MODELLING AND IDENTIFICATION OF NON-LINEAR UNSTEADY AERODYNAMIC LOADS BY NEURAL NETWORKS AND GENETIC ALGORITHMS

Flavio Marques and John Anderson

Department of Aerospace Engineering,
University of Glasgow, Glasgow G12 8QQ, UK

Abstract

Recent developments in neural networks and genetic algorithms have provided a unified framework for the identification of models of complex non-linear processes and for the subsequent design of appropriate controllers. The ability of neural networks to approximate arbitrary non-linear mappings and dynamical processes is particularly attractive in the context of unsteady aerodynamic problems where significant non-linearities and delays are present. The present work describes a procedure for the systematic identification of parameterized neural network models of unsteady aerodynamic loads exhibiting weak non-linearities. It is shown that a temporal finite impulse response (FIR) neural network model provides a reasonable approximation to the aerodynamic behaviour observed in mildly separated flow. The identification procedure is facilitated by an adaptive optimisation process based on a genetic algorithm in which both the network topology and network time-delays are optimised for multiple training data sets of input motion history and output aerodynamic response. The approach is shown to furnish a satisfactory generalisation property to different motion history input patterns.

Introduction

Prediction of motion induced unsteady aerodynamic loads in flow regimes exhibiting dynamic shock excursion or flow separation presents a significant challenge in aeroelastic design and analysis. The inherently non-linear relationship between the motion history and the aerodynamic response demands sophisticated modelling techniques. Recent research effort has focused on computational procedures based on numerical solution of the governing fluid dynamic equations[1] and on a range of semi-empirical methods[2,3]. While such methods are entirely appropriate as a basis for numerically computed response studies, their suitability for aeroservoelastic control system design is questionable. Here, the non-linear relationship between the motion history and the aerodynamic response is represented implicitly and is not easily accommodated by standard non-linear control synthesis procedures.

Empirical evidence suggests that, for weakly non-linear flow behaviour, a functional description of the aerodynamic force response is justified. The existence of a unique non-linear aerodynamic force response functional appropriate to a particular flow regime is, necessarily, inferential. Non-uniqueness of the aerodynamic force response is generally associated with certain types of degenerate flowfield behaviour. In particular, aerodynamic hysteresis, flow instability and bifurcation have been identified as key elements in the breakdown of unique, single-valued behaviour of the aerodynamic force response. Flows admitting shocks and gross separation offer a potentially rich source of mechanisms for the realisation of such phenomena. However, computational and experimental evidence suggests that, for certain classes of flows, unique single-valued behaviour of the aerodynamic force response is observed over a range of flow parameters and motion histories. The basic properties of the aerodynamic functional depend on the nature of the flow regime with which it is associated and the class of admissible motion histories for which it is defined.

In a series of papers[4,5], Tobak and co-workers have developed a hierarchical class of functional aerodynamic force response models sufficiently general to encompass a broad range of flow regimes and motion histories. Although explicit representation of the aerodynamic force response functional is generally unavailable, its notional existence permits a succinct representation of the aerodynamic force response. In addition, several methods exist to identify approximate aerodynamic force response functionals from known characteristics of the motion history and aerodynamic response. A convenient approach is based on the Volterra-Wiener theory of non-linear systems[6,7]. Here, the functional is approximated by an infinite series of multi-dimensional convolution integrals of increasing order (the Volterra series). For weakly non-linear systems, only the first few kernels of the Volterra series are required to accurately model the input-output

characteristics of the system. An important feature of the Volterra-Wiener theory of non-linear systems is that a bilinear state-space system can be realized once the kernels up to second-order have been identified. This bilinear state-space system can be used as a non-linear aerodynamic model for aeroservoelastic analysis and design.

Recently, an alternative approach to the approximation of non-linear functionals has been proposed in which the functional is represented as a non-linear combination of *linear* functionals[8]. For a large class of non-linear systems, these so-called multi-layer functionals can be shown to be universal approximators. Moreover, the approximate functional form is conveniently represented by a temporal neural network. The utility of neural networks in non-linear system modelling is well-documented[9,10]. Identification of an appropriate neural network model is achieved via a supervised learning process in which a limited sample of system input-output training sets is presented to the network. In general, the network architecture and parameters are adjusted to minimise a measure of the error between the network output and the sample outputs. For a sufficiently broad sample of training data, the inherent generalisation properties of the network enable extrapolation/interpolation to arbitrary inputs. The principal advantage of the neural network representation is that it readily accommodates multiple input/multiple output system descriptions. In addition, control system design is facilitated by a number of standard procedures. In the context of aeroservoelastic design, a neural network model of the unsteady aerodynamic response characteristics can be combined with a standard structural dynamic model for the purposes of control system design.

