Military Technical College
Kobry Elkubbah,
Cairo, Egypt

8th International Conference
On Aerospace Sciences
and Aviation Technology

On-line Identification of longitudinal Aircraft Dynamics Using Extended Neural Networks

A. M. Bayoumy*, F. I. Abdel-Ghany** and I. M. Ibrahim***

ABSTRACT

In this paper Artificial Neural Networks (ANN) approach is used to identify the longitudinal aircraft dynamics online. Memory Neuron NN (MNN) is used here as a model to perform online non-parametric identification of aircraft dynamics. The aircraft longitudinal motion is modeled as a linear system. The input and output of the aircraft model are used as training pair to the NN model. NN is extended by adding two more parameters, the maximum input and maximum output vectors, to learn the changes in I/O bounds dynamically. After the training phase the NN output shows good agreement with the aircraft output for the same input.

KEYWORDS

Aircraft, Neural Networks, online Identification.

1. INTRODUCTION

The development of Artificial Neural Networks (ANN), or simply Neural Networks (NN), began approximately fifty years ago, motivated by a hope to try both: understanding the brain and emulating some of its functions. The most famous NN is the Feed Forward Neural Network (FFNN). It is also known as Multi-Layer Precptron (MLP). It is trained using Error Back Propagation, or simply Backpropagation (BP) [1], [2]. NN is recently employed in the field of system identification and control. Neural networks have its particularities to be useful in this field. NN can learn by example, so no analytical knowledge about the system is required to perform the identification.
MNN is first suggested in [3], but in [4] it is suggested as an alternative to FFNN in identification and control of nonlinear dynamical elements. A variation of the BP algorithm called dynamic backpropagation (DBP) is used as the learning algorithm.
In this work the aircraft is modeled, for simulation purposes, as a Single Input Single Output (SISO) linear system. A NN model called neuroidentifier (NNI) is attached in

* Eng. Amgad M. Bayoumy, Military Technical College (MTC), Cairo, Egypt
** Assoc.Prof. Fawzy Ibrahim AbdelGhany, Chairman of Radar and Guidance Department, MTC, Cairo, Egypt
*** Prof. Ibrahim Mansour Ibrahim, Chairman of Design Center, Ministry of Military Production, Cairo, Egypt

parallel with the aircraft model. The instantaneous input and output of the aircraft model are used as a training pair to the NNI during the learning mode. After completing the online learning of the NNI (learning epochs to zero), the NNI is put in test mode. An arbitrary testing signal is applied as input to both aircraft model and NNI.
This paper is organized as follows. Section 2 introduces the aircraft model for the simulation purposes. Section 3 summarizes the NN model. Section 4 discusses the on-line identification scheme. Section 5 describes the necessity for extended NN, its mechanism and the AutoScaling algorithm used to train it. Section 6 analyzes the results of the computer simulation. Section 7 shows the main conclusions of this work.

2. AIRCRAFT MODEL

The aircraft model used here is a linear longitudinal model with elevator deflection (δ_e) as the input and aircraft pith rate (q) as the output. The aircraft model has four states [U W θ q] which are linear velocities in x and z direction, pitch angle and pitch rate, respectively. The input (u), output (y) and state vector (X) of the can be written as:

$$\begin{aligned} u &= [\delta_e] \\ X &= [U \quad W \quad \theta \quad q]^T \\ y &= [q] \end{aligned} \tag{1}$$

The model used here is of a general aviation aircraft [5], appendix B, pp. 252.
The linear model has the form:

$$\begin{aligned} \dot{X} &= AX + Bu \\ y &= CX \end{aligned} \tag{2}$$

Where A, B, C is the state, control and output matrices (Appendix A).

3. NEURAL NETWORK MODEL

Every NN model has three main characteristics: structure, processing algorithm and learning algorithm. The NN model used here is the MNN [4]. MNN is suggested as an alternative to FFNN in identification and control nonlinear dynamical elements. It has some characteristics similar to the FFNN and others similar to recurrent networks. In MNN every network neuron has, associated with it, a memory neuron whose single scalar output summarizes the history of past activation 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 now a recurrent one. The MNN can be sufficient for identifying nonlinear dynamical systems. A variation of the BP algorithm called dynamic backpropagation (DBP) is used as the learning algorithm.

