = Nominal Value; AVG = Average; STD = Standard Deviation; RSTD = Relative Standard Deviation;
EST = Estimate; AVG (0) = Estimate when all the neural inputs are set to zero; *Z_q is kept fixed at the nominal value for processing)

where the neural partial differential approach is likely to produce biased estimates, can be identified by the fact that the mean square error will not decay with iteration to the desired level. Moreover, the neural predicted force and moment coefficients will not match with the measured ones. However, for small noise to signal ratio, the mean square error will be small, therefore, the estimates will be reliable with small relative standard deviation.

In Figs. 13 to19, the plots for the more noisy data case (STD (2)) are presented. It can be observed from Fig.14 that the actual and the neural predicted values match with each other. This is due to relatively less effect of elevator deflection which is quite noisy on the normal acceleration N_z. However, in the case of $\dot{q}$, the match is poor as shown in Fig.17. In fact, $\dot{q}$ is majorly affected by the elevator deflection which is highly noisy in this case, hence, the neural predicted output gets affected by the noise in the elevator deflection. In Fig.19, variations in the estimated derivatives are easily visible, especially in the derivative $\dot{q}_{\delta e}$. Obviously, this is because of large noise in the input variable δ_e. All the results obtained are as per the theoretical deductions of the previous sections, and give more insight into the nature of the derivatives obtained using neural network. Thus, the neural estimates are very reliable and tend to get biased only when the input data to the neural network is quite noisy.

Conclusion

Neural partial differential approach was carried out for the estimation of the stability and control derivatives of unstable aircraft pertaining to the longitudinal dynamics. The test results for the unstable dynamics were generated using the simulated and the real flight data (numerical data was generated by digitizing the plots in the literature). The results were compared with the conventional methods such as the equation error and output error methods. A thorough theoretical analysis was carried out to ascertain quality of the estimates resulting from the neural partial differential approach. Theoretically, it was concluded that if the training is good, which is reflected in the small mean square error at the end of training, then quality of the estimates will be good yielding small relative standard deviation and the estimates will be unbiased also. Moreover, it was proved analytically that the relative standard deviation of the derivatives for the different inputs will be same. The same was verified through numerical simulation using the real and synthetic flight data for longitudinal dynamics of unstable aircraft. While the inputs were highly corrupted with noise then the estimates using the neural partial differential approach yielded slightly biased estimates, which was a direct consequence of poor learning indicated by large mean square error. The issue of neural architecture to be used for the estimation was resolved and was shown that keeping one neuron in the first hidden layer yielded low standard deviation and the mean square error was small at the end of the training. The theoretical analysis presented in this paper makes the neural partial differential approach more reliable and widely applicable. The parameter estimation, pertaining to the longitudinal motion, of the unstable aircraft X-31A (open loop) from the real flight data was also reported. A thorough analysis of the results, thus obtained, was carried

out to check the accuracy of the estimates, which confirmed the applicability of the neural partial differential approach to the parameter estimation problem for the unstable aircraft. The results obtained, using the neural approach from the simulated and real data, were compared with the actual values as well as with the results from the equation error and stabilized output error methods. It is concluded that performance of the neural partial differential approach is better than the other methods and it can be applied successfully for the parameter estimation of the unstable aircraft.

References

1. Jategaonkar, R. V., "Flight Vehicle System Identification: A Time Domain Methodology, Progress in Astronautics and Aeronautics", 1st ed., Vol.216, p. 9, AIAA, Reston, VA, 2006.

2. Hamel, P. G. and Jategaonkar, R. V., "Evolution of Flight Vehicle System Identification", Journal of Aircraft, Vol. 33, No.1, 1996, pp. 9-28.

3. Iliff, K. W., "Aircraft Parameter Estimation: AIAA Dryden Lecture in Research for 1987", NASA Technical Memorandum-88281, 1987.

4. Jategaonkar, R. V. and Thielecke, F., "Evaluation of Parameter Estimation Methods for Unstable Aircraft", Journal of Aircraft, Vol. 31, No. 3, 1994, pp. 51-519.

5. Hornik, K., Stinchcombe, M. and White, H., "Multilayer Feedforward Networks are Universal Approximators", Neural Networks, Vol.2, 1989, pp. 359-366.

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

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

8. 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.

