Defence Science Journal, Vol. 72, No. 5, September 2022, pp. 655-664, DOI : 10.14429/dsj.72.17547
© 2022, DESIDOC

Aircraft Parameter Estimation using Feedforward Neural Networks with Lyapunov Stability Analysis

Sara Mohan George,[#,*] S. Sethu Selvi[#] and J.R. Raol[$]

[#] *Dept. of Electronics and Communication, Ramaiah Institute of Technology, Bangaluru - 560 054, India*
[$] *Flight Mechanics and Control Division, NAL CSIR, Bangaluru - 560 017, India*
[*] *E-mail: saramohan@msrit.edu*

ABSTRACT

Aerodynamic parameter estimation is critical in the aviation sector, especially in design and development programs of defense-military aircraft. In this paper, new results of the application of Artificial Neural Networks (ANN) to the field of aircraft parameter estimation are presented. The performances of Feedforward Neural Network (FFNN) with Backpropagation and FFNN with Backpropagation using Recursive Least Square (RLS) are investigated for aerodynamic parameter estimation. The methods are validated on flight data simulated using MATLAB implementations. The normalized Lyapunov energy functional has been used to derive the convergence conditions for both the ANN-based estimation algorithms. The estimation results are compared on the basis of performance metrics and computation time. The performance of FFNN-RLS has been observed to be approximately 10% better than FFNN-BPN. Simulation results from both algorithms have been found to be highly satisfactory and pave the way for further applications to real flight test data.

Keywords: Aircraft parameter estimation; Neural networks; Lyapunov's method; Stability analysis

1. INTRODUCTION

The process of formulating a mathematical model that effectively describes and represents the characteristics of a dynamic system by using measurements from the system is known as parameter estimation.[1-2] Estimation of parameters and system identification are very important for the design and development of aircraft control systems.[3] The practice of determining the optimal values of coefficients and their partial derivatives of an airplane is critical as these coefficients help in flight testing-based data analysis and evaluation of the overall aircraft performance.[2] System performance under all regimes may be tested by examining an airplane prototype and also by performing wind tunnel tests.[4-6] Improved sensors, faster computational power, improvements in filtering and control theory have further contributed to successful flight vehicle system identification.[7-8] The Kalman filter is the optimal algorithm for a linear state space model with Gaussian noise assumption.[2] However, aircraft parameter estimation is a nonlinear filtering challenge due to the fact that the aerodynamic models are nonlinear and the measurements from the sensors are noisy and biased.

The Extended Kalman filter (EKF) is a commonly used method for dealing with nonlinear filtering and is frequently used to estimate aircraft aerodynamic parameters.[9-11] The key strategy used to develop EKF was to linearize the nonlinear process or measurement model around the current mean value. However, this might lead to linearization errors. Unlike EKF, Unscented Kalman Filter (UKF) put forward an intuitive method by which the Gaussian density, that represents the prior and posterior pdfs, is represented using a set of deterministic points and corresponding weights.[12] Several other algorithms for parameter estimation, such as iterated EKF and adaptive UKF, have also been developed.[11,13]

The application of Artificial Neural Networks (ANNs) in parameter estimation has been receiving a great deal of attention in recent times.[14-16] However, their usage for aircraft parameter estimation is relatively less. ANNs constitute neurons and connections between nodes/neurons where each connection has its own weight/coefficient. Appropriate choice of ANN and proper training of weights can help in achieving good parameter estimation. The training of conventional ANN is mainly based on the theory of optimization. The ability of ANN to estimate any given function to a desired level of accuracy by selecting the right number of hidden layers and neurons makes them a viable tool for parameter estimation.[17-20] Feedforward Neural Network (FFNN) and Recurrent Neural Network (RNN) are two forms of neural networks that can be used for estimation of parameters.

The FFNN is distinguished by its unidirectional signal flow.[21] FFNN estimates parameters by identifying the optimal weights that provide the best fit for a particular training data-set.[22,23] FFNN uses backpropagation, which is a technique that is based on the classical gradient algorithm that is used to minimize the mean-squared error between the estimated output and the actual output for a given input to the network. RNN can be applied on many types of data that evolve with time as well as ordered sequences. FFNN and RNN have been used in

Received : 5 October 2021, Revised : 31 March 2022
Accepted : 21 July 2022, Online published : 1 November 2022

various control and identification problems.[24-27]

The requirement of stability analysis of the parameter estimation algorithms is very important. In the case of aircraft parameter estimation, stability analysis is critical, especially if the determined aircraft mathematical models are to be used for fault identification and other safety-critical applications. There are various methods by which stability analysis can be performed.[28] The Lyapunov method is an established technique for studying the stability conditions of nonlinear systems[29].

Two types of FFNNs have been discussed in this paper, namely (i) backpropagation and (ii) backpropagation using Recursive Least Square Algorithm (RLS). The method convergence utilising the normalised Lyapunov energy functional (LEF) is investigated for both the variations of FFNNs. MATLAB implementations have been used to analyze the performance of the FFNN algorithms with realistically simulated data for estimation of aerodynamic parameters. The results are found to be highly satisfactory in terms of performance metrics.

2. FEEDFORWARD NEURAL NETWORK

FFNN is formed by neurons that are arranged into an input layer, one or more hidden layers and an output layer. The number of input and output variables in the given estimation problem determines the number of neurons in the input and output layers. There are various algorithms proposed to decide on the number of neurons in the hidden layer [30]. FFNN is normally used along with back propagation (BPN) to train neural networks. The basic goal of a FFNN-BPN is to learn and map the input–output relationship from the presented data for training. The forward step involves the computation of input weights, whereas the backward step deals with updating weights and computing errors. First, the training algorithms for the FFNNs are presented for the sake of completion, then the analytical results for deriving the conditions for the stability using the Lyapunov energy functional are derived. The parameter estimation procedure is outlined in Section 4.

2.1 FFNN-BPN

Consider the state space representation of a dynamical system as

$$\dot{x} = Ax + Bu \tag{1}$$

The main objective of parameter estimation is to find the numerical values of the elements of matrices A and B provided $\dot{x}, x$ and u are known.

$v_0 = [x, u]$ are the known inputs.

The estimation of parameters procedure using the FFNN-BPN can be divided into two steps. The first step is to train the network and hence obtain the weights of the network. The second step is to perform the delta method on the responses from the trained network to estimate the parameters.

Training of the network is achieved by giving the measured data to the network, comparing it with the predicted response and backpropagating the error of the output layer to estimate the weights.

The series of steps involved in training the network is as follows:

I. Initialize the weights randomly.

$W_1 = [w_{11}, w_{12}, \dots, w_{1n}]$, and $W_2 = [w_{21}, w_{22}, \dots, w_{2n}]$ are the guesstimates of the initial weights. These weights are later computed based on the recursive weight update rule given as[1]

$$W(j+1) = W(j) + \mu e v^t + \Omega(W(j) - W(j-1)) \tag{2}$$

where μ is the learning rate, e is the error computed in each step and $\grave{U}$ is the momentum constant which is used to expedite the convergence of the algorithm by smoothing the weight changes.