3.1. Structure

The architecture of a Memory Neuron Network (MNN) is shown in Fig. 1. The structure is the same as a Feed Forward NN except for the memory neurons. These neurons are shown as small filled circles attached to each network unit. Network neurons are shown as large open circles. To distinguish these two types of units, the terms network neuron and memory neuron are used. As can be seen at each level of the network except the output level, each of the network neurons has exactly one memory neuron connected to it. The memory neuron takes its input from the corresponding network neuron and it also has a self-feedback as shown in the legend. This leads to accumulation of past data of the network neuron in the memory neuron. All the network neurons and the memory neurons of each level send their outputs 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 output to that network neuron in the output layer. The shown network has two input nodes, one output node and a single hidden layer.

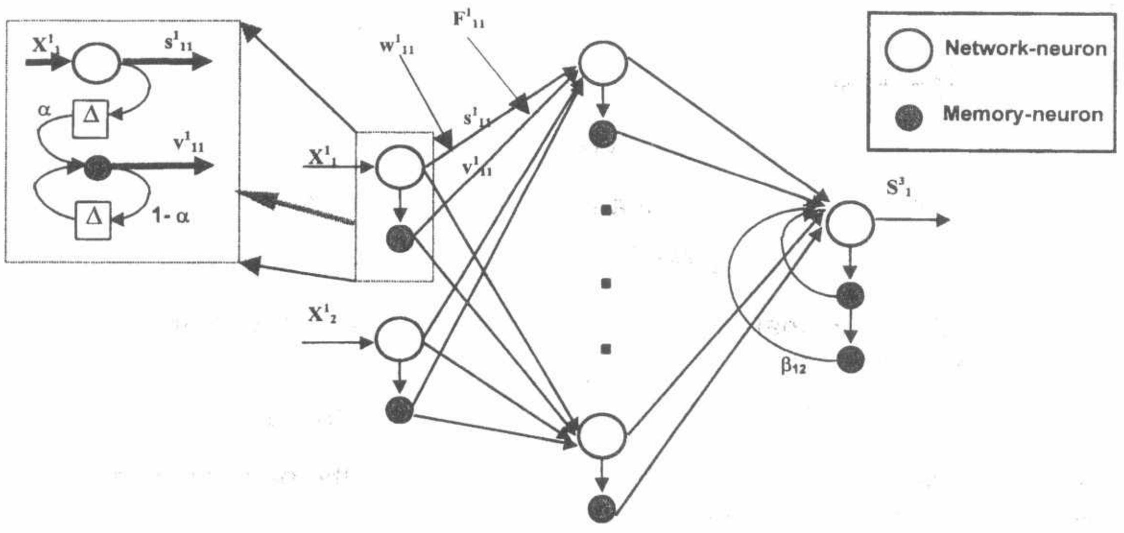

Fig.1. Architecture of Memory Neuron Neural Networks

Symbol list

The following notation will be used to describe the functioning of the network.

L	Is the number of layers of the network with layer 1 as the input layer and layer L as the output layer.
N_l	is the number of network neurons in layer l
$x^l_j(k)$	Is the net input to the j^{th} network neuron of layer l at time k
$s^l_j(k)$	Is the output of the j^{th} network neuron of layer l at time k
$v^l_j(k)$	Is the output of the memory neuron of the j^{th} network neuron in layer l at time $k.1 \le l \le L$

$w_{ij}^{l}(k)$	Is the weight of the connection from i^{th} network neuron of layer l to j^{th} network neuron of layer $l+1$ at time k
$f_{ij}^{l}(k)$	Is the weight of the connection from the memory neuron corresponding to the i^{th} network neuron of layer l to the j^{th} network neuron of layer $l+1$ at time k k
$\alpha_{j}^{l}(k)$	Is the weight of the connection from j^{th} network neuron in layer l to its corresponding memory neuron at time $k.1 \le l \le L$
$\alpha_{ij}^{L}(k)$	Is the weight of the connection from the $(j-1)^{th}$ memory neuron to the j^{th} memory neuron of the i^{th} network neuron in the output layer at time k^{3}
$v_{ij}^{L}(k)$	Is the output of the j^{th} memory neuron of the i^{th} network neuron in the output layer at time k
$\beta_{ij}^{L}(k)$	Is the weight of the connection from the j^{th} memory neuron of the i^{th} network neuron to the i^{th} network neuron in the output layer at time k
M_{j}	Is the number of memory neurons associated with the j^{th} network neuron of the output layer
$g(.)$	Is the activation function of the network neurons.