9. Basappa, K. and Jategaonkar, R. V., "Aspects of Feedforward Neural Network Modeling and its Application to Lateral-Directional Flight Data", DLR-IB 111-95/30, Braunschweig, Germany, September 1995.

10. Feteih, S. and Breckenridge, G., "Neural Network Based Estimator for a Maneuvering Aircraft", Proceedings of American Control Conference, Feb. 1993, pp. 1380-1384.

11. Raol, J. R., "Neural Network Based Parameter Estimation of Unstable Aerospace Dynamic Systems", IEE Proceedings-Control Theory Appl.,Vol.141, No.6, 1994, pp. 385-388.

12. Raisinghani, S. C., Ghosh, A. K. and Kalra, P. K., "Two New Techniques for Parameter Estimation using Neural Networks", The Aeronautical Journal, Vol. 102, No. 1011, 1998, pp. 25-29.

13. 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, 1998, pp. 876-881.

14. Singh, S. and Ghosh., A. K., "Estimation of Lateral-Directional Parameters Using Neural Network based Modified Delta Method", The Aeronautical Journal, October 2007, pp. 659-667.

15. Peyada, N. K. and Ghosh, A. K., "Aircraft Parameter Estimation Using a New Filtering Technique Based upon a Neural Network and Gauss-Newton Method", The Aeronautical Journal, Vol. 113, No. 1142, 2009, pp. 243-252.

16. Kumar, R., Ganguli, R. and Omkar, S. N., "Rotorcraft Parameter Estimation Using Radial Basis Function Neural Network", Applied Mathematics and Computation, 216, 2010, pp. 584-597.

17. Jain, A. K., Mao, J. and Mohiuddin, K. M., "Artificial Neural Networks: A Tutorial", IEEE Computer Society, March 1996, pp. 31-42.

18. Das, S., Kuttieri, R. A., Sinha, M. and Jategaonkar, R. V., "Neural Partial Differential Method for Extracting Aerodynamic Derivatives from Flight Data", AIAA Journal of Guidance, Control, and Dynamics, AIAA, Vol. 33, No. 2, 2010, pp. 376-384.

19. Sinha, M., Kumar, K. and Kalra, P. K., "Some New Neural Network Architecture with Improved Learning Schemes", Soft Computing, Vol. 4, No. 4, 2000, pp. 214-223.

20. Moller, M. F., "A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning", Neural Networks, Vol. 6, 1993, pp. 525-533.

21. Plaetschke, E., Mulder, J. A. and Breeman, J. H., "Results of Beaver Aircraft Parameter Identification", DFVLR-FB 83-10, March 1983.

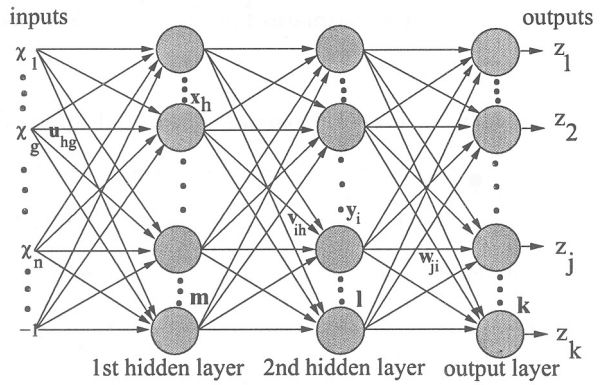

Fig.1 Schematic of a Feedforward Neural Network

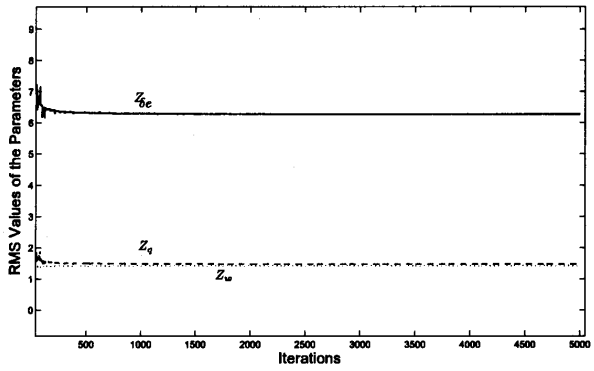

Fig.3 Convergence of the RMS Values of the Estimated Normal Acceleration Derivatives $Z_{\delta e}$, Z_q , and Z_w with Iterations for the Simulated Linear Aircraft Model