The aim of this paper is to describe an approach to the modelling and identification of unsteady non-linear aerodynamic loads via neural networks. A brief account of the approximation of non-linear functionals by multi-layer temporal neural networks is presented. This is followed by a description of a network adaptation procedure based on a genetic algorithm and a variation of the simulated annealing algorithm in which both the network architecture and network parameters are optimised for multiple training sets. Application of the scheme to representative motion history ~ aerodynamic response data for a 2-D aerofoil exhibiting mild separation in low subsonic flow is used to demonstrate the feasibility of the approach.

Multi-Layer Functional Approximation

The basic premise of multi-layer functional approximation is that all time-invariant systems characterised by continuous functionals can be approximated by multi-layer operators[8]. Multi-layer operators are input-output models that can be thought of as non-linear generalisations of convolution, or more generally as being composed of a number of linear systems.

Network Model

A practical realisation of the multi-layer functional approximation is the temporal neural network model[9,11]. The basic processing unit of a neural network is the *neuron* shown in Figure 1. The relationship between inputs and outputs is established via connections or *synapses*. The connections present weights which are used to modify the information between neurons. Each neuron modifies its inputs through an *activation function* (*e.g.*. a non-linear sigmoid function).

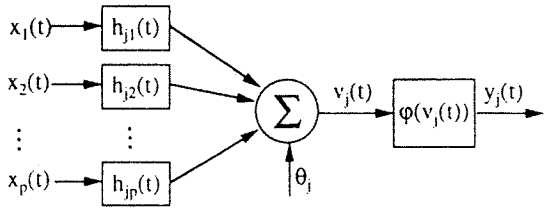

Figure 1. Neuron Model

To represent the temporal behaviour (or time-history effects) of the input data, each input connection or synapse is modelled by a linear, time-invariant filter. The characteristics of synapse i belonging to neuron j are described by an impulse response $h_{ji}(t)$. The response of the synapse i at time t to the input $x_i(t)$ is equal to the convolution of the impulse response $h_{ji}(t)$ with $x_i(t)$. Given a neuron j with a total of p synapses, the net activation potential $v_j(t)$ of the neuron due to the combined effect of all the inputs and the externally applied bias θ_j is given by

$$v_j(t) = \left[\sum_{i=1}^{p} \int_{-\infty}^{t} h_{ji}(\lambda)\, x_i(t-\lambda)\, d\lambda \right] - \theta_j \qquad (1)$$

The neuron output is the value of the activation function for $v_j(t)$,

$$y(t) = \varphi(v_j(t)) \qquad (2)$$

A multi-layer feedforward neural network or *multi-layer perceptron* is formed by interconnecting layers of neurons (Figure 2).

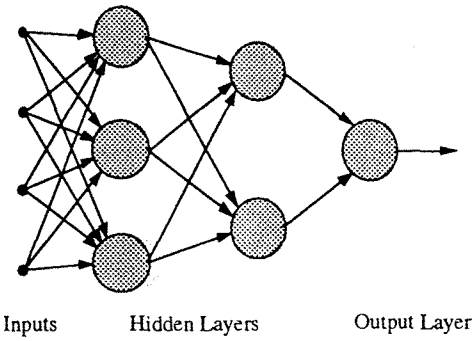

Figure 2. Example of Network Architecture

FIR Neural Network

From a computational viewpoint, it is convenient to assign a *finite* memory T to the synaptic filter and to approximate the convolution integral in equation (1) by a convolution sum. Consequently, the continuous-time variable t is replaced by a discrete-time variable n defined by $t = n\Delta t$ where n is an integer and Δt is the sample interval and equation (1) is approximated as

$$v_j(n) = \left[\sum_{i=1}^{p} \sum_{\ell=0}^{M} w_{ji}(\ell)\, x_i(n-\ell) \right] - \theta_j \qquad (3)$$

where $M = T/\Delta t$ is the number of delay units of the filter and $w_{ji}(\ell) = h_{ji}(\ell)\Delta t$ is the synaptic weight at time-delay ℓ.

Alternatively,

$$v_j(n) = \sum_{i=1}^{p} \mathbf{w}_{ji}^T \mathbf{x}_i(n) - \theta_j \qquad (4)$$

where

$$\mathbf{w}_{ji} = \left[w_{ji}(0)\ w_{ji}(1) \ldots w_{ji}(M) \right]^T$$

and

$$\mathbf{x}_i = \left[x_i(n)\ x_i(n-1) \ldots x_i(n-M) \right]^T$$

This model form is referred to as the *finite-duration impulse response (FIR) model* (see Figure 3). The neural network structure defined by a multi-layer perceptron whose hidden neurons and output neurons are all based on the FIR model is referred to as a *FIR multi-layer perceptron*.