3.2. Processing Algorithm

The output of network neurons is given as in [4]:

$$s_{j}^{l}(k) = g\left(x_{j}^{l}(k)\right), 1 \le l \le L \tag{3}$$

The output of memory neurons for any layer *l<L* is given by:

$$v_{j}^{l}(k) = \alpha_{j}^{l}(k) s_{j}^{l}(k-1) + (1-\alpha_{j}^{l}(k)) v_{j}^{l}(k-1) \tag{4}$$

In last layer the additional successive *m* memory neurons output are function of the previous memory neuron

$$v_{ij}^{L}(k) = \alpha_{ij}^{L}(k) v_{ij-1}^{L}(k-1) + (1-\alpha_{ij}^{L}(k)) v_{ij}^{L}(k-1) \tag{5}$$

The state of network neurons is given from the O/P of the neurons of the previous layer and the weight matrices W, F. for any layer *l<L*

$$x_{j}^{l}(k) = \sum_{i=0}^{N_{l-1}} w_{ij}^{l-1}(k) s_{i}^{l-1}(k) + \sum_{i=1}^{N_{l-1}} f_{ij}^{l-1}(k) v_{i}^{l-1}(k) \tag{6}$$

In last layer, *L*, the additional memory neurons and their weights β are added:

$$x_{j}^{L}(k) = \sum_{i=0}^{N_{L-1}} w_{ij}^{L-1}(k) s_{i}^{L-1}(k) + \sum_{i=1}^{N_{L-1}} f_{ij}^{L-1}(k) v_{i}^{l-1}(k) + \sum_{i=1}^{M_{j}} \beta_{ji}^{L}(k) v_{ji}^{L}(k) \tag{7}$$

3.3. Learning Algorithm

The error function of the network is defined as the Sum of Squared Errors (SSE), given as:

$$e(k) = \sum_{j=1}^{N_l} (s_j^L(k) - y_j(k))^2 \qquad (8)$$

Backward computing

$$e_j^L(k) = (s_j^L(k) - y_j(k))g'(x_j^L(k)) \qquad (9)$$

$$e_j^l(k) = g'(x_j^l(k))\sum_{p=1}^{N_{l+1}} e_p^{l+1}(k)w_{jp}^l(k),\ 1 \le l < L \qquad (10)$$

Weight Update

$$w_{ij}^l(k+1) = w_{ij}^l(k) - \eta e_j^{l+1}(k)s_i^l(k),\ 1 \le l < L \qquad (11)$$

$$f_{ij}^l(k+1) = f_{ij}^l(k) - \eta e_j^{l+1}(k)v_i^l(k),\ 1 \le l < L \qquad (12)$$

Memory coefficients update

$$\alpha_j^l(k+1) = \alpha_j^l(k) - \eta' \frac{\partial e}{\partial v_i^l}(k)\frac{\partial v_j^l}{\partial \alpha_j^l}(k),\ 1 \le l < L \qquad (13)$$

At the last layer L:

$$\alpha_{ij}^L(k+1) = \alpha_{ij}^L(k) - \eta' \frac{\partial e}{\partial v_{ij}^L}(k)\frac{\partial v_{ij}^L}{\partial \alpha_{ij}^L}(k), \qquad (14)$$

$$\beta_{ij}^L(k+1) = \beta_{ij}^L(k) - \eta' e_i^L v_{ij}^L \qquad (15)$$

Where:

$$\frac{\partial e}{\partial v_j^l}(k) = \sum_{s=1}^{N_{l+1}} f_{js}^l(k)e_s^{l+1}(k) \qquad (16)$$

