Systems Science, Vol. 36, No. 4, 2010

S. F. TOHA*, M. O. TOKHI*

ANFIS AND NEURAL NETWORK TECHNIQUES FOR NON-PARAMETRIC MODELLING OF A TWIN ROTOR SYSTEM

Interest in system identification especially for nonlinear systems has significantly increased in the past few decades. Soft-computing methods which concern computation in an imprecise environment have gained significant attention amid widening studies of explicit mathematical modelling. In this research, three different soft computing techniques that are multi-layered perceptron neural network using Levenberg–Marquardt (LM), Elman recurrent neural network and adaptive neuro-fuzzy inference system (ANFIS) network are deployed and used for modelling a twin rotor multi-input multi-output system (TRMS). The system is perceived as a challenging engineering problem due to its high nonlinearity, cross coupling between horizontal and vertical axes and inaccessibility of some of its states and outputs for measurements. Accurate modelling of the system is thus required so as to achieve satisfactory control objectives. It is demonstrated experimentally that soft computing methods can be effectively used for modelling the system with highly accurate results. The accuracy of the modelling results is demonstrated through validation tests including training and test validation and correlation tests.

Keywords: *Multi-layered perceptron neural network, Levenberg–Marquardt (LM), Elman recurrent neural network, adaptive neuro-fuzzy inference system, soft-computing, twin rotor multi-input multi-output system*

1. Introduction

System identification is the basis of designing control system, and it is very difficult to identify a nonlinear system today. In a model-based control framework, a pre-requisite to developing an effective control mechanism for a system is to model and predict the behaviours of the system based on given input–output data [1]. A high-fidelity system model is an important first step in control system design and analysis. A number of techniques have been devised by researchers to determine models that best describe input–output behaviour of a system. In model identification, traditional mathematical techniques are rather insufficient due to difficulty in the modelling of highly nonlinear components in the system. New methods of modelling based on soft computing techniques such as neural networks and fuzzy logic have shown promising results for modelling of nonlinear plants [2].

Soft computing is a practical alternative for solving computationally complex and mathematically intractable problems. The main components of soft computing namely neural network (NN) and fuzzy logic (FL) have shown great ability in solving complex nonlinear system identification problems [3, 4]. There has been an explosion in the literature on NN in the last decades or so, whose beginning was perhaps marked by the first IEEE International Conference on Neural Networks in 1987. It has been recognised that NN offer a number of potential benefits for applications in the field of control engineering, particularly for modelling non-linear systems. Some appealing features of NN are its ability for learning through examples, they do not require any *a priori* knowledge and can approximate arbitrary well any non-linear continuous function [5].

It is known that modelling and control of rotorcraft system is challenging due to its nature of nonlinearity, strong cross coupling between the two rotors and the difficulty in obtaining an accurate mathematical model. Previous researchers have successfully used feedforward multi-layered perceptron neural network (MLPNN) to model and control of rotorcraft system [6, 7, 8] using back propagation (BP) learning algorithm. However, it suffers from the problem of local minima and lower convergence rate because the gradient descent optimisation technique is used to mini-

* Department of Automatic Control and Systems Engineering, The University of Sheffield, UK, e-mail: cop06sft@sheffield.ac.uk

mise the error function [9]. A significant improvement on realization performance can be observed by using various second order approaches, namely Newton's method [10], conjugate gradient's [11], or the Levenberg–Marquardt (LM) [12] optimization technique. Among the methods mentioned, the LM algorithm is widely accepted as the most efficient one in the sense of realization accuracy [13]. The LM has also appeared to be the fastest method for training feedforward neural networks when the network contains no more than a few hundred weights [14]. It gives a good compromise between the speed of the Newton algorithm and the stability of the steepest descent method, and consequently it constitutes a good transition between these methods [15].

The limitation of using feedforward network is the absence of internal or hidden state, so that the processing of input patterns does not depend upon the order in which these patterns are represented during training process. This is called as static neural network, owing to the fact that these algorithms lack the necessary dynamical characteristics [16]. Elman recurrent neural network (ERNN) with internal dynamics has first been introduced by Elman [17] and is adopted in several recent works. Models with such network are shown to have the capability of capturing various plant nonlinearities and have a profound impact on the learning capability and performance of the network [18–20]. ERNN has shown more efficient than feedforward NN such as MLPNN and radial basis network in terms of the number of neurons required to model a dynamic system [21]. Furthermore, the level of error depends not only on the current parameter set, but also on the past parameter set [16]. Therefore, in this paper, a modified Elman recurrent neural network (ERNN) is proposed. It uses LM learning algorithm to calculate the error function with respect to the weights to perform the weight updates in times while the network runs as opposed to the standard network which is using gradient decent method.

An adaptive neuro-fuzzy inference system (ANFIS) is a combination between NN and fuzzy inference system (FIS) [22]. The objective of the synergy or hybridization (using neural networks and fuzzy logic) through ANFIS has been to overcome the weaknesses in one technology during its application, with the strengths of the other by appropriately integrating them. More often, the complexity surrounding a problem has called for a judicious combination of the technologies, when a technology individually applied has failed to obtain an efficient solution [23]. The advantages of fuzzy systems are: 1) Their ability to describe fuzzy rules which fit the description of real-world processor to a greater extent. 2) Their interpretability to explain why a particular value appeared at the output. In turn, some of the main disadvantages of fuzzy systems are that expert knowledge is needed to define fuzzy rules and it often requires long time to tune the parameters (e.g., parameters of membership functions) especially with a high number of fuzzy rules [24]. Fuzzy systems also lack the ability to learn and cannot adjust themselves to a new environment. Neural network on the opposite site has the ability to learn, where the user can train the network and the transformations of the variables are automated in the computational process. However, the major downside is that the individual relations between the input variables and the output variables are not developed by engineering judgment so that the model tends to be a black box without analytical basis [25]. Over this time, the research stream has gain momentum on ANFIS technique in time series prediction [26], control of flexible system [27], noise suppression [28], and clinical human model dynamics [29].

The modelling of a TRMS using neural networks is also reported in the literature. Ahmad et al. [30] have addressed nonlinear modelling of a TRMS using radial basis function network which presents nonlinear system identification method for modelling air vehicles of complex configuration. Dynamic modelling of a TRMS has also been presented in Aldebrez et al. [31], which has investigated the utilisation of NN with back propagation and parametric linear approaches for modelling the system. Feedforward neural network modelling of a TRMS has also been investigated by Shaheed [32]. Resilient propagation (RPROP) algorithm is used as learning algorithm to model the system. Rahideh et al. [33] have presented the modelling of a TRMS using both analytical and empirical approach. The NN model is proved to be superior to the analytical approach using Newtonian and Langrangian methods.

Therefore, this research will investigate the hovering position of a TRMS using real input–output data from the system. Such investigations in dynamic modelling of TRMS as well as similar system are very limited in the literature. It is also evident that the system has not been modelled using neural network with dynamic network, Elman recurrent NN and also adaptive neuro-fuzzy inference system (ANFIS). The rest of the paper is organised as follows. Section 2 describes the TRMS plant used in this work. Sections 3 through 5 describe the algorithms used in this work: a) MLPNN, b) ERNN and c) ANFIS. Non-parametric model identification validations are described in Sec-

tions 6 and 7, respectively. Results of comparison of the algorithms are presented, analysed and discussed well in Sections 8 and 9. Section 10 portrays the conclusion of the whole work. The responses of all the experiment based models are compared with those of the real TRMS to validate the accuracy of the model. Hence, the performances of the models are also compared with respect to each other. The models obtained for the TRMS will be used in subsequent investigations for the development of dynamic simulation, vibration suppression and control of the twin rotor system.

2. Twin rotor multi-input multi-output system

The TRMS is a laboratory set-up developed by Feedback Instruments Limited [34]. Due to the size, cost, ease of operation and interfacing facilities with personal computer, the TRMS has attracted many researchers and is being used as a "test rig" in aerodynamic control experiments. Although the TRMS does not fly, its behaviour in certain aspects resembles that of a helicopter. For example, like a helicopter it possesses a strong cross-coupling between the collective (main rotor) and the tail rotor. In a typical helicopter, the aerodynamic force is controlled by changing the attack of the blades. However, in the TRMS the aerodynamic force is controlled by varying the speed of the motors. It is driven by two DC motors. Its two propellers are perpendicular to each other and joined by a beam pivoted on its base that can rotate freely in the horizontal and vertical planes. The beam can thus be moved by changing the input voltage in order to control the rotational speed of the propellers. The articulated joint allows the beam to rotate in such a way that its ends move on spherical surfaces. The system is equipped with a pendulum counterweight hanging from the beam, which is used for balancing the angular momentum. Table 1 lists some characteristic parameters of the TRMS. A schematic diagram of the TRMS used in this work is shown in Fig. 1.

Table 1. Characteristic parameters of the TRMS.

Parameters	Value
Maximum input voltage range	+/–10 V
Length of the beam	0.49 m
Length of the counter balance	0.26 m
Mass of the main DC motor with main rotor	0.228 kg
Mass of the tail DC motor with tail rotor	0.206 kg

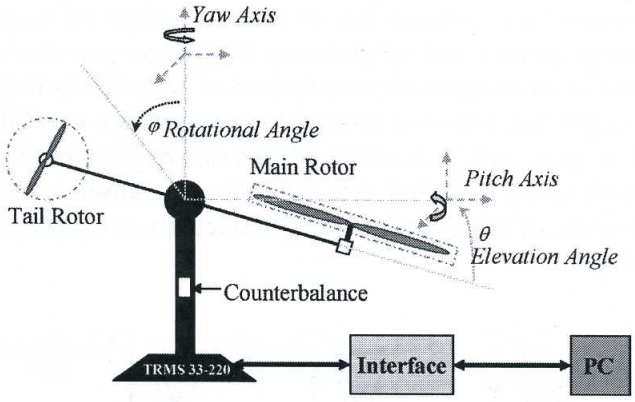


Fig. 1. Twin rotor multi-input multi-output system.

The system is balanced in such a way that when the motors are switched off, the main rotor end of the beam is lowered. The controls of the system are the supply voltages of the motors. It is important to note that the geometrical shapes of the propellers are not symmetric. Accordingly, the system behaviour in one direction is different from that in the other direction. Rotation of a propeller produces an angular momentum which, according to the law of conservation of angular momentum, is compensated by the remaining body of the TRMS beam. This results in interaction between the moment of inertia of the motors with propellers. This interaction directly influences the velocities of the beam in both vertical and horizontal planes. The system is interfaced with a personal computer through a data acquisition board, PCL-812PG. The measured signals are: position of the beam, which constitutes two position angles, and the angular velocities of the rotors. Angular velocities of the beam are obtained through software by differentiating and filtering the measured position angles of the beam. During the experimental procedure, the beam motion is allowed unrestricted in the pitch and the yaw planes, so that the motion of TRMS includes all the dynamic aspects of the system.