II. Calculate the intermediate vectors using available data.

y_1 is an intermediate vector between the input and hidden layer formed by combining the weights and known inputs.

$$y_1 = W_1 v_0 + W_{1b} \tag{3}$$

$$v_1 = f(y_1) \tag{4}$$

The input to the hidden layer is v_1 and the activation function applied on the intermediate values is $f(y_1)$

$$f(y_1) = \frac{1 - e^{-\lambda y_1}}{1 + e^{-\lambda y_1}} \tag{5}$$

λ represents the sigmoid slope parameter.

The intermediate vector y_2 between the hidden and output layers and the vector output layer v_2 is computed as

$$y_2 = W_2 v_1 + W_{2b} \tag{6}$$

$$v_2 = f(y_2) \tag{7}$$

III. Evaluate the error based on data from steps I and II.
Consider the quadratic cost function

$$E = \frac{1}{2} e e^T \tag{8}$$

where $e = \dot{x} - v_2$

Since, the rate of change in the values of parameters with respect to time $\frac{dW}{dt}$, is in the negative direction of the gradient of the cost function with respect to the parameter $\frac{\partial E(W)}{\partial W}$

$$\frac{dW}{dt} = -\mu(t) \frac{\partial E(W)}{\partial W} \tag{9}$$

Using Eqns. (6), (7) and (8)

$$\frac{\partial E}{\partial W_2} = -v_2' e v_1^T \tag{10}$$

v_2' is the derivative of the activation function and can be computed as follows:

$$v_2' = \frac{2\lambda e^{-\lambda y_2}}{\left(1+e^{-\lambda y_2}\right)^2} \tag{11}$$

The output layer error is denoted by e_{2b} and is given as

$$e_{2b} = v_2' e \tag{12}$$

IV. Update the weights.

From Eqn. (2) the weight updates for the output layer is expressed as

$$W_2(j+1) = W_2(j) + \mu e_{2b} v_1^T + \Omega(W_2(j) - W_2(j-1)) \tag{13}$$

The back propagated error of the hidden layer and the weight update for W_1

$$e_{1b} = v_1' W_2^T e_{2b} \tag{14}$$

where v_1' is the derivative of the activation function with respect to y_1

$$W_1(j+1) = W_1(j) + \mu e_{2b} v_0^T + \Omega(W_1(j) - W_1(j-1)) \tag{15}$$

V. The data is then presented again and the updated weights from the previous iteration is used. This process continues until convergence is reached.

Once the network is trained, the predicted responses from the network are perturbed one by one in order to estimate each parameter. This is called the Delta method.

2.2 FFNN-BPN using Recursive Least Square Algorithm

Feedforward Neural Network with Backpropagation using Recursive Least Square (FFNN-RLS) uses an enhanced form of the backpropagation technique. The conventional approach (section 2.1) minimises the mean-squared error with respect to the weights. Here, the algorithm minimizes the mean-squared error between the expected output and observed output with respect to the summation outputs[1,31]; and Kalman filter gain and covariance are computed in each layer to update the weights.

The first two steps, I and II, in the algorithms are the same as FFNN-BPN. Next, the Kalman filter equations are computed as follows:

The updates for filter gain K_j and covariance matrix P_j (j-for each layer) are given as:

$$K_j = P_j v_{j-1}\left(f_j + v_{j-1} P_j v_{j-1}\right)^{-1} \tag{16}$$

$$P_j = \frac{P_j - K_j v_{j-1} P_j}{f_j} \tag{17}$$

Here f_j represents the forgetting factor of the j^{th} layer.

The modified output error e_{2b} and the back propagation inner layer error e_{1b} is calculated in a similar way as FFNN-BPN Eqns. (12) and (14).

The weight updates for the output layer is given as:

$$W_2(j+1) = W_2(j) + (d - y_2) K_2^T \tag{18}$$

where, d is the summation of output and is calculated by using the inverse function as

$$d = \frac{1}{\lambda} ln \frac{1+\dot{x}}{1-\dot{x}} \tag{19}$$

Here λ is the sigmoid slope parameter and $\dot{x}$ is known as given in Eqn. (1).

The weight updates for the hidden layer is given as

$$W_1(j+1) = W_1(j) + \mu e_{1b} K_1^T \tag{20}$$

The updated weights are used and the algorithm is repeated till the desired convergence is achieved.

As explained in section 2.1, the delta method is used to obtain the parameters.

The major difference between the two algorithms is in the method of backpropagation of errors. As FFNN-RLS uses Kalman gain and covariance to update the weights, it is expected to achieve better network parameters in lesser number of iterations as compared to FFNN-BPN.

3. ASYMPTOTIC STABILITY ANALYSIS

Lyapunov stability provides a strong assurance on the convergence of the system for any state. It is well known that a system is Lyapunov stable if and only if there exists a Lyapunov energy function (LEF) that is strictly positive definite and the derivative of the LEF is strictly negative definite. In order to establish the conditions for the convergence of the algorithms, the Lyapunov-Krasovskii technique is used.

Let the error state be defined as follows:

$$e(\kappa+1) = Ge(\kappa) \tag{21}$$

A feasible normalized LEF is described as

$$V(e(\kappa)) = e^T(\kappa) P^{-1}(\kappa) e(\kappa) \tag{22}$$

Equation (22) is a positive definite matrix, and Eqn. (23) is used to obtain the derivative of the Lyapunov energy function:

$$\Delta\{V(e(\kappa))\} = V(e(\kappa+1)) - V(e(\kappa)) \tag{23}$$

$$\begin{aligned}
\Delta\{V(e(\kappa))\} &= V(e(\kappa+1)) - V(e(\kappa)) \\
&= e^T(\kappa+1) Ye(\kappa+1) - e^T(\kappa) Ye(\kappa) \\
&= e^T(\kappa) G^T YGe(\kappa) - e^T(\kappa) Ye(\kappa) \\
&= e^T(\kappa)\left[G^T YG - Y\right] e(\kappa) \\
\Delta\{V(e(\kappa))\} &= -e^T(\kappa)\left[-\left(G^T YG - Y\right)\right] e(\kappa)
\end{aligned} \tag{24}$$

Then, the condition for the convergence of the error dynamics can be stated as:

$$-\left(G^T YG - Y\right) = Y - G^T YG \tag{25}$$

$$\left|Y - G^T YG\right| > 0$$

$$Y - G^T YG > 0 \Rightarrow G^2 < 1$$

3.1 Stability Analysis of FFNN-BPN

Let $e(\kappa)$ represent the network error i.e., the difference between the actual network output and the desired output.

$$e(\kappa+1) = d(\kappa+1) - y(\kappa+1) \tag{26}$$

$y(\kappa+1)$ is the output of the neural network at the κ +1th time instant.

$$d(\kappa+1) = F_\kappa v_{0,\kappa+1} \tag{27}$$

where v_0 is the input to (the input layer) of the neural network as given in eqn. (27).

$$y(\kappa+1) = f\left(y_{2,\kappa+1}\right) \tag{28}$$

Substituting eqns. (27) and (28) in eqn. (26):

$$e_{\kappa+1} = F_\kappa v_{0,\kappa+1} - f\left(y_{2,\kappa+1}\right) \tag{29}$$

