FEED FORWARD NEURAL. NETWORKS FOR AERODYNAMIC MODELLING AND SENSOR FAILURE DETECTION+

J.R Raol*

Abstract

In this paper applications of feed forward neural networks for aerodynamic modelling and sensor failure detection are studied. These nehvorks are trained using fast back-propagation recursive least squares algorithms with forgetting factors. Specifically three training algorithms are implemented in PC MATLAB. The numerical results are presented using simulation of aircraft matliematical models.

Introduction

Recently the field ofartificial neural networks **(ANNs)** has fascinated scientists and engineers all over the world. ANN modeling has become very popular in the area of signal processing, pattern recognition, system identification and control [1,2]. Neural nehvorks have good ability to learn adaptively. In general the decision process in ANNs is based on inherently nonlinear operations. Such nonlinearities might be useful: to improve the convergence speed, to provide more general nonlinear mapping between the given input-output signals, and to reduce the influence of outliers in measurements.

Feed forward neural networks (FFNNs) have found some applications to flight data analysis and aircraft modeling and control [3,4]. However many of such studies generally use back-propagation training algorithm which has slow convergence rate requiring many training sessions. Many applications require online decisions which should be based on processing of the data in real-time for which fast training algorithms must be used.

In this paper application of FFNNs to aerodynamic modeling/prediction and aircraft sensor failure detection using simulated data is investigated. For training of FFNNs, three algorithms: i) back-propagation (BPN), ii) fast back-propagation recursive least squares (RLS) filter with nonlinear output block (BPNRLSFNL) [5] and iii) fast back-propagarion RLS filter with linear output block (BPNRLSFL) are implemented in PC MATLAB [6]. The results of aircraft sensor failuredetection [7] using the fast BPNRLSFL scheme are also presented.

Feed Forward Neural Networks

FFNN is an information processing system consisting of a large number **of** simple processing elements. The elements *are* called artificial neurons or nodes. These neurons are interconnected by links which are characterised by the so called weights and cooperate to perform parallel distributed processing in order to solve a desired computational task. The name neural network comes from the fact that the background of the early researchers, who were involved in the study of functioning of the human brain and modeling of nervous systems, was in the field of biology, physiology and psychology rather than in computer science or engineering [2]. Eventhough **ANNs** have some resemblance to real neural networks, they are more appropriately called massively parallel adaptive filters/circuits (MAPAFs), as ANNs have technical roots in the area of analog computing/circuits and signal processing [2].

The input-output subspace modeling using the FFNNs is feasible because the basic (neural network) functions can adequately approximate the system behaviour in an overall sense. FFNNs can be considered as nonlinear black-box model structures, the parameters (weights) of which can be estimated by conventional optimization methods. **They** are more suitable for system identification problems like: time-series modeling and prediction. sensor failure detection and estimation of aerodynamic coefficients.

* Scientist, Flight Mechanics and Control Division, National Aerospace Laboratories. Bangalore-560017, India

Manuscript received on 22 Aug 95; Paper reviewed and accepted on 23 Nov 95

+ *Paper to be presented at the 47th Annual General Meeting of the Aeronautical Society of India at IIT, Madras during 6-8 January 1996*

Algorithms for Training Feedforward Neural Networks

An FFNNs have non-cyclic layered topology (Fig. 1) and can be considered to give structure-free (in the sense of conventional polynomial model) nonlinear mapping between the input-output signals of a given system The F F '' is first trained using the so called training set data Then it is used for prediction using the same or different input set belonging to the same class of the data. This set is called validation set. The weights of the network are estimated using the so called back-propagatiodgradent based algorithm (BPN). Due to the layered structure, the estimation of weights of FFNN requirespropagation ofthe error of the output layer in backward direction and hence the name BPN. This algorithm has slow convergence rate. The algorithm based on Kalman filter (BPNRLSF) is known to be faster than the BPN [5]. In this paper the algorithms are described using the matrix/vector notation for the sake of comparison, clarity and ease of implementation in PC MATLAB. Even if one does not have the NN tool box of the PC MATLAB, these algorithm and the related simulation studies can be easily and efficiently carried out using the available (and newly formulated) dot-em (.m) files as is done for this paper. The variables of a typical F F '' are described below:

u_o = input to the nenvork (input layer)
n_i = the no. of input neurons (no. of inputs u_o)
n_h = the no. of neurons in the hidden layer
n_o = the no. of output neurons (no. of outputs z)
W_1 = n_h x n_i weight matrix between input and hidden layer
W_{10} = n_h x 1 bias weight vector
W_2 = n_o x n_h weight matrix between hidden and output layer
W_{20} = n_o x I bias weight vector
μ = the learning mre parameter or srep size

