Available online at www.sciencedirect.com

SCIENCE @ DIRECT®

Aerospace Science and Technology 7 (2003) 595–602

Aerospace Science and Technology

www.elsevier.com/locate/aescte

Lift coefficient prediction at high angle of attack using recurrent neural network

S. Suresh, S.N. Omkar *, V. Mani, T.N. Guru Prakash

Department of Aerospace Engineering, Indian Institute of Science, Bangalore-560012, India

Received 7 August 2002; received in revised form 6 March 2003; accepted 12 June 2003

Abstract

In this paper, identification of dynamic stall effect of rotor blade is considered. Recurrent Neural Networks have the ability to identify the nonlinear dynamical systems from training data. This paper describes the use of recurrent neural networks for predicting the coefficient of lift (C_Z) at high angle of attack. In our approach, the coefficient of lift (C_Z) obtained from the experimental results (wind tunnel data) at different mean angle of attack θ_{mean} is used to train the recurrent neural network. Then the recurrent neural network prediction is compared with experimental ONERA OA212 airfoil data. The time and space complexity required to predict C_Z in the proposed method is less and it is easy to incorporate in any commercially available rotor code.
© 2003 Elsevier SAS. All rights reserved.

Keywords: Unsteady rotor blade analysis; Dynamic stall; Memory neuron network; Recurrent multilayer perceptron network

1. Introduction

Research in the field of non-linear rotor dynamic analysis, continues to be strongly motivated by the performance of the rotorcraft. At the high flight speeds regime the linear analysis ceases to be valid. As a consequence of high speeds, the rotor retreating blade can penetrate locally in the region of dynamic stall, while the rotor advancing blade tip is subjected to unsteady transonic shock wave effects. Such aerodynamic non-linearties can only be treated, on theoretical grounds, by the general equations of fluid dynamics. Conventional rotor dynamic analysis does not include the effect of blade stall at high angle of attack. Conventional rotor dynamic analysis is based on a linearized aerodynamic model, which assumes a constant slope for lift-curve, throughout the operating range of angle of attack. Thus such analysis cannot produce an accurate representation of blade response, at high lift or high speed. There are however, more advanced analysis tools that attempt to include the non-linearity associated with stall. These analyses are generally computationally cumbersome, and rather difficult to use in research, or preliminary design applications.

Existing stall models usually involve the tabularization of large quantities of measured wind tunnel data. A look-up table and interpolation scheme is then used to obtain the data for each particular angle of attack that occur during the analysis. The major underlying drawbacks of these methods are the following: (1) Theoretical analysis is limited to only those test conditions, that were measured experimentally, thus making the interpolation for other conditions very difficult, (2) The tabular nature of model makes it difficult to obtain linearized equation for stability analysis and (3) The constant necessity of computer search methods on the available data set slows down the computation, thus making it more expensive. Hence, there exist a need for a simple analytical method, which can model the stall of oscillating airfoil, that can be easily incorporated into blade dynamic theory, and that will produce accurate results, over a wide range of aerodynamic parameters.

A semi-empirical analytical stall model is presented in [1]. This stall model is semi-empirically derived, using the measured wind tunnel data, for an oscillating aerofoil in conjunction with a parameter identification scheme. The result is a set of differential equation, which relates the normal lift coefficient of an airfoil to angle of attack and its time derivative. Mathematical models that attempt to predict the effects of dynamic stall currently range from relatively simple empirical or semi-empirical methods, to

* Corresponding author.
E-mail address: omkar@aero.iisc.ernet.in (S.N. Omkar).

1270-9638/$ – see front matter © 2003 Elsevier SAS. All rights reserved.
doi:10.1016/S1270-9638(03)00053-1

sophisticated computational fluid dynamics methods [2]. While semi-empirical methods are usually adequate for most rotor design purposes, they usually lack rigor and generality, when applied to different airfoils and at different Mach numbers, for which 2D experimental measurements may not be available [3,4].

Since, abundant experimental data are available, one of the best ways is that of intelligent interpolation, when measurements are not available. Neural Networks (NN) provides a way of obtaining intelligent interpolation [5]. Neural network gained prominence in 1980's due to its ability to learn from the past data, and able to generalize the knowledge obtained in the learning.

In the late 1980's, the research work on neural networks and its applications started attracting researchers from various fields of engineering. In fact IEEE started a separate transactions dealing with all the developments in this fields. Most of these work reported in the literature basically deals with obtaining optimal network architecture, new learning algorithm and the main application area targeted was pattern classification. There is a lot of literature available in neural network. It is difficult to present a complete literature survey on neural network here. So, we restrict our attention to only neural network application in problems associated with aerospace engineering. The applicability of neural network to a wide range of complex problems that arises in aerospace engineering is given in [6]. A survey of neural network application in diverse areas such as pattern classification, optimal control, and manufacturing are presented in [7–11]. These studies [7–11] provide a complete survey of the work done in the respective fields and will be useful.

In our study, because of the nature of the data, the problem reduces to nonlinear dynamical system identification using neural networks. The results shown in this paper clearly indicate that the Recurrent Neural Networks (RNN) models implemented are capable of predicting the dynamic stall with a good accuracy. Hence, modeling non-linear rotor dynamic analysis as high angle of attack, in a neural network framework, is a significant contribution in predicting the dynamic stall.