3. Multi-layered perceptron neural network

Multilayer perceptron (MLP) is a feedforward neural network consisting of an input layer, a number of hidden layers and an output layer. MLPNN is a computational model comprising numerous nonlinear processing elements arranged in patterns similar to biological neural networks. These computational models have now become exciting alternatives to

conventional approaches in solving variety of engineering and scientific problems. Multilayer perceptron NNs constitute an important class of NNs due to their simple topology and powerful approximation capability. They are supervised networks that require training with exact collected data [25]. A generalised architecture of MLP with its basic functions is shown in Fig. 2.

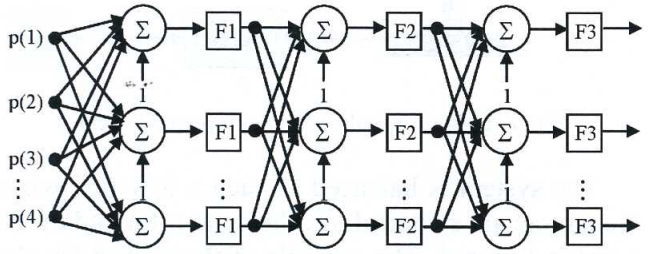


Fig. 2. A three-layered multiple layers of feedforward neural network.

The most common training methods in MLP is error backpropagation (BP) algorithm [35, 36]. In spite of the fact that BP is successfully used for various kinds of tasks, the training method is quenching for further improvement. One of the drawbacks of MLP-NN is the lack of standardization in choosing the number of hidden layers and hidden neurons per layer, which constitutes the architecture of an NN. Another drawback of using this architecture is that the NN with standard backpropagation learning method may get stuck in a shallow local minimum as the algorithm is based on the steepest descent (gradient) algorithm [37]. BP is also noted to have slow convergence and nonstability of convergence [38] as well as overfitting [39]. Since the local minimum is surrounded by a higher ground, once entered, the network usually does not leave a local minimum with a standard backpropagation algorithm.

3.1. Levenberg–Mardquardt learning algorithm

A highly popular algorithm known as the Levenberg–Marquardt algorithm is therefore employed to enable the MLPNN to slide through local minima and converge faster. It gives a good compromise between the speed of the Newton algorithm and the stability of the steepest descent method, and consequently it constitutes a good transition between these methods [15]. Consider a two-layered feedforward network such as the three layer network in Fig. 3 [14].

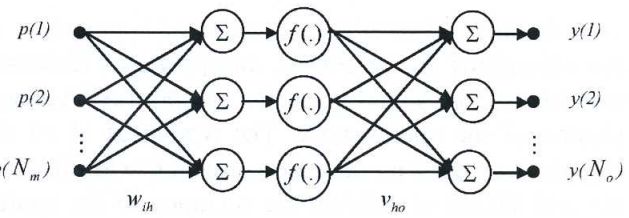


Fig. 3. Structure of a two-layered feedforward neural networks.

In the figure, the net input to unit h in the first layer is

$$z(h) = \sum_{i=1}^{N_m} w_{ih} x_i \,, \tag{1}$$

the net output from unit o in the second layer becomes

$$y(o) = \sum_{h=1}^{N_k} f(z(h)) v_{ho} \tag{2}$$

where w_{ih} is the weight between input x_i and hidden unit h, $f(.)$ is the activation function, v_{ho} is the weight between hidden unit h and output unit o, and N_k, and N_m are the numbers of hidden units and the dimension of input space, respectively. The three most commonly used activation functions are:

1. Sigmoid function: $f(x) = \dfrac{1}{1+e^{-x}}$. (3)

2. Hyperbolic tangent function:

$$f(x) = \tanh\left(\frac{x}{2}\right) = \frac{1-e^{-x}}{1+e^{-x}}. \tag{4}$$

3. Linear function: $f(x) = x$. (5)

The network can consist of any of the above activation functions with linear function normally used in the output layer, and sigmoid or hyperbolic tangent functions used in the hidden layers.

The network is to learn association between a fixed set of input/output pairs (P, T). In the Levenberg–Marquardt algorithm, the performance index for the network is

$$V = \frac{1}{2}\sum_{k=1}^{N_k} (t_k - y_k)^T (t_k - y_k) = \frac{1}{2}\sum_{k=1}^{N_k} e_k{}^T e_k \tag{6}$$

where t_k, y_k and e_k are the target, the output, and the error vector, respectively, when the k-th input vector p_k is presented.

Levenberg–Marquardt algorithm estimates the weight by approximating Newton's method. Assume that $V(w)$ required to be minimised with respect to

the parameter vector w, then Newton's method would be

$$\Delta \underline{w} = -[\nabla^2 V(w)]^{-1} \nabla V(w) \tag{7}$$

where $\nabla^2 V(w)$ is the Hessian matrix and $\nabla V(w)$ is the gradient. If we assume that $V(w)$ is a sum of square function

$$V(w) = \sum_{i=1}^{N} e_i^2(w) \tag{8}$$

then it can be shown that

$$\begin{aligned} \nabla V(w) &= J^T(w) e(w), \\ \nabla^2 V(w) &= J^T(w) J(w) + S(w), \end{aligned} \tag{9}$$

where $J(w)$ is the Jacobian matrix. The Jacobian matrix is calculated using the derivative of each error e_i with respect to each weight w or bias, and consequently the result will be an $N \times n$ matrix;

$$J(w(t)) = \begin{bmatrix} \frac{\partial e_1(w)}{\partial w_1} & \frac{\partial e_1(w)}{\partial w_2} & \cdots & \frac{\partial e_1(w)}{\partial w_n} \\ \frac{\partial e_2(w)}{\partial w_1} & \frac{\partial e_2(w)}{\partial w_2} & \cdots & \frac{\partial e_2(w)}{\partial w_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial e_n(w)}{\partial w_1} & \frac{\partial e_n(w)}{\partial w_2} & \cdots & \frac{\partial e_n(w)}{\partial w_n} \end{bmatrix} \tag{10}$$

and

$$S(w) = \sum_{i=1}^{N} e_i(w) \nabla^2 e_i(w) . \tag{11}$$

Then according to the Gauss–Newton method the update would be

$$\Delta w = [J^T(w) J(w)]^{-1} J^T(w) e(w) . \tag{12}$$

Thus, the Marquardt–Levenberg modification to the Gauss–Newton method is

$$\Delta w(t) = [J^T(w) J(w) + \mu I]^{-1} J^T(w) e(w) . \tag{13}$$

The new sum square error is then computed and compared to the previous one. If the performance becomes better, where the new error is less than the previous one, then the new parameters are defined as

$$w(t+1) = w(t) + \Delta w(t) \tag{14}$$

and the parameter μ is decreased using $\mu = \frac{\mu}{\beta}$. The new iteration is started if the stop criteria are not satisfied, otherwise μ is increased using $\mu = \mu\beta$ and $\Delta \underline{w}(t)$ is recalculated. A full description of Levenberg–Marquardt modification algorithm can be found in [40].

4. Elman recurrent neural network

Since all physical systems involve dynamics, modelling a physical system as a "natural" neural network should realistically include dynamical elements. Furthermore, from a control point of view, dynamical elements have to be included to have well posed problems [21]. If the feedback operation is added to static neural network, the network becomes dynamic neural network. It will make the network have short-term and long-term memories at the same time. The message at the present stage by way of feedback becomes the input information at the next stage or other neurons. It causes the time delay through feedback that network structure is used, and imitates the characteristic of brain which postpones and keeps the signal. Just like a biological information process system. Therefore, dynamic neural network can simulate complex systems such as time varying systems, which some static neural networks could not [16]. This dynamic neural network is called recurrent neural network (RNN). RNN offers a number of potential advantages over the use of static layered networks. RNN provides a means for encoding and representing internal or hidden states, albeit in a potentially distributed fashion, which leads to capabilities that are similar to those of an observer in modern control theory. RNN provides increasing flexibility for filtering noisy inputs. RNN feedback controllers are also more robust than static feedforward controllers to changes in plant dynamics and parameters [21].

Recurrent neural networks are different from feedforward network architecture in the sense that there is at least one feedback loop. Thus in those networks, there could exist one layer with feedback connections as well as there could also be neurons with self-feedback link where the output of a neuron is fed back into itself as the input. The presence of feedback loop has a profound impact on the learning capability of the network. Furthermore, these feedback loops involve the use of particular branches composed of unit delay elements that result in nonlinear dynamical behaviour by virtue of the nonlinear nature of neurons.

Nonlinear dynamics has a key role in the storage function of a recurrent network [15]. A representation

of a recurrent network as a block diagram is provided in Fig. 4. Contrary to feedforward networks, recurrent networks can be sensitive, and can be adapted to past inputs. Among the several NN architectures found in the literature, recurrent NNs (RNNs) involving dynamic elements and internal feedback connections have been considered as more suitable for modelling and control of nonlinear systems than feedforward networks [41].

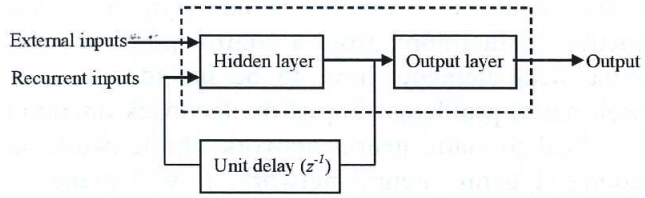


Fig. 4. Recurrent neural network architecture.

Elman [17] has proposed a partially RNN, where the feedforward connections are modifiable and the recurrent connections are fixed. It occupies a set of context nodes to store the *internal* states. Thus, it has certain unique dynamic characteristics over static NNs, such as the MLP-NN and radial basis function (RBF) networks [42]. The connections are mainly feedforward but also include a set of carefully chosen feedback connections that allow the network to remember cues from the recent past. The input layer is divided into two parts that are the true input units and the context units that hold a copy of the activations of the hidden units from the previous time step. As the feedback connections are fixed, backpropagation can be used for the training of the feedforward connections. The network is able to recognize sequences and also to produce short continuations of known sequences [17].