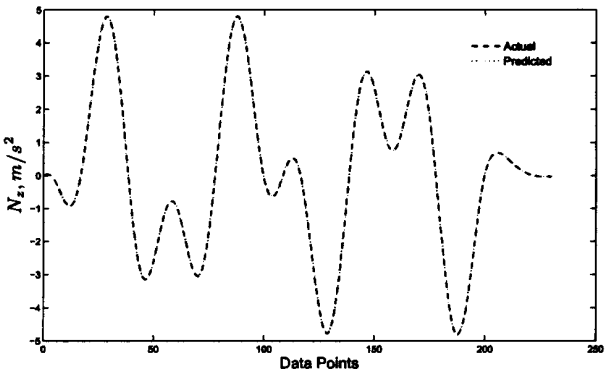

Fig.2 The Actual and Neural Predicted Values of the Normal Acceleration N_z for the Simulated Linear Aircraft Model

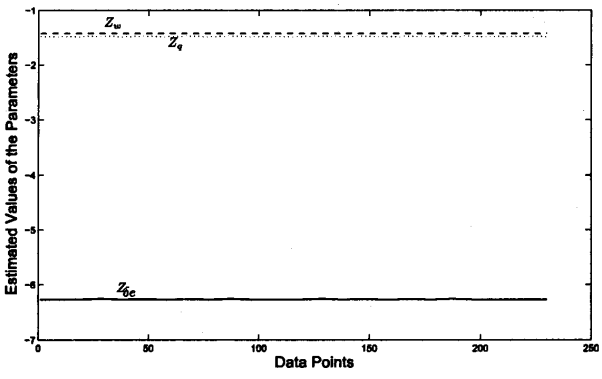

Fig.4 Estimated Values of the Normal Acceleration Derivatives $Z_{\delta e}$, Z_q , and Z_w Corresponding to Different Data Points for the Simulated Linear Aircraft Model

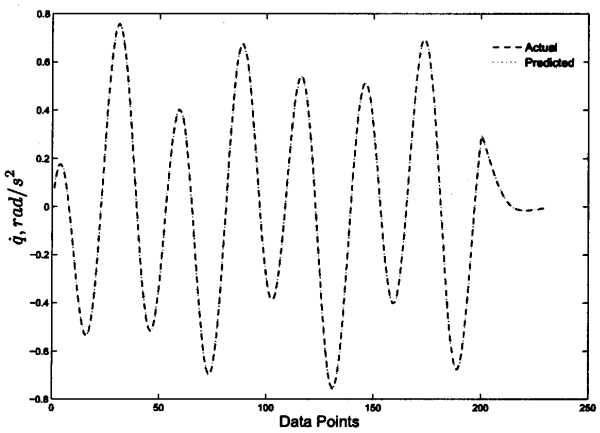

Fig.5 The Actual and Neural Predicted Values of the Pitch Acceleration $\dot{q}$ for the simulated Linear Aircraft Model

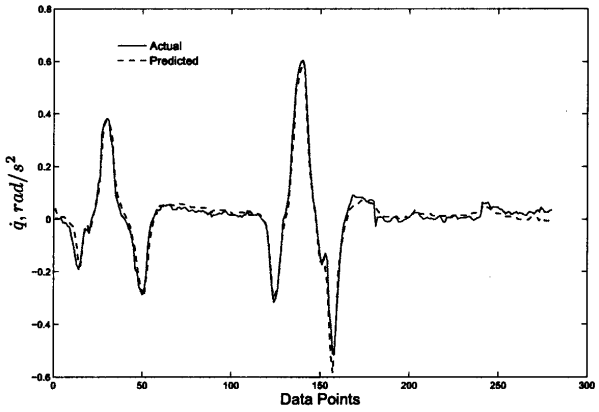

Fig.8 The Digitized (actual) and Neural Predicted Values of the Pitch Acceleration $\dot{q}$ on the Digitized Flight Dta of the X-31A Aircraft. Initial 180 data points are used for the training purpose. The remaining 100 data points are unused for training and are meant for testing the performance of the neural network.