2. Stall model

The analytical model used in this paper given in [1]. The stall model is semi-empirically derived using measured wind tunnel data for an oscillating airfoil in conjunction, with a parameter identification scheme [1]. The model consist of three differential equations, that relate the normal lift coefficient of an airfoil to its angle of attack and its time derivatives as follows,

$$C_{Z_1}^* + \lambda C_{Z_1} = \lambda\theta + (\lambda s + \delta)\theta^* + s^*\theta^*, \tag{1}$$

$$C_{Z_2}^{**} + 2\alpha\bar{\gamma} C_{Z_2}^* + \vec{\gamma}^2(1+\alpha^2)C_{Z_2} = -\bar{\gamma}^2(1+\alpha^2)\left[\Delta C_Z + \bar{c}\frac{\partial \Delta C_Z}{\partial\theta}\theta^*\right], \tag{2}$$

Table 1
Parameters for semi-empirical model

Parameter	Value
λ	0.20
S	$\frac{5\pi}{180}$
δ	$\partial C_{Z_l}/\partial\theta - (4\pi/180)[1 + 1.43\Delta C_Z]$
$\bar{\gamma}$	$0.10 + 0.023(\theta - 13°)u(\theta - 13°)$
α	$0.105/\bar{\gamma}$
$\bar{c}$	$2 - 5.1\tan^{-1}\{1.21(\theta - 13°)\}u(\theta - 13°)$

$u(\theta - 13°)$ is unit step function.

$$C_Z = C_{Z_1} + C_{Z_2}, \tag{3}$$

where

$(*)$	represents a derivative with respect to reduced time.
C_{Z_1}, C_{Z_2}	are the normal lift coefficients in the linear and non-linear regions of angle of attack.
θ	is the total aerodynamic angle of attack of the airfoil in degrees.
ΔC_Z	is the difference between the extended linear lift curve and the actual static lift curve.
λ	is the time delay parameter.
s	is the apparent mass quantity.
δ	parameter relate the lift coefficient and pitch rate of the airfoil.
α	is the damping factor.
$\overline{C}, \bar{\gamma}$	phase shift and natural frequency.

The parameter λ, s, δ, α, $\overline{C}$, $\bar{\gamma}$ are functions of angle of attack alone (for a given airfoil), and must be determined from the wind tunnel tests by parameter identification. The typical values of these parameters for an ONERA, OA212 airfoil is given in Table 1. We use the numerical values obtained from the wind tunnel experimental data for neural network approach.

3. Identification of dynamic stall using recurrent neural network

The main techniques available for nonlinear modeling can be classified into three categories: functional series, block oriented and black box methods (non-parametric method). Neural network models are used as black boxes for the identification and control of nonlinear systems due to their ability to approximate any continuous or discontinuous analytical function [5]. Identification of a system requires picking one of a class of functions (or models) so as to approximate the input-output behavior of the system in the best possible manner. The well known, multilayer feed forward neural network found wide application in many fields. Most of these applications primarily pertain to static systems. In static system, the output of a feed-forward neural network is a function of its input only and hence it can only model memoryless transformation.

Predicting the dynamic stall is not a static system, and the dynamics are involved. In dynamical system, the output is a function of not only inputs but also some past outputs. The fact that dynamics constitute an essential part of all physical systems instigated the researchers to include dynamics to neural networks. Considerable research has been carried out on the applications of neural networks, for identification and adaptive control of dynamical systems [12–14]. The identification of dynamical systems through neural network poses difficulty, since the output of such a physical system is a function of not only past inputs, but also past outputs. This implies that neural network should have some form of built-in memory to include the dynamics inside the network. The necessity of memory has been bypassed by using tapped delay lines [12]. This approach can succeed, when the order of the system is known a priori, in which case, all the necessary past inputs and outputs of the system, can be fed as explicit inputs to the network. Then the network can learn memoryless transformation, that captures the dependence of the output of the system, on the specified past inputs and outputs. Both from aesthetics and practicality, learning memoryless transformation in this manner may not lead to versatile dynamical models [15].

A scheme for using NN, as components of dynamical systems is proposed in [12]. In this, it has been shown that using NN's and linear filters in cascade and/or feedback configurations, can construct a rich class of models. The NN corresponds to the nonlinear part, and the linear filter gives the dynamics to the model. A method, called dynamic back propagation, has also been proposed, to backpropagate the errors through a linear dynamical system in [12]. It has been shown through simulations, that these models can identify some complicated nonlinear dynamical systems.

Although the above scheme is one elegant way to introduce dynamics into the model, it has certain drawbacks. For example, one needs a fairly good knowledge of the structure of the system, to decide what combination of memoryless transformations, and linear filters, is a good class of models to choose. To overcome these drawbacks, one possible solution is, to include the dynamics, directly into the network structure, so that it is possible to learn nonlinear dynamical systems, without assuming much knowledge of the systems. This problem can be avoided by using recurrent neural network. The ability of recurrent networks, to perform highly non-linear dynamic mappings make them particularly interesting, and quite useful in tasks, that have a temporal component, not easily handled through use of simple tapped delay lines [16].