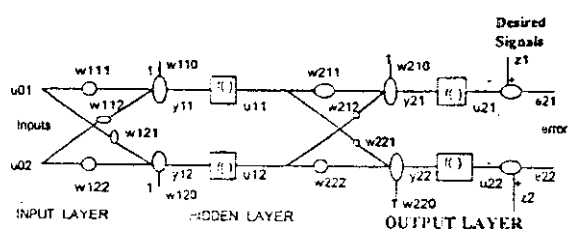

Fig. 1 Schematic of Feed Forward Neural Network

Back Propagation Algorithm

Subscripts 0, 1 and 2 indicate input-, hidden- and output- layers respectively. During a forward pass training ofthenetwork, intermediate signals are computed for each layer and each node oftbelayerandweights ofthe network are estimated. The signal computation is done using the following equations (between input and hidden layer) :

$$y_1 = W_1 u_0 + W_{10} \quad (1)$$

$$u_1 = f(y_1) \quad (2)$$

Here u_1 is the input to the hidden layer and $f(y_i) = \dfrac{1 - e^{-\lambda y_i}}{1 - e^{-\lambda y_i}}$ (a nonlinear activation function). The signal computation between hidden and output layer is done using the following equations:

$$y_2 = W_2 u_1 + W_{20} \quad (3)$$

$$u_2 = f(y_2) \quad (4)$$

Often, an unconstrained optimization problem for parameter estimation, is transformed into an equivalent system ofdifferentid equations which in turn constitute basic (neural network) algorithm to solve specified computational problems [6]:

$$\frac{dW}{dt} = -\mu(t)\,\nabla E(W) \quad (5)$$

where $\nabla(.)$ denotes the gradient of the cost function E wrt W (the unknown parameters). With the output error defined as $e_2 = z - u_2$, and a suitable quadratic cost function based on it, the evaluation of the gradient of E wrt W_2 yields the following expression:

$$\frac{\partial E}{\partial W_2} = -f(y_2)(z - u_2)u_1^T \quad (6)$$

where u_1^T is the gradient of y_2 wrt W_2 [see Eq. (3)], and the derivative of 'f' is given by:

$$f(y_i) = \frac{2\lambda e^{-\lambda y_i}}{(1 + e^{\lambda y_i})^2} \quad (7)$$

The modified error for the output layer can be expressed as:

$$e_{2b} = f(y_2)(z - u_2) \quad (8)$$

Combining Eqs. (5,6,8), the recursive weight-update rule for the output layer is obtained:

$$W_2^{l+1} = W_2^l + \mu\, e_{2b} u_1^T + \Omega(W_2^l - W_2^{l-1})$$

Here Ω is the momentum constant and the last term is used to smooth out the weight changes and to accelerate the convergence of the algorithm Similarly the back-propagation of the error and the weight-update rule (for inner/hidden layer) are obtained as:

$$e_{1b} = f(y_1)\, W_2^T e_{2b} \tag{10}$$

$$W_1^{l+1} = W_1^l + \mu\, e_{1b} u_0^T + \Omega\,(W_1^l - W_1^{l-1}) \tag{11}$$

Here l is the data/sampling time index for recursive update of the estimates. The input-output data are presented to the network in sequential manner. This process is called pattern learning in the ANN terminology. Once all the data are scanned, they are presented again but with initial weights as the output from the previous session. The process is continued untill the convergence is reached. One can call the entire process as the recursive-iterative method.

Back-propagation Recursive Least Squares Filter Algorithms

In these algorithms the weight update formulae for output layer and inner layers use the Kalman gains derived from the corresponding signals from the previous layers [5].