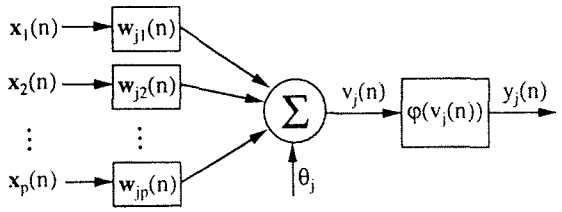

Figure 3. FIR Neuron Model

Network Adaptation and Genetic Algorithm

Identification of an appropriate neural network model is achieved via a supervised learning process in which a limited sample of system input-output training sets is presented to the network. Where the network architecture and network delays are prescribed, the synaptic weights can be trained using a temporal version of the standard back-propagation algorithm[9]. However, prescribed networks of this kind may exhibit poor convergence and generalisation properties and, in general, the network architecture and network parameters are adjusted to minimise some measure of the error between the network output and the sample outputs. An efficient network optimisation scheme can be formulated via a genetic algorithm[12,13].

Genetic algorithms are a type of evolution based search algorithm which manipulate sets of possible decoded solutions for a problem. As an evolution based technique or evolution program, genetic algorithms operate on the set of decoded solutions according to the principles of natural selection and the *survival of the fittest* premise. The elements of a conventional genetic algorithm comprise: *Individuals* - representing possible solutions to a problem, each of whose features are encoded in a chromosome; *Chromosomes* - the basic units of a genetic algorithm which encode how each individual is to be constructed (a chromosome is normally represented by a string of binary numbers, but other representations may also be used); *Genes* - subsets of a chromosome which maintain a particular feature of an individual; *Population* - a complete set of individuals for the search process; *Fitness Function* - a value assigned to each individual which represents how good an individual is as a solution to the given problem.

A conventional genetic algorithm generally starts with a randomly initialised population of individuals. Each individual is evaluated by decoding its chromosome and applying the fitness function. The new individuals

are the result of combining individuals (ranked by fitness) from the original population, in a process called *reproduction*. Reproduction in a genetic algorithm is facilitated by the operations of: *Selection* - to choose the individuals for combination; *Crossover* - to create new individuals by swapping genes from the selected individuals; *Mutation* - to guarantee that occasionally (with low probability) a few genes are modified and therefore a new search space is explored, thereby increasing the chance of achieving the global minimum. The reproduction process is repeated until a new complete population is established. The process is further iterated only if the fitness of the best individual has not achieved a goal value, or if other termination criteria have not been satisfied. For each new population, the process steps a generation.

The genetic algorithm is used, as part of a supervised training process, to obtain an optimal architecture and time-delay distribution for the FIR neural network while, simultaneously, training the network (that is, identifying the synaptic weights). To achieve this, the genetic algorithm interprets each FIR neural network as an individual belonging to a population. The associated chromosome is a sequence comprising the time-delays and weights per connection. The measure of the network fitness, f, is defined by the inverse of the sum of squared errors between the desired outputs and the actual neural network outputs; that is,

$$f = \frac{1}{\sum_{k=1}^{N}\sum_{n=1}^{L}\left(d_k(n) - y_k(n)\right)^2} \qquad (5)$$

where N is the number of training sets, L is the total number of time steps, $d_k(n)$ is the desired output at time n of training set k and $y_k(n)$ is the corresponding neural network output.

The networks in the population are constrained to maintain certain basic features; for example, the network is composed of a multi-layered architecture of biased neurons without missing connections between hidden layers, all hidden neurons are non-linear (sigmoid activation function), and all output neurons are linear.

The chromosome is represented by a string of constant length irrespective of the network architecture encoded within it. This is achieved by assuming an FIR neural network architecture with bounded structure and parameters. The chromosome size depends on the limiting FIR neural network considered. It is a string which records the information necessary to decode any feasible network within the pre-defined bounded architecture. For each neuron of the limiting architecture, the string is the sequence of all time-delay values of the previous hidden layer to the neuron itself. The complete chromosome is, therefore, the sequence described above for all neurons of the limiting architecture. A schematic representation of the chromosome is illustrated in Figure 4.