Recurrent Multilayer Perceptron (RMLP) networks are by virtue of their ability to perform temporally extended tasks [16], thus adding considerable power beyond the static mapping, performed by the multilayer feed forward networks. Dynamics can be directly included in to the system so that network can identify the dynamical system, directly without assuming the structure [16]. Real Time Recurrent Learning (RTRL) algorithm is used to train such networks [16]. Though RMLP can be used for identifying the non-linear dynamical systems, they require at least some knowledge of the systems, that is, one must be able to specify the required level and recurrence for a given problem.

One such kind of recurrent network with internal memory is called the Memory Neuron Network (MNN) [17]. Here each unit of neuron (network neuron) has, associated with a memory neuron, whose single scalar output summarizes the history of past activations of that unit. These memory neurons (or more precisely the weights of connection into them) represent trainable dynamical elements of the model. Since the connections between a unit and its memory neuron, involve feedback loops, the overall network is a recurrent one. Through extensive simulations [17], it has been shown that this level of internal memory and trainable temporal elements, are sufficient for identifying nonlinear dynamical systems. These networks are fairly small and the learning algorithm (which is an approximation to the gradient descent method) is robust. Because of this advantage for identifying the dynamical system problem, we employ Memory Neuron Network and Recurrent Multilayer Perceptron in this paper.

3.1. Memory Neuron Networks (MNN)

We will briefly explain the memory neuron network for our problem. The architecture of a memory neuron network is similar to the one given in [17], and is shown in Fig. 1. In Fig. 1, the memory neurons are represented as small shaded circles and the network neurons are represented as circles. At each level of the network, except at the output level, the network neurons have exactly one memory neuron, connected to them. The memory neuron, takes its input from the corresponding network neuron, and it also has a self-feedback, as shown in the inset of Fig. 1. This leads to storage of past values of the network neuron, in the memory neuron. All the network neurons, and the memory neurons, send their output, to the network neurons of the next level. In the output layer, each network neuron can have a cascade of memory neurons, and each of them send their outputs, to that network neuron, in the output layer. The memory neuron network model is often found to be more useful, for generating stable adaptive laws. To identify an m-input, p-output plant, we will use a network with $m + p$ inputs, and p-outputs. This will be the case irrespective of the order of the system. The actual outputs of the plant, at each instant, are used as for training the network.

In our study the mean angle of attack ($\theta_{\text{mean)}}$ and the cyclic angle of attack (θ) are used as input, and the coefficient of lift (C_z), is used as output of the network. The memory neuron network is trained using Back Propagation Through Time (BPTT) as described in [17]. In this training algorithm, the error is computed for each network neuron in output layer. The weight adjustments for each copy of net are determined individually, and all copies of each weight

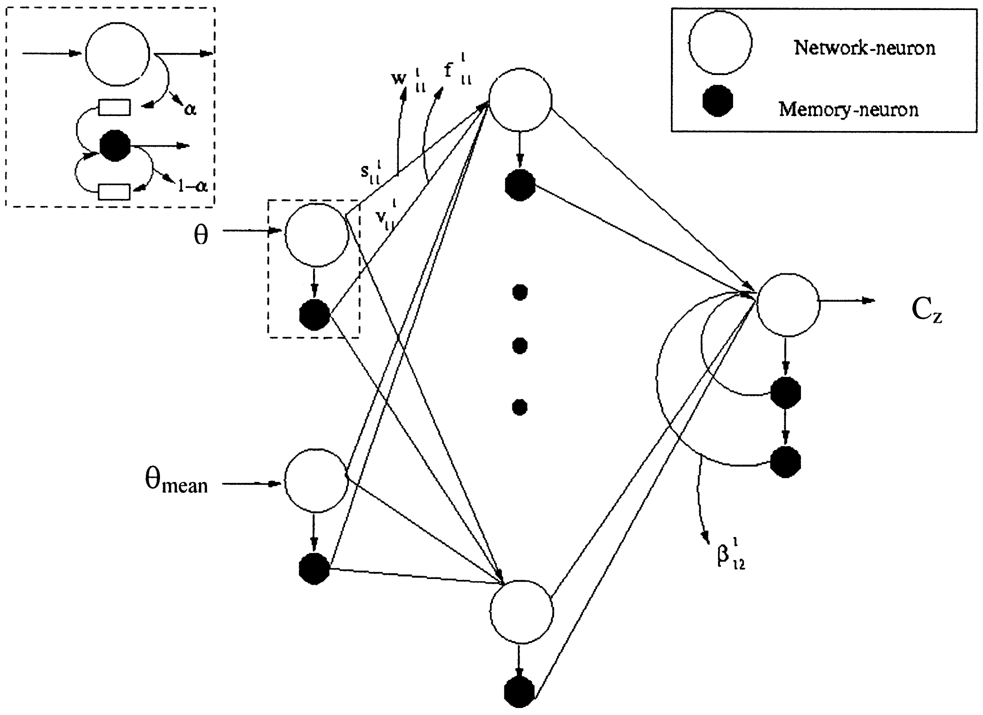

Fig. 1. Architecture of memory neuron network.

are updated. Thus, only the derivatives of error with respect to weights in the network are needed. Due to the presence of memory neurons, it is not easy to find out exact partial derivatives, through back propagation at single time step. So an approximation is used where by the network is unfolded, by exactly one time step, and the error is back propagated. This means that all the weights can be updated at time t, without needing any extra storage of the past activation of the nodes. A detailed explanation of obtaining the weights for this network is given in [17].

