Military Technical College
Kobry Elkubbah,
Cairo, Egypt

8th International Conference
On Aerospace Sciences
and Aviation Technology

On-line Trajectory Control of Aircraft Using Neural Networks

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

ABSTRACT

In this paper an artificial Neural Network (NN) is used to control an aircraft along a predetermined trajectory. The inverse dynamics concept is used. Two NN models are used: one for forward identification NNI) nd he ther ne or nverse odel identification and forward control (NNC). After training of both models the NNC is used as a forward controller to the aircraft model. The simulation results shows good agreement between required and achieved trajectory.

KEY WORDS

Aircraft, Neural Networks, Inverse Dynamics, Trajectory Control

1. INTRODUCTION

The concept of aircraft trajectory control was imported from the field of robotics. The term *'trajectory control'* can be interchanged with the term 'inverse dynamics'. The use of inverse dynamics approach to aircraft maneuvers was started thirty years ago, but many advances have been made since this date. This approach was found to give answers to the questions like *"How an airplane should be controlled when its flight maneuver is given?"* In 1959, Etkin introduced, to the first time, the inverse problems of aircraft maneuvers [1]. He gave an application to single degree of freedom pure rolling maneuver. In 1981, a pioneer work was done by Meyer *et al.* [2] who used the concept of nonlinear inverse dynamics to construct a transformation from natural nonlinear representation of the system into an equivalent linear constant-coefficient reference model. In 1993, Al-Bahi and Abdel-Rahman presented a general formulation of the inverse problem [3]. The method was applied to the case of circular loop prescribed maneuvers. In 1994, Al-Bahi and Abdel-Rahman presented a generalized technique for inverse simulation along predetermined

* 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

trajectory [4]. The prescribed technique can be given either in analytic form or as a succession of points.
Recently, neural networks approach was employed to find the exact inverse neurocontroller of the aircraft [5]. The main interesting feature of the neural networks is its ability to learn. This method is superior to the previous methods in many aspects: the mathematical model is not needed and any changes in the system parameters can be tracked and overcome, so, It can be used with high uncertainties, actuator sluggishness, and battle damaged or partially missing control surfaces.
In this work the NN was employed to perform online trajectory control. Two NN models were used the first one is called the neuroidentifier (NNI) and the other is called the neurocontroller (NNC).

2. AIRCRAFT MODEL

The used aircraft model is a single input single output (SISO) discrete linear system. It models the longitudinal dynamics of a fighter aircraft trimmed at horizontal steady level flight with constant velocity and altitude [6] pp. 173.
The input of the model is the elevator deflection angle (δ_e) and the output is the pitch rate (q). The model is given as a transfer function in the s-domain, as:

$$\frac{q}{\delta_e} = \frac{-10.45s(s+0.9871)(s+0.02179)}{(s+1.204 \pm j1.492)(s+0.007654 \pm j0.07812)} \cdot \frac{\text{deg}/s}{\text{deg}} \qquad (1)$$

3. ON-LINE CONTROL SCHEME

In order to achieve on-line control, a neural network model, called neuroidentifier