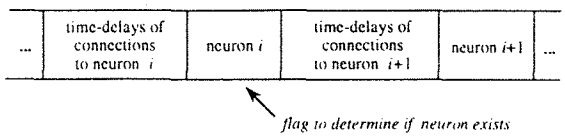

Figure 4. Generic Representation of the Chromosome

Flags are used in the chromosome to signify which neurons and connections do not exist. For each connection which exists a weight vector (whose size depends on the time-delay value) and a bias value are assigned to each neuron. This information is recorded separately in each case to avoid working with a very lengthy chromosome. However, the weight and bias values must always be related to their respective connection and neuron, whatever the genetic operation.

The training process commences with an initial population of individuals. Each individual is created with a randomly generated architecture and receives random weight values from a uniformly distributed source in the range -1.0 to 1.0. The entire population is evaluated through a feedforward pass of each individual to produce a fitness distribution. The values of the minimum (f_{min}), average (f_{ave}), and maximum (f_{max}) fitness values are evaluated for further genetic operations.

The next step is to apply the genetic algorithm. Parents are selected and, using the fitness values f_{min}, f_{ave} and f_{max} determined previously, the selection operator re-scales the fitness values of the population via a linear rule and then conventional roulette wheel selection is applied. Selected parents produce new individuals by the crossover operator. A conventional crossover operator is used in which multiple crossover points may be chosen. Some care must be taken after the production of new individuals. Encoded FIR neural networks of different architectures may present problems during genetic operations. The gaps left inside the chromosomes by non-existent neurons or connections may lead to an inconsistent new individual after the crossover operation.

The need for a checking routine following any operation on the chromosomes is clear. The checking procedure attempts to correct distortions of the architecture of the new individuals. The procedure provides a systematic means of identifying anomalies and enables the chromosome strings to be re-arranged in the best way possible. If this is not feasible, then the new individual is discarded. To facilitate the checking

procedure, a number of pre-conditions are assumed to apply:

- Each existing neuron must receive connections from all existing neurons of the previous layer and it must send connections to all existing neurons of the next layer;
- All connections must be present between two adjacent layers;
- Networks must have at least one hidden layer;
- Output neurons must always exist;
- Networks are not allowed to have hidden layers with only one neuron;
- No one individual can be equal to any other individual of the population.

If a new individual is accepted there is also a possibility of that individual being mutated. The new individual's chromosome is swept gene by gene and for each one the mutation operator changes its value with respect to a user-defined probability distribution. Only time-delay values and neuron existence parameters are mutated, although the respective changes in weight and bias values must be carried out. The mutation operator is not allowed to change time-delay values over the limit of the memory span and it is forbidden to mutate output neurons.

A final operation is applied to the accepted new individuals to update their weight and bias values. This operation consists of perturbing each weight and bias value by a normally distributed random value multiplied by a proportionality constant; that is,

$$\begin{aligned} \mathbf{w}_{new} &= \mathbf{w}_{old} + \beta\, \mathbf{N}(0,1) \\ \boldsymbol{\theta}_{new} &= \boldsymbol{\theta}_{old} + \beta\, \mathbf{N}(0,1) \end{aligned} \qquad (6)$$

Analogously to the simulated annealing algorithm[9], the new values of weights and biases are only accepted if they lead to a fitter FIR neural network. This process of updating the weight and bias values is repeated several times before returning the modified individual to the population. In the simulated annealing algorithm, it is possible that poor perturbations may be accepted, but this circumstance occurs with a probability which depends on the system's temperature parameter. This feature is not used in the present procedure.

After one generation, the new and old individuals of the population are compared in terms of their fitness values and the best ones retained for the next generation. With this form of non-replacement, the expectation is that the good genes will not disappear during the genetic search. This procedure also provides a means of accelerating the convergence.

The simple application of mutation to the new individuals, or even the use of a greater number of crossover points, cannot guarantee that the process will not be trapped in a sub-optimal condition. When this occurs, the process appears to stagnate and, if the members of the population are inspected, it is observed that they are all practically equal. In fact, for a good implementation of a genetic algorithm, such behaviour is desired. However, because of the size and complexity of the chromosome encoding a FIR neural network, it is appropriate to implement routines to help the process to escape from the stagnation.

Here, the avoidance of stagnation is treated by another mutation operator. The operator is called *forced mutation* and it starts to work only after a pre-defined number of generations present the same fitness value for the best individual. For each member of the population, the operator randomly chooses a gene to be modified. The gene is then modified and the individual is tested to check if its new fitness is greater than the old one. If this is the case, the individual is accepted, otherwise it is only accepted with a probability of 0.01%. The routine is repeated a number of times and the final mutated individual returns to the population. Such a scheme is basically a variation of the simulated annealing algorithm, but considering constant temperature.