The function $f(.)$ is a nonlinear sigmoidal activation function as mentioned in Eqn. (5)

$$y_{2,\kappa+1} = \sum w_{2,\kappa+1} v_{1,\kappa+1} \tag{30}$$

$v_1 = f(y_1)$, where v_1 is the input to the hidden layer and y_1 is a vector of intermediate values.

$$y_{1\kappa} = \sum w_{1,\kappa+1} v_{0,\kappa+1} \tag{31}$$

Substituting Eqns. (30) and (31) in Eqn. (29)

$$e_{\kappa+1} = F_\kappa v_{0,\kappa+1} - f\left(\sum w_{2,\kappa+1} \times f\left(\sum w_{1,\kappa+1} v_{0,\kappa+1}\right)\right) \tag{32}$$

Rewriting $w_{2,\kappa+1}$ in terms of $w_{2,\kappa}$ with respect to Eqn. (13), we get

$$w_{2,\kappa+1} = w_{2,\kappa} + \mu e_{2b,\kappa} v_{1,\kappa}^T + \Omega\left(\Delta w_{2,\kappa}\right) \tag{33}$$

From Eqn. (12)

$$e_{2b,\kappa} = f'\left(y_{2,\kappa}\right)\left(d_\kappa - y_\kappa\right) = f'\left(y_{2,\kappa}\right) e_\kappa \tag{34}$$

Similarly,

$$w_{1,\kappa+1} = w_{1,\kappa} + \mu e_{1,\kappa} v_{0,\kappa}^T + \Omega\left(\Delta w_{1,\kappa}\right) \tag{35}$$

Using Eqn. (14)

$$e_{1,k} = f'\left(y_{1,k}\right) w_{2,\kappa+1}^T e_{2,\kappa} = f'\left(y_{1,\kappa}\right)\left[w_{2,k} + \mu e_{2,\kappa} v_{1,\kappa}^T + \Omega\left(\Delta w_{2,\kappa}\right)\right]^T f'\left(y_{2,\kappa}\right) e_k \tag{36}$$

From eqn. (32)

$$e(\kappa+1) = F_\kappa v_{0,\kappa+1} - f\left(\begin{matrix} w_{2,k} + \mu f'\left(y_{2,\kappa}\right) e_\kappa v_{1,\kappa}^T + \Omega\left(\Delta w_{2,\kappa}\right) \times \\ f\left(w_{1,k} + \mu f'\left(y_{1,\kappa}\right)\left[\begin{matrix} w_{2,\kappa} + \mu f'\left(y_{2,\kappa}\right) e_\kappa v_{1,\kappa}^T \\ + \Omega\left(\Delta w_{2,\kappa}\right)\end{matrix}\right] f'\left(y_{2,\kappa}\right) e_\kappa v_{0,\kappa}^T + \\ \Omega\left(\Delta w_{1,\kappa}\right)\right) v_{0,\kappa+1}\end{matrix}\right) \tag{37}$$

To simplify the error dynamics, the following bounds on the norms of the following vectors are required. $\left\|Fv_0\right\| = \rho\left\|e\right\|^2$

$$\left\|e_k\right\| \le \varepsilon\ ;\ \left\|f'(y)\right\| \le h\ ;\ \left\|w\right\| \le \omega\ ;\ \left\|v\right\| \le \upsilon \tag{38}$$

Substituting, $1+\Omega = k$ and solving,

$$e(\kappa+1) = \rho\varepsilon^2 - f(\omega k + \mu h\upsilon\varepsilon \times f(\omega\upsilon k + \mu h^2\upsilon^2\varepsilon[\omega k + \mu h v\varepsilon])) = Ge_\kappa \tag{39}$$

$$G^2 \equiv \left(\rho\varepsilon - \frac{1}{\varepsilon} f\left(\omega k + \mu h\upsilon\varepsilon \times f\left(\omega\upsilon k + \mu h^2\upsilon^2\varepsilon\left[\omega k + \mu h v\varepsilon\right]\right)\right)\right)^2 < 1 \tag{40}$$

$$\rho\varepsilon - \frac{1}{\varepsilon} f\left(\omega k + \mu h\upsilon\varepsilon \times f\left(\omega\upsilon k + \mu h^2\upsilon^2\varepsilon\left[\omega k + \mu h v\varepsilon\right]\right)\right) < sqrt(1) = \pm 1 \tag{41}$$

$$\rho\varepsilon < 1 + \frac{1}{\varepsilon} f\left(\omega k + \mu h\upsilon\varepsilon \times f\left(\omega\upsilon k + \mu h^2\upsilon^2\varepsilon\left[\omega k + \mu h v\varepsilon\right]\right)\right) \tag{42}$$

By defining

$$f_{\max} = \max\left(\omega k + \mu h\upsilon\varepsilon \times f\left(\omega\upsilon k + \mu h^2\upsilon^2\varepsilon\left[\omega k + \mu h v\varepsilon\right]\right)\right) \tag{43}$$

$$\rho\varepsilon < 1 + \frac{f_{max}}{\varepsilon} \tag{44}$$

3.2 Stability Analysis of FFNN-RLS

Let the normalized LEF be defined as Eqn. (22) and the output error state as Eqn. (21).

The steady state solution is defined as P(∞) and this must be in the range:

$$p_l I \le P \le p_u I \tag{45}$$

Here, $p_u, p_l > 0$ are positive numbers, P is a positive definite matrix by definition and the bounds are specified by the constants.

Also, the following bounds are needed to simplify the derivative:

$$\left\|G^T G\right\| \le g^2; \left\|e_\kappa\right\|^2 \le \varepsilon^2; \text{and } f_1 \text{ is the forgetting factor} \tag{46}$$

$$\Delta\left\{V\left(e(\kappa)\right)\right\} = V\left(e(\kappa+1)\right) - V\left(e(\kappa)\right) \tag{47}$$

$$= e^T(\kappa+1) P^{-1}(\kappa+1) e(\kappa+1) - e^T(\kappa) P^{-1}(\kappa) e(\kappa) \tag{48}$$

From eqn. (17)

$$P(\kappa+1)=\frac{P(\kappa)-K(\kappa+1)v_1(\kappa)P(\kappa)}{f_1} \tag{49}$$

$$K(\kappa+1)=P(\kappa)v_1(\kappa)\left(f_1+v_1^T(\kappa)P(\kappa)v_1(\kappa)\right)^{-1} \tag{50}$$

v_1 is the input to the hidden layer which is obtained after performing the nonlinear activation function on y which is the summation of the product of weights and input vector.

$$v_1=\frac{1-e^{-\lambda y}}{1+e^{-\lambda y}} \tag{51}$$

λ is the sigmoid slope parameter of the hidden layer.