The structure of the Elman recurrent NN (ERNN) is illustrated in Fig. 5 where z^{-1} is a unit delay. Elman networks are two-layer backpropagation networks with addition of a feedback connection from the output of the hidden layer to its input. The Elman network has tansig neurons in its hidden (recurrent) layer, and purelin neurons in its output layer. This combination is special in that two-layer networks with these transfer functions can approximate any function (with a finite number of discontinuities) with arbitrary accuracy. The only requirement is that the hidden layer must have enough neurons. More hidden neurons are needed as the function being fitted increases in complexity [43]. It is easy to observe that the Elman network consists of four layers: input layer, hidden layer, context layer, and output layer. There are adjustable weights connecting every two adjacent layers. Generally, it can be considered as a special type of feedforward NN with additional memory neurons and local feedback [44]. The distinct self-connections of the context nodes in the Elman network make it sensitive to the history of input data, which is essentially useful in the modelling of dynamic systems [17].

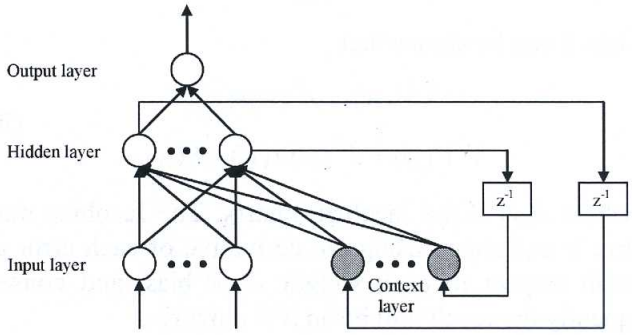


Fig. 5. Elman recurrent neural network architecture.

5. Adaptive neuro-fuzzy inference system

Neuro-fuzzy systems are hybrids of fuzzy systems and neural networks. The goal of neuro-fuzzy systems is to combine the learning capability of a neural network with the intuitive representation of knowledge found in a fuzzy system. This may be accomplished by designing network architecture to mimic a fuzzy system, by incorporating linguistic terms into the computations performed by the network, by means of an explanation mechanism for the network, and so forth [28]. ANFIS is the well-known neuro-fuzzy system, which mimics the operation of a Takagi–Sugeno–Kang (TSK) fuzzy system [45]. An ANFIS network makes use of a supervised learning algorithm to determine a nonlinear model of the input–output function, which is represented by a training set of numerical data where under proper conditions it can be used as a universal approximator [46].

The ANFIS uses a hybrid learning rule combining back-propagation, gradient-descent (GD), and a least squares error (LSE) algorithm to identify and optimize the Sugeno system's signals. The ANFIS type-3 fuzzy reasoning and the equivalent ANFIS architecture of a first order Sugeno fuzzy model with two inputs are shown in Fig. 6. The model has five

layers and every node in a given layer has a similar function. The architecture of ANFIS with two inputs, four rules and one output, where each input is assumed to have two associated membership functions (MFs). The four fuzzy IF-THEN rules can be expressed as [47]

$$\begin{aligned}
&R_1: \textbf{If } x_1 \text{ is } A_1 \textbf{ And } x_2 \text{ is } B_1 \textbf{ Then } f_{11}=a_{11}x_1+b_{11}x_2+c_{11}\\
&R_2: \textbf{If } x_1 \text{ is } A_1 \textbf{ And } x_2 \text{ is } B_2 \textbf{ Then } f_{12}=a_{12}x_1+b_{12}x_2+c_{12}\\
&R_3: \textbf{If } x_1 \text{ is } A_2 \textbf{ And } x_2 \text{ is } B_1 \textbf{ Then } f_{21}=a_{21}x_1+b_{21}x_2+c_{21}\\
&R_4: \textbf{If } x_1 \text{ is } A_2 \textbf{ And } x_2 \text{ is } B_2 \textbf{ Then } f_{22}=a_{22}x_1+b_{22}x_2+c_{22}
\end{aligned} \tag{15}$$

where x_1 and x_2 are the input variables to the ANFIS. A_1, A_2, B_1 and B_2 are the linguistic terms of input membership function for each rule. Also a_{ij}, b_{ij} and c_{ij}, where $i, j = 1, 2$, are the coefficients of output membership function f_{ij}. The node functions in the same layer are of the same function family as described below.

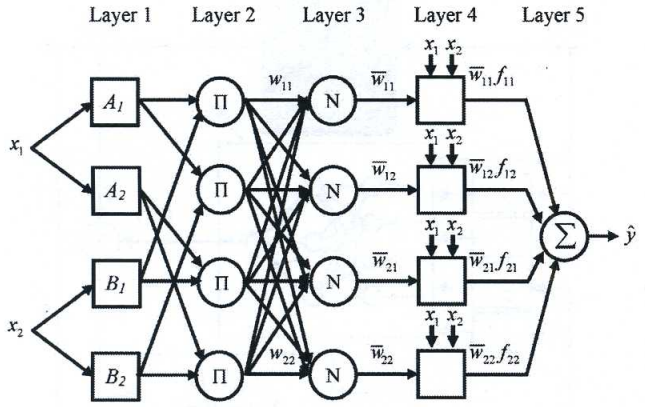


Fig. 6. Adaptive neuro-fuzzy inference system architecture.

Layer 1: is the **membership layer**. The output of any node in this layer will give the membership degree of an input (crisp). Every node i in this layer is a square node with a node function

$$\begin{aligned}
O^1_{A_i} &= \mu_{A_i}(x_1), \quad i=1,2,\\
O^1_{B_j} &= \mu_{B_j}(x_2), \quad j=1,2,
\end{aligned} \tag{16}$$

where x is the input to node I, and A_i and B_j the linguistic labels (such as big, small, etc.) associated with this node function. In other words, O^1_i is the membership function of A_i and it specifies the degree to which the given x satisfies the qualifier A_i. $\mu_{A_i}(x)$ is Gaussian shaped with the maximum equal to 1 and minimum equal to 0. The μ_{A_i} and μ_{B_j} are input MFs of Gaussian type of the form

$$\begin{aligned}
\mu_{A_i}(x_1, a_i, b_i, c_i) &= \exp\left\{-\frac{1}{2}\left(\frac{x_1-m_i}{\sigma_i}\right)^2\right\}, \quad i=1,2,\\
\mu_{B_j}(x_2, a_j, b_j, c_j) &= \exp\left\{-\frac{1}{2}\left(\frac{x_2-m_j}{\sigma_j}\right)^2\right\}, \quad j=1,2,
\end{aligned} \tag{17}$$

where $\{m_i, \sigma_i\}$ and $\{m_j, \sigma_j\}$ represent the centre and width of the fuzzy sets A_i and B_j, respectively. Parameters in this layer are referred to as *antecedent* parameters.

Layer 2: is the **multiplication layer**. Every node in this layer multiplies the inputs of membership degrees with a circle node labelled Π and produces the firing strength of the rule or the degree in which the corresponding rule is fired. For instance,

$$O^2_{ij} = w_{ij} = \mu_{A_i}(x_1)\mu_{B_j}(x_2), \quad i,j=1,2, \tag{18}$$

where each node output represents the firing strength of a rule.

Layer 3: is just a **normalisation layer**. Every node in this layer is a circle node labelled **N**. The i-th node calculates the ratio of the particular rule-firing degree to the sum of all rule degrees,

$$O^3_{ij} = \overline{w}_{ij} = \frac{w_{ij}}{\sum_{i=1}^{2}\sum_{j=1}^{2} w_{ij}}, \quad i,j=1,2. \tag{19}$$

Outputs of this layer are known as *normalized firing strengths.*

Layer 4: applies Sugeno's processing rule. It is also the **defuzzification** layer. Every node i in this layer is a square node with a node function

$$O^4_{ij}\overline{w}_{ij}f_{ij} = \overline{w}_{ij}(a_{ij}x_1 + b_{ij}x_2 + c_{ij}), \quad i,j=1,2, \tag{20}$$

where $\overline{w}_{ij}$ is the output of Layer 3, and $\{a_{ij}, b_{ij}, c_{ij}\}$ are the parameter set. Parameters in this layer are known as *consequent parameters.*

Finally, the single node of **layer 5**: calculates the overall output as the sum of all incoming signal. The single node in this layer is a circle node labelled Σ that computes the overall output as the summation of all incoming signals, i.e.,

$$O^5 = \hat{y} = \sum_{i=1}^{2}\sum_{j=1}^{2} \overline{w}_{ij}f_{ij}, \quad i,j=1,2. \tag{21}$$

The hybrid learning algorithm using ANFIS technique, thus described can be formulated as:

Input data is divided into two parts
Initialise input data (training data)
Assign number of iterations (epoch) and tolerance (error) value
Repeat
Forward-pass learning using LSE optimisation
Calculate value of consequent parameters (equation 20)
Backward-pass learning using gradient descent optimisation
Calculate value of antecedent parameters (equation 17)
Until convergence criteria is satisfied
Validate with independent data (checking data)

6. Non-parametric modelling

Black box and some non-parametric modelling techniques are gaining considerable interest among researchers due to their simplicity [48]. Non-parametric identification methods are techniques to estimate model behaviour without necessarily using a given parameterised model set. These methods avoid aerodynamics/kinetics of the system and can derive model based on data collected at several input–output terminals. The success of such approaches depends on pre-experimentation techniques through which the system is expected to reveal all the important dynamics, suitable selection of sensors to measure signals. Data acquisition system, accuracy of collected data, and all the estimation or prediction technique, commonly known as algorithm.

6.1. Model structure

The most basic relationship between the input and output of a nonlinear system can be represented in a non-linear autoregressive model with exogenous inputs (NARX) form [48], as given by

$$\hat{y}(t) = f[u(t-1), u(t-2), ..., u(t-n_u), y(t-1), y(t-2), ..., y(t-n_y)] \quad (22)$$

where $\hat{y}(t)$ is the predicted output and $f(.)$ is some vector valued nonlinear function of $y(t)$ and $u(t)$. n_u and n_y are the maximum lags in the output and input vectors. For this work, the input $u(t)$ indicating the voltage of main rotor, $V_v(t)$, the output $y(t)$ indicating the actual pitch angle of the beam and $\hat{y}(t)$ is the predicted pitch angle of the beam.