$$\frac{\partial v_j^l}{\partial \alpha_j^l}(k) = s_j^l(k-1) - v_j^l(k-1)$$

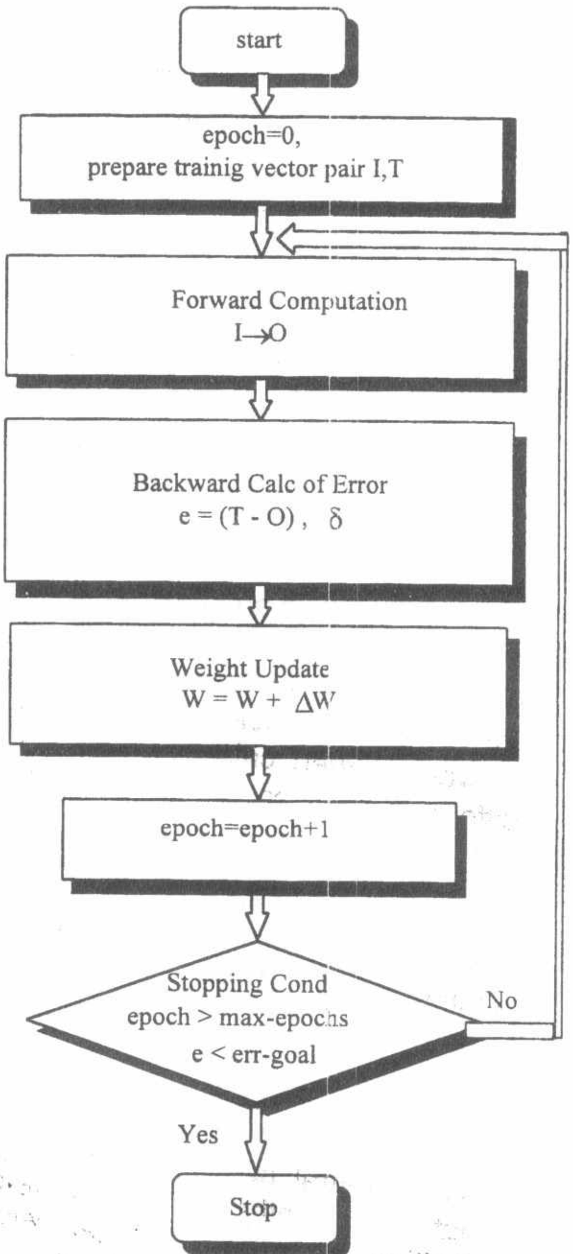
start
epoch=0, prepare trainig vector pair I,T
Forward Computation I→O
Backward Calc of Error e = (T - O), δ
Weight Update W = W + ΔW
epoch=epoch+1
Stopping Cond epoch > max-epochs e < err-goal
No
Yes
Stop

Fig.2. Flow chart of NN on-line learning

$$\frac{\partial e}{\partial v_{ij}^L(k)} = \beta_{ij}^L(k)e_i^L(k) \qquad (17)$$

$$\frac{\partial v_{ij}^L}{\partial \alpha_{ij}^L}(k) = v_{ij-1}^L(k-1) - v_{ij}^L(k-1) \qquad (18)$$

4. ON-LINE NN IDENTIFICATION SCHEME

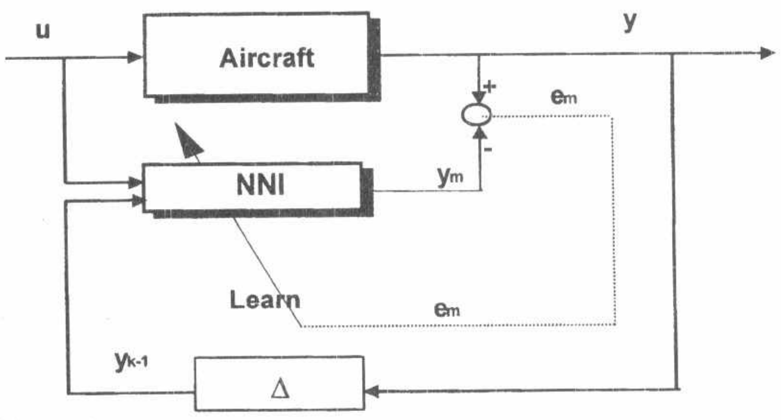

Fig.3. Basic Scheme of NN on-line identification

The basic architecture is very simple; the control signal and the plant states (suitably delayed) are sampled and this forms the network-input vector. The output of the network is then calculated and compared with the measured plant output, and the difference between these two quantities is used to adjust the weight vector to reduce the network output error:

$$e_m(t) = y(t) - y_m(t) \tag{19}$$

The learning rules are typically formulated so that they minimize the Mean Square Output Error (MSE):