This can be re-written as $v_1=f\left(\sum_{j=1}^{n} w_{1,j}v_0\right)=f(y)$ where n is the number of hidden nodes; $w_{1,j}$ are the weights between input and hidden layer; and v_0 is the input vector to the neural network.

$$f(y)=y_m \tag{52}$$

$$\Delta\{V(e(\kappa))\}=e_k^T G^T\left(\frac{P_k-\left[P_k f(y_\kappa)\left(f_1+f(y_\kappa)^T P_k f(y_\kappa)\right)^{-1}\right]f(y_\kappa)P_\kappa}{f_1}\right)^{-1} Ge_\kappa - e_\kappa^T P_\kappa^{-1} e_\kappa \tag{53}$$

After substituting the bounds defined in Eqns. (45), (46) and (52),

$$\Delta\{V(e(\kappa))\}=\varepsilon^2 g^2\left(\frac{p_l-\dfrac{p_l^2 y_m^2}{f_1+y_m^2 p_l}}{f_1}\right)^{-1}-\frac{\varepsilon^2}{p_l}$$

$$=\varepsilon^2\left[g^2\frac{f_1+y_m^2 p_l}{p_l}-\frac{1}{p_l}\right] \tag{54}$$

$$\Delta\{V(e(\kappa))\}<0 \tag{55}$$

$$\left[g^2\frac{f_1+y_m^2 p_l}{p_l}-\frac{1}{p_l}\right]<0 \tag{56}$$

$$g^2\left(f_1+y_m^2 p_l\right)<1 \tag{57}$$

All the coefficients appearing in Eqns. (44) and (57) are precisely defined as these are the constants with positive bounding. If the conditions of Eqns. (44) and (57) are met, the time derivative of the Lyapunov energy function, represented in Eqn. (23), will be negative definite and the FFNNs will converge. This is a novel approach to establishing the asymptotic stability of the FFNN-BPN and FFNN-RLS. If these conditions are met, then the aircraft parameter algorithms using the FFNNs would also converge and the time histories obtained from the converged FFNN-algorithms can be used for the computation of parameters as outlined in section 4.

4. AIRCRAFT PARAMETER ESTIMATION

The following postulated aircraft model is used to depict aircraft longitudinal motion.[1-2]

$$\dot{u}=\frac{\bar{q}S}{m}C_X-qw-g\sin\theta$$

$$\dot{w}=\frac{\bar{q}S}{mV}C_Z+qu+g\cos\theta \tag{58}$$

$$\dot{q}=\frac{\bar{q}S\bar{c}}{I_{yy}}C_m$$

$$\dot{\theta}=q$$

In Eqn. (58), C_x, C_z and C_m are the non-dimensional aerodynamic coefficients expressed as:

$$C_z=C_{Z_0}+C_{Z_\alpha}\alpha++C_{Zq}\frac{q\bar{c}}{2V}+C_{Zq\delta_e}\delta_e$$

$$C_x=C_{X_0}+C_{X_\alpha}\alpha+C_{X_{a^2}}\alpha^2$$

$$C_m=C_{m_0}+C_{m_\alpha}\alpha++C_{m_{\alpha^2}}\alpha^2+C_{mq}\frac{q\bar{c}}{2V}+C_{m_{\delta_e}}\delta_e \tag{59}$$

Here, the state variables are the velocity components in the longitudinal and vertical axis represented by u and w respectively, pitch angle θ, and pitch rate q. The various parameters used for aircraft data simulation are listed in Table 1. The control input to aircraft dynamics is elevator deflection and is denoted by δ_e. β is the vector of unknown parameters that is to be estimated and it comprises of the aerodynamic derivatives:

$$\beta=\left[C_{x0},\ C_{x\alpha},\ C_{x_{\alpha^2}},\ C_{z0},\ C_{z\alpha},\ C_{z\delta e},\ C_{m0},\ C_{m\alpha},\ C_{m_{\alpha^2}},\ C_{mq},\ C_{m\delta e}\right] \tag{60}$$

5. RESULTS AND DISCUSSIONS

MATLAB implementation is performed to obtain the short period data of a light transport aeroplane and non-dimensional longitudinal characteristics are computed to assess the efficacy of FFNN-BPN and FFNN-RLS. The Eqns. (58) and (59) are utilised to estimate the 4-degrees of freedom (4-DOF) longitudinal axis model. In order to validate the algorithms for aircraft parameter estimation, two datasets/examples were simulated.

The two examples are denoted as Ex1 and Ex2. The training dataset for Ex1 is generated for a duration of nearly 7 seconds with a sampling interval of 0.03 seconds by giving a doublet input to the elevator control surface. The second dataset (for Ex2) is also generated for a duration of approximately 7 seconds with a sampling interval of 0.03 seconds by giving a 3211 input to the aircraft control surface. The initial values for the states $[u, w, q, \theta]$ are chosen as [36,7,0,0.15] for both examples.[2] The time history of input states and control input for Ex1 and Ex2 are shown in Fig. 1 and Fig. 3 respectively. Aircraft parameters used to generate both the data sets are given in Table 1.

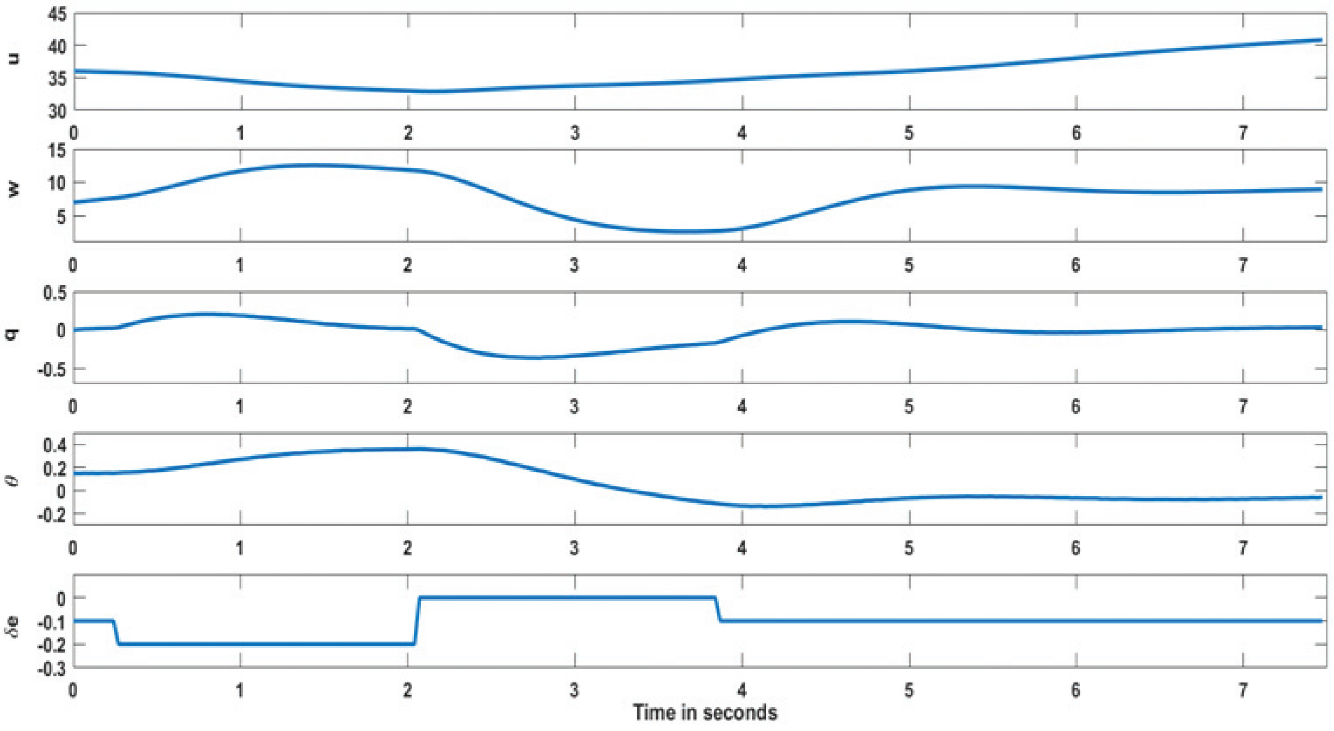

Figure 1. Time series plot of inputs (Ex1).