3.2. Recurrent multilayer perceptron (RMLP)

Another network used for this problem is recurrent multilayer perceptron neural network architecture as shown in the Fig. 2. In this, the training is done using the algorithm described in [16]. Let the network have n hidden units, with m external input lines. $y(t)$ denotes the n-tuple of output units in the network at time t, and $x(t)$ denotes m-tuple of external input signal to the network at time t. We concatenate $y(t)$ and $x(t)$ to form $m+n$ tuple $z(t)$. The $z(t)$ is given by,

$$\begin{aligned} z(t) &= x_k(t) \quad \text{if } k \in I, \\ & y_k(t) \quad \text{if } k \in U, \end{aligned} \tag{4}$$

where I and U are the number of inputs and hidden state, respectively. In our problem, we use the mean angle of attack (θ_{mean}) and cyclic angle of attack (θ) as input to the network, and the coefficient of lift (C_z) as output. The output of the network is delayed by one time step, and fed back to the input layer. First hidden unit is considered, as output neuron.

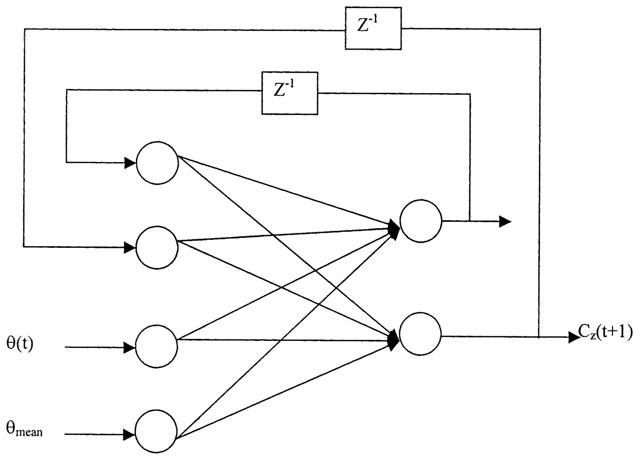

Fig. 2. Architecture of recurrent multilayer perceptron network.

A complete detail and training algorithm for recurrent neural network is given in [16].

4. Simulation using recurrent neural network

In this section, we present the simulation results in predicting the coefficient of lift (C_z) using recurrent neural networks. For training and validation of neural network approach, in predicting C_z, we use the experimental data given in [1]. This set of data containing the wind tunnel test for an OA012 airfoil has been used by various researchers [18–20]. Experimental results given in [1] are as follows: For a given mean angle of attack (θ_{mean}), and a cyclic angle

Table 2
Recurrent neural network parameters

Parameters	Memory neuron network	Recurrent multilayer perceptron network
No. of input nodes	3	6
No. of hidden layers	1	1
No of hidden nodes	25	3
Learning rate	0.03	0.01
Epoch	1000	500
Recurrence	–	0ne time step delay
No. of memory neuron in output node	1	–
Total training time	211 s	475 s
Time taken for forward pass	0.020 s	0.018 s

of attack (θ), the coefficient of lift (C_z) is obtained. The cyclic angle of attack is varied from $\theta_{\text{mean}} \pm 6°$, and the corresponding coefficient of lift (C_z) is also given. This forms a data set for one mean incidence of angle of attack, the data set contains 40 cyclic angle of attack trained (θ) in the range $\theta_{\text{mean}} \pm 6°$, and the corresponding coefficient of lift (C_z). For training the recurrent neural network four data sets (with θ_{mean} equal to 0°, 6°, 12°, and 16°) are used. After training is complete, the value of θ_{mean} at 9° and 14° are used for validation. The recurrent neural networks predicted value of C_z, for a θ_{mean} of 9° and 14° (for various cyclic angle of attack θ) are compared with the experimental results.

In this problem, the recurrent multilayer perceptron network is trained with real time recurrent algorithm, and the memory neuron network is trained by back propagation. Since this problem is dynamic, one past output of the system is also fed to the network as input. In our study, we have explicitly made the input and output of the network as zero, after every data set (data corresponding to mean angle of attack θ_{mean}). In case of recurrent multilayer perceptron network trained with real time recurrent learning algorithm, the output (C_z) is delayed by three-time step, so that it can identify the system properly. For training memory neuron network, we do not need any such delays. The network by itself will evolve to map the dynamic behavior of the system. The initial weights are set randomly (very close to zero) and training is performed until the mean square error is minimum. Thus post-training, the network will predict the coefficient of lift corresponds to cyclic angle of attack for a given mean angle of attack. Table 2 shows the network parameters for recurrent multilayer perceptron network trained with real time recurrent learning algorithm and memory neuron network.

