# Identification of a non-linear F/A-18 model by the use of fuzzy logic and neural network methods

**N Boely, R M Botez***, and **G Kouba**
Laboratory of Active Controls, AeroServoElasticity and Avionics, Ecole de Technologie Supérieure, Montréal, Québec, Canada

*The manuscript was received on 24 January 2010 and was accepted after revision for publication on 22 September 2010.*

DOI: 10.1177/2041302510392871

**Abstract:** The aim of this article is to determine the mathematical model between control deflections and structural deflections in an F/A-18 modified aircraft in the active aeroelastic wing programme. One future application would be the design of a flutter suppression model based on flight flutter tests. Five excited sources were provided by NASA Dryden Flight Research Center from flight flutter tests. These excitations, given by aircraft control surfaces, are: differential and collective ailerons, collective and differential stabilizers, and rudders. The neural network and fuzzy logic algorithms were chosen in order to identify the multi-input multi-output system for the F/A-18 aircraft. One main contribution of this article is the mapping of fuzzy logic algorithm results into neural network data. Then, these methods were applied for the F/A-18 model identification and validation for sixteen flight conditions expressed in terms of Mach numbers variations between 0.85 and 1.30 and altitudes varying between 5000 and 25 000 ft. Accurate results were obtained, expressed in terms of fit coefficients between estimated and measured signals greater than 99 per cent, which allows one to conclude that these new methodologies are very efficient for an aircraft identification and validation.

**Keywords:** fuzzy logic, neural network, model identification, flight tests

## 1 INTRODUCTION

Aeroelasticity studies concern the investigation of the aircraft structural surface deformations of aircraft.

Both neural network (NN) field and flight flutter domain have been studied for the last two decades. Radial basis function network were used to map aircraft pitch dynamics. Pitch dynamics was modelled with a non-linear system based on the F-16 A/C non-linear model [**1**]. Online learning is used with the NN theory. De Weerdt *et al.* [**2**] used the NN model to compute input/output (I/O) mapping of F-16 A/C aerodynamic coefficients. They have found accurate results which were higher than 99 per cent. The mapping methods were used on a reconfigurable flight control for actuators failures and structural failures. I/O mapping were also built for unstable system identification [**3**]. The authors used the model reference indirect adaptive control and system theory to prove that a unique I/O mapping exists. Direct model reference adaptive controllers were established to control flexible space structures, such as aeroelastic wings [**4**]. A direct adaptive disturbance rejection control was also used to decrease flutter effects on aircraft structures.

Aeroelastic wings were also studied for their behaviour and for flutter suppression analysis. Castravete [**5**] modelled non-linear flutter with perturbation techniques and Monte–Carlo simulations whereas Gujjula [**6**] explored flutter suppression with robust and adaptive control in two degrees of freedom. Zink [**7**] explained a structural design methodology in order to use it on active aeroelastic wing (AAW). This structural design decreased the wing box skin weight.

NN algorithms are increasingly applied to aerospace identifications and controllers. Wu [**8**] discussed and explained the application of artificial intelligence for modelling and control. Neuro-fuzzy methods were developed in order to get intelligent systems to control active vibration on flexible structures [**9**]. This

**Corresponding author: Laboratory of Active Controls, AeroServo-Elasticity and Avionics, Ecole de Technologie Supérieure, Montréal, Québec, Canada H3C 1K3.*
*email: ruxandra@gpa.etsmtl.ca*

neural-fuzzy study highlighted the advantages of both techniques. He proposed a new strategy to simplify the architecture of the controllers.

In this article, a non-linear system is built that relates the control deflections due to pilot manoeuvres to structural deflections. This system is identified and validated from flight flutter tests for the NASA F/A-18 AAW Research Aircraft. The NN method is used first to identify the highly non-linear open-loop model for the F/A-18 aircraft. The open-loop identification model gives good results between 98.03 per cent and 99.84 per cent whereas, in closed loop, it gives poor results. Then, a fuzzy logic (FL) algorithm is used to optimize computational time, and finally to provide accurate NN results for 16 flight flutter test (FFT) cases characterized by various Mach numbers and altitudes. The closed-loop identification model gives more accurate results (higher than 99 per‘cent).

## 2 DEFINITION OF THE SYSTEM

The first six inputs considered in the work are the left and right aileron positions $AIL_L$ and $AIL_R$, the left and right stabilizer positions $STB_L$ and $STB_R$, and the left and right vertical tail positions $VERT_L$ and $VERT_R$. The four outputs are the left and right wing positions $WING_L$ and $WING_R$, and the left and right trailing edge flap positions $TEF_L$ and $TEF_R$. In order to improve the match between the model and the FFT data, the number of inputs was increased by adding nine non-linear inputs of second degree to the six original inputs, and then a total of 15 inputs were obtained [**10**].

In order to obtain the recorded FFTs data, the flight control computer for the F/A-18 AAW aircraft was modified by adding a research flight control system (RFCS) to generate the Schroeder frequency sweep control inputs. The software used by the RFCS to control the actuators was called the on-board excitation system (OBES). The Schroeder frequency sweep generated by the OBES is a sum of harmonics, equally spaced in the frequency domain. An example of the OBES control inputs time history is shown in Fig. 2.

The advantage of using Schroeder signal is that it minimizes its peak-to-peak amplitude, which is very useful in aircraft inputs excitation. The OBES-generated Schroeder signal is sent to the aircraft actuators to generate the F/A-18 control surfaces oscillations. For each flight test record, the excited control surfaces may be one of the following: *differential aileron*, *collective aileron*, *collective stabilizer*, *differential stabilizer*, or *rudders*. The outputs of the mathematical model are the structural deflections of both wings and trailing edge flaps. The tests were performed for a combination of Mach numbers from 0.85 to 1.20 and for altitudes from 5000 to 25 000 ft. At each flight condition, characterized by an altitude and a Mach number, all the five different excitation inputs mentioned above were performed to generate different records with a time length of 30 s. In order to capture all the system dynamics when building the mathematical model, manoeuvres corresponding to a

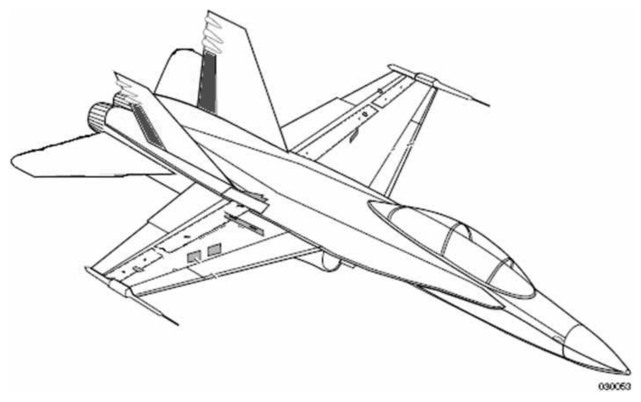

**Fig. 1** Left- and right-vertical tail positions as inputs on the F/A-18 aircraft

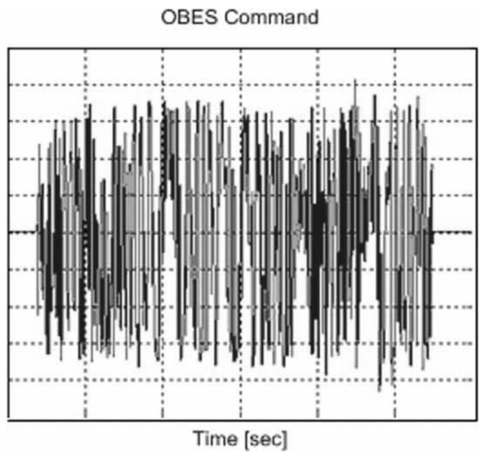

**Fig. 2** OBES control inputs versus time

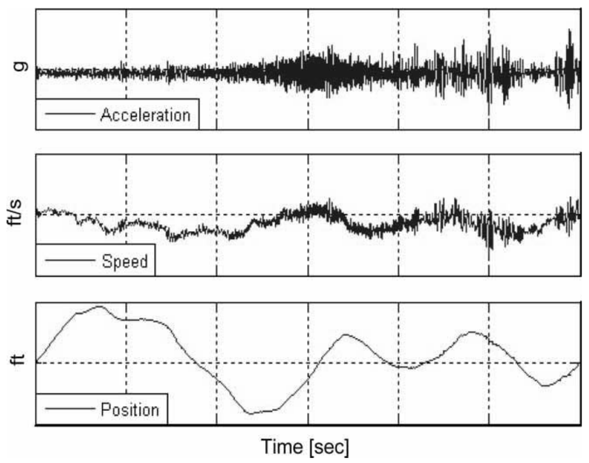

**Fig. 3** Left-wing accelerations and their single and double integrations with time, which give the velocity and the deflection in time

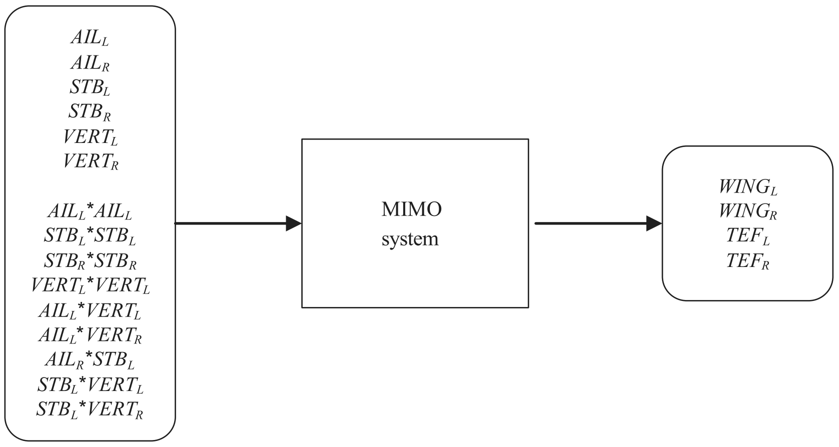

**Fig. 4** MIMO model with 15 inputs and four outputs

given altitude and Mach number were concatenated to generate a single 150 s time record.