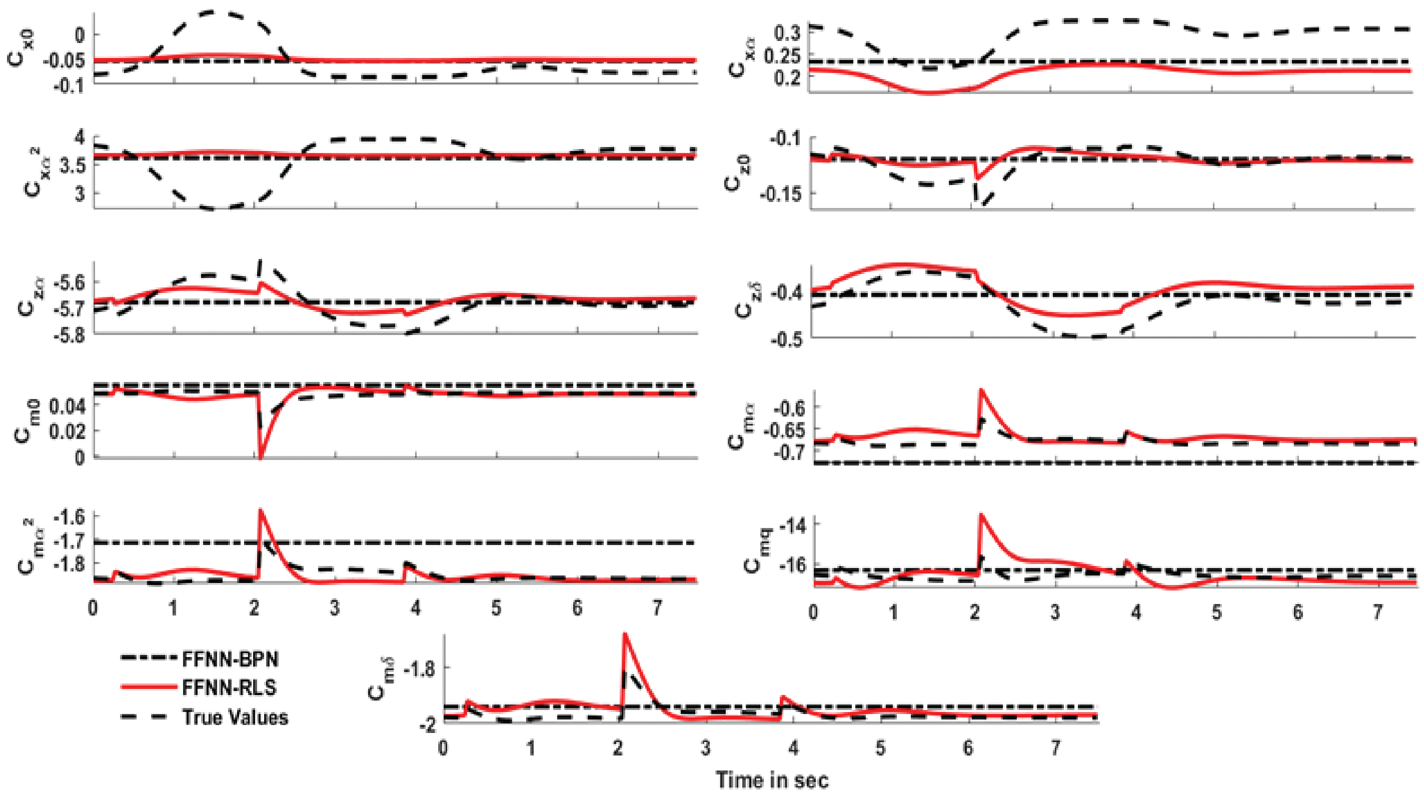

Figure 2. Time history match of parameters estimated using FFNN-BPN/FFNN-RLS (Ex1).

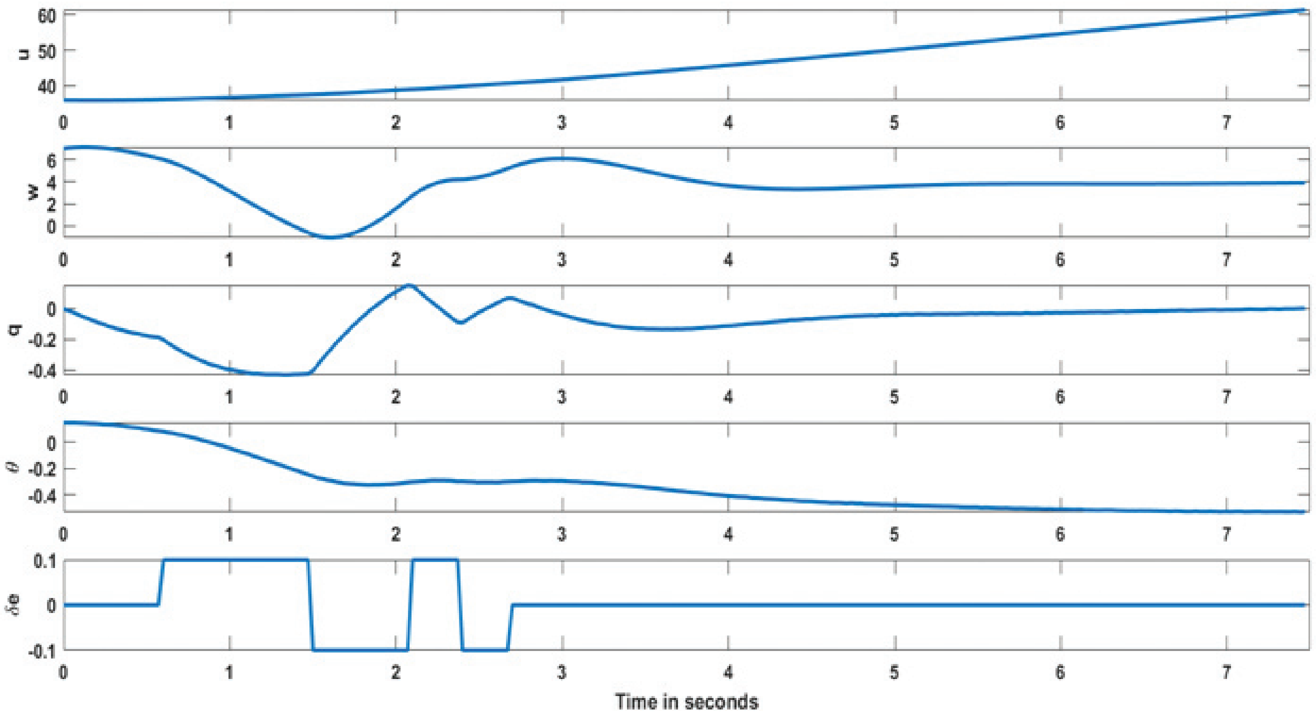

Figure 3. Time series plot of inputs (Ex2).