4.1. Evaluation criterion

In our study, we use two qualitative performance measures describing the learning and generalization abilities of a given trained neural network as in [12]. Training error is the mean sum of squared residuals (error) in the training data and generalization error is the mean sum squared error in the (validating) testing data.

$$TE = \frac{1}{N_1} \sum_{k \in N_1} \bigl(t - y(x_k)\bigr)^2, \tag{5}$$

$$GE = \frac{1}{N_2} \sum_{k \in N_2} \bigl(t - y(x_k)\bigr)^2, \tag{6}$$

where t is the time value of the output, and $y(x_k)$ is the estimated output of the recurrent neural network for the kth input x_k. N_1 is the number of data points used in the training set. In our case, t is the value of C_z obtained form the experiments for the given value of θ_{mean} and θ. The value of $y(x_k)$ is the predicted value of C_z for a given x_k and x_k (input) is θ_{mean} and θ. When the training error approaches zero, then the network tends to interpolate the data.

Generalization error indicates how well a trained RNN behaves on new data, which are not used in training. This can be thought of as a curve fitting in which the training error indicates the average deviation from the fitted curve to the original curve. When the generalization error is small, we can say that a good fitting is achieved. N_2 is the number of points in the testing (validation) set. In our case, there are 160 data points are used (N_1) and 80 data (N_2) data points are used for testing. Thus after training, it is possible to accurately determine not only how well models predicted the training data but also how well the recurrent neural network models could generalize the coefficient of lift (C_z) to both mean angle of attack (θ_{mean}), and cyclic angle of attack (θ).

Neural network with too many hidden nodes may over fit the network and cause unrealistic oscillation between the training samples. On the other hand network with small number of hidden nodes may fail to approximate/generalize the complex underlying relationship in the data with fine fidelity [21]. If the training error approaches to zero than the network is able to interpolate the training data very well. When the number of training epochs is increased during the training process, the training error approaches to zero, but the problem of excessive training occurs and the network converges to 'memorization' of the training data. In such situation, the generalization error will be very high. The training error indicates the average deviation from the actual values used for training. Obviously, when noise is present in the training data set, an extremely small TE gives warning for possibilities of over training the neural network to the noise in the data. So in our simulations we have fixed the training error as 0.002. Once, the training error is less than 0.002 the training process is stopped. The training error (TE) and generalization error (GE) are calculated using Eqs. (5) and (6). The GE indicates the average deviation from the actual values not used in training. A small GE suggests that the network has captured the underlying relationship between the input-output data. Hence, both the training as well as generalization error should be less than the prescribed value to obtain the better result.

The following steps are used to obtain the optimal network configuration such that TE and GE are less than the prescribed value:

Step 1. Select network with minimal configuration.
Step 2. Train the network until $TE < 0.002$,
Step 3. Calculate the GE.
Step 4. If $GE > 0.002$ than increase the number of hidden node go to step 2 else stop.

This process is repeated until TE and GE is less than the certain prescribed value. The optimal configurations of neural networks are listed in Table 2.

As mentioned earlier, two different conventional methods exists for obtaining the coefficient of lift. One of the methods is based on look-up table, which has lot of disadvantages as mentioned in the Section 1. The other method is by solving numerically the governing differential equations [1]. In this, the nonlinear differential equations are linearized about θ_{mean}. These linear perturbed differential equations are used for calculating the C_z for various θ for a given θ_{mean}. Apart from the efforts involved in setting up the linear perturbed equations, the time taken to solve them is approximately 150 times that of RNN method, which is of the order of few milli-seconds (for single forward pass) as shown in Table 2. Both these conventional methods need more memory and computational time than the RNN approach. The training and testing times for MNN and RMLP are shown in Table 2.

4.2. Simulation results

The predicated value of the coefficient of lift (C_z), using the recurrent neural networks is shown in Figs. 3–8. We have trained two neural networks namely Memory Neuron Networks (MNN) and Recurrent Multilayer Perceptron Network (RMLP) for predicting the value of coefficient of lift (C_z). The results obtained for Memory Neuron Network (MNN) are shown in Figs. 3–5. In all these figures, the coefficient of lift (C_z) is plotted against cyclic angle of attack (θ) for a given mean angle of attack (θ_{mean}). For example, consider Fig. 3, in this figure, the mean angle of attack (θ_{mean})

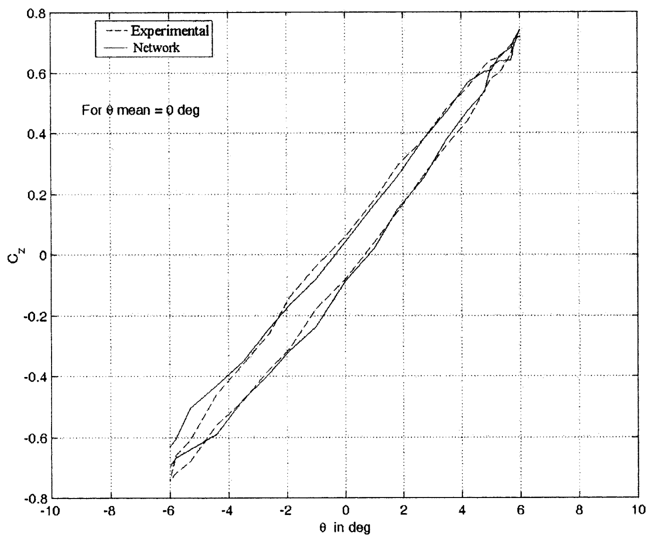

Fig. 3. Memory neuron network output for $\theta_{\text{mean}} = 0°$.