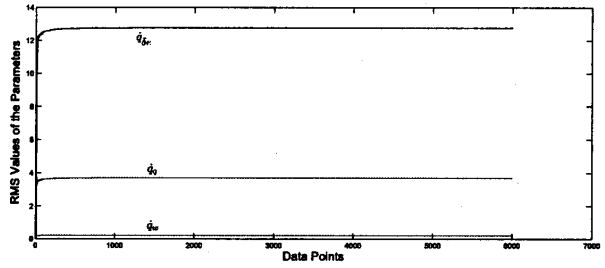

Fig.6 Convergence of the RMS Values of the Estimated Pitch Acceleration Derivatives $\dot{q}_{\delta e}$, $\dot{q}_q$, and $\dot{q}_w$ with Iterations for the Simulated Linear Aircraft Model

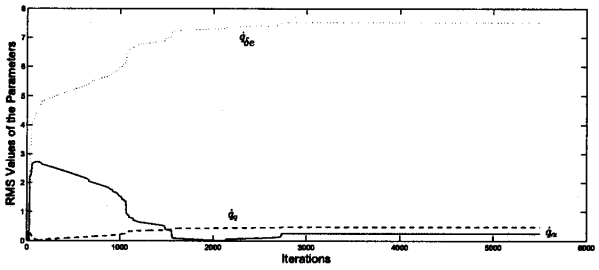

Fig.9 Convergence of the RMS values, of the estimated pitch acceleration derivatives $\dot{q}_{\delta e}$, $\dot{q}_q$, and $\dot{q}_\alpha$, with iterations using 180 data points on the digitized flight data of X-31A aircraft

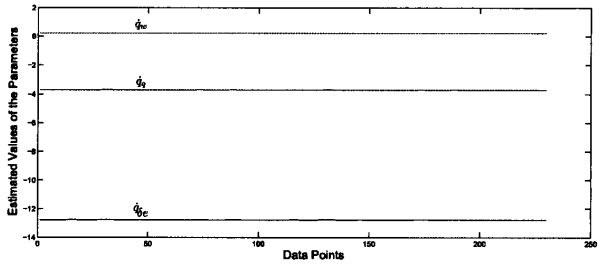

Fig.7 Estimated Values of the Pitch Acceleration Derivatives $\dot{q}_{\delta e}$, $\dot{q}_q$, and $\dot{q}_w$ Corresponding to Different Data Points for the Simulated Linear Aircraft Model

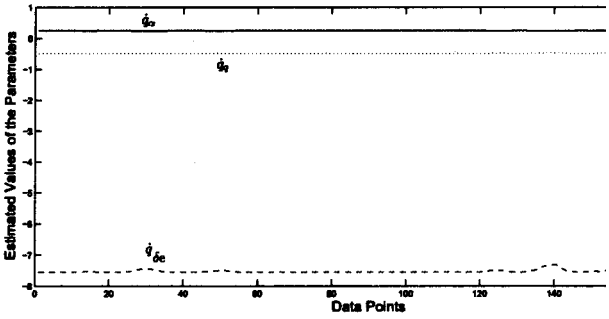

Fig.10 Estimated values of the pitch acceleration derivatives, $\dot{q}_{\delta e}$, $\dot{q}_q$, and $\dot{q}_\alpha$ corresponding to different data points, of the X-31A aircraft from the digitized flight data

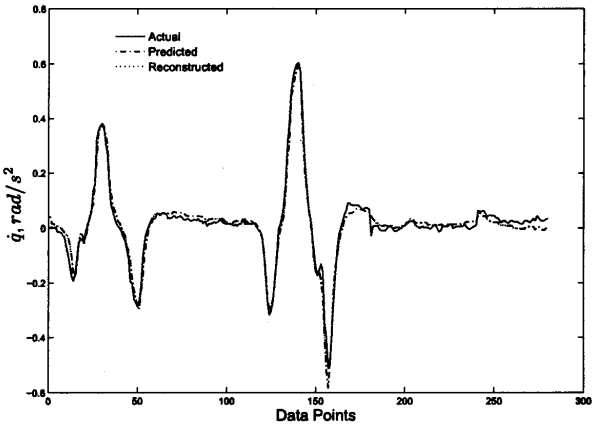

Fig.11 Plots of $\dot{q}$ reconstructed using the estimates from the neural partial differential approach, the actual, and neural predicted values for the X-31A aircraft