Table 1. Aircraft parameters used for simulation

Parameter	Values used
Mass m	2280 kg
Wing Chord $\bar{c}$	1.5875 m
Wing Span S	23.23 m^2
Dynamic Pressure $\bar{q} = \frac{1}{2}\rho V^2$	6.1042 Pa
Initial velocity in x axis u0	36 m/s
Initial velocity in z axis w0	7 m/s
True air speed $V = \sqrt{u^2 + w^2}$	36.6742 m/s
Angle of attack $\alpha = tan^{-1}\left(\frac{w}{u}\right)$	0.1973 °
Moment of inertia I_{yy}	6940 kg/m^2

A 4th order Runge-Kutta integration method is used to obtain the longitudinal flight variables u, w, q and θ from eqn. (58). The performance metric used is parameter estimation error norm (PEEN). PEEN is defined as:

$$PEEN = \frac{norm(\beta_{true} - \beta_{est})}{norm(\beta_{true})} * 100 \quad (61)$$

where β_{true} is the vector of true values of the aircraft parameters and β_{est} is the vector of estimated values of the aircraft parameters. PEEN is calculated for each method. Table 2 presents the estimated parameter values for Ex1 and Ex2. The comparison of the computational times and the PEENs is depicted in Table 4.

In order to train the network, simulations are carried out with specifications as in Table 1 to generate time histories for the variables $\alpha, \alpha^2, q, \delta_e$ and coefficients C_x, C_m and C_z using the (56) and (57). The data $\alpha, \alpha^2, q, \delta_e$ is given as input to the network and C_x, C_m and C_z is presented as output. The algorithms are used to train the network. FFNN-BPN and FFNN-RLS are used to obtain the updated weights which can be used to estimate the parameters. Once the network is trained, delta method is used to estimate the aerodynamic parameters such as $C_{x\alpha}$, $C_{x\alpha^2}$, $C_{z\alpha}$, $C_{z\delta_e}$, $C_{m\alpha}$, $C_{m\alpha^2}$, $C_{m\delta_e}$, C_{mq} . In the delta method, the input variables are perturbed to cause a change in the output variable. For example, the value of q is changed to $q + \Delta q$ for all the data points and a corresponding change in C_m^+ is obtained. Similarly, q is changed to $q - \Delta q$ for all the data points and corresponding change C_m^- is obtained. The derivative C_{mq} will be $C_{mq} = \frac{C_m^+ - C_m^-}{2\Delta q}$. The remaining three parameters C_{x0}, C_{m0} and C_{z0} can be calculated by substituting the parameters estimated using the delta method and the available coefficients i.e., C_x, C_m and C_z into Eqn. (59). The estimated parameter values are compared with the true values that are given in the 2nd column of Table 2. It is observed that the estimated values are quite close to the true values even when there is noise of SNR=10 in the data. The PEEN related to Ex1 is 1.82% for FFNN-BPN and 1.69 % for FFNN-RLS, which is reasonably small. Similarly, simulations were performed for Ex2 and the PEEN was observed to be 1.98% and 1.71% for the FFNN-BPN and FFNN-RLS algorithms respectively.

Table 2. Estimated aerodynamic parameters

Parameter	True Values	FFNN-BPN Ex1	FFNN-RLS Ex1	FFNN-BPN Ex2	FFNN-RLS Ex2
C_{X0}	- 0.0540	- 0.0560	- 0.0515	-0.05381	-0.05457
$C_{X\alpha}$	0.2330	0.2919	0.2159	0.25124	0.2407
$C_{X\alpha^2}$	3.6089	3.5830	3.6416	3.5567	3.56543
$C_{Z\alpha}$	- 0.1200	- 0.1226	- 0.1209	-0.1271	-0.1330
$C_{Z\alpha}$	- 5.6800	- 5.6799	- 5.6687	-5.70397	-5.5722
$C_{Z\delta e}$	- 0.4070	- 0.4231	- 0.3944	-0.3165	-0.41148
C_{m0}	0.0550	0.0479	0.0481	0.05131	0.05522
$C_{m\alpha}$	- 0.7290	- 0.6803	- 0.6819	-0.6718	-0.73905
$C_{m\alpha^2}$	- 1.7150	- 1.8535	- 1.8182	-1.9878	-1.4706
C_{mq}	- 16.300	-16.5652	-16.0477	-16.4310	-16.22079
$C_{m\delta e}$	- 1.9400	- 1.9669	- 1.9374	-1.9425	-1.94366
PEEN(%)		1.886	1.693	1.97761	1.71047