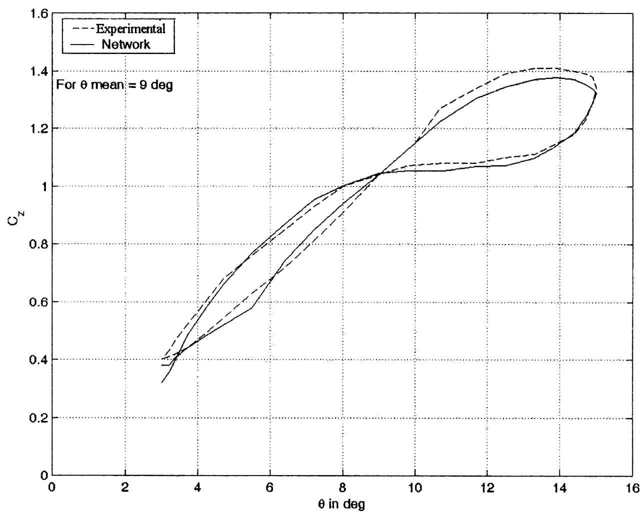

Fig. 4. Memory neuron network output for $\theta_{\text{mean}} = 9°$.

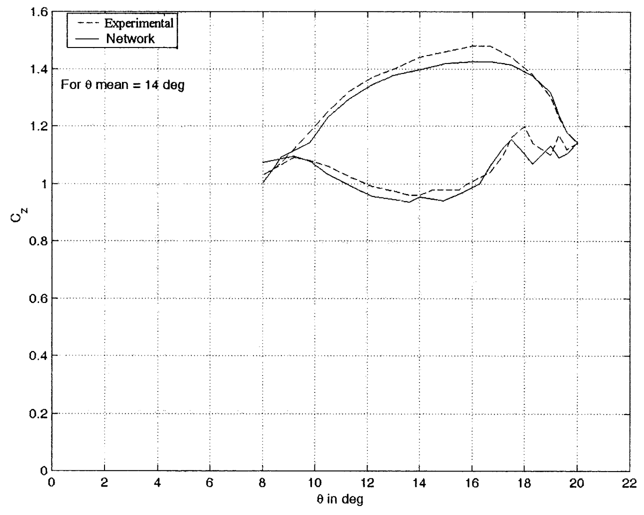

Fig. 5. Memory neuron network output for $\theta_{\text{mean}} = 14°$.

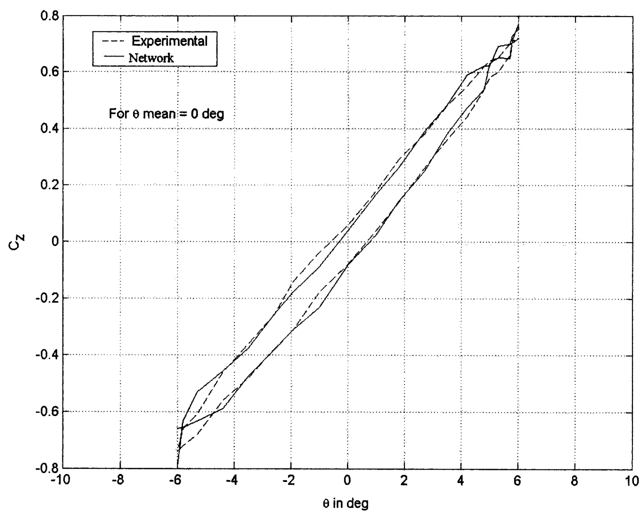

Fig. 6. Recurrent multilayer perceptron network output for $\theta_{\text{mean}} = 0°$.

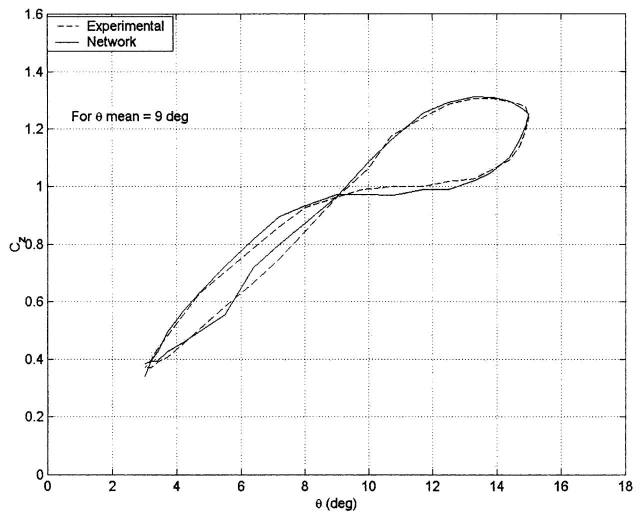

Fig. 7. Recurrent multilayer perceptron network output for $\theta_{\text{mean}} = 9°$.