The NARX model structure is shown in Fig. 7. The signal vector applied to the input layer of the non-parametric model consists of a data window made up of the following components:

(i) Present and past values of the input, namely, $u(t-1)$, $u(t-2)$, ..., $u(t-n_u)$, which represent *exogenous* inputs originating from outside the network;

(ii) Delayed values of the output, namely, $y(t-1)$, $y(t-2)$, ..., $y(t-n_y)$, on which the predicted model output $\hat{y}(t)$ is *regressed.*

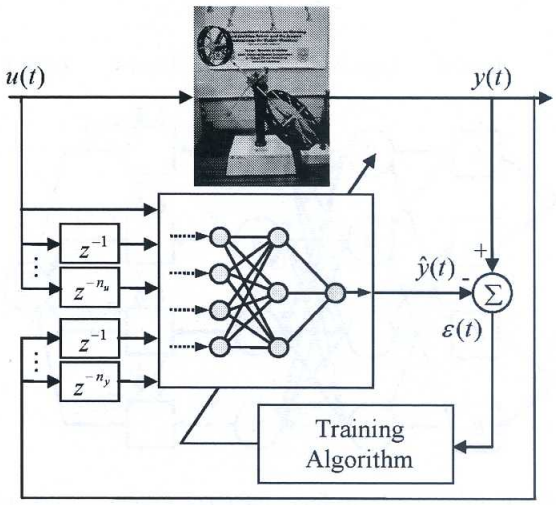


Fig. 7. NARX model structure for non-parametric modelling.

6.2. Excitation signal

The TRMS set-up is very sensitive to atmospheric disturbance, hence it was ensured that the identification experiments are conducted in calm air. The test signal was designed separately and read from the workspace in the MATLAB/Simulink environment. This is analogous to automation of the test signal, which ensures the experiments to be sufficiently controlled, be repeatable, and guarantees the desired spectral content.

Theoretically, the TRMS will have an infinite number of resonances with associated frequencies. It is observed from the power spectral density of the system that the significant mode lies in the 0–1 Hz bandwidth, with the main resonance mode at 0.34 Hz

which can be attributed to the main body dynamics [30]. During the experimental procedure, the beam motion is allowed unrestricted in the pitch and the yaw planes, so that the motion of TRMS includes all the dynamic aspects of the system.

To investigate variations in the detected vibration modes of the TRMS, modelling exercise is carried out based on the actual system response to the applied excitation signal. A pseudo-random binary signal (PRBS), covering the dynamic range of interest of the TRMS flexible system is used as an excitation signal. This experiment was carried out using a sampling time of $T = 0.1$ s with 5 Hz bandwidth. Input–output data necessary to perform parametric identification were collected, where a 5 Volt input PRBS indicates the voltage of main rotor, $V_v(t)$ was used to excite the TRMS system. The output data collected was the pitch angle of the beam, $\alpha_v(t)$.

The measured data was sampled and recorded on a PC using the real-time kernel (RTK) software. To overcome the problem of obtaining biased model, the empirical input and output data are divided into two data sets, 1) training data and 2) checking data sets. The training data set is fed into the algorithm to develop a model. The model usually tracks the system output well and converges to a target error value. However, the checking data, which is different from the training data set, is presented to the developed model to verify its performance against the actual data. The optimisation function utilised is the mean squared error (MSE) between the actual output $y(t)$ of the system and the predicted output $\hat{y}(t)$, for N number of data.

$$\text{Objective function} = \frac{1}{N}\sum_{t=1}^{N}\left|y(t)-\hat{y}(t)\right|^2. \quad (23)$$

6.3. MLPNN modelling using Levenberg–Marquardt training algorithm

The TRMS is modelled using an MLP-NN with a configuration of two-layered network with 2 × 1 neurons. The neural network was trained by using a sound Levenberg–Marquardt algorithm. The NN-based model is designed to have 5 inputs, 2 neurons in the hidden layer and 1 neuron in the output layer. Figure 8 shows the MLPNN structure for modelling the TRMS. The input data structure comprises the voltage of main rotor at present time, $V_v(t)$, voltage of main rotor at previous time, $V_v(t-1)$, voltage of main rotor at 2 previous sample times, $V_v(t-2)$, pitch angle of the beam at previous time, $\alpha_v(t-1)$, pitch angle of the beam at 2 previous sample times, $\alpha_v(t-2)$. In this non-parametric modelling, the activation functions used for the hidden and output layers are sigmoid and linear functions, respectively.

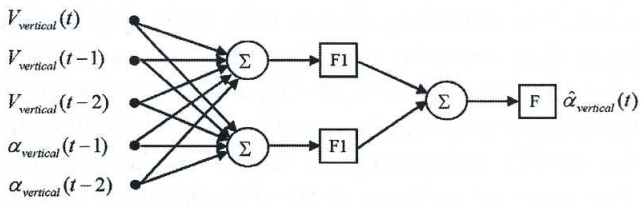


Fig. 8. MLPNN structure for modelling the TRMS.

6.4. ERNN modelling using Levenberg–Marquardt training algorithm

The functionality of the recurrent network is determined by specifying the choice of network architecture, that is, the number and type of neurons, the location of feedback loops and the development of a suitable training algorithm. The Elman network was achieved with a configuration of two hidden layers, each having two sigmoid neurons and one output layer with linear neuron. The neural network was trained by using a sound Levenberg-Marquardt algorithm. This architecture was obtained by trial and error process in order to achieve a good result. Figure 9 shows the ERNN structure for modelling the TRMS. The input data structure comprises the voltage of main rotor at present time, $V_v(t)$, voltage of main rotor at previous time, $V_v(t-1)$, pitch angle of the beam at previous time, $\alpha_v(t-1)$. In this non-parametric modelling, the activation functions used for the hidden and output layers are sigmoid and linear functions, respectively.

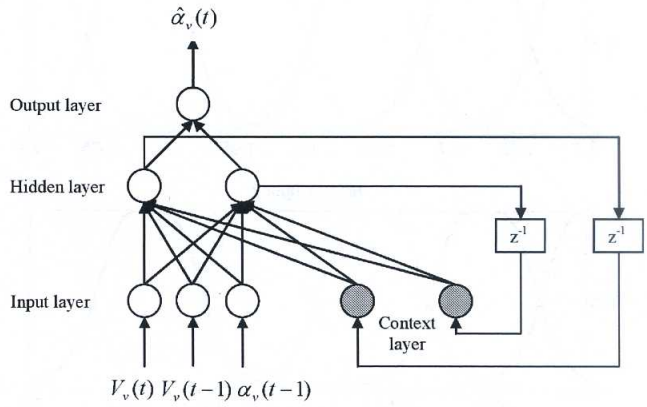


Fig. 9. ERNN structure for modelling the TRMS.

6.5. ANFIS modelling

The adaptive network-based fuzzy inference system (ANFIS) using gradient-descent (GD), and a least squares error (LSE) algorithms is used for the non-parametric modelling of the twin rotor system. The input data structure comprises the voltage of main rotor at previous time, $V_v(t-1)$, and the pitch angle of the beam at previous time, $\alpha_v(t-1)$. The output from the ANFIS non-parametric model is the predicted pitch angle of the beam $\hat{\alpha}_v(t)$ at the hovering motion. The number of the input membership function assigned to each input of the ANFIS was arbitrarily set to six. Therefore, the ANFIS structure with first-order Sugeno model containing 36 rules is shown in Fig. 10.

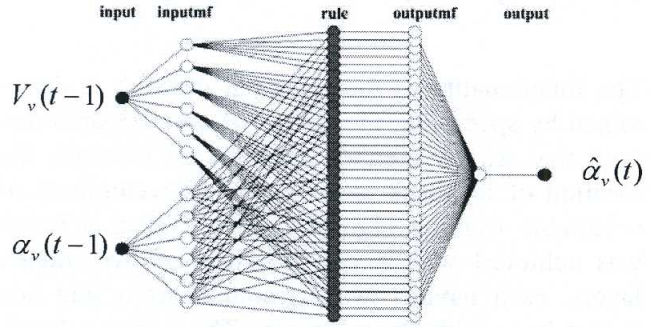


Fig. 10. ANFIS structure for modelling the TRMS.

Gaussian membership functions with product inference rule were used at the fuzzification level. The ANFIS Gaussian function used for both input as well as the 36 rules is shown in Fig. 11. The fuzzifier outputs the firing strengths for each rule. The vector of firing strengths is normalized. The resulting vector is defuzzified by utilizing the first-order Sugeno model.

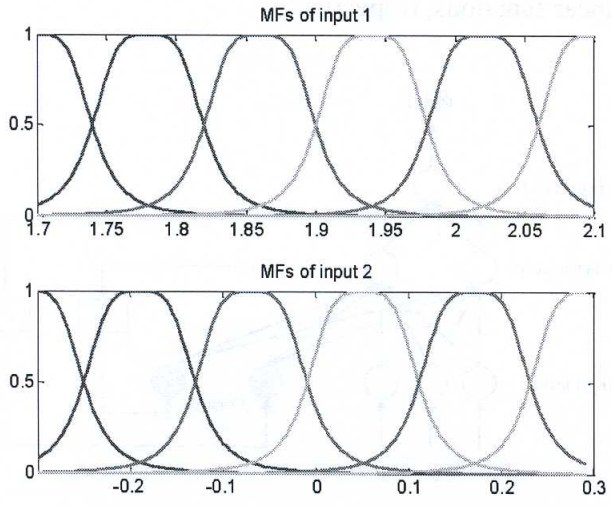


Fig. 11. ANFIS membership function for input1 and input 2.

7. Model validation

In system identification, it is crucial to validate whether the model obtained is good enough to replicate the real system. The use of plots and common sense, as well as statistical test are the two ways to investigate the validity of a model [49]. In this work, model validity tests are carried out as below:

(i) Use of plots and common sense
  a. Time domain representation
  b. Power spectral density representation
(ii) Use of statistical tests
  a. One-step ahead prediction
  b. Correlation tests
  c. Percentage of accuracy

For a good model, the predicted output should resemble the actual output. Such plots and the numerical fits associated with them are of course most useful and intuitively appealing for evaluating a given model [48]. A more convincing method of model validation is to use statistical methods. If a model of a system is adequate, then the residuals $e(t)$ should be unpredictable from past inputs and outputs. In other words, the residuals should not depend on something that is likely to change.

7.1. One-step-ahead prediction

The one-step-ahead (OSA) prediction of the system output is a common measure of predictive accuracy used in control and system identification [50]. This approach is adopted in this work and expressed as

$$\hat{y}(t) = f[u(t), u(t-1), u(t-2), ..., u(t-n_u), y(t-1), y(t-2), ..., y(t-n_y))] \quad (24)$$

where $f(\bullet)$ is a nonlinear function, u and y are the input and output, respectively. The residual or prediction is the difference between the measured output and the predicted output, given by

$$e(t) = y(t) - \hat{y}(t) . \quad (25)$$

Often $\hat{y}(t)$ will be a relatively good prediction of $y(t)$ over the estimation set even if the model is biased because the model was estimated by minimizing the prediction errors.