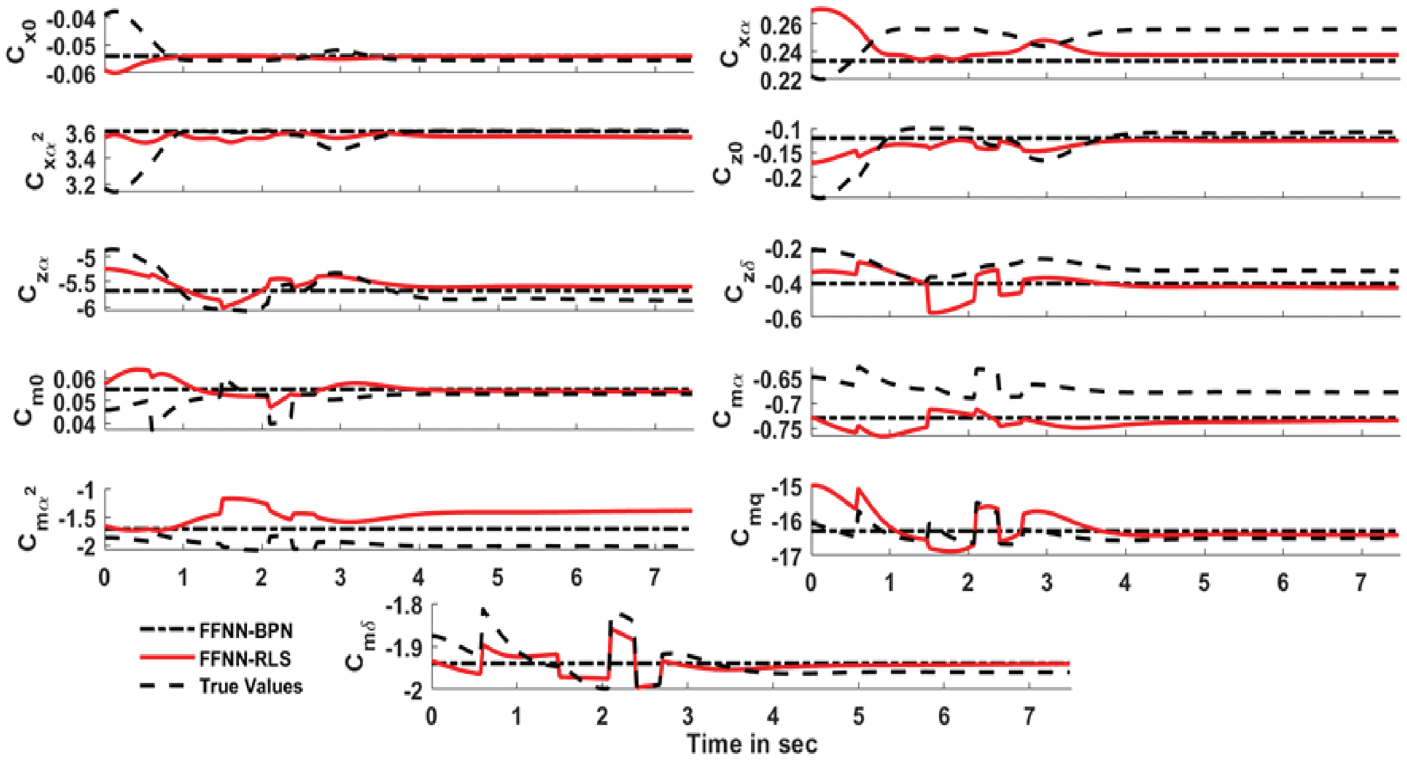

Figure 4. Time history match of parameters estimated using FFNN-BPN/FFNN-RLS (Ex2).

The tuning parameters that have been used for training the network are listed in Table 3. The time histories of the estimated parameters in Figure 2 (for Ex1) and Figure 4 (for Ex2) also show that the estimated values are significantly close to the true values and the estimates using FFNN-RLS have approximately 10% better PEEN than FFNN-BPN.

Table 3. Tuning parameters for FFNN-BPN/FFNN-RLS

Tuning Parameters	FFNN-BPN	FFNN-RLS
No. of hidden layers	1	1
No. of nodes in hidden layer	6	6
Sigmoid Slope parameter λ_1	0.8	0.8
Sigmoid Slope parameter λ_2	0.75	0.75
Learning Parameter	0.2	0.2
Momentum parameter	0.4	-
No. of training iterations	1000	200

The comparative CPU time requirements for the two methods are shown in Table 4. Computation of Kalman gain is needed in FFNN-RLS. However, this method, which uses an improved form of backpropagation using Kalman gain (and covariance) requires lesser number of training iterations to converge. Hence, the overall computational time is seen to be about 60% lesser in FFNN-RLS as compared to FFNN-BPN.

Table 4. Comparison of execution times and PEENs

	PEEN in %		Execution Time in seconds	
	Ex1	Ex2	Ex1	Ex2
FFNN-BPN	1.886	1.9776	3.912173	4.364223
FFNN-RLS	1.693	1.71047	1.129653	1.15643

5. CONCLUDING REMARKS

Two variants of FFNN, namely FFNN-BPN and FFNN-RLS have been presented and the performance of these algorithms have been evaluated for aircraft parameter estimation. The major difference between these two algorithms is in the method by which the errors are propagated back to update the network weights. The simulation results show that the estimation results of FFNN-RLS is better than FFNN-BPN by approximately 10%. Although the number of steps in the RLS algorithm is more than the BPN algorithm due to the computation of Kalman gain and covariance, iterations required for convergence have been observed to be lesser in FFNN-RLS making the algorithm faster on the whole. The conditions for asymptotic stability for both the techniques have been derived by proposing a Lyapunov energy function and using Lyapunov-Krasovskii's approach. The simulation results show that the estimation accuracy obtained by both the methods IS encouraging, and this paves the way to further consider these techniques for parameter estimation of fighter aircrafts and helicopters. Now, that the FFNN based approach is validated analytically as well as with simulated data, it can be further applied to more complex models.

REFERENCES

1. Raol, J.R.; Girija, G. & Singh, J. Modelling and parameter estimation of dynamic systems. The IEE/IET Control Series, 2004, **65**.
2. Raol, J.R.; Twala, B. & Gopalratnam, G. (2017). Nonlinear filtering: Concepts and engineering applications. CRC Press, Taylor & Francis, 2017.
3. Iliff, K.W. Parameter estimation for flight vehicles. *AIAA J. Guidance and Control*, 1989, **12** (5), 609-622. doi: 10.2514/3.20454
4. Milenković-Babić, M.; Samardžić, M.; Antonić, V.; Marjanović, M. & Stefanović-Gobeljić, V. Longitudinal stability characteristics of the LASTA airplane. *Aircraft Eng. and Aerosp. Technol.*, 2017, **89** (6), 911-919. doi: 10.1108/AEAT-02-2016-0026
5. Wolowicz, H. & Yancey, R.B. Longitudinal aerodynamic characteristics of light, twin-engine, propeller-driven airplanes. NASA Technical Note, 1972.
6. Nicolosi, F.; Corcione, S. & Della Vecchia, P. Commuter aircraft aerodynamic characteristics through wind tunnel tests. *Aircraft Eng. and Aerosp. Technol.: An Int. J.*, 2016, **88**(4), 523-534. doi: 10.1108/AEAT-01-2015-0008
7. Chauhan, K. & Singh, S. Review of aerodynamic parameter estimation techniques. *In International Conference on Infocom Technologies and Unmanned Systems (Trends and Future Directions) (ICTUS)*, 2017, 864-869. doi: 10.1109/ICTUS.2017.8286127
8. Chen, R.T.N.; Eulrich, B.J. & Lebacqz, J.V. Development of advanced techniques for the identification of V/STOL aircraft stability and control parameters. Cornell Aeronautical Lab., Inc., USA, Report No. BM-2820-F-1, 1971.
9. Jategaonkar, R.V. & Plaetschke, E. Algorithms for Aircraft Parameter Estimation Accounting for Process and Measurement Noise. *J. of Aircraft*, 1989, **26**(4), 360-372.
10. Garcfa-Velo, J. & Walker, B.K. Aerodynamic Parameter Estimation for High Performance Aircraft Using Extended Kalman Filtering. *J. of Guidance, Control, and Dynamics*, 1997, **20**(6), 1257-1259.
11. Chowdhary, G. & Jategaonkar, R. Aerodynamic Parameter Estimation from Flight Data Applying Extended and Unscented Kalman Filter. *Aerosp. Sci. and Technol.*, 2010, **14**(2), 106-117. doi: 10.1016/j.ast.2009.10.003
12. Julier, S.J. & Uhlmann, J.K. Unscented filtering and nonlinear estimation. *In* Proceedings of the IEEE, 2004, **92**(3), 401-422. doi: 10.1109/JPROC.2003.823141
13. Chaabane, M.; Baklouti, I.; Mansouri, M.; Jaoua, N.; Nounou, H.; Nounou, M. & Destain, M.F. Nonlinear state and parameter estimation using iterated sigma point kalman filter: Comparative studies. *Nonlinear Systems-Design, Analysis, Estimation and Control*, IntechOpen, 2016. doi: 10.5772/61494
14. Gutierrez-Villalobos, J. M.; Rodríguez-Reséndiz, J.; Rivas-Araiza, E. A. & Mucino, V. H. A review of parameter estimators and controllers for induction motors based on artificial neural networks. *Neurocomputing*, 2013, **118**, 87-100. doi: 10.1016/j.neucom.2013.02.018
15. Sadeghi-Goughari, M.; Mojra, A. & Sadeghi, S. Parameter