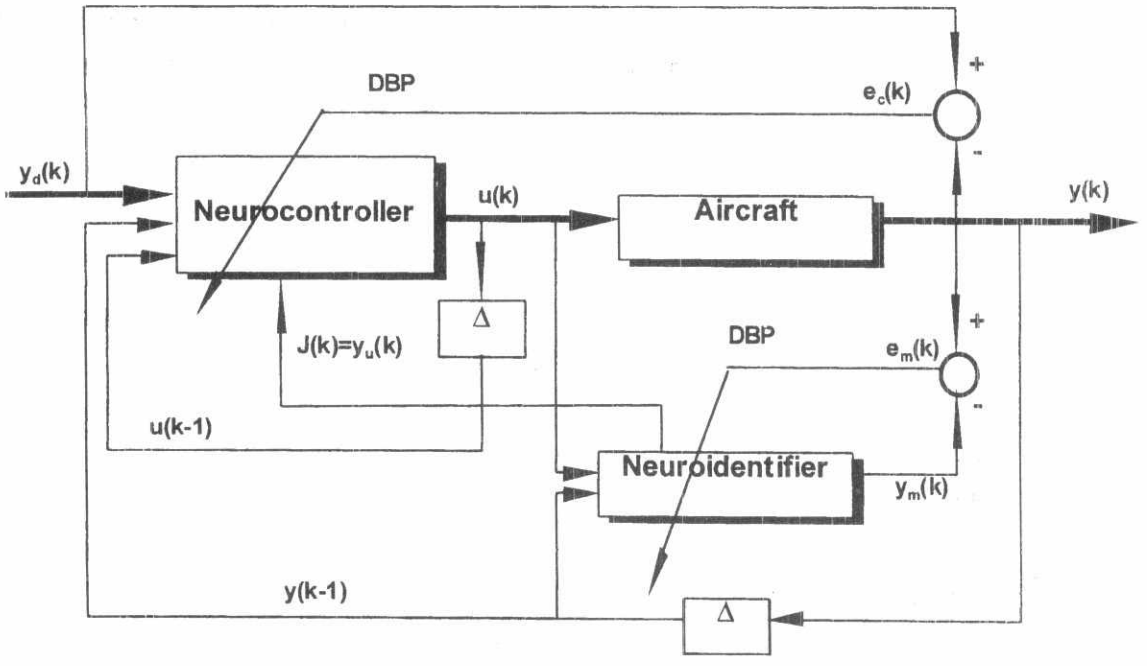

Fig.1. Basic scheme of NN on-line control

(NNI), is selected, implemented and used to identify the aircraft forward dynamics online. An arbitrary-chosen teaching signal is used as a control input to activate the dynamics of the aircraft model. The neuroidentifier is put in the learn mode. At every time step a learning algorithm, called dynamic back propagation (DBP), is used to adjust the weights of the neuroidentifier to minimize the local error between the neuroidentifier and the aircraft model responses. This local learning is assumed to minimize the overall error between the aircraft and the neuroidentifier. After the complete learning process, another neural network model, called neurocontroller, is constructed to model the inverse dynamics of the aircraft online. The neurocontroller is put as a feedforward controller. The neurocontroller is put in learn mode. The dynamic backpropagation is used to minimize the error between the neurocontroller input, which is the desired output, and the actual output of the aircraft model. To avoid disturbances to the aircraft during the learning phase of the neurocontroller, the output of the neuroidentifier is used instead of it. The neurocontroller uses the jacobian or sensivity matrix computed by the neuroidentifier to compute the error in the control-input signal. When the local error between the desired and achieved output tends to zero the learning of the neurocontroller is completed. The neurocontroller is then used to control the aircraft over two specified trajectories. Good agreement is noticed between the specified and achieved trajectories.

The three phases of operation of the above scheme can be summered as:

I. Identification of the forward dynamics of the A/C. (learn NNI)
II. Identification of the inverse dynamics model of the A/C. (learn NNC)
III. Control of the aircraft over the specified trajectory. (test NNC)

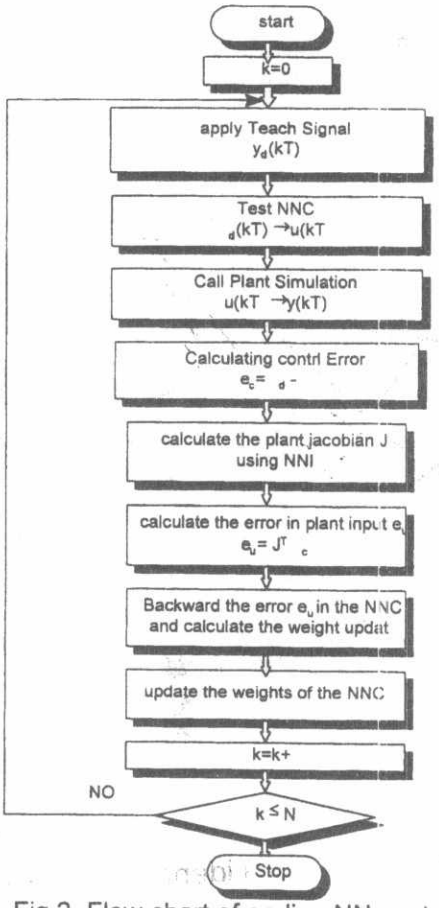

Fig.2. Flow chart of on-line NN control

Notes:

In real conditions, the learning phases, phase I and phase II, occurred during normal flight conditions with aircraft controlled by another (conventional, non-intelligent) control system. In the off-design conditions, emergency, the scheme learns the deviations and controls the aircraft along the predetermined trajectory. For study purpose we give the aircraft a teaching signal (sinusoidal or pulse) as control input u(k)and desired output yd(k) during learning phases (phase I and II).