Algorithm with Nonlinear Output Layer (BPNRLSFNL)

Due to the presence of output nonlinearity, its inversion is required in order to employ the linear KF concepts to obtain this algorithm [S]. During the forward pass training of the network the signals y and u are computed for each layer as per the BPN algorithm using Eqs. (1-4). The Kalman gains, $K_{1,2}$ are computed for both the layers. The forgetting factors $h_{1,2}$ are chosen. The formulation is in the usual scalar data processing form.

For layer 1 the Kalman gain and covariance matrix update are given as [5]:

$$K_1 = P_1 u_0 (h_1 + u_0 P_1 u_0)^{-1} \tag{12a}$$

$$P_1 = (P_1 - K_1 u_0 P_1)/h_1 \tag{12b}$$

For layer 2 the Kalman gain and covariance matrix update are given as:

$$K_2 = P_2 u_1 (h_2 + u_1 P_2 u_1)^{-1} \tag{13a}$$

$$P_2 = (P_2 - K_2 u_1 P_2)/h_2 \tag{13b}$$

The modified output error is given as in the case of BPN algorithm by:

$$e_{2b} = f'(y_2)(z - u_2) \tag{14}$$

Back-propagation of the output error (to the inner/hidden layer) gives inner layer error:

$$e_{1b} = f'(y_1)\, W_2^T e_{2b} \tag{15}$$

Finally the weight-update rule (for output layer) is given as:

$$W_2^{l+1} = W_2^l + (d - y_2)\, K_2^T \tag{16}$$

where the components of d are given as:

$$d_i = \frac{1}{\lambda} \ln((1 + z_i)/(1 - z_i)) \tag{17}$$

with $z_i \neq 1$. For the innerhidden layer, the weight updation rule is given as :

$$W_1^{l+1} = W_1^l + \mu e_{1b} K_1^T \tag{18}$$

Except for the additional computation of Kalman gains for each layer, the procedure far training is the same as that for the BPN algorithm.

Algorithm with Linear Output Layer (BPNRLSFL)

In this algorithm the nonlinearities are used only for the inner layers and the output layer is kept as a linear block. The advantage of this modification is that it avoids the inversion of the output stage nonlinear function which might cause some ill-conditioning. In the present case, the use of linear KF concept is directly applicable. During the forward pass training of the network the signals y and u are computed for each layer as per the BPN algorithm using Eqs. (1-3). Since the output black is assumed to be linear, output signal is computed as follows:

$$u_2 = y_2 \tag{19}$$

The Kalman gain related computations are done as per the CPNRLSFNL algorithm. Since there is no output nonlinearity, the error for output layer is given by:

$$e_{2b} = e_2 = (z - y_2) \tag{20}$$

Back-propapion of the output error (to the inner/hidden layer) gives inner layer error :

$$e_{1b} = f(y_1)\, W_2^T\, e_{2b} \tag{21}$$

The weight-update rules are given as:

for output layer : $W_2^{l+1} = W_2^l + e_{2b}\, K_2^T$ (22)

for the inner/hidden layer: $W_1^{l+1} = W_1^l + \mu e_{1b} K_1^T$ (23)

Once the data has been scanned and the convergence achieved, the estimated weights of the last iteration cycle are used and the input data are presented again to the nenvork to obtain the predicred output. This output is compared with the desired/available output 'in order to judge the ability of the network for prediction. Although the mathematical description is given for FFNNs with one hidden layer, the algorithm are implemented for maximum of two hidden layers.

Application **of** the Feed forward Neural Network **Algorithms**

The simulation dam an: generated with sinusoidal signal input *to* the aircraft model [6]:

$$\begin{aligned}
\dot{V} &= -\frac{\bar{q}S}{m} C_D - g \sin(\theta - \alpha) \\
\dot{a} &= -\frac{\bar{q}S}{mV} C_L + \frac{g}{V} \cos(\theta - \alpha) + q \\
\dot{q} &= \frac{\bar{q}S\bar{c}}{I_y} C_m \\
\dot{\alpha} &= q
\end{aligned} \tag{24}$$