$$E = \frac{1}{2} e_m^2 \tag{20}$$

Static neural network is the network that could only learn a static mapping between patterns. In other words it is the feedforward only networks. An example of this is the multi-layer precptron (MLP) networks with the backpropagation algorithm. These networks are more common and well understood.
Any observable controllable dynamic system can be represented as:

$$y(k) = h(\bar{y}(k-1), \bar{u}(k)) \tag{21}$$

Where:

$$\bar{y}(k-1) = [y(k-1), y(k-2), \ldots.., k(k-n)]^T$$

$$\bar{u}(k) = [u(k), u(k-1), \ldots.., u(k-m)]^T$$

Where: n and m are the numbers of delays necessary for determining the next plant output. These variables (n, m) vary from one system to another and are called the plant order.

In order to represent a dynamic plant with the static networks, one should use taped delays of the previous system outputs and inputs by exactly the order of the system.

The order of the plant (the number of delays: *n* and *m*) must be known exactly. Over-estimating their value results in poor convergence rates and generalization as the model is over-parameterized. Choosing too small a value means that unmodeled dynamics exist that may affect the stability of any learning control system.
The *dynamic neural network*, *neural network with feedback*, or *Recurrent Neural Network* in the other hand is the network that uses its previous output as an input,. The dynamic NN itself can be considered as a dynamic system. So, trying to identify a dynamic plant with it is not a big problem. For the previous reasons the dynamic NN is employed in the system identification used in this work.
The flowchart of the NN online identification is shown in Fig.4. At each time step *k* a teaching signal *u* is applied as an input to both the aircraft and the NNI. The difference of outputs is then used to adjust the weights of the NNI until the local error em tends to zero at each time step. After some time steps, the global error between the A/C and the NNI tends to zero and the learning process terminated.

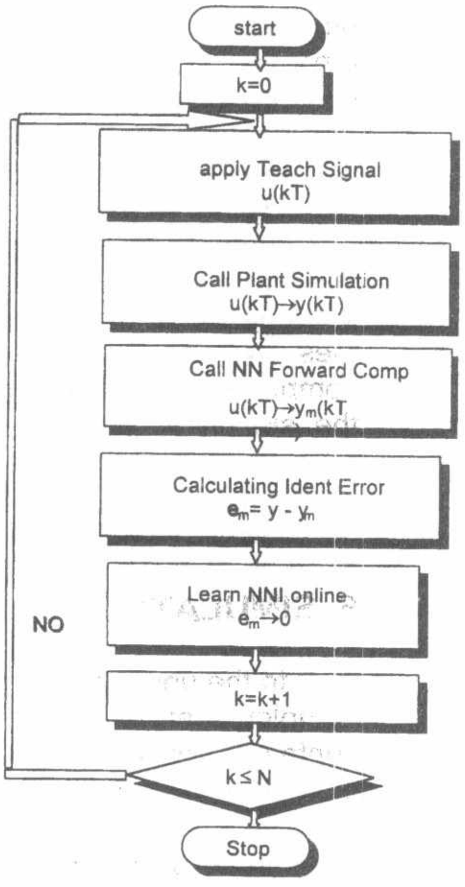
start
k=0
apply Teach Signal u(kT)
Call Plant Simulation u(kT)→y(kT)
Call NN Forward Comp u(kT)→y_m(kT
Calculating Ident Error e_m= y - y_m
Learn NNI online e_m→0
k=k+1
k≤N
NO
Stop

Fig.4. On-line NN Identification Flow Chart

5. EXTENDED NEURAL NETWORKS

The extended NN scheme is shown in Fig.5. NN is extended by two more trainable blocks. The first block is a divisor block with I_{max} , which is vector of max components of the input vector. The second block is a multiplier with O_{max} and added after the standard NN block. The function of these blocks is to work as interface between the scaled NN and the real world. The input and output pair of the standard NN block, $\bar{I},\bar{O}$, are scaled in the range [-1,1]. I_{max} and O_{max} are trained in the learning phase of the NN to capture the I/O bounds and any changes in them, online. The algorithm used to train these blocks is introduced here as AutoScaling algorithm.