In this article, the authors have used the measured structural accelerations provided by NASA Dryden. The measured accelerations on the structural surfaces are very noisy. The authors remove the noise in order to identify the F/A-18 AAW model by performing a double integration on the surface accelerations to obtain the surface deflections. The effect of integration on the accelerations is shown in Fig. 3 where velocity and position time histories are represented for the left-wing surface over the 150 s concatenated time interval. Only the structural surface deflection, velocity, and acceleration for the left wing are shown to illustrate the way in which integration operations remove the unwanted noise.

By observing Fig. 3, the double integration has produced smoother signals which can be easily estimated. The scheme for the multiple inputs–multiple outputs (MIMO) model with 15 inputs and four outputs is shown in Fig. 4.

**Table 1** Flight test cases as a function of Mach numbers and altitudes

| Flight case | Mach | Altitude (ft) |
|---|---|---|
| 1 | 0.85 | 5000 |
| 2 | 0.85 | 10 000 |
| 3 | 0.85 | 15 000 |
| 4 | 0.9 | 5000 |
| 5 | 0.9 | 10 000 |
| 6 | 0.9 | 15 000 |
| 7 | 1.1 | 10 000 |
| 8 | 1.1 | 15 000 |
| 9 | 1.1 | 20 000 |
| 10 | 1.1 | 25 000 |
| 11 | 1.2 | 10 000 |
| 12 | 1.2 | 15 000 |
| 13 | 1.2 | 20 000 |
| 14 | 1.2 | 25 000 |
| 15 | 1.3 | 20 000 |
| 16 | 1.3 | 25 000 |

Sixteen FFTs were realized at NASA Dryden Flight Research Center, for altitudes varying from 5000 to 25 000 ft, and for Mach numbers between 0.85 and 1.30. In Table 1, the FFT cases are numbered as a function of Mach number and altitude values.

## 3 ARCHITECTURE

The MIMO model is made of two main black boxes (see Fig. 5). Both are made of neurons. The only method which makes them different is how to calculate the NN data. The first black box is computed using the Levenberg–Marquardt back propagation (LMBP) algorithm. The second one is calculated using the Sugeno FL method, and the results are converted to NN data in order to implement them in neurons.

The proposed MIMO model is further detailed by five blocks (see Fig. 6). Its main characteristics are that the first block, Block 1, is compiled using an NN optimization algorithm (which is also Block 1 in Fig. 5), whereas the other four blocks (Block 2, Block 3, Block 4, and Block 5) are computed using an FL algorithm (shown by the Block 2 in Fig. 5).

Contrary to the classical FL method, the blocks computed using an FL algorithm in Fig. 6 are made up of neuron layers, in the same way in which blocks are normally computed using the NN method. A neuron layer is composed of a weight matrix **M**, a bias $b$, a sum $(+)$, and a function $F$ [**11**], as shown in Fig. 7.

From Fig. 7, it can be seen that the $n$th neuron layer is composed of the $n$th weight matrix $\mathbf{M}^n$, the $n$th bias $b^n$, the sum $(+)$, and the $n$th function $F^n$ [**11**]. Therefore, the $n$th output of the neural layer is given by the

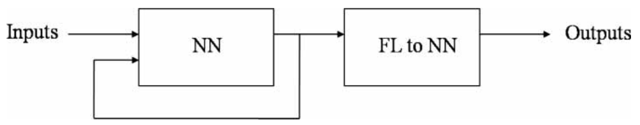

**Fig. 5** MIMO identification subsystem architecture

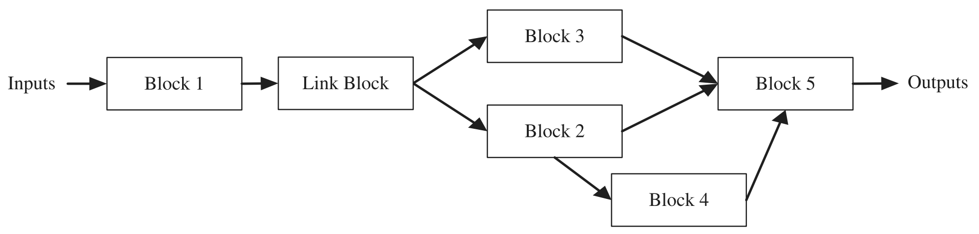

**Fig. 6** MIMO identification system black boxes

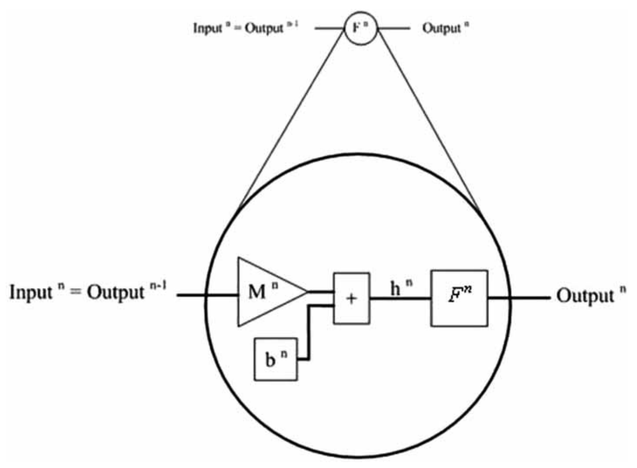

**Fig. 7** The $n$th neuron layer

following equations

$$\text{Output}^n = F^n(\mathbf{M}^n \text{Input}^n + b^n) \tag{1a}$$

$$\text{Output}^n = F^n(\mathbf{M}^n \text{Output}^{n-1} + b^n) \tag{1b}$$

If the previous data signals are model inputs (Input$^n$), equation (1a) is used, while if the previous data signals are provided by a previous layer (Output$^{n-1}$), equation (1b) is used.

## 4 THE LMBP METHOD

An $N$-layer network is used with $A$ neurons on the $n$th layer and $B$ neurons on the $(n+1)$th layer (see equation (10)) to explain the LMBP algorithm. From Fig. 7, it is seen that the $k$th network output of the $n$th layer is given by the following equation

$$\text{Output}_i^n = F^n(h_k^n) \quad k = 1, \ldots, A \tag{2a}$$

where

$$h_k^n = \sum_{i=1}^{n-1} (M_{k,i}^n \text{Output}_i^{n-1} + b_i^n) \tag{2b}$$

The performance $P$ of the gradient descent algorithm used in the LMBP method can be approximated using the following equation

$$P = \frac{1}{2} \sum_{q=1}^{Q} (z_q - \widehat{y}_q)^{\mathrm{T}} (z_q - \widehat{y}_q) \tag{3}$$

where $z_q$ is the desired output of the $q$th network, $\widehat{y}_q$ is the $q$th NN output, and $Q$ is the number of outputs.

Equation (3) can also be written in the following discrete time domain

$$P(t) = \frac{1}{2}[z(t) - \hat{y}(t)]^{\mathrm{T}}[z(t) - \hat{y}(t)] \tag{4}$$

At each iteration of a gradient descent algorithm, matrices and biases are updated. Therefore, the weight matrix $M^n$ and the bias $b^n$ at time $(t+1)$ are written as a function of $M^n$, $b^n$, and performance $P$ at previous time $t$ [**12**]

$$M_{k,i}^n(t+1) = M_{k,i}^n(t) - \eta \frac{\partial P(t)}{\partial M_{k,i}^n(t)} \tag{5a}$$

$$b_k^n(t+1) = b_k^n(t) - \eta \frac{\partial P(t)}{\partial b_k^n(t)} \tag{5b}$$

where $\eta$ is the network learning rate.

The current errors between training results and FFT data at time $t$ are reduced using the learning rate method. The global error reduction is created by modifying the weight matrices $M_{k,i}^n(t)$ until network accuracy, defined by the user, is reached. The weight matrices' values are kept frozen once global error reduction has been accomplished. The global error is represented by the performance function $P$.

For a multi-layered network, the updates of$M^n$ and $b^n$ at time $(t+1)$ (see equations (5a) and (5b)) are indirect functions of the previous weight matrices and biases at time $t$, and therefore they are calculated differently, using the following equations [**12**]

$$\frac{\partial P(t)}{\partial M_{k,i}^n(t)} = \frac{\partial P(t)}{\partial h_k^n(t)} \cdot \frac{\partial h_k^n(t)}{\partial M_{k,i}^n(t)} \tag{6a}$$

$$\frac{\partial P(t)}{\partial b_k^n(t)} = \frac{\partial P(t)}{\partial h_k^n(t)} \cdot \frac{\partial h_k^n(t)}{\partial b_k^n(t)} \tag{6b}$$

The derivatives of $h_k^n(t)$ given by equation (2b) are

$$\frac{\partial h_k^n(t)}{\partial M_{k,i}^n(t)} = \text{Output}_i^{n-1}(t) \tag{6c}$$

$$\frac{\partial h_k^n(t)}{\partial b_k^n(t)} = 1 \tag{6d}$$

Therefore, using equations (6c) and (6d), equations (6a) and (6b) become

$$\frac{\partial P(t)}{\partial M^n_{k,i}(t)} = \frac{\partial P(t)}{\partial h^n_k(t)} \cdot \text{Output}^{n-1}_i(t) \tag{6e}$$

$$\frac{\partial P(t)}{\partial b^n_k(t)} = \frac{\partial P(t)}{\partial h^n_k(t)} \tag{6f}$$

The performance sensitivity $s^n_k(t)$ is defined in the next equation [**12**]

$$s^n_k(t) = \frac{\partial P(t)}{\partial h^n_k(t)} \tag{6g}$$

$P(t)$ derivatives are necessary to update the new values of $M^n_{k,i}(t+1)$ and $b^n_k(t+1)$ (see equations (8a) and (8b)). Replacing the definition given by equation (6g) in equations (6e) and (6f), the following expressions for the $P(t)$ derivatives are obtained

$$\frac{\partial P(t)}{\partial M^n_{k,i}(t)} = s^n_k(t)\text{Output}^{n-1}_i(t) \tag{7a}$$

$$\frac{\partial P(t)}{\partial b^n_k(t)} = s^n_k(t) \tag{7b}$$

Using the derivatives given by equations (7a) and (7b), equations (5a) and (5b) can be rewritten as

$$M^n_{k,i}(t+1) = M^n_{k,i}(t) - \eta s^n_k(t)\text{Output}^{n-1}_i(t) \tag{8a}$$

$$b^n_k(t+1) = b^n_k(t) - \eta s^n_k(t) \tag{8b}$$

Moreover, the recurrence relationships for layer sensitivities are used, and therefore equation (6g) can be written as follows

$$s^n_k(t) = \frac{\partial P(t)}{\partial h^{n+1}_k(t)} \cdot \frac{\partial h^{n+1}_k(t)}{\partial h^n_k(t)} \tag{9}$$

The diagonal terms $((\partial h^{n+1}_k(t))/(\partial h^n_k(t)))$ of the Jacobian matrix $((\partial h^{n+1}(t))/(\partial h^n(t)))$ used in equation (9) is defined as follows

$$\frac{\partial h^{n+1}(t)}{\partial h^n(t)} = \begin{pmatrix} \frac{\partial h^{n+1}_1(t)}{\partial h^n_1(t)} & \cdots & \frac{\partial h^{n+1}_1(t)}{\partial h^n_A(t)} \\ \vdots & \ddots & \vdots \\ \frac{\partial h^{n+1}_B(t)}{\partial h^n_1(t)} & \cdots & \frac{\partial h^{n+1}_B(t)}{\partial h^n_A(t)} \end{pmatrix} \tag{10}$$

where $A$ is the number of neurons on the $n$th layer and $B$ is the number of neurons on the $(n+1)$th layer.