7.2. Correlation tests

The most useful statistical methods through which one can be roughly convinced about the validity of a model

are the auto-correlation and cross-correlation tests. The auto-correlation test examines the correlation among the residuals themselves. On the other hand, the cross-correlation tests investigate the correlation between the residuals and past inputs. In order to check these correlations, it is reasonable to study the covariance among the residuals and changeable parameters [49].

If a system is linear, then two simple covariance tests can be applied to detect unmodelled linear terms in the residuals [51]. If the process and noise model estimates are correct, it can be shown that

$$\phi_{ee}(\tau) = E[\varepsilon(t-\tau)\varepsilon(t)] = \delta(\tau), \quad (26)$$

$$\phi_{ue}(\tau) = E[u(t-\tau)\varepsilon(t)] = 0 \quad \forall\tau, \quad (27)$$

where $\varepsilon(t)$ represents the residual sequence. If the process model is correct but the noise model is incorrect, the residuals will be autocorrelated such that $\phi_{ee}(\tau) \neq \delta(\tau)$ but they will be uncorrelated with the input such that $\phi_{ue}(\tau) = 0 \ \forall\tau$. Alternatively, if the noise model is correct and the process model is biased, then the residuals are both autocorrelated such that $\phi_{ee}(\tau) \neq \delta(\tau)$ and correlated with the input such that $\phi_{ue}(\tau) \neq 0$. It is possible using the simple correlations, therefore, to distinguish between deficiencies in the process and noise models.

Billing and Voon [52] had shown that the traditional tests used for linear system in equations (26) and (27) are not sufficient. During the model validation stage, only a little control over the form of the input signals and the residuals are known. Therefore, few tests are derived that work under the worst possible combinations of signal properties. It is assumed that $u(t)$ and $\varepsilon(t)$ are independent zero-mean process, that $\varepsilon(\cdot)$ is white and that $u(\cdot)$ may be white. The non-linear model estimated will therefore only be unbiased providing

$$\phi_{u^2e}(\tau) = E[(u^2(t-\tau) - \overline{u}^2(t)\varepsilon(t)] = 0 \quad \forall\tau, \quad (28)$$

$$\phi_{u^2e^2}(\tau) = E[(u^2(t-\tau) - \overline{u}^2(t)\varepsilon^2(t)] = 0 \quad \forall\tau, \quad (29)$$

$$\phi_{e(eu)}(\tau) = E[\varepsilon(t)\varepsilon(t-1-\tau)u(t-1-\tau)] = 0 \quad \tau \geq 0. \quad (30)$$

7.3. Percentage of accuracy

Accuracy of a result or experimental procedure can refer to the percentage difference between the experimental result and the accepted value. In this work, the percentage of accuracy is calculated to measure the relationship between the actual output $y(t)$ and predicted output $\hat{y}(t)$ according to equation (25). The higher the value of percentage of accuracy to 100% replicates an accurate time domain output mapping of actual output and predicted model output of the system.

$$\text{Percentage of accuracy} = 100\% - \text{Percentage of error} = \left(1 - \left(\frac{1}{N}\sum_{t=1}^{N}\frac{y(t)-\hat{y}(t)}{y(t)}\right)\right) \times 100. \quad (31)$$

8. Results and discussion

This section presents the results of non-parametric modelling of a TRMS using three different intelligent techniques that are MLPNN using LM training algorithm, ERNN using LM training algorithm and hybrid ANFIS GD + LSE. In order to establish reputability, the hybrid modelling methods are run ten times independently. All simulations are carried out on a PC (Intel Pentium ®, 3.40 GHz, RAM 1 GB).

The correspondence time domain mappings between the actual and predicted output of the pitch angle (rad) are shown in Fig. 12, Fig. 13 and Fig. 14. For the 700 training data, it is shown that all the three different modelling methods can obtain a good output mapping. However, the zoom-up figure shows that MLPNN gives the worst result as compared to ERNN and ANFIS GD+LSE. It is further explained, where the percentage of accuracy (equation (31)) indicates some variations in non-parametric modelling with 96.68%, 97.51% and 98.06% using MLPNN, ERNN and ANFIS GD+LSE.

The power spectral densities of the actual and predicted non-parametric model of the system are shown in Fig. 15. Since the significant mode lies in the 0–1 Hz bandwidth, it is clearly evidence that the non-parametric modelling techniques used are able to capture the main resonance mode at 0.34 Hz. From the modelling work carried out, it is also observed that with correct network architecture for MLPNN and ERNN, as well as correct number of rules for ANFIS GD+LSE, they can estimate the system well and manage to locate resonance frequencies of the system.

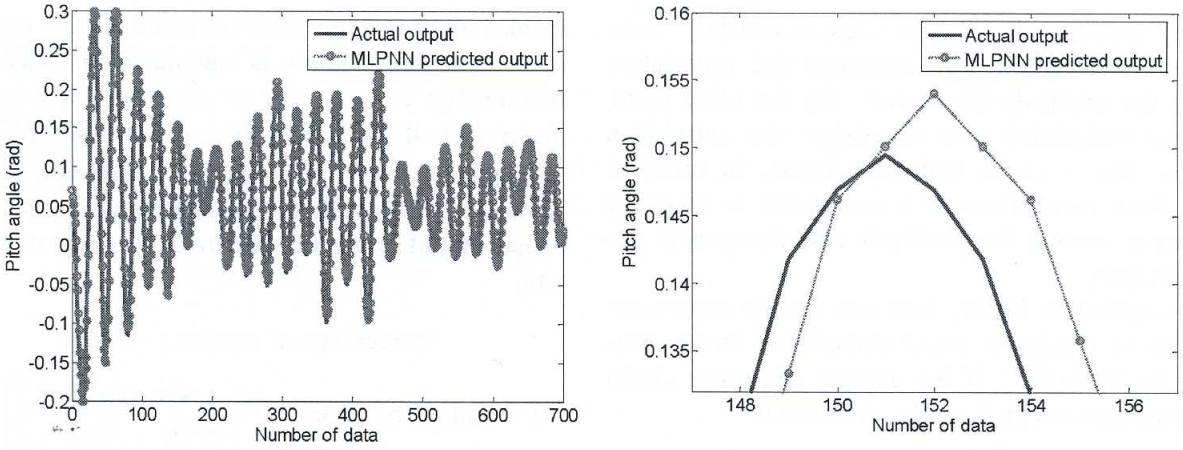


Fig. 12. Actual and MLPNN predicted output of the system.

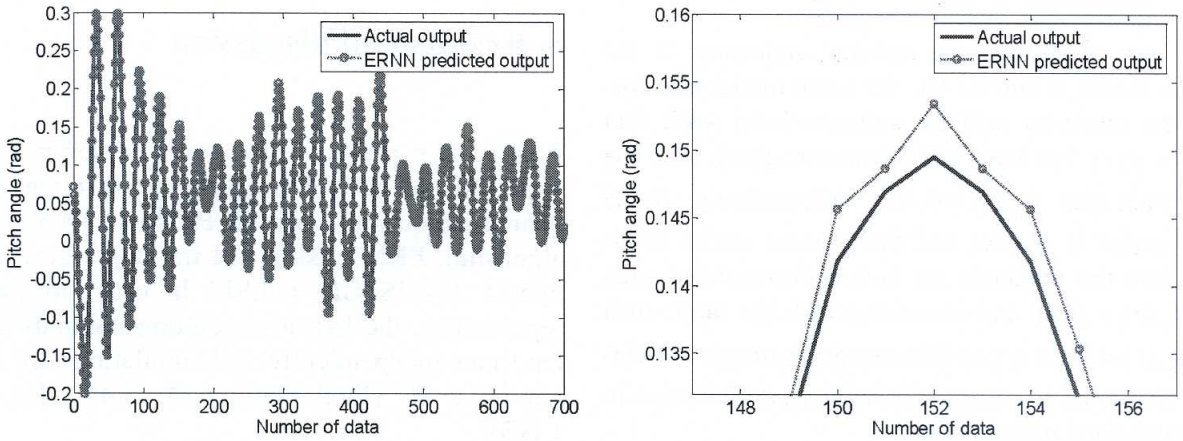


Fig. 13. Actual and ERNN predicted output of the system.

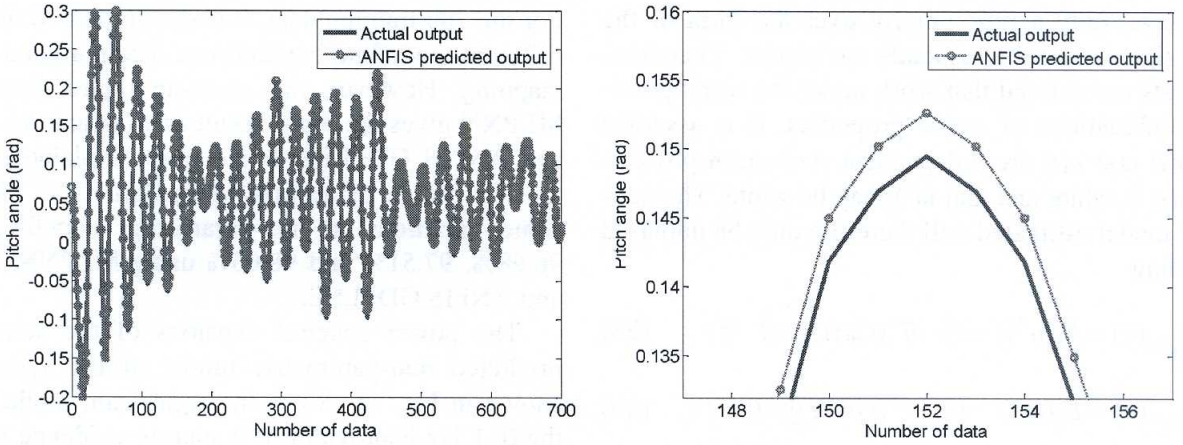
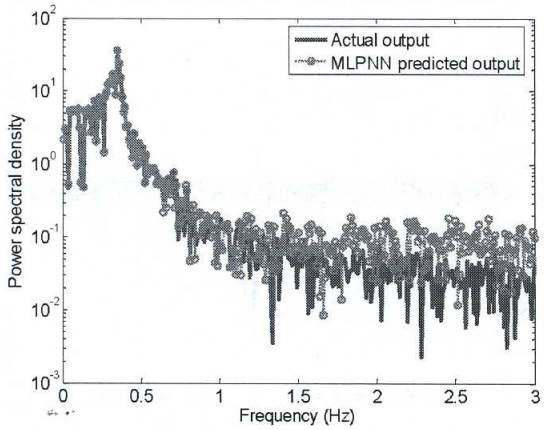
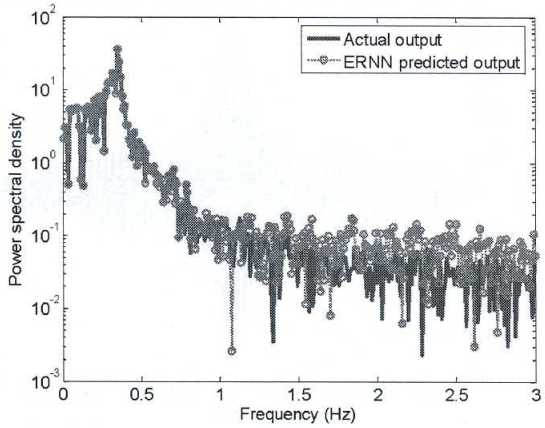


Fig. 14. Actual and ANFIS GD+LSE predicted output of the system.