A scheme for the complete training process is summarised in Figure 5.

Step 1 Initialisation
Step 2 Evaluation - Fitness values
Step 3 Genetic Algorithm
    3.1 Selection (roulette wheel)
    3.2 Crossover
    3.3 New Individual OK?
        yes - Mutate and Update Weight/Bias
        no - next step
    3.4 New Population complete?
        yes - next step
        no - back to *3.1*.
Step 4 Evaluation - Fitness values
Step 5 Fitness of the best individual is unchanged
    after a pre-defined number of generations?
    yes - Forced Mutation and back to ***Step 3***.
    no - Stop criteria ?
        yes - end of process
        no - back to ***Step 3***.

Figure 5. Training Process

Functional Approximation of Unsteady Aerodynamic Response

To illustrate the application of the neural network identification procedure to an unsteady aerodynamic system, the low-Mach number unsteady aerodynamic normal force response of a 2-D NACA0012 aerofoil in pitch is considered. The aerodynamic force response is assumed to depend on all past values of the pitch motion and may be interpreted as a non-linear functional of the motion history. Only motion histories compatible with the assumption of weak aerodynamic non-linearities and continuous functional behaviour are considered (motion histories resulting in gross flow separation or deep dynamic stall are therefore excluded). The data used to train the FIR neural network is generated by a semi-empirical unsteady aerodynamic model originally proposed by Beddoes[3,14]. The model uses various input parameters obtained from steady and unsteady wind tunnel data and simple mathematical formulations based on observation of the physical aspects of the flow. For the training process the pitch incidence, $\alpha(t)$, is the prescribed input and the normal force coefficient, $C_N(t)$, evaluated from the Beddoes model is the desired output.

For a range of motion histories, the aerodynamic normal force coefficient exhibits continuous weakly non-linear behaviour. The limit of continuous non-linear behaviour of C_N is assumed as the boundary for network training purposes. For the freestream conditions considered in the present example, numerical experimentation indicates that weakly non-linear behaviour is present for angles of attack in the range 10° to 14° and that discontinuous behaviour becomes apparent for motion histories involving angles of attack greater than 14°.

Identification of the FIR neural network characterising the non-linear relationship between the motion history and the aerodynamic response demands that the network be exposed to a broad range of motion induced unsteady aerodynamic responses. To meet this requirement, a variety of motion histories are presented to the network for training. In the present example, three training cases are considered: sinusoidal motion, ramp-up motion, and ramp-down motion. In each case, the motion history is normalised with respect to the maximum incidence prior to training.

The network architecture and parameters are identified using the genetic algorithm described previously. The following parameters define the maximum complexity FIR neural network in the population:

- 2 hidden layers;
- 10 neurons per hidden layer;
- 4 time-delays per connection.

A population of 40 FIR neural networks is used for training. Five points are applied to the crossover operator. This number is adopted after comparing the performance of the process for other numbers of crossover points. It is observed that better results occur when the fitnesses increase more smoothly. If a greater number of crossover points is used there is more chance of bigger jumps in the fitness values which, in most of the cases examined, leads to stagnation. The stagnation arises because the total replacement approach is not applied to the process. If a greater amount of pressure is induced in the process, sub-optimal solutions may occasionally appear and the process is driven to a local minima.

Other parameters that may influence the performance and accelerate the process are the mutation probability values. These values are applied to the mutation operator as a reference to modify or retain a time-delay value, or to create or delete a neuron. If the chance of such modifications is great, the pressure on the process is proportionally greater, increasing the chance of achieving a local minima. The mutation probability values adopted for training are 0.3% to mutate a time-delay value and 0.1% to create or delete a neuron. The selection operator assumes a scaling coefficient equal to 2.0. Larger values do not improve the process performance.

Weight and bias values are modified considering a perturbation constant $\beta = 0.01$ (*cf.* equation (4)). This parameter is similar to the learning rate parameter commonly applied to the general back-propagation algorithm; therefore, it has a substantial influence on the training performance. Another factor affecting the performance is the number of steps in the routine to update the weight and bias values. Both a large β value and large number of steps in the routine increase the pressure on the process. It is observed that the convergence rate for optimising the weight and bias values is smaller than the convergence rate in the genetic algorithm for the network architecture. Increasing the pressure on updating the weight and bias values does not tune the two algorithms to the same convergence rate. In fact, a larger pressure on the weight and bias value modifications leads only to local minima. It appears to be more appropriate to update the weight and bias values for a few steps and return the modified individual to the population. Such an approach also helps to accelerate the procedure. In the present application, the number of steps is set equal to five. The training procedure has two termination criteria. The first criterion checks if the fitness of the best individual achieves a value of 10^2. The second criterion terminates training after a fixed number of generations. In the present application, the maximum number of generations is 10^5.