Based on equation (2b), the elements of the Jacobian matrix $((\partial h^{n+1}_i(t))/(\partial h^n_j(t)))$ are determined, where $i$=1, 2,...,$A$ and $j = 1, 2, \ldots, B$

$$\frac{\partial h^{n+1}_i(t)}{\partial h^n_j(t)} = M^{n+1}_{i,j}(t)\frac{\partial \text{Output}^n_j(t)}{\partial h^n_j(t)} \tag{11a}$$

which can be written in the following form, using equation (2a)

$$\frac{\partial h^{n+1}_i(t)}{\partial h^n_j(t)} = M^{n+1}_{i,j}(t)\,\dot{F}^n[h^n_j(t)] \tag{11b}$$

Equation (11b) is generalized for all terms as follows

$$\frac{\partial h^{n+1}(t)}{\partial h^n(t)} = M^{n+1}(t)\,\dot{F}^n[h^n(t)] \tag{11c}$$

where

$$\dot{F}^n[h^n(t)] = \begin{pmatrix} \dot{F}^n[h^n_1(t)] & 0 & 0 \\ 0 & \ddots & 0 \\ \cdots & & \cdots \\ 0 & 0 & \dot{F}^n[h^n_A(t)] \end{pmatrix} \tag{12}$$

Equation (9) can also be written in the following generalized form, also considering the properties of transpose matrices

$$s^n(t) = \frac{\partial P(t)}{\partial h^{n+1}(t)}\frac{\partial h^{n+1}(t)}{\partial h^n(t)} = \left(\frac{\partial h^{n+1}(t)}{\partial h^n(t)}\right)^{\mathrm{T}} \frac{\partial P(t)}{\partial h^{n+1}(t)} \tag{13a}$$

By replacing $((\partial h^{n+1}(t))/(\partial h^n(t)))$ and $s^{n+1}(t)$given by equations (11c), (12), and (6g), respectively, in equation (13a), the following equation is obtained

$$s^n(t) = \dot{F}^n[h^n(t)][M^{n+1}(t)]^{\mathrm{T}} s^{n+1}(t) \tag{13b}$$

where $n = 1, 2, 3, \ldots, N-1$ and $N$ is the number of layers. Then, $P(t)$ given by equation (4) is substituted into equation (6g) and the output layer $s^N(t)$ is obtained

$$s^N_k(t) = \frac{\partial P(t)}{\partial h^N_k(t)} = \frac{\partial\left[(1/2)\sum_{j=1}^{K}(z_j - \widehat{y}_j)^2\right]}{\partial h^N_k(t)} \tag{14a}$$

$$s^N_k(t) = \frac{\partial P(t)}{\partial h^N_k(t)} = \frac{\partial\left[(1/2)\sum_{j=1}^{K}(z_j - \widehat{y}_j)^2\right]}{\partial \widehat{y}_k}\frac{\partial \widehat{y}_k}{\partial h^N_k(t)} \tag{14b}$$

By differential properties, it can be written that

$$\frac{\partial\left[(1/2)\sum_{j=1}^{K}(z_j - \widehat{y}_j)^2\right]}{\partial \widehat{y}_k} = -(z_k - \widehat{y}_k) \tag{15}$$

Owing to the fact that the sensitivity of the last layer of neurons is calculated, the NN output $\widehat{y}_k$ is equal

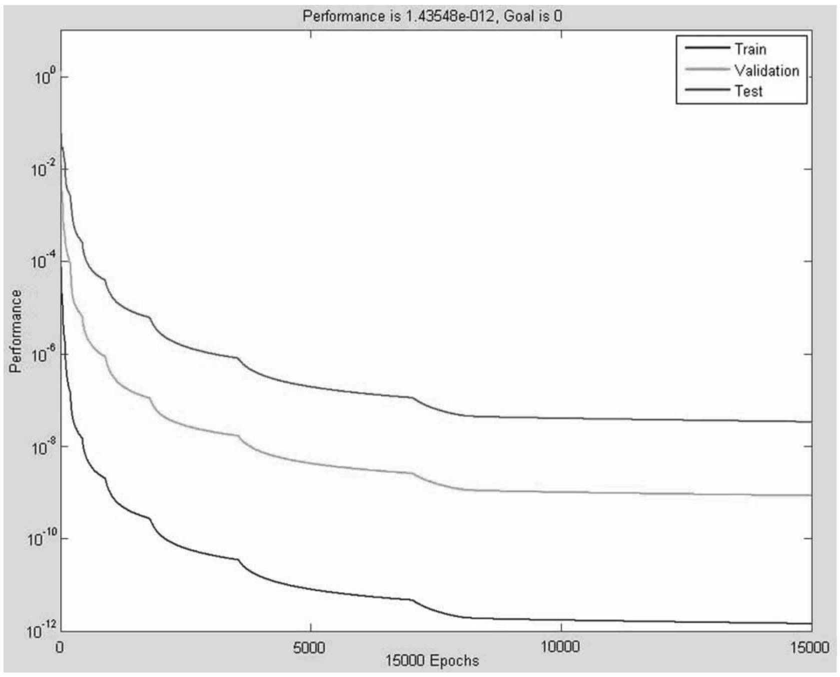

**Fig. 8** Performance evolutions during neural network training

to the $k$th network output of the $N$th layer. Then, equations (14b) and (15) are written as follows

$$s_k^N(t) = -(z_k - \widehat{y}_k)\frac{\partial \mathrm{Output}_k^N(t)}{\partial h_k^N(t)} \tag{16}$$

From equation (2a), it is obtained that

$$\frac{\partial \mathrm{Output}_k^N(t)}{\partial h_k^N(t)} = \dot{F}^N[h_k^N(t)] \tag{17}$$

Therefore, equation (16) can be rewritten as

$$s_k^N(t) = -(z_k - \widehat{y}_k)\,\dot{F}^N[h_k^N(t)] \tag{18}$$

Using equations (8a) and (8b), the new values of matrices and biases are calculated, while the sensitivity values for each output and each layer in real time are found using equations (13b) and (18).

Results expressed in terms of performance evolutions (three curves) are obtained using this algorithm, and are presented in Fig. 8. All input and output data are divided into three parts as described in the following paragraph. For the learning phase, all the input and output data are concatenated and split into three parts, which do not correspond to the third of the signals. The first part of the performance evolutions achieved with the first half (1/2) of the original signal is used to train the network (*blue* curve). The second part is the third fourth (1/4) of the original signal, which is used to validate the optimization of the training (*green* curve). The last part, which corresponds to the last fourth (1/4) of the signal, is used to test the efficiency of learning with unknown system data (red curve). Figure 8 represents the decreasing of the NN outputs error versus time. The train curve always has the best result. It can be explained because the first half of all input data are used to produce the network. The validation curve and the test curve have worse results than the train curve. The last two curves (test and train) are used to validate the model that explains the high differences between their errors.

With supervised learning, the artificial NN must be trained before it becomes useful. Training consists of giving input and output data to the network; these data are often referred to as the *training set*. For each input set provided to the system, the corresponding desired output set is also provided. This training is considered complete when the NN reaches a user-defined performance level, which signifies that the network has achieved the desired statistical accuracy as it produced the required outputs for a given sequence of inputs. The most important performance level concerns the test set. Notwithstanding high performance levels in the training and validation sets, the performance levels achieved by the network correspond to the test set performance levels. Only this parameter indicates whether the NN method can adapt online to changing input conditions. The validation set performance is always a little bit better than the test set performance. Thus, its analysis is less important.

## 5 NN ALGORITHM: BLOCK 1

Any system can by approximated by a two-layer NN with one non-linear function and one linear function [**11**]. The authors' goal is to approximate the FFT

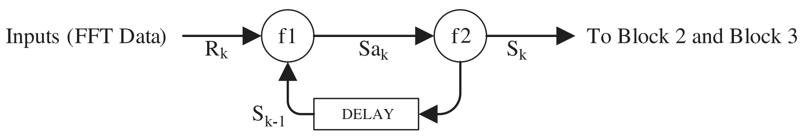

**Fig. 9** Block 1 architecture (closed loop)

model in an open loop. Block 1 consists of two layers expressed by the non-linear function $f_1$ and the linear function $f_2$. Each layer is composed in the same manner as shown in Fig. 7.