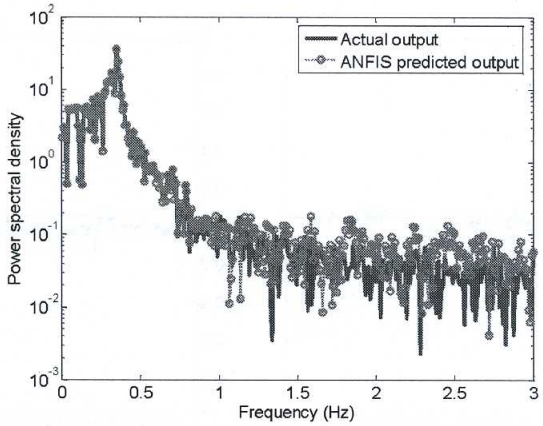
The convergence representation of mean-squared error for 100 epochs using non-parametric modeling techniques is shown in Fig. 16. It is clearly evidence that the convergence curve using MLPNN and ERNN techniques are still not converge until the predefined number of epoch. However, the dynamic ERNN wins in the convergence competition with smaller value of MSE as compared to static feedforward MLPNN. It is opposed to the ANFIS GD+LSE, where the synergies between neural



(a) MLPNN



(b) ERNN



(c) ANFIS GD+LSE

Fig. 15. Power spectral density of actual and non-parametric modelling predicted output.

network and fuzzy logic compensate the weaknesses between both techniques. It gives faster convergence in only 14th epochs with the best value of MSE, 3.304×10^{-4}.

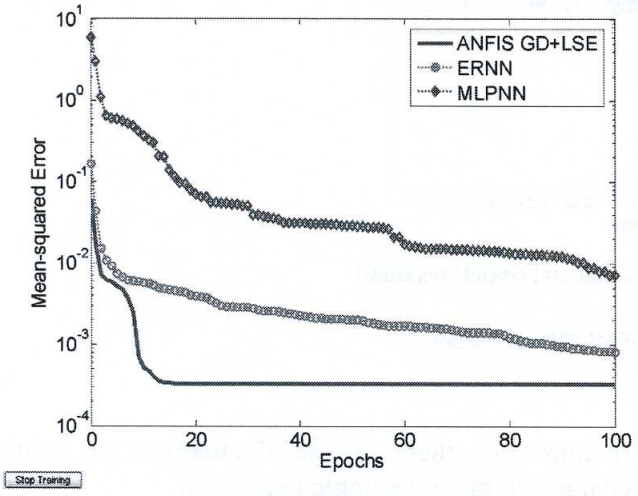


Fig. 16. Convergence of non-parametric modelling algorithms over 100 epochs.

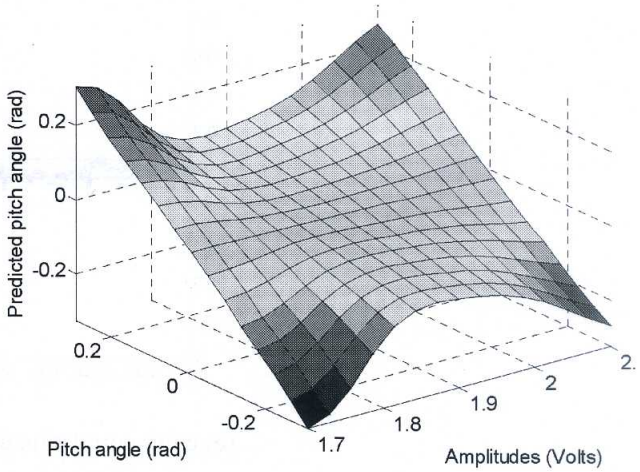
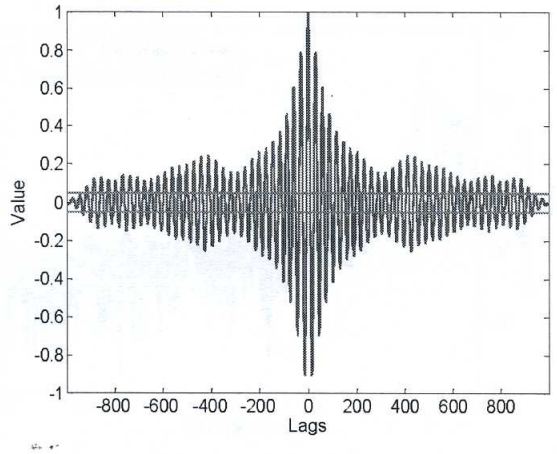
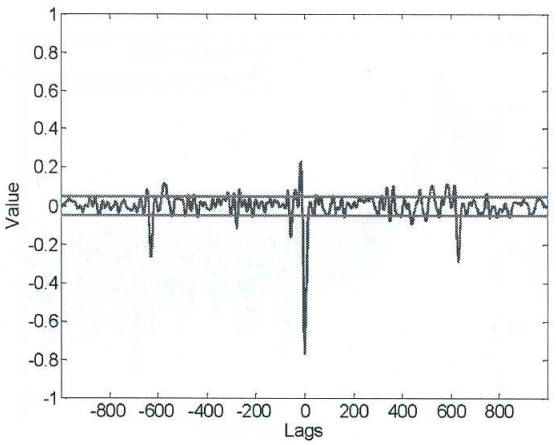


Fig. 17. The 3D-surface of the ANFIS GD+LSE modelling.

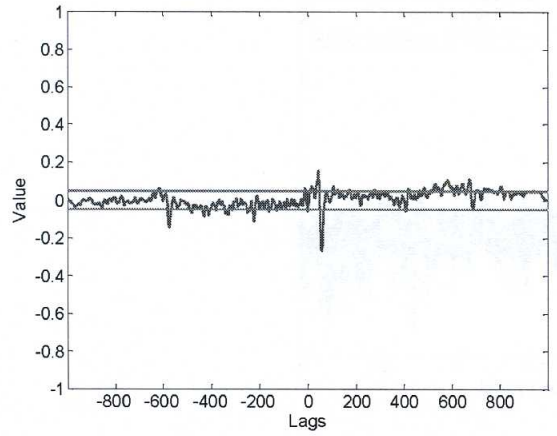
Figure 17 shows the 3D-surface of the hybrid ANFIS modelling (which is responsible for predicting the pitch angle output of the TRMS) using GD+LSE training algorithm. From the 3D-surface, it can be noticed that this ANFIS predictor is stable



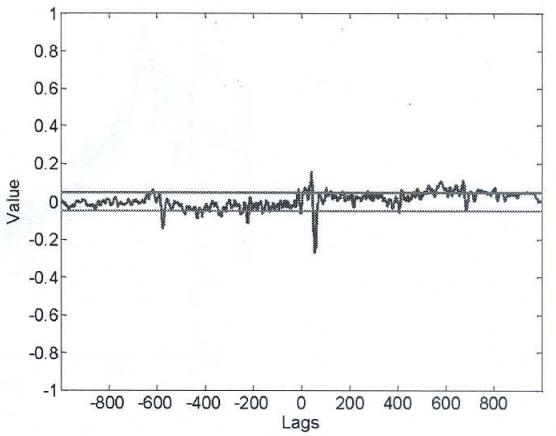
(a) auto-correlation of residuals



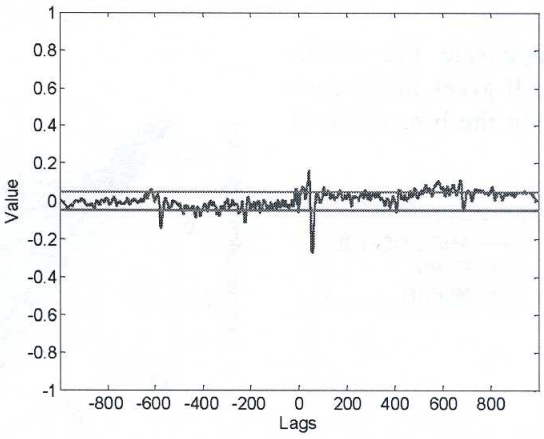
(b) cross-correlation of input and residuals



(c) cross-correlation of input square and residuals



(d) cross-correlation of input square and residuals square

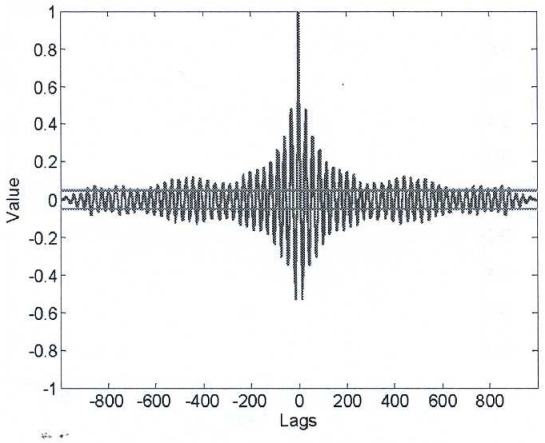


(e) cross-correlation of residual and (input*residual)

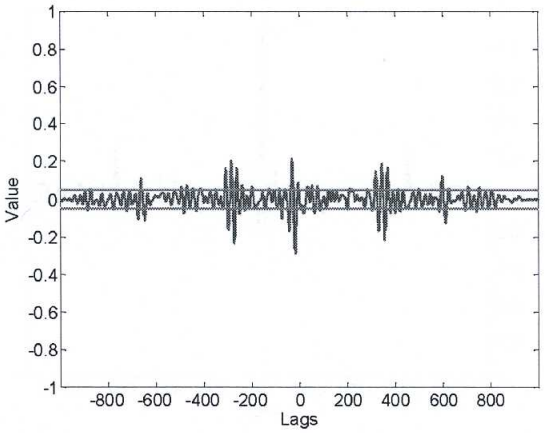
Fig. 18. Correlation test of MLPNN estimation.

because along the domain of discourse, there are no predicted pitch angle output values lying beyond the upper or lower limit. Furthermore, the surfaces are smooth which means that the rule-base is continuous, i.e., there are no fluctuations or sudden changes in the pitch angle prediction.