Results

The network training sets are presented in Figure 6. Each of the training sets corresponds to a constant freestream Mach number of 0.117 and aerofoil chord of 0.55 m. The pitch incidence in the ramp-up case (Figure 6a) ranges from 0° to 13.5°, while in the ramp-down case (Figure 6b), the incidence ranges from 13.5° to -5°. The sinusoidal input case (Figure 6c) has a mean pitch incidence of 0°, amplitude 13.5° and a frequency of 2.0 Hz. The reconstructed aerodynamic normal force coefficients generated by the adapted FIR neural network model for the same motion histories are also shown for comparison. The sample interval in the discrete-time model is 0.01 s.

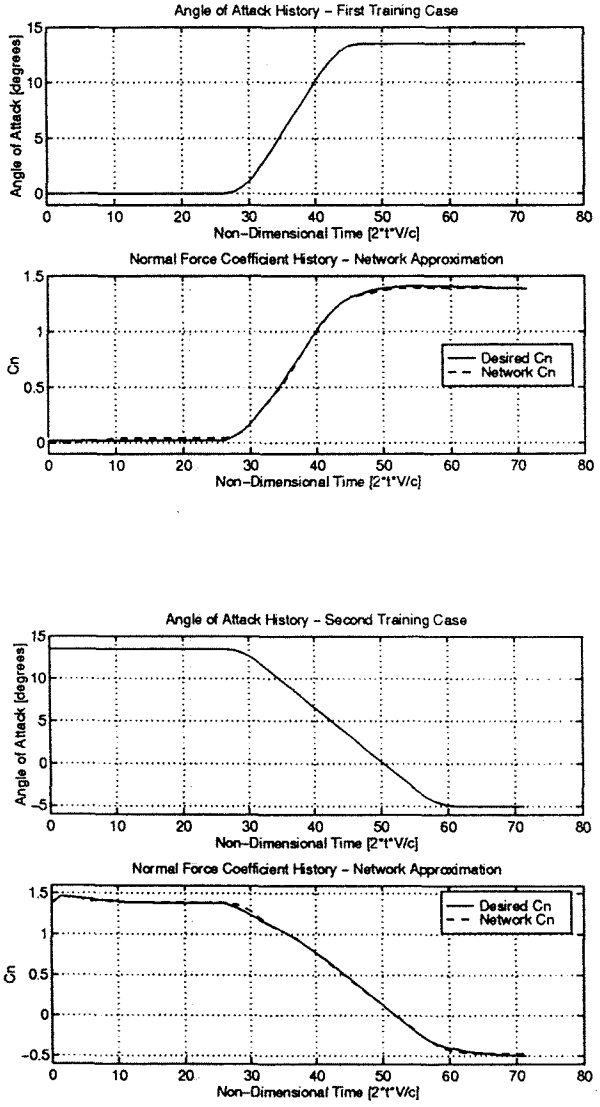

Figure 6. Training Sets and Network Output

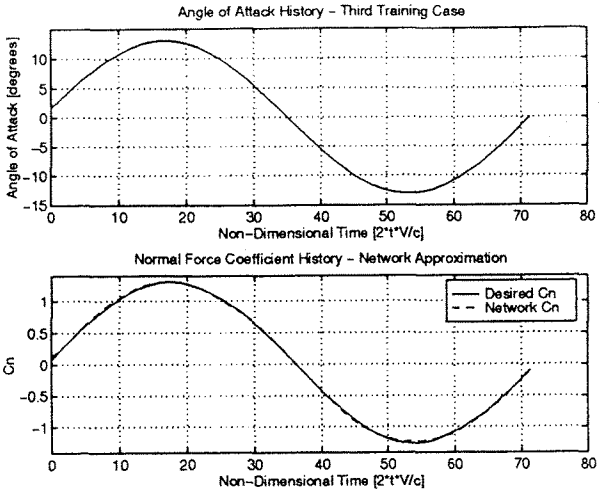

Figure 6. (cont'd)

The form of the adapted network (including the number of time-delays per connection) is illustrated in Figure 7. This network represents the best FIR neural network in the population after the maximum number of generations. The dependence of the aerodynamic response on past values of the motion history is immediately apparent. In addition, the presence of two hidden layers in the network ensures functional complexity.