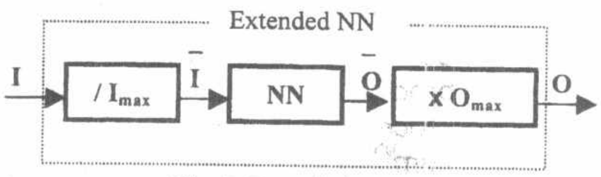

Fig. 5. Extended NN Scheme

5.1. AutoScaling Algorithm

To overcome the problem of network saturation which bound the output between the range: [-1.0,+1.0] one may scale the input output pairs given to the network to be in the given interval. The scaling of the input vectors may be linear or nonlinear. The nonlinear scaling (e.g. with *tanh(.)*) forces the input space to be bounded but in the other hand cause the distortion of the input vectors, so usually the linear scaling is performed. (e.g. dividing each element of the input vector with a max. value). A problem appears is that before identification nobody is assumed to know anything about the system so assume the I/O bounds to be known is not fair. For this purpose a technique called AutoScaling is proposed, implemented and tested in this study.
The AutoScaling is used only in the training phase of the neuroidentifier. The NN parameters are increased by two more parameters: I_{max} and O_{max}, which are the max input and output vectors respectively. Before training the I/O pair is compared with these max. vectors and if they are greater the new pair is used as the max I/O. Every component in the I/O pair is divided by these max vectors. Due to the saturation of the sigmoid functions at about 0.9 the max. values are multiplied by factor 1 / 0.9 $\approx$1.1.

6. SIMULATION AND RESULT ANALYSIS

Due to the unavailability, to authors, of ready-made neurocontrol software package, a complete set of C++ library programs is built and developed during this work. It contains Matrix and Vector library, Chart drawing library, Numerical algorithms, ... and much more. All of these libraries are built using the Object-Oriented Programming concept, which has several advantages over the classical procedural-oriented programming approach.
The simulation is performed on IBM PC-Pentium machine of 100 MHz speed.
In this simulation the aircraft model (described in section 2) is attached in parallel with the NN model (described in section 3) as explained in section 4. The Extended NN (described in section 5) is used here to perform online scaling of the I/O pairs.
The simulation is carried out with integration step (T) equals 0.01 sec.
The choice of learning rates is a critical job. Large learning rates give short learning time but the possibility of learning instability increases. On the other hand, small learning rates increases the total time of learning. Adaptive learning rate selection is a good trade-off. The learning rates are magnified when the error is decreased with time, and decreased when the error grows.
Stopping condition of the online learning is another critical factor. The learning process at each time step is stopped when decreasing the error but if the learning process does not converge, the learning process is terminated after a predetermined number of epochs in order to follow the changes in the aircraft. These two factors play a great role in the online identification process. The stopping conditions should not be very severe in order to get global rather than local learning.
The teaching signal used here is a bounded signal (either sinusoidal or square signal) in the full range of the input. The testing signal is an arbitrary signal.

The simulation is carried out into two phases:

The first one is the '*learn phase*' in which the teaching signal is applied to the aircraft and the NNI. The learning algorithm is used to adjust the weights of NNI. The second one is the '*test phase*' in which the weights of the NNI are kept fixed.
The NN structure is:

- **Input layer:** two input neurons (u_k , y_{k-1})
- **One hidden layer:** 10 neurons
- **Output layer:** has 1 output neuron ($y_{m\,k}$), and 4 memory neurons.

Fig.6 shows the output of the aircraft model and the NNI, put in '*test mode*'. The simulation shows good agreement between the response of the aircraft model and