Here V is the aircraftvelocity, a the angle of attack, θ the pitch attitude and $\bar{q}$ is the dynamic pressure. The constants S, $\bar{c}$, m and I_y are geometry and mass related parameters of the simulated aircraft. Aerodynamic Model *is* given as:

$$\begin{aligned}
C_D &= C_{D0} + C_{D\alpha}\alpha + C_{D\delta_e}\delta_e \\
C_L &= C_{L0} + C_{L\alpha}\alpha + C_{L\delta_e}\delta_e \\
C_m &= C_{m_0} + C_{m_\alpha}\alpha + C_{m_q}\frac{qc}{2V} + C_{m_{\delta_e}}\delta_e
\end{aligned}$$

The aircraft aerodynamics is represented fur the purpose of modelling using FFNN as:

$$= h_1(V,\alpha,q,\delta_e) \tag{26}$$

$$\dot{q} = h_2(V,\alpha,q,\delta_e) \tag{27}$$

Here h indicates some nonlinear functional relation. The signals V, a, q and δ e are presented to the network as input patterns and signals a and q as output patterns. The FFNN is trained using the BPN and BPN-RLSF algorithms. In general BPN algorithm required more training sessions (10-15) than the fast BPN-RLSF algorithms.

Figure 2 shows the plots of time history match for prediction phase of FFNN using BPN algorithm for μ =0.5, Ω =0.6, ten hidden layer neurons (in a single hidden layer) and with scaled signals. Although the prediction is largely satisfactory, the algorithm is not able to predict the signals very accurately during the interval of 15 to **35** secs. Reasonably optimum values of training parameters and number of hidden layer neurons were usedinall the cases. In Fig. 3 the results of the BPNRLSFL for the same data set **are** presented for μ = 40. Six hidden layerneurons in a single hidden layer were used with scaled signals. The sigmoidal slope parameter λ was chosen as **0.2.** The performance of this algorithm is better than that of the BPN algorithm and only one training iteration was required. This is significant improvement over the classical gradient based BPN algorithm as far **as** the speed and accuracy areconcemed.

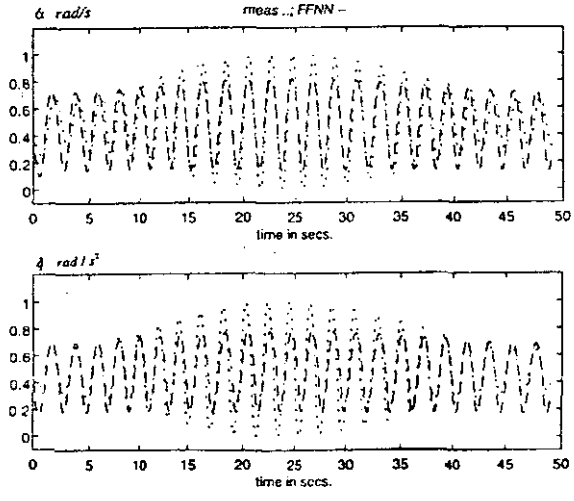

*Fig.2 FFNN-BPN Algorithm Time History Match, **Prediction** Phase*

Subsequently the FFNN **was** used for prediction of the aerodynamic coefficients : C_L and C_m asafunctionof a. The values of computed (simulated) coefficients, **Eq.** (25),

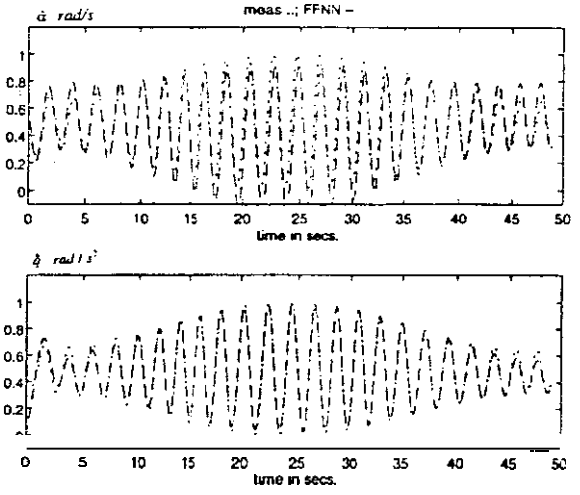

Fig.3 FFNN-BPNRLSF Algorithm Time History Match, Prediction Phase

are used as the output and a and δ_e as the input to the network. The results of this exercise using BPNRLSFNL algorithm are shown in Fig. 4 (for μ =40 and six hidden layer neurons in a single hidden layer) with unscaled signals. Aiso the sigmoidal gain factors for the output layers were chosen different for each output : for C_L, λ =0.1 and for C_m , λ = 0.5. For the inner layer the gain factor was chosen as 0.2. The C_L, , plotted in Fig. 5 wrt angle of attack, tends to follow the pattern of the measured coefficient. Forgetting factors for RLS based algorithms were chosen as 0.99 for FFNN training in all the examples.