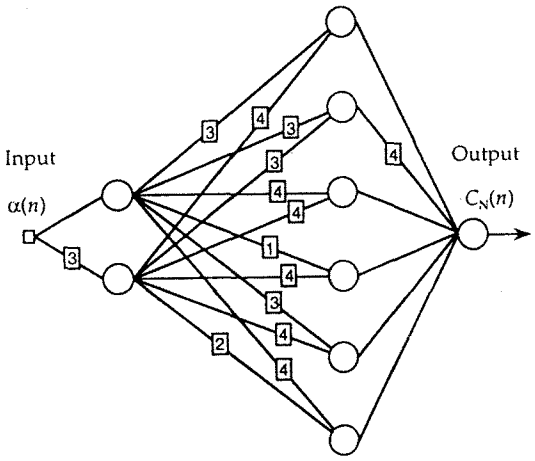

Figure 7. Adapted FIR Network

The robustness of the adapted network identified from the prescribed training sets is examined in Figure 8. Here, the network response to a range of other motion histories is shown to be satisfactory (the freestream conditions and network sampling are unchanged) although minor discrepancies do exist in the early transient phases of the motion. The ability of the network to capture the essential features of both the linear and weakly non-linear aerodynamic behaviour with only a few training patterns is particularly noteworthy.