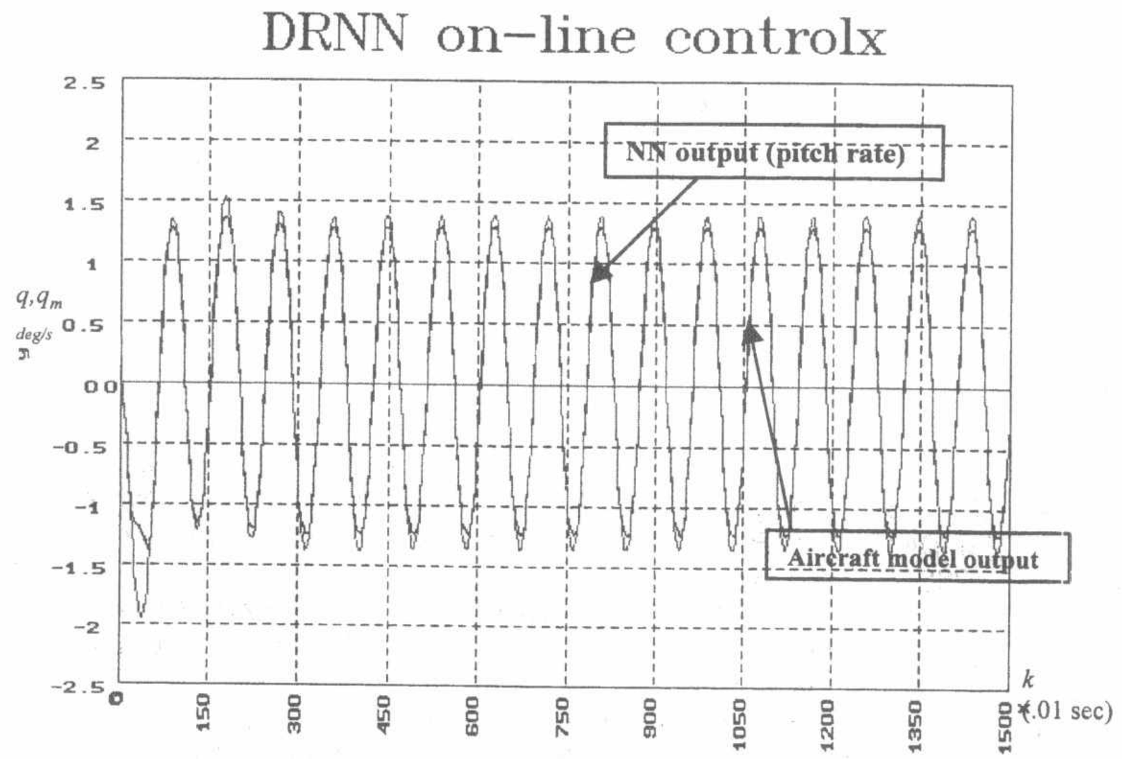

Fig.6. Comparison of the neuroidentifier (NNI) output and the aircraft response for a sinosoidal signal (Test mode)

the NN identified model for the same input.

7. CONCLUSION

In this paper neural networks are used as a tool for online identification of aircraft longitudinal dynamics. Neural network is generally a very good tool in the field of system identification. NN can even identify the system while it runs without any lost time. Any sudden change in the model could be identified without much effort. But there are some constrains exist. First, the plant must be Bounded Input Bounded Output (BIBO) stable. Second, the learning rate must be chosen carefully. Greater learn rate cause learning oscillations while small learn rate may cause the learning to increase learning time. But adaptive learning rate selection reduces this constrain.

The extended NN is useful in the field of online identification. I/O bounds are learned with the AutoScaling algorithm, so, no previous knowledge about them is assumed before identification.

Appendix A

The aircraft model matrices are given in [5] as:

A=

-0.04409	0.035268	0	-32.185
-0.3615	-1.97941	176	0
0.001827	-0.03888	-2.92085	0
0	0	1	0

B=

0
27.54415
-11.7653
0

C=

0	0	0	1

References

[1] Rumelhart, D. E., G. E. Hinton, & R. J. Williams. "Learning Internal Representations by Error propagation." In D. E. Rumelhart & J. L. McClelland, eds., Parallel Distributed Processing, vol. 1 chapter 8. reprinted in Anderson & Rosenfeld (1988), pp. 675-695.

[2] Rumelhart, D. E., G. E. Hinton, & R. J. Williams. "Learning Representations by Back-Propagation Error." Nature, 323: 533-536. Reprinted in Anderson & Rosenfeld (1988), pp. 696-699.

[3] P. Poddar and K. P. Unnikrishnan, "Memory neuron networks: A prolegomenon," Tech. Rep. GMR-7493, General Motors Research Laboratories, (1991)

[4] Sastry, P. S., Santharam G., and Unnikrishnan, K. P., "Memory Neuron Networks for Identification and Control of Dynamical Systems", IEEE Trans. On Neural Networks, Vol. 5, No. 2, March (1994).

[5] Nelson, R. C., Flight Stability and Automatic Control, McGraw Hill, (1990).

[6] A. M. Bayoumy, "Neural Network Approach to Aircraft Inverse Dynamics Control", M.Sc. Thesis Report, MTC, (1997)