The first layer is composed of the real non-linear function $f_1$

$$f_1 : \mathbb{R} \to \mathbb{R}, \quad x \to \frac{1 - \mathrm{e}^{-2x}}{1 + \mathrm{e}^{-2x}} \tag{19a}$$

and of the real linear function $f_2$

$$f_2 : \mathbb{R} \to \mathbb{R}, \quad x \to x \tag{19b}$$

As each layer has a matrix $M$ and a bias $b$, the output equations given by Block 1 are

$$Sa_k = f_1(\mathbf{M}_1 S_{k-1} + \mathbf{M}_0 R_k + b_1) \tag{20a}$$

and

$$S_k = f_2(M_2 \mathrm{Sa}_k + b_2) \tag{20b}$$

By introducing $S_{k-1}$ from equation (20a) into S$a_k$ given by equations (20b), the output equation of the closed-loop Block 1 becomes

$$S_k = f_2[\mathbf{M}_2 f_1[\mathbf{M}_1 S_{k-1} + \mathbf{M}_0 R_k + b_1] + b_2] \tag{20c}$$

All matrix values ($\mathbf{M}_0$, $\mathbf{M}_1$, and $\mathbf{M}_2$), and bias values ($b_1$ and $b_2$) are calculated by the LMBP algorithm described in section 4 [**11**]. The open-loop Block 1 is trained to match the FFT data; however, this block is used to introduce a closed loop in the identification model. The closed-loop Block 1 is expressed by equation (20c). The output $S_k$ is a function of the previous output $S_{k-1}$ and the input $R_k$. The delay of the back propagation in the closed loop is given by the term $S_{k-1}$, as shown in Fig. 9.

The first layer should contain three, four, or five neurons according to the orthogonal least squares method

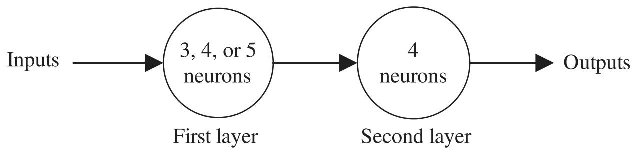

**Fig. 10** Open-loop Block 1 architecture

[**13**]. Block 1 identifies the FFT system in an open loop, while the second layer, which is the last, must contain a certain number of neurons as outputs. Therefore, the second layer has four neurons because there are four outputs, as shown in Fig. 10.

Thus, NN Block 1 is trained using the LMBP algorithm for 500 and 5000 iterations, and for three, four, and five neurons on its first layer.

The criterion used to evaluate the model's precision is the fit coefficient, which is expressed with equation (21)

$$\text{Fit coefficient} = 100\left(1 - \sqrt{\frac{\sum_1^n [z - \hat{y}]^2}{\sum_1^n [z - \text{mean}(z)]^2}}\right) \tag{21}$$

where $z$ is the measured output, $\hat{y}$ is the estimated output, and $n$ represents the number of points of the signal $y$.

The NN systems composed of three, four, or five neurons on the first layer for 500 and 5000 iterations are trained, and the fit coefficients are calculated. Results are expressed in terms of fit coefficients calculated for the left and right wing and left and right trailing edge versus the number of neurons (3, 4, and 5), ##are shown in Table 2 for 500 and 5000 iterations, respectively.

From Table 2, it is clear that the fit coefficients obtained using the NN method by the use of only three neurons on its first layer are smaller than the fit coefficients obtained using four or five neurons.

**Table 2** Values of fit coefficients (per cent) for 500 and 5000 iterations (iteration)

| | Three neurons | | Four neurons | | Five neurons | |
|---|---|---|---|---|---|---|
| | 500 iteration | 5000 iteration | 500 iteration | 5000 iteration | 500 iteration | 5000 iteration |
| Left wing | 28.16 | 28.17 | 99.26 | 99.99 | 99.34 | 99.98 |
| Right wing | 89.50 | 89.60 | 99.60 | 99.98 | 99.20 | 99.21 |
| Left trailing edge flap | 71.77 | 71.87 | 99.36 | 99.98 | 99.30 | 99.31 |
| Right trailing edge flap | 88.57 | 88.10 | 99.31 | 99.99 | 97.67 | 99.96 |

**Table 3** Training time (hours) versus the number of iterations for four and five neurons (hours)

| | Four neurons | Five neurons |
|---|---|---|
| 500 iterations | 0.21 | 0.32 |
| 5000 iterations | 2.14 | 3.14 |
| 15 000 iterations | 7.56 | 11.78 |

**Table 4** Fit coefficients for the number of 15 000 iterations versus the number of neuronss for $WING_L$, $WING_R$, $TEF_L$, and $TEF_R$

| | Three neurons | Four neurons | Five neurons |
|---|---|---|---|
| Left wing | 28.17 | 99.99 | 99.98 |
| Right wing | 89.60 | 99.98 | 99.21 |
| Left trailing edge flap | 71.87 | 99.98 | 99.31 |
| Right trailing edge flap | 88.10 | 99.99 | 99.96 |

Next, it is needed to decide between the use of four and five neurons on the first layer, based on the training time. In Table 3, the training time (hours) versus the number of neurons (4 and 5) is highlighted for 500, 5000, and 15 000 iterations. The results show that the training time with four neurons is always less time-consuming than with five neurons and, for this reason, four neurons are used on the first layer.

Results, expressed in terms of fit coefficients as functions of the number of neurons, are shown for the left wing, right wing, left trailing edge flap, and right trailing edge flap in Table 4.

From the results shown in Table 4, it is concluded that is better to use 5000 rather than 500 iterations in order to obtain more accurate fit coefficient values. It is not worth using 15 000 iterations, because the same results are obtained with only 5000 iterations; in addition, the execution time for 15 000 iterations is

**Table 5** Fit coefficients for 16 flight cases of the open-loop Block 1 (model results versus FFT data)

| Flight case | Mach number | Altitude | $WING_L$ | $WING_R$ | $TEF_L$ | $TEF_R$ |
|---|---|---|---|---|---|---|
| 1 | 0.85 | 5000 | 99.07 | 99.51 | 99.57 | 99.56 |
| 2 | 0.85 | 10 000 | 98.99 | 99.39 | 99.22 | 99.57 |
| 3 | 0.85 | 15 000 | 98.93 | 99.77 | 99.67 | 99.67 |
| 4 | 0.9 | 5000 | 99.65 | 99.71 | 99.72 | 99.72 |
| 5 | 0.9 | 10 000 | 98.50 | 99.76 | 99.74 | 99.69 |
| 6 | 0.9 | 15 000 | 99.47 | 99.84 | 99.47 | 99.76 |
| 7 | 1.1 | 10 000 | 99.60 | 99.40 | 99.74 | 99.37 |
| 8 | 1.1 | 15 000 | 99.32 | 99.69 | 99.59 | 99.66 |
| 9 | 1.1 | 20 000 | 99.68 | 99.52 | 99.55 | 99.60 |
| 10 | 1.1 | 25 000 | 99.74 | 99.22 | 98.94 | 98.52 |
| 11 | 1.2 | 10 000 | 98.86 | 98.03 | 98.94 | 99.09 |
| 12 | 1.2 | 15 000 | 99.30 | 99.67 | 99.56 | 99.68 |
| 13 | 1.2 | 20 000 | 99.41 | 99.70 | 99.65 | 99.77 |
| 14 | 1.2 | 25 000 | 99.77 | 99.17 | 99.44 | 99.76 |
| 15 | 1.3 | 20 000 | 99.36 | 99.74 | 99.36 | 99.74 |
| 16 | 1.3 | 25 000 | 98.90 | 98.99 | 99.58 | 99.80 |

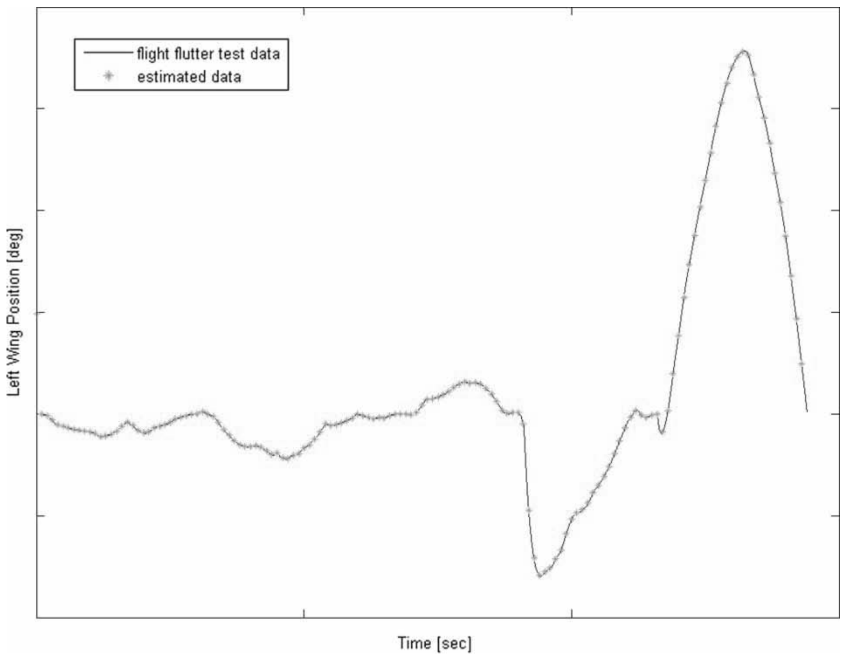

**Fig. 11** Time variation of the left wing $WING_L$ angle

3.5 times longer than for 5000 iterations, as shown in Table 3.

The fit coefficient values for 16 flight flutter cases are shown in Table 5, where it can be seen that the smallest fit coefficient is obtained for flight flutter case 11 (which is, for this reason, considered to be the worst case). In Figs 11 to 14, results are expressed in terms of time variations of aircraft surfaces angles (left and right wing, left and right trailing edge flap); the FFT data are shown in blue, while the model results are represented by green stars.

## 6 THE LINK BLOCK BETWEEN NN AND FL NETWORKS

The first block of the MIMO architecture (see Fig. 6), Block 1, is computed using the NN method, while the rest of the system identification is computed by the FL method. Thus, a method is found to connect the NN to the FL algorithm.