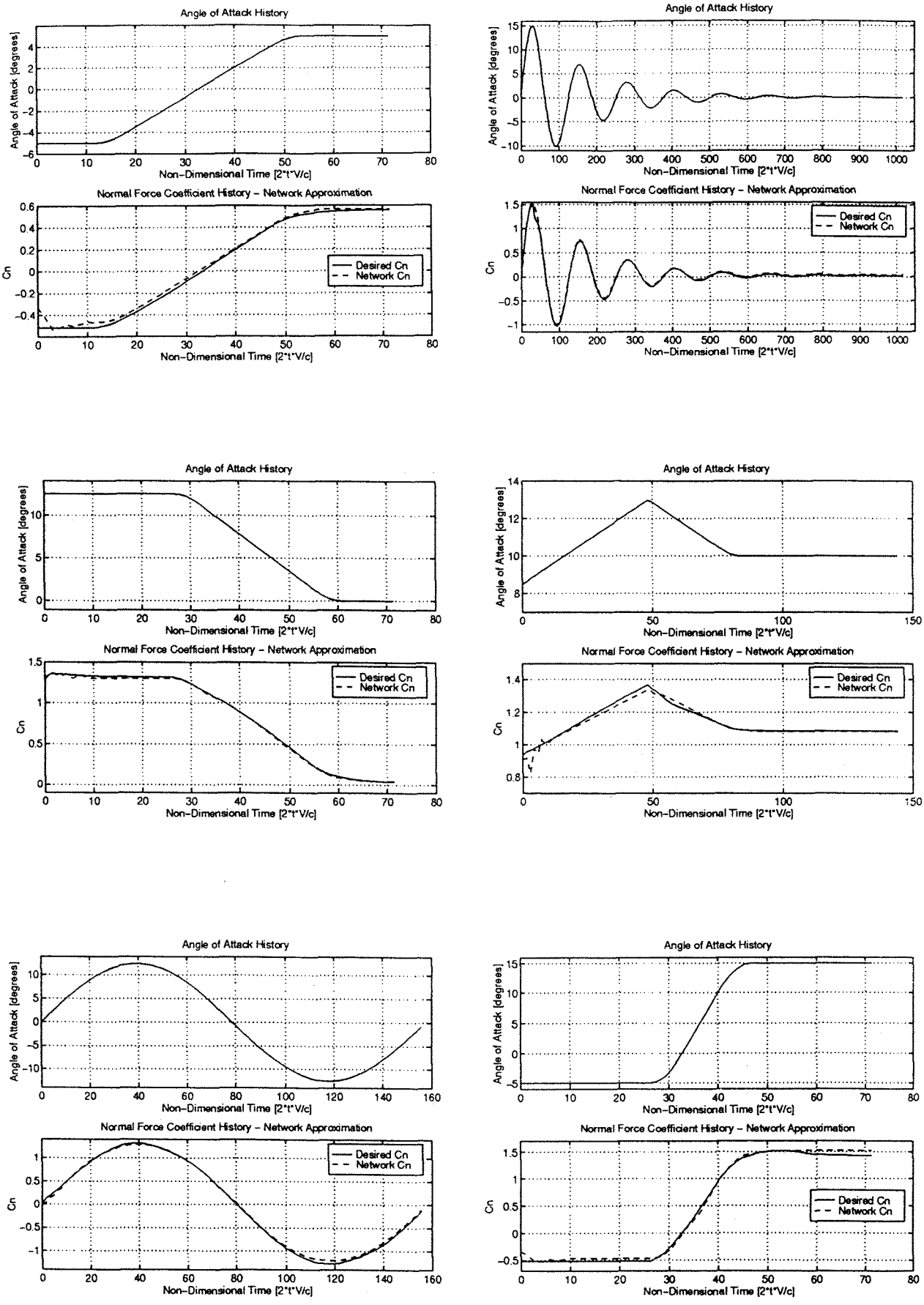

Figure 8. Network Response to Arbitrary Motion Histories

Conclusions

The functional representation of unsteady aerodynamic response characteristics is an efficient model form for weakly non-linear aerodynamic systems. Approximation of the aerodynamic response functional can be realised via a discrete-time multi-layer FIR temporal neural network.

Identification of the network architecture and network parameters is achieved via a supervised learning process using mutliple training sets. The identification procedure is based on a genetic algorithm and a variation of the simulated annealing algorithm. This approach overcomes many of the difficulties associated with the standard temporal back-propagation algorithm, and allows each connection to be assigned a different time-delay.

The use of multiple input/output training sets ensures that the adapted network models a broad range of system characteristics and presents good generalisation features. In addition, multiple input and multiple output variables are readily accommodated by the network model. Furthermore, static parameters, such as Reynolds number or Mach number, can be used as inputs to the neural model, thereby increasing the range of flow conditions and, as consequence, the applicability of neural model.

A major limitation of the FIR neural network approximation is the restriction to continuous functionals. To extend the capabilities of the neural network modelling approach to unsteady aerodynamic systems exhibiting strong non-linearities associated with discontinuous functional behaviour, it is necessary to incorporate additional logic. A continuous-time neural model is also desired to overcome some of the limitations of the discrete version.

Acknowledgements

The authors would like to thank Dr. F. Coton of the Department of Aerospace Engineering, University of Glasgow for his assistance and advice relating to the generation of the unsteady aerodynamic data.

The first author (FM) acknowledges the financial support of the Brazilian government during the tenure of a postgraduate research studentship.

References

(1)NIXON, D. (ed.) *Unsteady Transonic Aerodynamics*, Progress in Astronautics and Aeronautics, Vol. 120, AIAA, Washington DC, 1989.

(2)BEDDOES, T.S., Representation of Airfoil Behaviour, *AGARD Specialists Meeting on Prediction of Aerodynamics Loads on Rotorcraft, AGARD CP-334*, 1982.

(3)BEDDOES, T.S., Practical Computation of Unsteady Lift, *Eight European Rotorcraft and Powered Lift Aircraft Forum*, Aix-En-Provence, France, 1982.

(4)TOBAK, M. and Schiff, L.B., The Role of Time-History Effects in the Formulation of the Aerodynamics of Aircraft Dynamics, *AGARD CP-235, Dynamic Stability Parameters*, Paper 26, May 1978.

(5)TOBAK, M. and Schiff, L.B., Aerodynamic Mathematical Modeling - Basic Concepts, *AGARD Lectures Series*, n. 114, Paper 1, March 1981.

(6)SILVA, W.A., Application of Nonlinear Systems Theory to Transonic Unsteady Aerodynamic Responses, *Journal of Aircraft*, Vol. 30, n. 5, pp. 660-668, 1993a.

(7)VOLTERRA, V., *Theory of Functionals and of Integral and Integro-Differential Equations*, Dover, New York, 1959.

(8)MODHA, D.S. and Hecht-Nielsen, R., Multilayer Functionals, in: *Mathematical Approaches to Neural Networks*, ed. Taylor, J.G., Noth-Holland mathematical library, Elsevier Science Publishers B.V., Amsterdam, The Netherlands, 1993.

(9)HAYKIN, S., *Neural Networks - A Comprehensive Foundation*, Macmillan College Publishing Company, Inc., New York, USA, 1994.

(10)HUSH, D.R. and Horne, B.G., Progress in Supervised Neural Networks: What's New Since Lippmann?, *IEEE Signal Processing Maganize*, Vol. 10, pp. 8-39, January 1993.

(11)WAN, E.A., Temporal Backpropagation for FIR Neural Networks, In: 1990 International Joint Conference on Neural Networks (*IJCNN 90*), San Diego, CA, USA, 17-21 June, 1990a, Vol. 1, pp.575-580.

(12)GOLDBERG, D.E., *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, 1989.

(13)DAVIS, L., ed., *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York, 1991.

(14)NIVEN, A.J. and Galbraith, R.A.McD., *A User's Guide to the Beddoes Model*, University of Glasgow, G.U. Aero Report, May 1991.