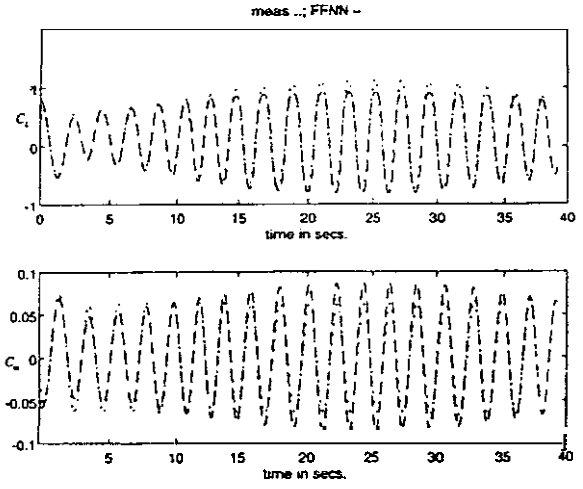

Fig.4 Aerodynamic Coefficient Modelling Using FFNN-BPNRLSNL Algorithm-Prediction Phase (with a ond δ_e as Input to FFNN)

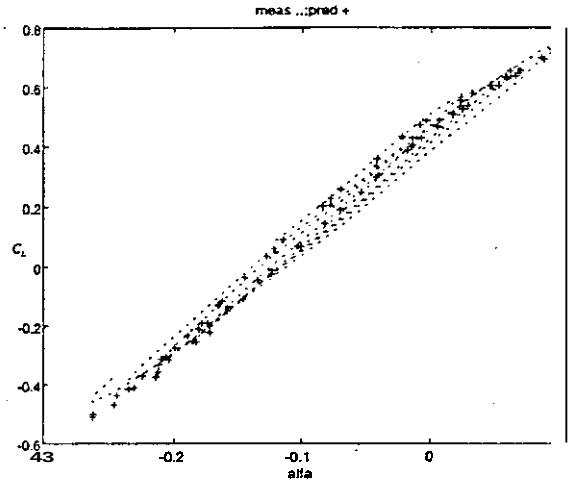

Fig.5 Aerodynamic Coefficient (for o Segment of Time Scale) Prediction

The main aim of this work is to illustrate the feasiblity of using FFNNs for non-linear mapping of the aerodynamic coefficients (as a function of angie of attack, mach number, altitude, etc.) rather than estimating the aerodynamic derivatives. The approach should be useful in obtaining accurate non-linear aerodynamic models based on aerodynamic data (obtained from wind tunnels and analytical predictions) [4,8] in order to, perhaps, replace the time-consuming table look-up methods for nonlinear flight simulation exercises and related work. Accurate estimation of the aerodynamic derivatives using FFNNs requires : major modification in the cost function, training strategy and validation This work has been recently initiated. Alternatively, for estimation of aerodynamic derivatives (like , C_{L_α}, C_{m_α} etc.) reccurrent neural networks (FFNNs with some feed hack from output to the input of the network) can be considered [9].

Next, a scheme based on FFNN-BPNRLSFL algorithm is presented for detection of fault in aircraft sensors. Fig. 6 shows the scheme in comparison to one based on Kalman filter. The Kalman Filter/Observer based approach requires: i) specification of a model of the system, ii) integration of the system's dynamics, iii) computation of Kalman gain and covariances, iv) tuning of the filter (based on process/measurement covariance matrices), v) extended Kalman filter for nonlinear systems. However it is recursive and amenable to real-time online detectiom of faults. The F F '' based method has the following features: i) model of the system is not required, ii) hence the integration of the system equations is avoided, iii) computationofweights of the FFNN, iv) tuning of the nonlinear-