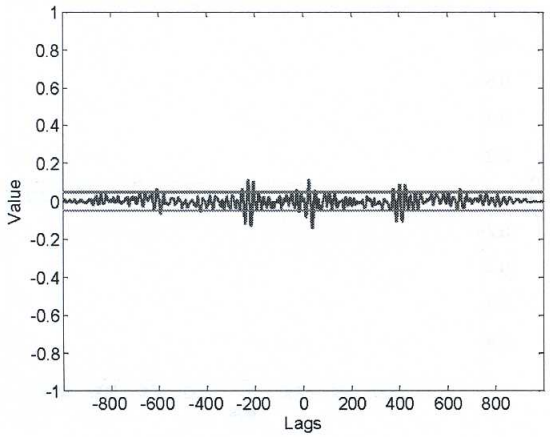
Normalization ensures that all the correlation functions lie in the range [1, –1] irrespective of the



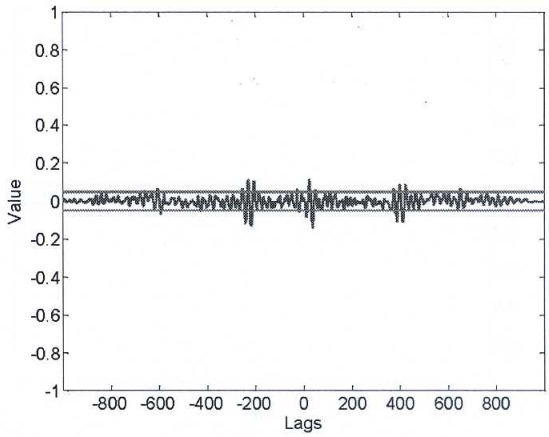
(a) auto-correlation of residuals



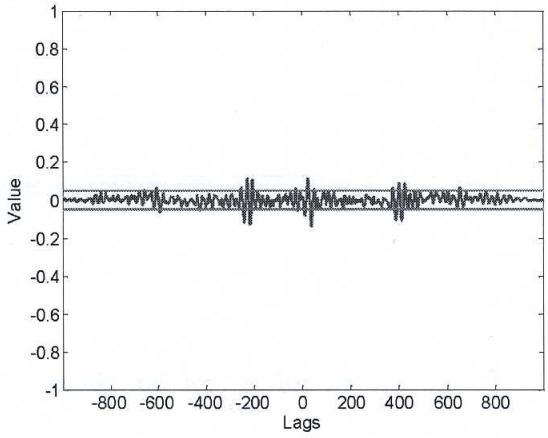
(b) cross-correlation of input and residuals



(c) cross-correlation of input square and residuals



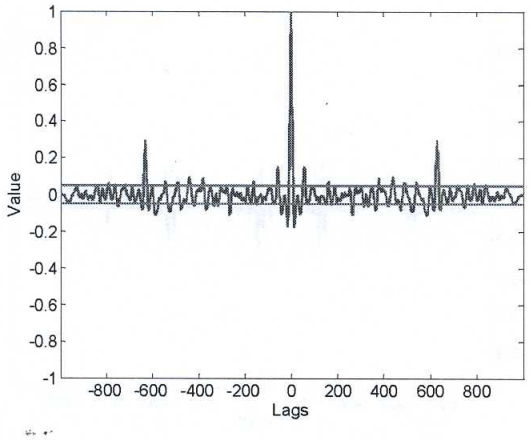
(d) cross-correlation of input square and residuals square



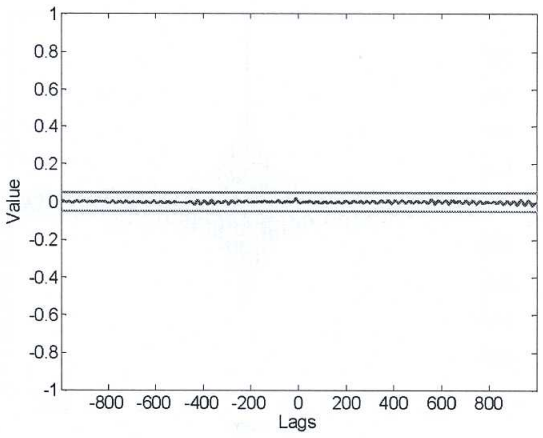
(e) cross-correlation of residual and (input*residual)

Fig. 19. Correlation test of ERNN estimation.

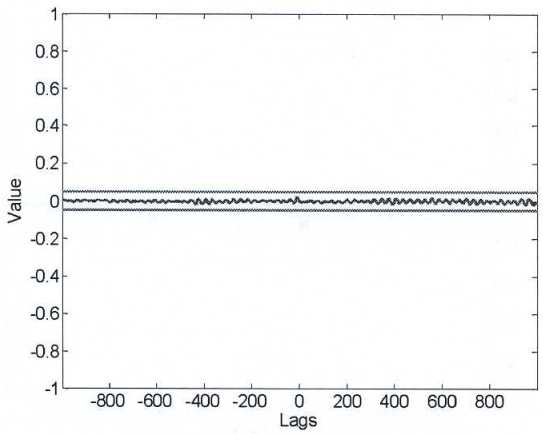
signal strengths. The correlations will never be exactly zero for all lags and the 95% confidence bands defined as $|r|<1.96\sqrt{N}$ are used to indicate whether the estimated correlations are significant or not, where N is the data length and r is the correlation function [52]. Using the five equations in Section 7, the correlation tests are carried out to determine the effectiveness of the non-parametric modelling techniques. The



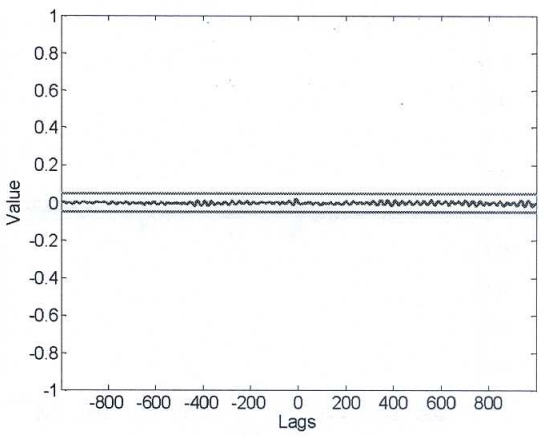
(a) auto-correlation of residuals



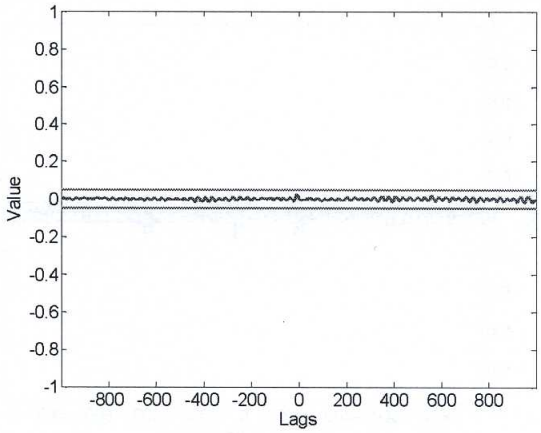
(b) cross-correlation of input and residuals



(c) cross-correlation of input square and residuals



(d) cross-correlation of input square and residuals square



(e) cross-correlation of residual and (input*residual)

Fig. 20. Correlation test of ANFIS GD+LSE estimation.

results of correlation tests are shown in Fig. 18, Fig. 19 and Fig. 20 using MLPNN, ERNN and ANFIS GD+LSE, respectively. It is evidenced that model ANFIS GD+LSE technique complies with all the five correlation tests indicating that the model behaviour is unbiased and close to that of the real system.

9. Comparative assessment

This section will discuss the comparative assessment of using MLPNN, ERNN and ANFIS GD+LSE to obtain the nonlinear model of the TRMS in Table 2 and Table 3. In this work, LM has been used to overcome the slow convergence rate problem in BP algorithm. Both the feedforward MLPNN and dynamic ERNN are trained using LM algorithm. The crucial drawbacks of this method, however, are that in many applications too much computation per pattern is required so that the storage and memory requirements go up as the square of the size of the network [53]. Thus, the number of epochs obtained in each run is very high which is approaching the predefined number of epochs.

Table 2. Performance of the employed non-parametric modelling techniques with 100 epochs (training data).

Non-parametric Modelling	Mean-squared Error (MSE)	No. of Epochs	CPU Execution Time (s)
MLPNN	7.269×10^{-3}	80	2.689
ERNN	8.448×10^{-4}	92	2.238
ANFIS GD+LSE	3.304×10^{-4}	29	2.473

Table 3. Performance of the employed non-parametric modelling techniques with 100 epochs (checking data).

Non-parametric Modelling	Mean-squared Error (MSE)	No. of Epochs	CPU Execution Time (s)
MLPNN	7.538×10^{-3}	88	2.125
ERNN	8.762×10^{-4}	85	1.952
ANFIS GD+LSE	3.483×10^{-4}	32	2.275

Table 4. Percentage of accuracy for parametric modelling of the system.

Non-parametric Modelling	Dataset	Main Resonance Mode (Hz)	% of accuracy
MLPNN	Training	0.3479	96.68
	Testing	0.3479	96.09
ERNN	Training	0.3479	97.51
	Testing	0.3479	96.92
ANFIS GD+LSE	Training	0.3479	98.06
	Testing	0.3479	97.63

ANFIS GD+LSE technique on the other hand, is able to produce the best MSE value, within less than 34 epochs (according to the average number of epochs) in a comparable CPU execution times as compared to the other methods. The number of membership functions for each input is set to be six which is equivalent to 36 rules to obtain the nonlinear model of the system. It is also shown in Table 2 and Table 3 that ANFIS GD+LSE technique gives the smallest value of standard deviation, indicating that the algorithm is consistent. According to Table 4, the highest percentage of accuracy also obtained using ANFIS GD+LSE technique which corroborates that the estimated model is adequate to replicate well the real system.