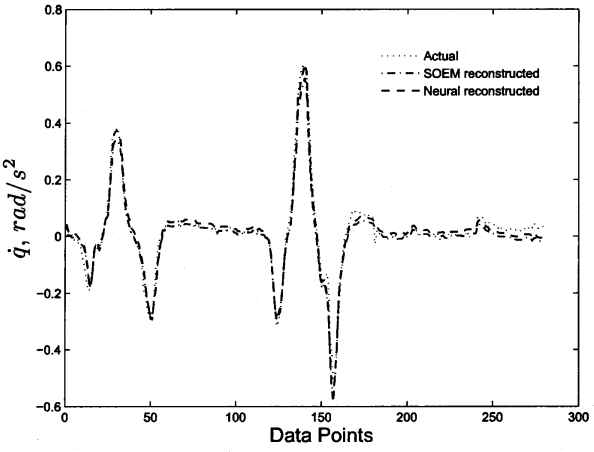

Fig.12 Plots of $\dot{q}$ reconstructed using the estimates from the neural partial differential approach, the actual, and reconstructed using the estimates from the stabilized output error method for the X-31A aircraft

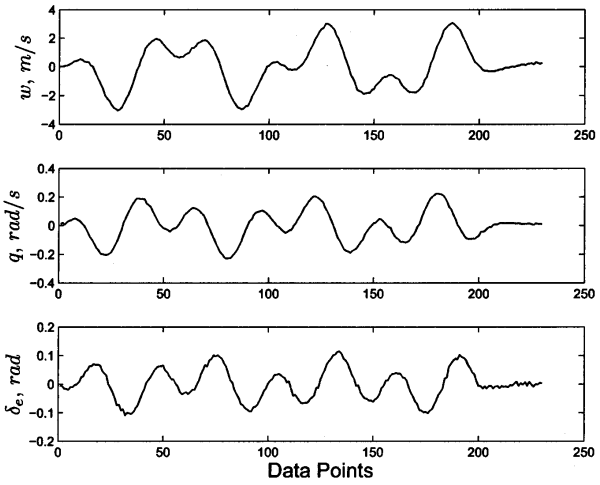

Fig.13 Simulated highly noisy motion and control variables (data points simulated at an interval of 0.5 seconds for a duration of 11.5 seconds with total 230 points to which zero mean white Gaussian noise is added)

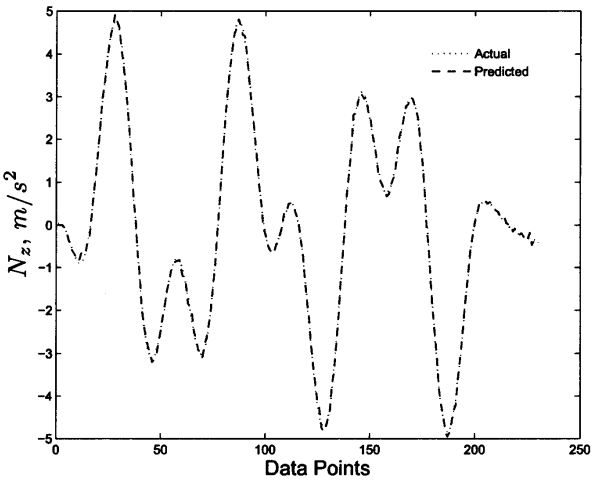

Fig.14 The actual and neural predicted values of the normal acceleration N_z on the highly noisy data set

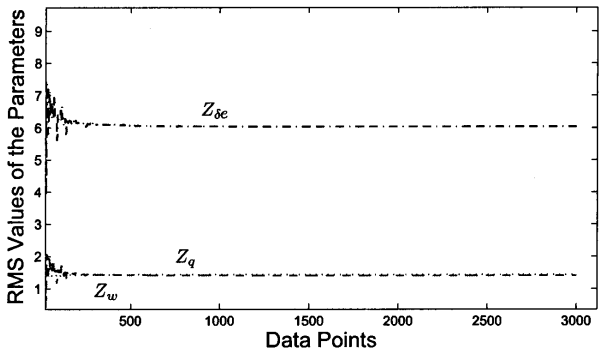

Fig.15 Convergence of the RMS values of the estimated normal acceleration derivatives $Z_{\delta e}$, Z_q, and Z_w with iterations on the highly noisy data set