estimation of brain tumors using intraoperative thermal imaging based on artificial tactile sensing in conjunction with artificial neural network. *J. Phy. D: Appl. Phy.*, 2016, **49**(7), 075404.
doi: 10.1088/0022-3727/49/7/075404
16. Jamili, E. & Dua, V. Parameter estimation of partial differential equations using artificial neural network. *Comp. & Chem. Eng.*, 2021, **147**, 107221.
doi: 10.1016/j.compchemeng.2020.107221
17. Agrawal, S.; Gobiha, D. & Sinha, N. Nonlinear parameter estimation of airship using modular neural network. *The Aeronaut. J.*, 2020, **124**(1273), 409-428. doi:10.1017/aer.2019.125
18. Gabrielli, L.; Tomassetti, S.; Squartini, S.; Zinato, C. Introducing Deep Machine Learning for Parameter Estimation in Physical Modelling. *In* Proceedings of the 20th International Conference on Digital Audio Effects (DAFx-17), 2017.
19. Sinha, M.; Kalra, P.K. & Kumar, K. Parameter estimation using compensatory neural networks. Sadhana, 2000, **25,** 193–203.
doi: 10.1007/BF02703759
20. Yang ,Y. & Shi, Y. State and Parameter Estimation Algorithm for State Space Model Based on Linear Neural Network and Kalman Filter. *In IEEE International Conference on Mechatronics and Automation (ICMA),* 2314-2318, 2019.
doi: 10.1109/ICMA.2019.8816520.
21. Raol, J.R. & Jategaonkar, R.V. Aircraft parameter estimation using recurrent neural networks – a critical appraisal. *In* AIAA Atmospheric Flight Mechanics Conference, 1995, Paper No AIAA 95-3504.
doi: 10.2514/6.1995-3504
22. Alessandri, A; Cirimele, G.; Cuneo, M.; Pagnan, S. & Sanguineti, M. EKF learning for feedforward neural networks. *In European Control Conference (ECC), 2003,* 1990-1995.
doi: 10.23919/ECC.2003.7085258
23. Narendra, K. S. & Parthasarathy, K. Identification and control of dynamical systems using neural networks. *IEEE Trans. on Neural Networks*, 1990, **1**(1), 4– 27.
doi: 10.1109/72.80202
24. Raol, J.R; Madhuranath,H. Neural network architectures for parameter estimation of dynamical systems. *IEE Proceedings-Control Theory and Appl.*,1996, **143** (4), 387-394.
doi: 10.1049/ip-cta:19960338
25. Raol, J.R. Parameter estimation of state space models by recurrent neural networks. *IEE Proceedings-Control Theory and Applications*,1995, **142**(2), 114-118.
doi: 10.1049/ip-cta:19951733
26. Ogunmolu, O.; Gu, X; Jiang, S.; Gans, N.R. Nonlinear Systems Identification Using Deep Dynamic Neural Networks. *ArXiv, 2016.*
doi: 10.48550/arXiv.1610.01439
27. Rajagopal, R.B; Harika,P.; BhanuPrasad, C. & Sonam, S.K. State Estimation and Tracking using Recurrent Neural Networks. *Int. J. of Eng. Res. and Technol.*, 2017, **6**(5), 545-549.
doi: 10.17577/IJERTV6IS050294
28. Rhudy, M.B. Sensitivity and stability analysis of nonlinear Kalman filters with application to aircraft attitude estimation. West Virginia University, 2013.
29. Khalil, H. K. Lyapunov stability. Control systems, robotics and automation, 2009, **12**, 115.
30. Sheela, K.G. & Subramaniam N.D. Review on methods to fix number of hidden neurons in neural networks, *Math. Problems in Eng.*, 2013.
doi: 10.1155/2013/425740
31. Scalero, R.S. & Tepedelenlioglu, N. A fast new algorithm for training feedforward neural networks. *IEEE Trans. on Signal Process.*, 1992, **40**(1), 202-210.
doi: 10.1109/78.157194

CONTRIBUTORS

Ms Sara Mohan George is currently pursuing her PhD from Visvesvaraya Technological University. She is working as an Assistant Professor in the Department of Electronics and Communication, Ramaiah Institute of Technology, Bangalore. She received M. Tech and B.Tech degrees in Electronics and Communication from Visvesvaraya Technological University and Mahatma Gandhi University respectively. Her research interest includes nonlinear filtering, signal processing.
She has contributed towards the study and simulation of Aircraft example in MATLAB, deriving the stability conditions and generation of results. She as the first author has drafted & structured the paper.

Dr S. Sethu Selvi is Professor at Department of ECE, Ramaiah Institute of Technology. She obtained her PhD from the Indian Institute of Science in the year 2001. She completed her B. E from Thiagarajar College of Engineering, Madurai and ME from Anna University. She has numerous publications to her name in the field of Machine Learning, Pattern Recognition and Signal and Image Processing. Her fields of interests are Digital Image Processing, Machine/Deep Learning, Video Processing, Character Recognition and Biometrics. She has authored a chapter titled “Image Algebra and Image Fusion” in the book “Data Fusion Mathematics: Theory and Practice”, CRC Press, 2017 and has been listed as a noteworthy technical contributor by Marquis Who’s Who (World), 2009.
She has contributed towards checking the ANNs training alorithms, overall guidance to the first author and checking the entire MATLAB code and simulations.

Dr Jitendra R. Raol obtained his PhD from McMaster University, Canada, 1986. He completed his BE and ME degrees, in electrical engg., from M. S. University of Baroda, Vadodara (1971/1974). He worked in CSIR-NAL, Bangalore from 1975 to 1981 on human pilot modelling and from 1986 to 2007 on parameter estimation, filtering and data fusion. He retired as Scientist-G & Head, FMCD (CSIR-NAL) in July 2007. He has been a senior member of the IEEE (USA), a fellow of IEE/IET (UK), and he is a life fellow of Aero. Soc. of India, and a life member of Syst. Soc. of India. He has authored (singly and jointly) seven books, published by IEE/IET (UK) and CRC Press, USA. He has published nearly 150 research papers.
He has contributed towards conceptualization of the work, Lyapunov method approach for stability analysis, validation of the analytical results and the parameter estimation results; and overall text refinement and modifications.