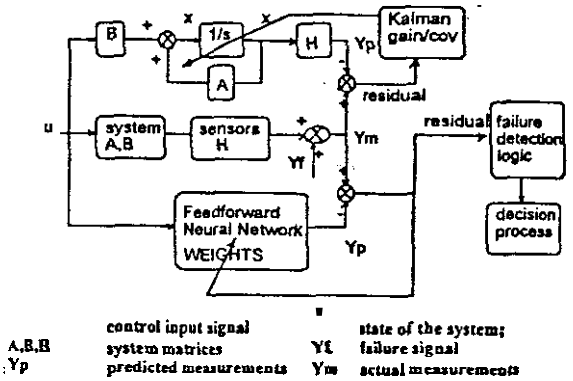

Fig.6 Scnematic Comparison of Sensor Fault Detection Schemes

ity, step size, and choice of neurons in the hidden layer is needed, v) direct applicability to nonlinear systems, vi) recursive and amenable to real-time online detection of faults, vii) relatively easy to implement in hardware and sofhvare. The scheme in this paper utilizes the FFNN as the residual generator [7] and fast BPNRLS [5] Filterwith linear output block for training. The input pattem used is [a(t-1) a (t-2) q(t-1) q(t-2) δe(t-1)] and the out put is [a (t)q (t)]. The sensor fault is simulated as an additional signal and the angle of attack time history is accordingly corrupted. The NW during the training phase itself is used for fault detection. The NW is allowed to train for some initial period Since the BPNRLS filter is used for the training, it takes only a few seconds for the NW to train and is able to predict the data. When the fault occurs at time t, the NW predicted output is compared with the measured data at time t and the residual is compared with the pre-asssigned threshold. The range of residuals is determined under normal no-fault condition. The threshold (for each type of signal/residual) is set at a slightly higher value than the maximum absolute value of the residuals. Alternatively it can be taken as 2 to 3 times the standard deviation of the residul signal. For each type of system under consideration, ihe thresholds should be detrmined based on realistic simulation of die probem [10].

An estimate of the threshold is obtained in real-timeas follows:

$$v(k) = \frac{1}{k-1}(k-2)v(k-1) + r^2(k)] \; ; k = t_1, \ldots, t_1 + m$$

with m = 40 to 50 (28)

$$\alpha_{th} = 3\,[sqrt\,(v\,(t_1 + m))] \quad (29)$$

Here r is the residual generated by the FF '' . About 50 data points were required to obtain the steady value of the threshold during the normal operating condition, after the F F '' has been sufficiently trained. This method is suitable for real-time online operation of the method

Various phases of the NW operation for the sensor failure detection are shown in Fig. 7. The data used are the same as used in the exercises discussed above. The trapezoidal fault signal was added in the angle of attack time history. It can be seen from the figure that the NW is able to immediateiy detect the fault For this study various parameters are: μ =40, λ =0.2, forgetting factors =0.99, hidden layers neurons =6 in single hidden layer. The unscaled signals were used The Fig. 7 shows all the phases of the NW operation during single session or iteration and it can be easily seen that the present scheme which combines the FFNN with BPNRLS filter training algorithm is amenable to online detection of sensor failures. Fig. 8shows the expanded view of the NW operation during the occurrence and detection phase (approx. 24-30 seconds interval) of the fault

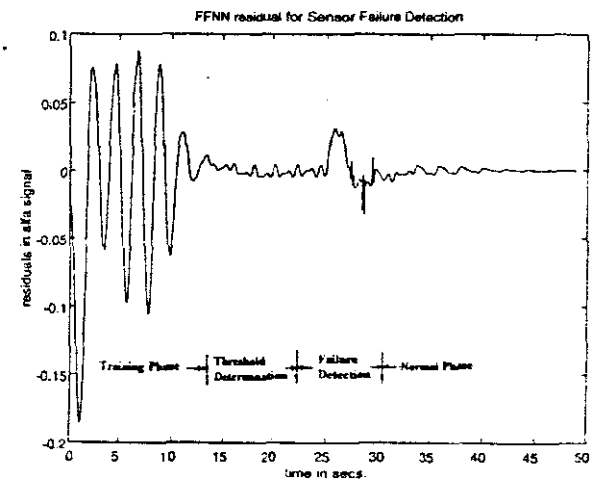

Fig.7 FFNN-BPNRLSF On-line Fault Detection

For fault-identification cross-correlation functions between various significant variables (like angular rates, etc.) can be evaluated and used [9]. Thus the scheme can be extended to detect faults in aircraft sensors as well as actuators. Any indication of fault thus determined will be useful to piiot to monitor the health of the aircraft flight control systems. Further validation is necessary for this