The FL network is composed of Block 2, Block 3, Block 4, and Block 5. Two cases are chosen using different types of inputs for the FL network. For the first

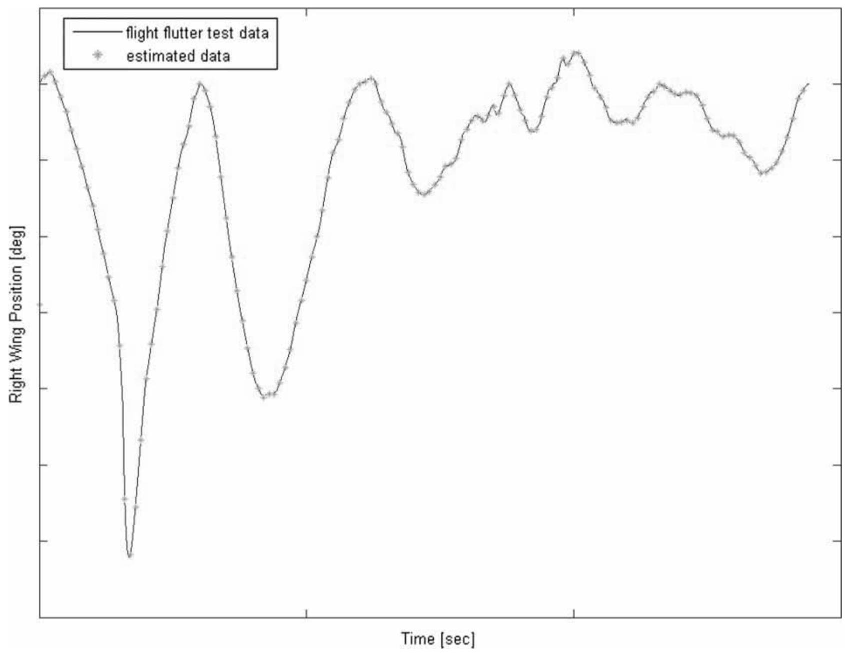

**Fig. 12** Time variation of the right wing WING$_R$ angle

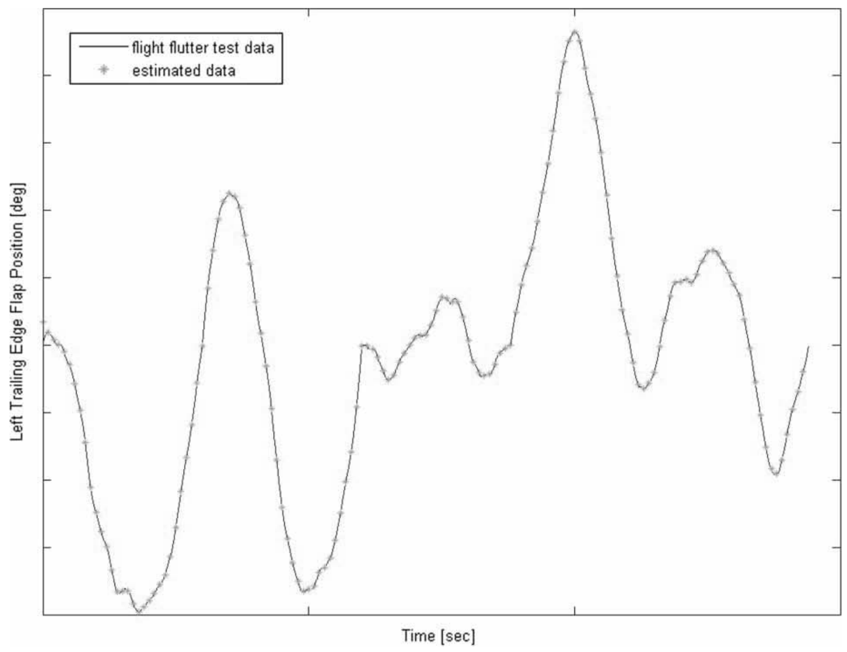

**Fig. 13** Time variation of the left trailing edge flap TEF$_L$ angle

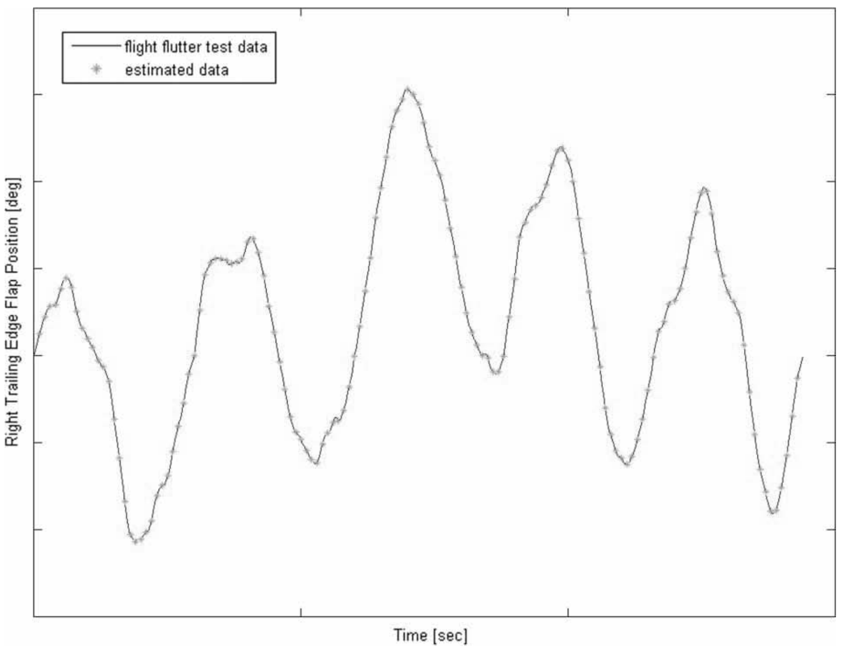

**Fig. 14** Time variation of the right trailing edge flap $TEF_R$ angle

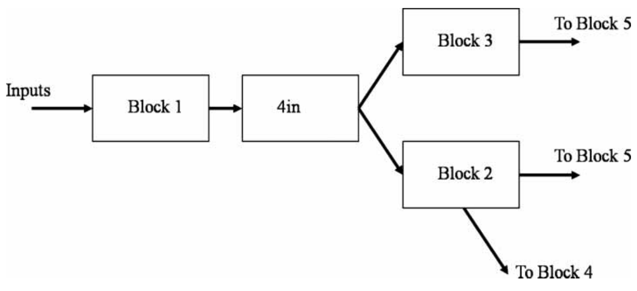

**Fig. 15** 4in link block architecture

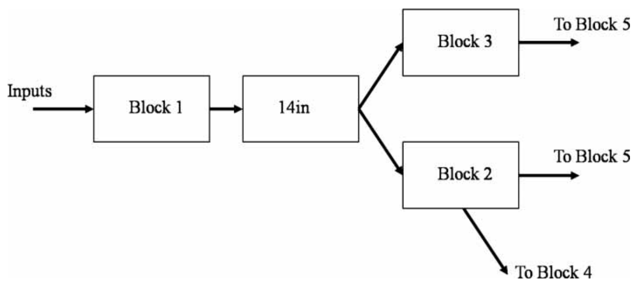

**Fig. 16** 14in link block architecture

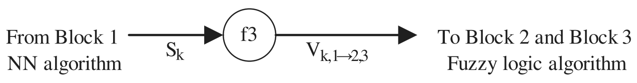

**Fig. 17** Link block architecture

case, denoted by 4in, the Block 1 outputs are chosen as Block 2 and Block 3 inputs. For the second case, denoted by 14in, then Block 1 outputs and ten Block 1 output products are chosen as Block 2 and Block 3 inputs (see Figs 15 to 17).

**Table 6** Fit coefficients obtained for $WING_L$ and $WING_R$ for both input cases

| | $WING_L$ | | $WING_R$ | |
|---|---|---|---|---|
| Flight case | 4in method | 14in method | 4in method | 14in method |
| 1 | 96.71 | 99.24 | 97.78 | 99.55 |
| 2 | 95.78 | 99.75 | 97.75 | 99.85 |
| 3 | 96.88 | 99.65 | 97.35 | 99.60 |
| 4 | 97.23 | 99.71 | 98.26 | 99.81 |
| 5 | 97.75 | 99.76 | 98.36 | 99.78 |
| 6 | 96.47 | 99.57 | 98.72 | 99.78 |
| 7 | 97.60 | 99.79 | 97.60 | 99.71 |
| 8 | 96.87 | 99.71 | 97.86 | 99.72 |
| 9 | 96.70 | 99.64 | 96.47 | 99.67 |
| 10 | 95.90 | 99.63 | 97.99 | 99.83 |
| 11 | 97.04 | 99.81 | 95.55 | 99.56 |
| 12 | 95.10 | 99.50 | 97.25 | 99.72 |
| 13 | 96.96 | 99.77 | 98.51 | 99.90 |
| 14 | 98.04 | 99.78 | 97.97 | 99.81 |
| 15 | 93.79 | 99.38 | 95.46 | 99.61 |
| 16 | 91.47 | 99.72 | 95.54 | 99.70 |

Then the FL and the NN algorithms are computed to determine which case gives the best system identification. The fit coefficients for $WING_L$ and $WING_R$ are shown in Table 6, and the fit coefficients for $TEF_L$ and $TEF_R$ are shown in Table 7 for both cases, 4in and 14in.

From Tables 6 and 7, it was found that the 14in fit coefficients were higher and thus, better than the 4in fit coefficients, which means that adding non-linear products to Block 1 outputs improves the fit coefficient values, for wings and trailing edge flaps.

Thus, fit coefficients are increased up to 99 per cent, from 1per cent to 8 per cent for any flight case and any output. For system identification, the 14in method was chosen as the link to connect Block 1 to Blocks 2 and 3.