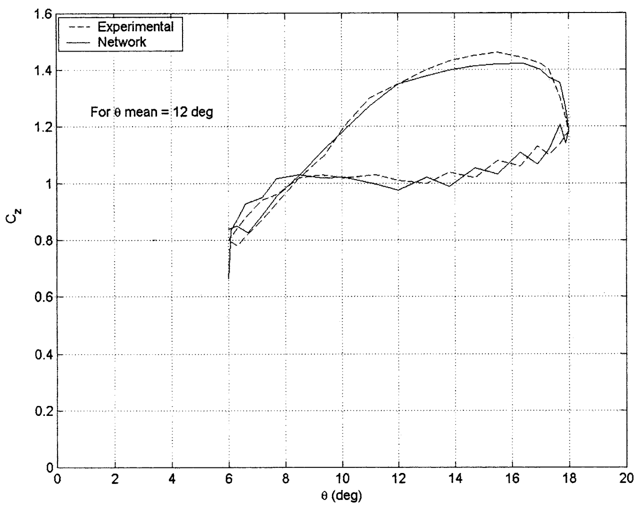

Fig. 8. Recurrent multilayer perceptron network output for $\theta_{\text{mean}} = 14°$.

is 0°, and cyclic angle of attack (θ) varies from $-6°$ to $+6°$. The value of C_z for both the increasing portion of the angle of attack and decreasing portion of the angle of attack are shown. In this figure, the value of C_z obtained from experiments is shown as a dotted line and network predicated as a solid line. From this figure we can say that the network is able to predict, the value of C_z to good degree of accuracy, except at the beginning of cyclic angle of attack. Fig. 3 shows the predicted value of C_z for mean angle of attack 0° which is used for training the network. The mean squared error for $\theta_{\text{mean}} = 0°$ is 0.001261 and total training error is equal to 0.00115. We have used the trained network, for predicting the value of coefficient of lift (C_z), for the mean angle of attack (θ_{mean}) at 9° and 14°, for all the cyclic angle of attack (θ) in the range of $\theta_{\text{mean}} \pm 6°$. Note that θ_{mean} at 9° and 14° are not used in training. The predicted value of C_z is shown along with the experimental value of C_z, for θ_{mean} equal to 9° and 14° are shown in Figs. 4 and 5, respectively. From these figures we can say that the recurrent neural network prediction is good. The mean squared error (generalization error) in this prediction using Memory Neuron Network is 0.00133. This shows that the network is able to generalize well. The same type of study is conducted using recurrent multilayer perceptron neural networks. The results obtained using recurrent multilayer perceptron neural networks are shown in Figs. 6–8. The total training error and the generalization error for recurrent multilayer perceptron neural network are 0.001615 and 0.000919, respectively. The R^2 estimate for training and testing data is calculated. The values of R^2 for different θ_{mean} are calculated for training set using the following equation,

$$R^2 = 1 - \frac{SS_T}{SS_R}, \tag{7}$$

where

$$SS_R = \sum_{t=1}^{N} \left(C_z(t) - \overline{C}_z\right)^2, \qquad SS_T = \sum_{t=1}^{N} \left(C_z(t) - \widehat{C}_z(t)\right)^2,$$

where $\overline{C}_z$ – mean value of the output for a given θ_{mean} and $\hat{C}_z(t)$ – network predicted value. The R^2 estimates for training data are given in Table 3 and for testing data are given in Table 4. The R^2 estimate for $\theta_{\text{mean}} = 0°$ is equal to 0.99125. Since the R^2 estimate is equal to 1, the network is able to predict the output very well. Similarly, we can observe from the R^2 estimates of C_z for various θ_{mean} that neural network predictions are very close to actual values. These results clearly indicate, that the recurrent neural networks, is able to capture the dynamics very well but the training time is higher than the memory neuron network. Also these results clearly indicate that real-time

Table 3
Quantitative analysis for training the network

Network type	Training			
	θ_{mean} (°)	R^2 estimate	MSE*	Total error
MNN	0	0.99125	0.001261	0.00115
	6	0.98647	0.001223	
	12	0.96703	0.001281	
	16	0.99898	0.000829	
RMLP	0	0.99552	0.001194	0.001615
	6	0.97647	0.002498	
	12	0.93703	0.001837	
	16	0.99761	0.000931	

* Mean square error.

Table 4
Quantitative analysis for testing the network

Network type	Generalization			
	θ_{mean} (°)	R^2 estimate	MSE*	Total error
MNN	9	0.99751	0.001041	0.00113
	14	0.99111	0.001221	
RMLP	9	0.99911	0.000409	0.000919
	14	0.97751	0.001429	

* Mean square error.

models, of dynamic response of rotor blade, can be done using recurrent neural networks.