For the purpose of robustness, NNI continues to learn in both the phase II and phase III for online adaptation of the aircraft inverse.

The approach for control and system identification using diagonal recurrent neural network (DRNN) [12-15] is used in this work. A system identifier, called the diagonal recurrent neurocontroller (DRNI) identifies the unknown plant, the neurocontroller is used to drive the unknown dynamic system

such that the error between plant and desired output is minimized. A generalized algorithm, called the dynamic backpropagation (DBP), is developed to train both DRNC and DRNI.

4. NEURAL NETWORKS MODEL

A recurrent neural network (RNN) is defined as a feedforward network which allows feedback for some of the nodes, i.e., the output of a node becomes input to a node in a preceding or in the same layer [7-10]. The output of a RNN, therefore, depends not only on the inputs to the network, but also on the current state of the network. Thus, the RNN is a (nonlinear) dynamic network, while the FFNN is a static network. The RNNs have important capabilities not found in FFNN, such as their dynamics and the ability to store information for later use. Of particular interest is their ability to deal with time varying input or output through their own natural temporal operation [11]. Thus the RNN is a dynamic mapping and is better suited for dynamic systems than the FFNN.

4.1. Structure

With the objective of a simple RNN and a shorter training time for the neural network model, a diagonal recurrent neural network (DRNN), Fig.3, is developed [15]. It can be shown that the DRNN model is a dynamic mapping in a way the Fully connected Recurrent NN (FRNN) is dynamic. Since there is no interlinks among neurons in the hidden layer, the DRNN has considerably fewer weights than the FRNN and the network is simplified considerably. The hidden layer is a recurrent one, but the output