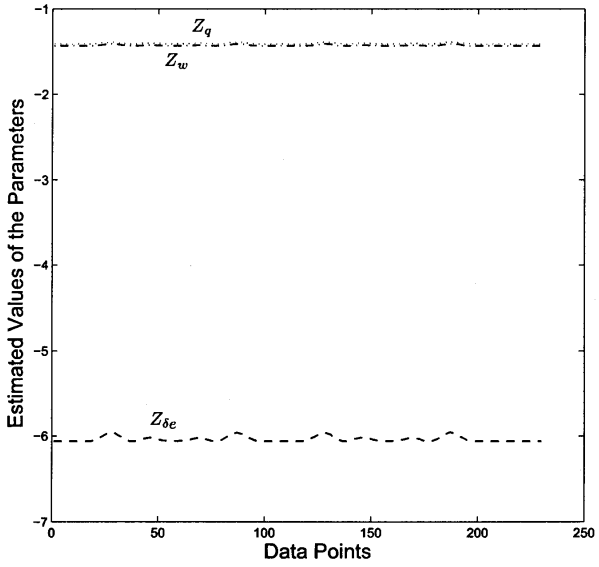

Fig.16 Estimated values of the normal acceleration derivatives $Z_{\delta e}$, Z_q, and Z_w, corresponding to the different data points, for the highly noisy data set

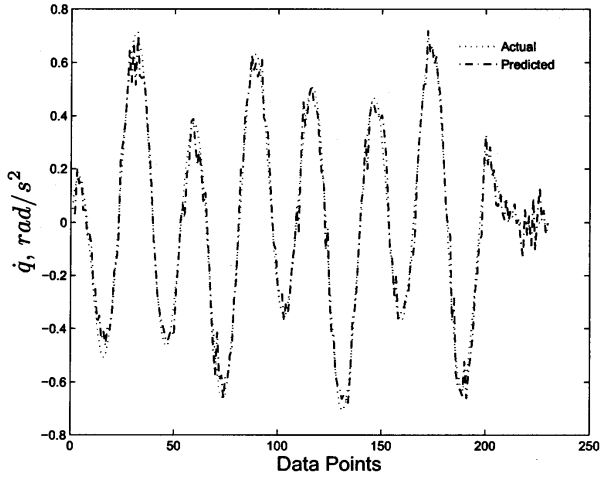

Fig.17 The actual and neural predicted values of the pitch acceleration $\dot{q}$ for the highly noisy data set

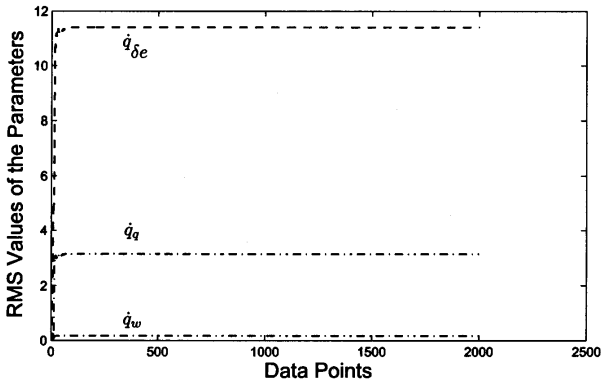

Fig.18 Convergence of the RMS values of the estimated pitch acceleration derivatives $\dot{q}_{\delta e}$, $\dot{q}_q$, and $\dot{q}_w$ with iterations for the highly noisy data set

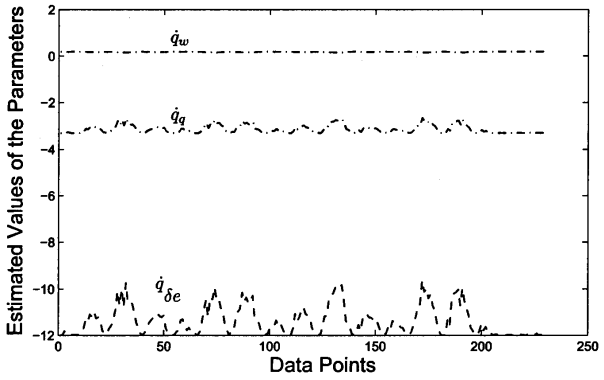

Fig.19 Estimated values of the pitch acceleration derivatives $\dot{q}_{\delta e}$, $\dot{q}_q$, and $\dot{q}_w$ corresponding to different data points for the highly noisy data set