5. Conclusion

Recurrent Neural network is used to identify the dynamic stall effect of rotor blade. Rotating angle of attack at different mean angle of attack and corresponding coefficient of lift are obtained from wind tunnel experiment for ONERA OA212 airfoil. Recurrent multilayer perceptron neural network and memory neuron network are used to identify the dynamic stall effect of rotating blades. Four sets of data correspond to mean angle of attack is used for training the network. Inputs to the networks are mean angle of attack and rotating angle of attack. The analysis for predicting coefficients of lift C_z for a given mean angle of attack ($\theta_{\text{mean}} = 0°$) is shown in Figs. 3 and 6. Fig. 3 shows coefficient of lift predicted by the memory neuron network and Fig. 6 shows the coefficient of lift predicted by recurrent multilayer perceptron network. Similarly for testing data, the networks responses with actual output are plotted in Figs. 4, 5 and Figs. 7, 8. In all these cases, the result clearly indicates that the recurrent multilayer perceptron neural network accurately predicts the dynamic stall response of the rotor blade (OA212 airfoil). This is verified quantitatively in Tables 3 and 4 and graphically by plotting the predicted and experimental values in Figs. 3–8. These results are clearly indicated those recurrent neural networks are able to predict the coefficient of lift with less than 1% deviation from experimental values. Since the recurrent neural network based method able to predict C_Z very well and time required to predict is very less, it can be easily incorporated in any commercially rotor codes. Since our idea is to show the applicability of this method, we have used OA212 airfoil experimental data because this data is well known and available in the literature. This approach can be easily applied to the experimental data obtained for any other airfoil.

Acknowledgement

The authors are very thankful to Aeronautical Research and Development Board (ARDB), India for the financial support to do the analytical study. The authors also like to thank the reviewers for their comments and suggestions, which has improved the quality of presentation.

References

[1] C.T. Tran, D. Petot, Semi-empirical model for the dynamic stall of airfoils in view of the application to the calculation of responses of a helicopter blade in forward flight, in: Sixth European Rotorcraft and Powered Lift Forum, Bristol, England, 1980.
[2] L.E. Ericsson, J.P. Reding, Fluid mechanics of dynamic stall. Part I. Unsteady flow concepts, J. Fluid Structures 2 (1988) 1–33.
[3] G.J. Leishman, Principles of Helicopter Dynamics, Cambridge University Press, Cambridge, UK, 2000.
[4] J.A. Ekaterinaris, G.R. Srinivasan, W.J. Croskey, Present capabilities of predicting two-dimensional dynamic stall, AGARD CP-552, 1994.
[5] S. Haykins, Neural Network: A Comprehensive Foundation, Macmillan, 1994.
[6] W.E. Faller, S.J. Schreck, Neural networks: Applications and opportunities in aeronautics, Progress in Aerospace Sciences 32 (1996) 433–456.
[7] B.K. Wrong, S.L. Vincent, Lam J., A bibliography of neural network business applications research: 1994–1998, Computers and Operation Research 27 (2002) 1045–1076.
[8] K.J. Hunt, D. Sbarbaro, R. Zbikowski, P.J. Gawthrop, Neural networks for control systems – a survey, Automatica 28 (1992) 1083–1112.
[9] H.J. Udo, Neural networks application in manufacturing process, Computers and Industrial Engineering 23 (1–4) (1992) 97–100.
[10] B.K. Wong, T.A. Bodnovich, Y. Selvi, Neural network application in business: a review and analysis of the literature (1988–1995), Decision Support Systems 19 (1997) 301–320.
[11] D. Chen, P. Burrel, On the optimal structure design of multilayer feedforward neural networks for pattern recognition, Int. J. Pattern Recognition and AI 16 (4) (2002) 375–398.
[12] K.S. Narendra, P. Parthasarathy, Identification and control of dynamical system using neural networks, IEEE Trans. on Neural Networks (1990) 4–27.
[13] K.S. Narendra, P. Parthasarathy, Neural network and dynamical systems, Part III: control, Report No. 8909, May 1989.
[14] M.T. Miller, R.S. Sutton, P.J. Werbose, Neural Networks for Controls, MIT Press, Cambridge, MA, 1990.
[15] R.J. Williams, D. Zipser, Gradient based learning algorithms for recurrent connectionist networks, Technical Report NU-CCS-90, Boston, North Eastern University, College of Computer Science, 1990.
[16] W.R. Jand, D. Zipser, A learning algorithm for continually running fully recurrent networks, Neural Computation 1 (1989) 270–280.
[17] P.S. Sastry, G. Santharam, K.P. Unnikrishnan, Memory neuron networks for identification and control of dynamical systems, IEEE Trans. on Neural Networks 5 (2) (1994) 306–319.
[18] D.J. Rudy, Comparison of rotor flapping response with three different dynamic stall models. St. Louis section of the AHS society (presented in competition for Robert L. Lichten Award), 1983.
[19] J.P. Rogers, Application of an analytical stall model to dynamic analysis of rotor baldes, Master of Science thesis, Washington University, 1982.
[20] D.J. Rudy, Three interpretations of the ONERA dynamic stall model with application to rotor blade flapping response, Master of Science thesis, Washington University, 1983.
[21] D.S. Chan, R.C. Jain, A robust back propagation learning algorithm for function approximation, IEEE Trans. on Neural Networks 5 (3) (1994) 467–479.