**Table 7** Fit coefficients for the simulation results for $TEF_L$ and $TEF_R$ for both input cases

| | $TEF_L$ | | $TEF_R$ | |
|---|---|---|---|---|
| Flight case | 4in method | 14in method | 4in method | 14in method |
| 1 | 96.57 | 99.67 | 97.92 | 99.59 |
| 2 | 92.38 | 99.62 | 93.62 | 99.69 |
| 3 | 97.97 | 99.68 | 97.68 | 99.74 |
| 4 | 98.09 | 99.82 | 97.55 | 99.75 |
| 5 | 96.73 | 99.68 | 97.33 | 99.67 |
| 6 | 96.82 | 99.61 | 97.32 | 99.60 |
| 7 | 97.98 | 99.61 | 97.01 | 99.55 |
| 8 | 96.76 | 99.74 | 97.78 | 99.79 |
| 9 | 94.91 | 99.50 | 95.53 | 99.69 |
| 10 | 96.05 | 99.60 | 98.54 | 99.77 |
| 11 | 98.15 | 99.85 | 97.53 | 99.73 |
| 12 | 95.05 | 99.48 | 94.07 | 99.42 |
| 13 | 97.79 | 99.81 | 97.39 | 99.76 |
| 14 | 97.14 | 99.56 | 98.62 | 99.71 |
| 15 | 94.84 | 99.76 | 94.80 | 99.54 |
| 16 | 95.03 | 99.61 | 98.47 | 99.88 |

Consequently, the NNs are connected to the FL algorithms, using a neuron layer.

This layer is defined by the following equation

$$V_{k,1\to 2,3} = f_3(S_k) \tag{22}$$

where the function $f_3$ is defined as follows

$$\begin{aligned} &f_3 : \mathbb{R}^4 \to \mathbb{R}^{14}, \\ &(x_1, x_2, x_3, x_4) \to (x_1, x_2, x_3, x_4, x_1^2, x_2^2, x_3^2, x_4^2, x_1.x_2, \\ &\quad x_1.x_3, x_1.x_4, x_2.x_3, x_2.x_4, x_3.x_4) \end{aligned} \tag{23}$$

## 7 FL ALGORITHM

The FL method is used to identify non-linear relationships between input and output data. The FL algorithm used in the identification system is the Sugeno algorithm. This method calculates the memberships and the constant coefficients [**14**]. Their definitions are given next.

A membership is determined by a mathematical statistic law. The Gaussian distribution is chosen as the statistical law. In this method, linear combinations of statistical parameters are used to calculate non-linear system outputs. Thus, each membership is represented by a mean and a standard deviation.

This algorithm has three distinct parts. The standard deviations are computed in the first part, then the means are calculated, and, in the last part, the constant coefficients, which depend on previous values, are computed.

To prevent an excess of Gaussian distribution computation on signal data, three different parameters are stated. The first parameter, *cluster proximity* (CP), is used to keep two statistical laws from becoming too close to each other. The other two parameters, *accept ratio* (AR) and *reject ratio* (RR), are used to accept or reject data as Gaussian distribution means. Before computing statistical laws, all data must be gathered as shown in equations (24) and (25), and then normalized between 0 and 1 using equation (26)

$$\text{Xin} = \left[\text{inputs data}\right] \quad \text{Xout} = \left[\text{outputs data}\right] \tag{24}$$

where Xin is the concatenation of input data and, Xout is the concatenation of output data

$$\mathbf{X} = [\text{Xin} \quad \text{Xout}]_{[\text{numPoints}\times\text{numParams}]} \tag{25}$$

where $\mathbf{X}$ is the concatenated matrix of input and output data, numPoints is the data length, and numParams is the number of input and output signals

$$\mathbf{X}_{\text{normalized}} = \frac{\mathbf{X} - \min(\mathbf{X})}{\max(\mathbf{X}) - \min(\mathbf{X})} \tag{26}$$

To calculate these laws, a potential value, potVals, is allocated to each signal's data as follows [**15**]

$$\text{potVals}(i) = \sum_{i=1}^{\text{numPoints}} \mathrm{e}^{-4\sum_{j=1}^{\text{numParams}}(\mathrm{d}x_{i,j})^2} \tag{27}$$

where $\mathrm{d}x_{i,j}$ is the term of the $i$th row and the $j$th column of the matrix

$$\mathrm{d}x = \begin{bmatrix} \begin{array}{c} X(i,1) - X \\ (1,1) \end{array} & \cdots & \begin{array}{c} X(i, \text{numParams}) - X \\ (1, \text{numParams}) \end{array} \\ \cdots & \cdots & \cdots \\ \begin{array}{c} X(i,1) - X \\ (\text{numPoints}, 1) \end{array} & \cdots & \begin{array}{c} X(i, \text{numParams}) \\ -X(\text{numPoints}, \\ \text{numParams}) \end{array} \end{bmatrix}$$

The first high-potential value maxPotVal, called refPotVal, is used as a reference. For any iteration, the highest potential value maxPotVal is selected to calculate maxPotRatio as follows

$$\text{maxPotRatio} = \frac{\text{maxPotVal}}{\text{refPotVal}} \tag{28}$$

This ratio is used to determine whether the selected data are the mean of a Gaussian distribution. To know whether the selected data can be accepted, it is compared with two thresholds, AR, and RR. If the value of maxPotRatio is higher than AR, the data are accepted as a mean. If the maxPotRatio value is lower than RR, the data are rejected and the next data are chosen by the iterative process. However, if the maxPotRatio value is between AR and RR, a new value minDist must be computed

$$\text{minDist} = \sqrt{\begin{array}{c}[(\mathbf{X}(\text{maxPotRatio}) - \text{mean}) \\ \times\text{accumMultp}]^2\end{array}} \tag{29}$$

where accumMultp is an FL ratio which is a function of CP ratio.

This value highlights *two criteria* used to evaluate the validity of these data for identifying the system. On the one hand, this distance is used to determine whether the data have a *high enough potential value*. On the other hand, this value will avoid the selection of a new Gaussian distribution mean that is too close to the previous ones. If the value maxPotRatio + minDist is less than 1, the data are not a cluster and the potential value is set to zero. Otherwise, the data are added to the list of the cluster centres.

Then, all potential values are modified subsequently to these obtained results. Using an iterative method, the higher values are selected as the means, and the potential values are updated according to the membership values. The new potential values $\mathrm{potVals}_{\mathrm{new}}$ are functions of the previous potential values $\mathrm{potVals}_{\mathrm{old}}$, as shown in the following equations

$$\mathrm{potVals}_{\mathrm{new}} = \mathrm{potVals}_{\mathrm{old}} - \mathrm{maxPotVal}_{\mathrm{old}} \times \mathrm{e}^{-4\sum_{i=1}^{\mathrm{numParams}}(\mathrm{dd}x_i)^2} \tag{30}$$

where $\mathrm{dd}x_i$ is the $i$th term of the column vector $\boldsymbol{ddx} = (\mathbf{X}(\mathrm{maxPotRatio}) - \mathbf{X}) \times \mathrm{new\_sqshMultp}$ and, new_sqshMultp is an FL ratio which is a function of CP ratio.

If, after the updates of the potential values, a new value remains negative, it is immediately set to zero. The iterative loop for computing cluster centres stops when all potential values are equal to zero.

At the end of the algorithm, a defined number of Gaussian distributions are obtained and each law is composed of its mean and its standard deviation. Only the standard deviations of the previously calculated Gaussian distribution law are yet to be determined. All the standard deviations are computed as a function of the minimum and maximum of each signal's data before its normalization, as follows

$$\sigma_{[\mathrm{numParams}\times 1]} = \frac{\mathrm{CP}[\max(X) - \min(X)]}{2\sqrt{2}} \tag{31}$$

When this step is completed, the laws required to identify the non-linearities of the studied system are calculated. To finalize the identification with the Sugeno FL method, there is the need to calculate the constant coefficients relating linearly the input data with the output data.

Chiu [**16**] compares the fuzzy C-means (FCM) with the Gaussian definitions of the membership functions. The Gaussian membership function produces more accurate optimized models than the FCM. Hence, the FL rules are defined, using Gaussian membership functions, as follows

$$\mathbf{muVals}_{[\mathrm{numPoints}\times 1]} = \mathrm{e}^{-((X\mathrm{in}-\mathrm{means})/\sqrt{2}\cdot\mathrm{Std})^2} \tag{32}$$

The values **muVals** are used to define the normalized matrix **muMatrix** that highlights the linear rules between the inputs and the outputs, weighting each term according to its importance in the linear rules

$$\mathbf{muMatrix}(i, (j-1)(\mathrm{numInp}+1)+1 : (\mathrm{numInp}+1)j) = \left[\frac{X\mathrm{in}(i,:) \times \mathbf{muVals}(i, (\mathrm{numInp}+1)j)}{\sum_{k=1}^{\mathrm{numRule}} \mathbf{muVals}(i, (\mathrm{numInp}+1)k)} \quad \frac{\mathbf{muVals}(i, (\mathrm{numInp}+1)j)}{\sum_{k=1}^{\mathrm{numRule}} \mathbf{muVals}(i, (\mathrm{numInp}+1)k)}\right]_{[\mathrm{numPoints}\times(\mathrm{numRule}\cdot(\mathrm{numInp}+1))]} \tag{33}$$

It is possible to find the coefficients of the linear algebraic equation connecting the input to the output data, using this matrix. These coefficients outEqns are obtained by equation (34), since **muMatrix** is not square

$$\mathrm{outEqns} = (\mathrm{muMatrix}^{\mathrm{T}}\mathrm{muMatrix})^{-1}\mathrm{muMatrix}^{\mathrm{T}} \cdot \mathrm{Xout} \tag{34}$$

## 8 FROM FL RESULTS TO NN DATA

The rest of the identification system was compiled, not with an NN optimization algorithm, but with the FL algorithm. The new method involves mixing both algorithms and converting FL results into NN data; the FL compilation time is shorter than the NN compilation time.

Different numbers of memberships depend on the chosen flight case. The notation $R$ is chosen for the number of memberships, depending on the flight cases. All descriptions of matrices and biases are generalized. The authors choose the number of inputs $I = 14$ (due to the function $f_3$) and the number of outputs $\Theta = 4$.

### 8.1 Statistical coefficients: Block 2

First of all, in the FL algorithm, the authors calculate the statistic coefficients from Gaussian distributions (i.e. a mean ($\beta_i$) and a standard deviation ($\alpha_k$)). A Gaussian distribution $\Gamma$ given by a mean $\beta_i$ and a standard deviation $\alpha_k$ is defined as follows

$$\Gamma(\beta_i, \alpha_k) = \mathrm{e}^{-((x-\beta_i)/\alpha_k)^2} \tag{35}$$

For each pair of membership and input, a mean and a standard deviation are computed. The $f_2$ function is used to deduct the mean values, whereas the $f_4$ function layer is used to divide by the standard deviation. These Gaussian distribution results are provided by the $f_4$ outputs, $\mathrm{Yb}_{k,1}$.