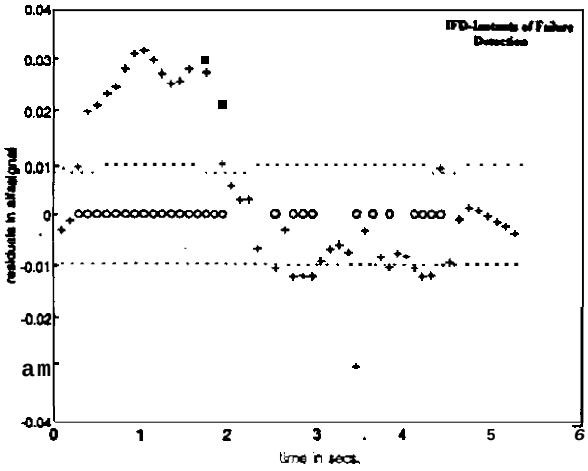

Fig.8 Expanded View of Fault Defection (a FD Phase of Fig.7)

purpose. Although the results presented establish feasibility of using the fast FFNN-BPNRLS filter based schemes for aerodynamic modeling/prediction and aircraft sensor failure detection and look promising, further investigations using real data might be very useful.

. Concluding Remarks

In this paper feasiblity of fast FFNN-BPNRLS filter schemes has been demonstrated for aerodynamic modeling/prediction and aircraft sensor failure detection Quick online detection is feasible because of the use of fast BPNRLS filter which is used for training feedfonvard neural network to generate the residuals as validated by numerical simulation.

Acknowledgements

A part of the work was carried out by the author at the Institute of Flight Mechanics (IFM) under the DLR-CSIR collaborative exchange programme. The author is very grateful to Dr.-Ing. SM, P. Hamel, Director, DLR IFM, and Dr.-Ing. K.-F. Doherr, Head, Mathematical Methods and Data Handling Branch, DLR-IFM, for their support. Technical discussions with Dr. R.V. Jategaonkar, DLR-IFM, were very useful.

1. K. Warwick, G.W. Irwin and K.J. Hunt (Eds.). Neural Networks for Control and Systems. IEE series, London, U.K., 1992.

2. R.C. Eberhart and R.W. Dobbins. Neural Network PC Tools- A Practical Guide. Academic Press Inc. New York, 1992.

3. RA. Hess. On the use of Back Propagation with Feed Forward neural Networks for the Aerodynamic Estimation Problem. AIAA-93-3638-CP, 1993, pp. 233-241.

4. H.M. Youssef and J.C. Juang. Estimation of Aerodynamic Coefficients Using Neural Networks. AIAA-93-3639-CP, pp. 242-245, 1993.

5. Scalero R.S and Tepedelenlioglu N. A Fast New Algorithm for Training Feedforward Neural Networks. IEEE Trans. on Signal Processing, Vol. SP 40, No. 1, Jan. 1992, pp. 202-210.

6. J. R. Raol and R.V. Jategaonkar. Artificial Neural Networks for Aerodynamic Modeling. DLR IB 111-94/41, October, 1994.

7. M.R. Napolitano, C.I. Chen and S. Naylor. Aircraft Failure Detection and Identification using N e d Networks. Jl. of Guid., Contr. and Dynamics, Vol 16, No. 6, Nov.-Dec. 1993.

8. DJ. Linse and R.F. Stengel. Identification of Aero-Dynamic Coefficients Using Computational Neural Neworks. AIAA paper No. 92-0172.

9. J.R. Raol. Parameter Estimation of State Space Models by Recurrent Neural Networks. IEE Proc.-Control Theory Appl., Vo. 142, No. 2, March 1995.

10. V. Krishnaswami, G.C. Luh and G. Rizzoni. Nonlinear Parity Equation Based Residual Generation for Diagnosis of Automative Engine Faults. Control Eng. Practice, Vol. 3, No. 10, October 1995.