10. Conclusion

In this work, a non-parametric modelling approach for characterising the TRMS has been carried out based on MLPNN, ERNN and ANFIS GD+LSE. It is observed from the results and discussion (Section 8) that ANFIS GD+LSE gives the best result for modelling such systems. The underlying theme of this work has been to demonstrate the nonlinear modelling technique, which has various other applications apart from its utility as a "true" representation of the plant for controller evaluation. For instance, nonlinear models are essential for the design of nonlinear control laws. The ANFIS GD+LSE technique has performed well in approximating the system. In the time domain the developed model has predicted the system output closely and in the frequency domain the resonance frequencies of the model have matched those of the actual system very well.

References

[1] Åström K.J., Eykhoff P., *System identification – A survey*, Automatica, 1971, 7(2), 123–162.

[2] Elanayar V.T.S., Shin Y.C., *Radial basis function neural network for approximation and estimation of nonlinear stochastic dynamic systems*, IEEE Transactions on Neural Networks, 1994, 5(4), 594–603.

[3] Zadeh L.A., *Fuzzy logic, neural network and soft computing*, Communication of the ACM, 1994a, 37(3), 77–84.

[4] Zadeh L.A., *Soft computing and fuzzy logic*, IEEE Software, 1994b, 11(6), 48–56.

[5] Hornik K., Stinchcombe M., White H., *Multilayer feedforward networks are universal approximators*, Neural Networks, 1989, 2(5), 359–366.

[6] Rajpal P.S., Shishodia K.S., Sekhon G.S., *An artificial neural network for modeling reliability, availability and maintainability of a repairable system*, Reliability Engineering & System Safety, 2006, 91(7), 809–819.

[7] Miller N.A., Kunz D.L., *A comparison of main rotor smoothing adjustments using linear and neural network algorithms*, Journal of Sound and Vibration, 2008, 311(3–5), 991–1003.

[8] Putro I.E., Budiyono A., Yoon K.J., Kim D.H., *Modeling of unmanned small scale rotorcraft based on Neural Network identification*, IEEE International Conference on Robotics and Biomimetics (ROBIO 2008) Bangkok, Thailand, 22–25 February 2009, 1938–1943.

[9] Baldi P., Hornik K., *Neural networks and principal component analysis: Learning from examples without local minima*, Neural Networks, 1989, 2(1), 53–58.

[10] Long N., Zhang F., *Novel Newton's learning algorithm of neural networks*, Journal of Systems Engineering and Electronics, 2006, 17(2), 450–454.

[11] Rahideh A., Shaheed M.H., *Neural network-based modelling of a two-degrees-of-freedom twin rotor multiple input, multiple output system using conjugate gradient learning algorithms*, Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering, 2008, 222(6), 757–771.

[12] Toha S.F., Tokhi M.O., *MLP and Elman recurrent neural network modelling for the TRMS*, 7th IEEE International Conference on Cybernetic Intelligent Systems, (CIS 2008), London, U.K., 9–10 September 2008, 1–6.

[13] Wilamowski B.M., Iplikci S., Kaynak O., Efe M.O., *An algorithm for fast convergence in training neural networks*, International Joint Conference on Neural Networks (IJCNN01), Washington, DC, USA, 15–19 July 2001, 1778–1782.

[14] Cheol-Taek K., Ju-Jang L., *Training Two-Layered Feedforward Networks With Variable Projection Method*, IEEE Transactions on Neural Networks, 2008, 19(2), 371–375.

[15] Haykin S., *Neural networks and learning machines*, Third ed. (New Jersey, Pearson International Edition, 2009).

[16] Juang J.-G., Chiou H.-K., Chien L.-H., *Analysis and comparison of aircraft landing control using recurrent neural networks and genetic algorithms approaches*, Neurocomputing, 2008, 71(16–18), 3224–3238.

[17] Elman J., *Finding Structure in Time*, Cognitive Science, 1990, 14(2), 179–211.

[18] Gonzalez-Olvera M.A., Yu T., *Black-Box Identification of a Class of Nonlinear Systems by a Recurrent Neurofuzzy Network*, IEEE Transactions on Neural Networks, 2010, 21(4), 672–679.

[19] Al Seyab R.K., Cao Y., *Nonlinear system identification for predictive control using continuous time recurrent neural networks and automatic differentiation*, Journal of Process Control, 2008, 18(6), 568–581.

[20] Becerra V.M., Garces F.R., Nasuto S.J., Holderbaum W., *An efficient parameterization of dynamic neural networks for nonlinear system identification*, IEEE Transactions on Neural Networks, 2005, 16(4), 983–988.

[21] Hsu C.F., Lin C.M., Chen T.Y., *Neural-network-identification-based adaptive control of wing rock motions*, IEE Proceedings on Control Theory and Applications, 2005, 152(1), 65–71.

[22] Jang J.S.R., *ANFIS: adaptive-network-based fuzzy inference system*, Systems, Man and Cybernetics, IEEE Transactions on, 1993, 23(3), 665–685.

[23] Rajasekaran S., Vijayalakshmi Pai G.A., *Neural Networks, Fuzzy Logic, and Genetic Algorithms*, Synthesis and Applications Prentice Hall of India, Private Limited, 2005.

[24] Bawane N., Kothari A.G., Kothari D.P., *ANFIS based HVDC control and fault identification of HVDC converter*, HAIT Journal of Science and Engineering B, 2005, 2(5–6), 673–689.

[25] Negnevitsky M., *Artificial intelligence: a guide to intelligent systems*, Second ed, Addison Wesley, Pearson education, 2005.

[26] Zounemat-Kermani M., Teshnehlab M., *Using adaptive neuro-fuzzy inference system for hydrological time series prediction*, Applied Soft Computing, 2008, 8, 928–936.

[27] Alavandar S., Nigam M.J., *Adaptive neuro-fuzzy inference system based control of six DOF robot manipulator*, Journal of Engineering Science and Technology Review, 2008, 1, 106–111.

[28] Civicioglu P., *Using uncorrupted neighborhoods of the pixels for impulsive noise suppression with ANFIS*, IEEE Transactions On Image Processing, March 2007, 16(3), 759–773.

[29] Denai M.A., Palis F., Zeghbib A., *Modeling and control of non-linear systems using soft computing techniques*, Applied Soft Computing, 2007, 7, 728–738.

[30] Ahmad S., Shaheed M., Chipperfield A., Tokhi M., *Nonlinear modelling of a one-degree-of-freedom twin-rotor multi-input multi-output system using radial basis function networks*, Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering, 2002, 216(4), 197–208.

[31] Aldebrez F.M., Mat Darus I.Z., Tokhi M.O., *Dynamic modelling of a twin rotor system in hovering position*, First International Symposium on Control, Communications and Signal Processing, Hammamet, Tunisia, 2004, 823–826.

[32] Shaheed M.H., *Feedforward neural network based nonlinear dynamic modelling of a TRMS using RPROP algorithm*, International Journal on Aircraft Engineering and Aerospace Technology, 2005, 77(1), 13–22.

[33] Rahideh A., Shaheed M.H., Huijberts H.J.C., *Dynamic modelling of a TRMS using analytical and empirical approaches*, Control Engineering Practice, 2008, 16(3), 241–259.

[34] Feedback Instrument Ltd. Twin Rotor MIMO System manual 33-007-0. (Sussex, UK, 1996).

[35] Hinton G.E., *How neural networks learn from experience*, Scientific American, September 1992, 267(3), 145–151.

[36] Rumelhart D.E., Hinton G.E., Williams R.J., *Learning representation by back propagation errors*, Nature, 1986, 323, 533–536.

[37] Hamey L.G.C., *Comments on; Can backpropagation error surface not have local minima?*, IEEE Transactions on Neural Networks, 1994, 5(5), 844–845.

[38] Beale R., Jackson T., *Neural computing an introduction*, Bristol, U.K., Institute of Physics, 1990.

[39] Lean Y., Shouyang W., Lai K.K., *An integrated data preparation scheme for neural network data analysis*, IEEE Transactions on Knowledge and Data Engineering, 2006, 18(2), 217–230.

[40] Hagan M.T., Menhaj M.B., *Training feedforward networks with the Marquardt algorithm*, Neural Networks, IEEE Transactions on, 1994, 5(6), 989–993.

[41] Linkens D.A., Nyongesa H.O., *Learning systems in intelligent control: an appraisal of fuzzy, neural and genetic algorithm control applications*, Control Theory and Applications, IEE Proceedings, 1996, 143(4), 367–386.

[42] Moody J., Darken C.J., *Fast learning in networks of locally-tuned processing units*, Neural Computation, 1989, 1, 281–294.

[43] Demuth H., Beale M., Hagan M., *Neural network toolbox 6.0 user's guide*, The Mathworks, 2008.

[44] Sastry P.S., Santharam G., Unnikrishnan K.P., *Memory neuron networks for identification and control of dynamical systems*, Neural Networks, IEEE Transactions on, 1994, 5(2), 306–319.

[45] Yen J., Wang L., Gillespie C.W., *Improving the interpretability of TSK fuzzy models by combining global learning and local learning*, IEEE Transactions on Fuzzy Systems, November 1998, 6(4), 530–537.

[46] Kosko B., *Fuzzy systems as universal approximators*, IEEE Transactions on Computers, November 1994, 43(11).

[47] Hamidian D., Seyedpoor S.M., *Shape optimal design of arch dams using an adaptive neuro-fuzzy inference system and improved particle swarm optimization*, Applied Mathematical Modelling, 2010, 34(6), 1574–1585.

[48] Ljung L., *System identification: theory for the user*, Englewood Cliffs, New Jersey, Prentice-Hall, 1999.

[49] Söderström T., Stoica P., *System Identification*, New Jersey, Prentice-Hall, 1988.

[50] Shaheed M.H., *Neural and genetic modelling, control and real-time finite element simulation of flexible manipulator*, The University of Sheffield, U.K., 2000.

[51] Box G.E.P., Jenkins G.M., *Time series analysis: Forecasting and control*, San Francisco, Holden-Day, 1976.

[52] Billings S.A., Voon W.S.F., *Correlation based model validity tests for non-linear models*, International Journal of Control, 1986, 44(1), 235–244.

[53] Behera L., Kumar S., Patnaik A., *On adaptive learning rate that guarantees convergence in feedforward networks*, IEEE Transactions on Neural Networks, 2006, 17(5), 1116–1125.

Received April 14, 2011