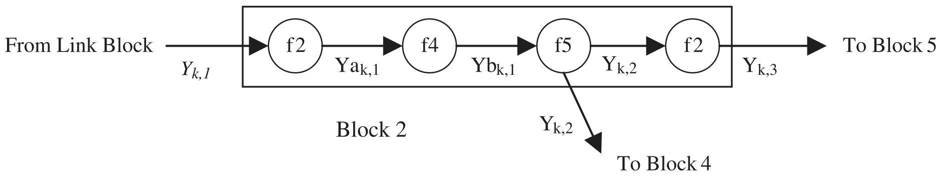

**Fig. 18** Block 2 architecture

The $f_5$ function gives the products of the inputs. Those results are used in Block 4 to calculate a global coefficient (see section 8.3) for all outputs.

In Fig. 18, Block 2 is summarized.

$\mathrm{Ya}_{k,1}$ and $\mathrm{Yb}_{k,1}$ are defined, and further the two outputs $Y_{k,2}$ and $Y_{k,3}$ are given by Block 2. These outputs are defined in Fig. 18, as follows

$$\mathrm{Ya}_{k,1} = f_2(\mathbf{M}_3 V_{k,1\to 2,3} + b_3) \tag{36a}$$

$$\mathrm{Yb}_{k,1} = f_4(M_4 \mathrm{Ya}_{k,1}) \tag{36b}$$

$$Y_{k,2} = f_5(M_5 \mathrm{Yb}_{k,1}) \tag{36c}$$

$$Y_{k,3} = f_2(M_6 Y_{k,2}) \tag{36d}$$

where the $f_4$ function is defined as a negative exponential function

$$f_4 : \mathbb{R} \to \mathbb{R}, \quad x \to \mathrm{e}^{-x^2} \tag{37}$$

and the $f_5$ function is defined as a product function

$$f_5 : \mathbb{R}^{R.I} \to \mathbb{R}^R, \quad \mathbf{X} \to Y$$
$$\text{telque } Y(i) = \prod_{k=(i-1)\cdot I+1}^{i.I} \mathbf{X}(k) \tag{38}$$

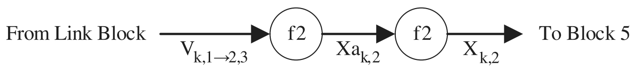

**Fig. 19** Block 3 architecture

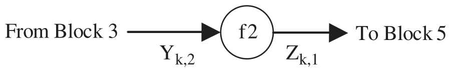

**Fig. 20** Block 4 architecture

## 8.2 Linear combinations: Block 3

The linear coefficients with mean and standard deviation values are computed here. Using the results obtained in the 14in link block, the coefficients of linear combinations ($\chi_{ik}$) are calculated for each input, and also constant coefficients $\gamma_i$ independent of the number of inputs. Actually, the outputs of the second $f_2$ function layer are combinations of linear inputs. Unfortunately, system identification is far from linear, and therefore not only Block 2 is required for system identification, but the other blocks also are.

From the Block 3 architecture seen in Figs 19 to 21, the first layer is defined as follows

$$\mathbf{Xa}_{k,2} = f_2(\mathbf{M}_7 \cdot V_{k,1\to 2,3} + b_4) \tag{39a}$$

The second layer, which has the same transfer function $f_2$ (see section 5), is defined as

$$\mathbf{X}_{k,2} = f_2(\mathbf{M}_8 \cdot \mathbf{Xa}_{k,2}) \tag{39b}$$

## 8.3 Global ratio coefficients: Block 4

The Block 4 contains only one function $f_2$. Its purpose is to compute a global coefficient, $Z_{k,1}$, to match the model's outputs with the FFT data. This global coefficient is calculated by linear computation of the input data. Each output has the same ratio between its real value and calculated value. Thus, the global coefficient has a unique value.

Block 4 is defined as a one-neuron layer block, written in the following form

$$\mathbf{Z}_{k,1} = f_2(\mathbf{M}_9 Y_{k,2}) \tag{40}$$

## 8.4 Final calculus: Block 5

The final calculus realized in the identification system method can be split into three steps. The first step consists of multiplying the Block 2 and Block 3 outputs.

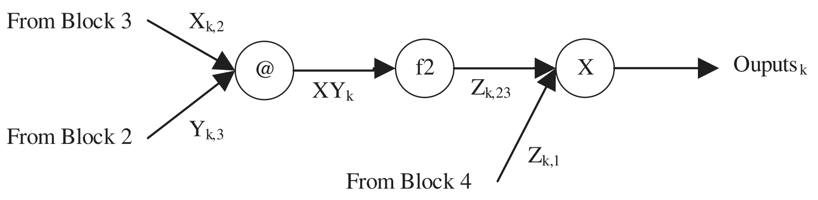

**Fig. 21** Block 5 architecture

Both outputs' vectors have the same size ($[\![R\Theta \times 1]\!]$). Each of the outputs' vector values are multiplied, term by term, in order to construct signals that are close to linear combinations of an FFT data signal.

The Block 5 architecture can be split into three parts. The first part gives the following equations from Block 2 and Block 3 outputs

$$\mathbf{X}Y_k = \mathbf{X}_{k,2} @ Y_{k,3} \tag{41a}$$

where the operator @ is defined as a term-by-term product as follows

$$A@B = C \leftrightarrow a_{ij}b_{ij} = c_{ij} \quad \text{with}$$
$$\dim(A) = \dim(B) = \dim(C)$$

The second layer can be written as a neuron layer under the following form

$$\mathbf{Z}_{k,23} = f_2(W_{10} \cdot \mathbf{X}Y_k) \tag{41b}$$

The third layer uses the global coefficient, $Z_{k,1}$, to compensate the ratio between the linear combination estimation of inputs and the FFT system in real time

$$\text{Outputs}_k = Z_{k,23} \times Z_{k,1} \tag{41c}$$

## 9 RESULTS

NN and FL algorithms were used for the 16 flight test cases. The size of each neuron layer depended on the computed flight test case. The fit coefficients (see section 5 – equation (21)) were calculated for the four outputs ($\text{WING}_\text{L}$, $\text{WING}_\text{R}$, $\text{TEF}_\text{L}$, and $\text{TEF}_\text{R}$) for all flight test cases. Table 8 was obtained.

**Table 8** Fit coefficients values for $\text{WING}_\text{L}$, $\text{WING}_\text{R}$, $\text{TEF}_\text{L}$, and $\text{TEF}_\text{R}$, for 16 flight test cases

| Flight case | $\text{WING}_\text{L}$ | $\text{WING}_\text{R}$ | $\text{TEF}_\text{L}$ | $\text{TEF}_\text{R}$ |
|---|---|---|---|---|
| 1 | 99.24 | 99.55 | 99.67 | 99.59 |
| 2 | 99.75 | 99.85 | 99.62 | 99.69 |
| 3 | 99.65 | 99.60 | 99.68 | 99.74 |
| 4 | 99.71 | 99.81 | 99.82 | 99.75 |
| 5 | 99.76 | 99.79 | 99.68 | 99.67 |
| 6 | 99.57 | 99.79 | 99.61 | 99.60 |
| 7 | 99.79 | 99.71 | 99.61 | 99.55 |
| 8 | 99.71 | 99.72 | 99.74 | 99.79 |
| 9 | 99.64 | 99.67 | 99.50 | 99.69 |
| 10 | 99.63 | 99.83 | 99.60 | 99.77 |
| 11 | 99.81 | 99.56 | 99.85 | 99.73 |
| 12 | 99.50 | 99.72 | 99.48 | 99.42 |
| 13 | 99.78 | 99.90 | 99.81 | 99.76 |
| 14 | 99.78 | 99.81 | 99.56 | 99.71 |
| 15 | 99.38 | 99.61 | 99.76 | 99.54 |
| 16 | 99.72 | 99.70 | 99.61 | 99.88 |

The worst-fit coefficient was obtained for the FFC (Mach number = 0.85 and altitude = 5000 ft) for the left-wing position ($\text{WING}_\text{L}$). The best-fit coefficient was obtained for the 13 flight case (Mach number = 1.2 and altitude = 20 000 ft) for the right-wing position ($\text{WING}_\text{R}$).

## 10 CONCLUSION

By using NN and FL algorithms, regardless of the flight test case, the fit coefficient error for FFT data were always small – less than 0.8 per cent (see Table 8). The computation time was optimized by converting FL results into NN data. It is concluded that the identification method is more accurate than the subspace identification method [10], because of the higher values of the fit coefficients calculated using the identification method.

## ACKNOWLEDGEMENTS

Thanks are dues to Mr Marty Brenner, from NASA Dryden Flight Research Center, for his collaboration on this work. This study was possible due to funds received from the National Sciences and Engineering Research Council of Canada (NSERC) and from the Ministère du Développement Economique, de l'Innovation et de l'Exportation (MDEIE).

## REFERENCES

1 **Ahmed-Zaid, F., Ioannou, P. A., Polycarpou, M. M.,** and **Youssef, H. M.** Identification and control of aircraft dynamics using radial basis function networks. In Proceedings of the IEEE International Conference on *Control applications*, Vancouver, Canada, 1993, vol. 2, pp. 567–572.

2 **De Weerdt, E., Chu, Q. P.,** and **Mulder, J. A.** Neural networks aerodynamic model identification for aerospace reconfiguration. In Proceedings of the AIAA Guidance, Navigation and Control Conference, San Francisco, California, USA, 2005, vol. 8, pp. 5962–5991.

3 **Suresh, S., Kannan, N., Omkar, S. N.,** and **Mani, V.** Time bounded neural network command 2004-773 control design for unstable aircraft. In Proceedings of the 42nd AIAA Aerospace Sciences Meeting and Exhibit, Reno, Nevada, USA, 2004, pp. 8884–8889.

4 **Mehiel, E. A.** *On direct model reference adaptative controller design for flexible space structure.* PhD, University of Colorado, Boulder, 2003, 217 pp.

5 **Castravete, S. C.** *Nonlinear flutter of a cantilever wing including the influence of structure uncertainties.* PhD, Wayne State University, 2007, 318 pp.

6 **Gujjula, S.** *Suppresion of limit cycle oscillations in an aeroelastic system using robust and adaptative control.* PhD, University of Nevada, Las Vegas, 2004, 94 pp.

7 **Zink, P. S.** *A methodology for robust structural design with application to active aeroelastic wing.* PhD, Georgia Institute of Technology, 2001, 282 pp.

8 **Wu, Y.** *Artificial intelligence methodologies for aerospace and other control systems*. DSc, Washington University, 1993, 165 pp.

9 **Ji, X.** *Intelligent systems for active vibration control in flexible engineering structures*. MSc, Lake head University, Canada, 2008, 135 pp.

10 **De Jesus Mota, S., Nadeau-Beaulieu, M., Botez, R.,** and **Brenner, M.** Modeling of structural deflections on a F/A-18 aircraft following flight flutter tests by use of subspace identification method. *Proc. IMechE, Part G: J. Aerospace Engineering*, 2007, **221**(G5), 719–731. DOI: 10.1243/09544100JAER0219.

11 **Hagan, M. T.** and **Demuth, H. B.** Neural networks for control, invited tutorial. In Proceedings of the 1999 American Control Conference, San Diego, 1999, pp. 1642–1656.

12 **Hagan, M. T.** and **Menhaj, M.** Training feed-forward networks with the Marquardt algorithm. *IEEE Trans. Neural Netw.*, 1994, **5**(6), 989.

13 **Yang, Z. J.** Hidden-layer size reducing for multilayer neural methods using the orthogonal least-squares method. *Trans. Soc. Instrum. Control Engr*, 1997, **33**(3), 216–223.

14 **Tagaki, T.** and **Sugeno, M.** Fuzzy identification of system and its application of modeling and control. *IEEE Trans. Syst. Man Cybern.*, 1985, **15**, 116–132.

15 **Chiu, S.** and **Liu, K.** Matlab, version R2008a, MathWorks, 2005.

16 **Chiu, S. L.** A cluster estimation method with extension to fuzzy model identification. In Proceedings of the IEEE International Conference on *Fuzzy system*, San Diego, California, 1994.

## APPENDIX 1

### Notation

| | |
|---|---|
| accumMultp | FL algorithm ratio which is a function of CP ratio |
| $\mathrm{AIL}_L$ | left aileron position |
| $\mathrm{AIL}_R$ | right aileron position |
| AR | AR for clusters centres |
| $b^n$ | bias of the *n*th layer |
| $b_i^n$ | bias coefficient for the *i*th input of the *n*th layer |
| $b_{p\_}$ | bias, $p = 0, 1, 2, 3, 4$ |
| CP | ratio for determining the proximity of two clusters centres |
| $f_p$ | transfer function, $p = 1, 2, 3, 4, 5$ |
| $F^n$ | transfer function of the *n*th layer |
| $\dot{F}^n$ | derivative function of the transfer function of the *n*th layer |
| $h^n$ | data of the *n*th layer before activation |
| $h_k^n$ | *k*th data of the *n*th layer before activation |
| $\mathrm{Input}^n$ | input of the *n*th layer |
| max(**X**) | maximum value of the variable **X** |
| maxPotRatio | ratio between the maximum value of the potential value and the reference potential value |
| maxPotVal | maximum value of the potential value |
| mean($X$) | mean value of the variable $X$ |
| min($X$) | minimum value of the variable $X$ |
| minDist | criterion taking in account all clusters means |
| muVals | exponential functions of Gaussian means and standard deviations |
| $\mathbf{M}^n$ | weight matrix of the *n*th layer |
| $M_{k,i}^n$ | weighting coefficient between the *i*th input and the *k*th output of the the *n*th layer |
| new_sqshMultp | FL algorithm ratio which is a function of CP ratio |
| numInp | size of the inputs' vector |
| numParams | size of the input and output data |
| numPoints | length of signals data in time |
| numRules | number of clusters |
| outEqns | matrix of the constant coefficients computed using the Sugeno FL algorithm |
| $\mathrm{Output}^n$ | output of the *n*th layer |
| $\mathrm{Output}_k^n$ | *k*th output of the *n*th layer |
| potVals | potential value of a signal's data |
| $P$ | performance coefficient |
| refPotVal | reference potential value of a signal's data |
| RR | Reject ratio for clusters centres |
| $s_k^n$ | performance sensitivity of the *k*th output of the the *n*th layer |
| $\mathrm{STB}_\mathrm{L}$ | left stabilizer position |
| $\mathrm{STB}_\mathrm{R}$ | right stabilizer position |
| Std | standard deviation of a Gaussian distribution |
| $\mathrm{TEF}_\mathrm{L}$ | left trailing edge flap position |
| $\mathrm{TEF}_\mathrm{R}$ | right trailing edge flap position |
| $\mathrm{VERT}_\mathrm{L}$ | left rudder position |
| $\mathrm{VERT}_\mathrm{R}$ | right rudder position |
| $W_p$ | weight matrix, $p = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10$ |
| $\mathrm{WING}_\mathrm{L}$ | left wing position |
| $\mathrm{WING}_\mathrm{R}$ | right wing position |
| Xin | input data matrix |
| Xout | output data matrix |
| $\hat{y}$ | estimated output signal |
| $\boldsymbol{z}$ | desired output vector |
| $\alpha_k$ | standard deviation of a Gaussian distribution |
| $\beta_i$ | mean of a Gaussian distribution |
| $\eta$ | learning rate |

## APPENDIX 2

$$\begin{array}{ll}
\mathbf{M}_3 = (m_{3_{ij}}) & b_3 = (b_{3_i}) \\
\dim(\mathbf{M}_3) = [RI \times I] & \dim(b_3) = [RI \times 1] \\
\forall i \in \llbracket 1; RI \rrbracket, \forall j \in \llbracket 1; I \rrbracket & \forall i \in \llbracket 1; RI \rrbracket \\
\text{if}(Rj - i) \in \llbracket 0; R-1 \rrbracket & b_{3_i} = -\beta_i \\
\quad \Rightarrow w_{3_{ij}} = 1 & \\
\text{else } m_{3_{ij}} = 0 &
\end{array}$$

$$\begin{array}{l}
\mathbf{M}_4 = (m_{4_{ij}}) \\
\dim(\mathbf{M}_4) = [RI \times RI] \\
\forall i \in \llbracket 1; RI \rrbracket, \forall j \in \llbracket 1; RI \rrbracket, \forall k \in \llbracket 0; I-1 \rrbracket \\
i \in \llbracket kR+1; (k+1)R \rrbracket \Rightarrow m_{4_{ij}} = \alpha_k \delta_{ij}
\end{array}$$

$$\begin{array}{l}
\mathbf{M}_5 = (m_{5_{ij}}) \\
\dim(\mathbf{M}_5) = [RI \times RI] \\
\forall i \in \llbracket 1; RI \rrbracket, \forall j \in \llbracket 1; RI \rrbracket \\
m_{5_{ij}} = \delta_{(i-RI)(RI-j)} + \delta_{(i-1)(1-j)} \\
\forall i \in \llbracket 1; RI-1 \rrbracket, \forall j \in \llbracket 1; RI-1 \rrbracket \\
m_{5_{ij}} = 1 \Rightarrow m_{5_{(i+I)(j+1)}} = 1 \text{ and } m_{5_{(i+1)(j+R)}} = 1
\end{array}$$

$$\begin{array}{l}
\mathbf{M}_6 = (m_{6_{ij}}) \\
\dim(\mathbf{M}_6) = [R\Theta \times R] \\
\forall i \in \llbracket 1; R\Theta \rrbracket, \forall j \in \llbracket 1; R \rrbracket \\
\text{if}(\Theta j - i) \in \llbracket 0; \Theta - 1 \rrbracket \Rightarrow m_{6_{ij}} = 1 \\
\text{else } m_{6_{ij}} = 0
\end{array}$$

$$\begin{array}{ll}
\mathbf{M}_7 = (m_{7_{ik}}) & b_4 = (b_{4_i}) \\
\dim(\mathbf{M}_7) = [R\Theta \times I] & \forall i \in \llbracket 1 \times R\Theta \rrbracket \\
\forall i \in \llbracket 1 \times R\Theta \rrbracket, \forall k \in \llbracket 1 \times I \rrbracket & b_{4_i} = \gamma_i \\
m_{7_{ik}} = \chi_{ik} &
\end{array}$$

$$\begin{array}{l}
\mathbf{M}_8 = (m_{8_{ij}}) \\
\dim(\mathbf{M}_8) = [R\Theta \times R\Theta] \\
\forall i \in \llbracket 1; R\Theta \rrbracket, \forall j \in \llbracket 1; R\Theta \rrbracket \\
m_{8_{ij}} = \delta_{(i-1)(1-j)} \\
\forall i \in \llbracket 1; R\Theta - 1 \rrbracket, \forall j \in \llbracket 1; R\Theta - 1 \rrbracket \\
m_{8_{ij}} = 1 \Rightarrow m_{8_{(i+\Theta)(j+1)}} = 1 \text{ and } m_{8_{(i+1)(j+R)}} = 1
\end{array}$$

$$\begin{array}{l}
\mathbf{M}_9 = (m_{9_j}) \\
\dim(\mathbf{M}_9) = [1 \times R] \\
\forall j \in \llbracket 1 \times R \rrbracket \\
m_{9_j} = 1
\end{array}$$

$$\begin{array}{l}
\mathbf{M}_{10} = (m_{10_{ij}}) \\
\dim(\mathbf{M}_{10}) = [\Theta \times R\Theta] \\
\forall i \in \llbracket 1 \times \Theta \rrbracket, \forall j \in \llbracket 1 \times R\Theta \rrbracket, \forall k \in \llbracket 1 \times (R-1) \rrbracket \\
m_{10_{ij}} = \delta_{\Theta k + i, j}
\end{array}$$