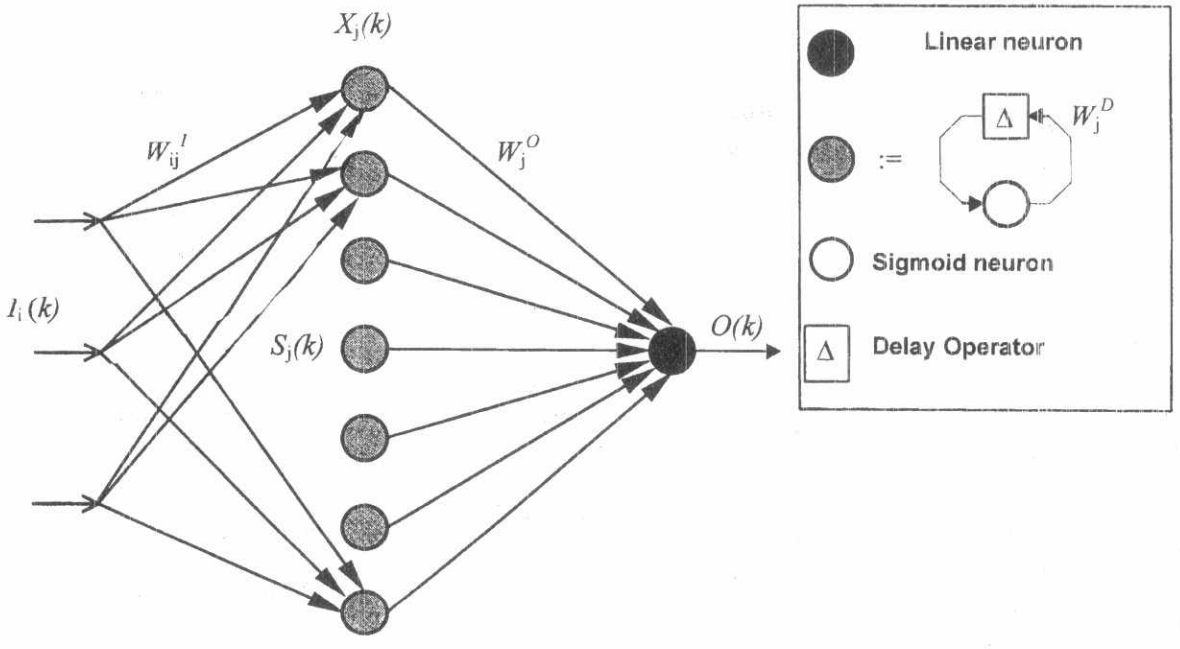

Fig.3. Structure of DRNN

of every neuron in the hidden layer is fed-backed only to itself [12-15]. DRNN shows good characteristics in identification and control of nonlinear dynamic systems.

4.2. Processing Algorithm (Forward Computation)

$$S_j(k) = \sum_i W_{ij}^I I_i(k) + W_j^D X_j(k-1) \tag{2}$$

$$X_j(k) = f(S_j(k)) \tag{3}$$

$$O(k) = \sum_j W_j^o X_j(k) \tag{4}$$

Where for each discrete time k, $I_i(k)$ is the i^{th} input to the DRNN, $S_j(k)$ is the sum of inputs to the j^{th} recurrent neuron, $X_j(k)$ is the output of the j^{th} recurrent neuron, and $O(k)$ is the output of the DRNN. The bias term can be included as one of the inputs as usual. Here $f(.)$ is the usual sigmoid function, and W^I, W^D, and W^O, are input, recurrent, and output weight vectors, respectively, in $\Re^{ni}$, $\Re^{nd}$, and $\Re^{no}$.

4.3. Learning Algorithm

The NNC error is defined as:

$$E_c = \frac{1}{2}(y_d(k) - y(k))^2 \tag{5}$$

And for NNI:

$$E_m = \frac{1}{2}(y(k) - y_m(k))^2 \tag{6}$$

Where $y_m(k) = O(k)$ of (4). The gradient of error in (5) with respect to an arbitrary weight vector $W \in \Re^n$ is represented by:

$$\begin{aligned} \frac{\partial E_c}{\partial W} &= -e_c(k) y_u(k) \frac{\partial u(k)}{\partial W} \\ &= -e_c(k) y_u(k) \frac{\partial O(k)}{\partial W} \end{aligned} \tag{7}$$

where $e_c(k)=y_d(k)-y(k)$ is the error between the desired and output responses of the plant, and the factor $y_u(k) \equiv \frac{\partial y(k)}{\partial u(k)}$ represents the sensitivity of the plant with respect to its input.

Since the plant is normally unknown, the sensitivity needs to be estimated for DRNC. However, in the case of the DRNI, the gradient of error in (6) simply becomes:

$$\frac{\partial E_m}{\partial W} = -e_m(k) \frac{\partial y_m(k)}{\partial W} = -e_m(k) \frac{\partial O(k)}{\partial W} \tag{8}$$

where $e_m(k)=y(k)-y_m(k)$ is the error between the plant and the DRNI responses.

The output gradient $\partial O(k)/\partial W$ is common for both DRNC and DRNI.

Dynamic Backpropagation For DRNI

$$\frac{\partial O(k)}{\partial W_j^O} = X_j(k) \tag{9}$$

$$\frac{\partial O(k)}{\partial W_j^D} = W_j^O P_j(k) \tag{10}$$

$$\frac{\partial O(k)}{\partial W_{ij}^I} = W_j^O Q_{ij}(k) \tag{11}$$

Where:

$$P_j(k) \equiv \frac{\partial X_j(k)}{\partial W_j^D} \text{ and } Q_{ij} \equiv \frac{\partial X_j(k)}{\partial W_{ij}^I} \tag{12}$$

And satisfy

$$\begin{aligned} &P_j(k) = f'(S_j)(X_j(k-1) + W_j^D P_j(k-1)), \\ &P_j(0) = 0. \end{aligned} \tag{13}$$

$$\begin{aligned} &Q_{ij}(k) = f'(S_j)(I_i(k) + W_j^D Q_{ij}(k-1)), \\ &Q_{ij}(0) = 0. \end{aligned} \tag{14}$$

The negative gradient of the error with respect to a weight vector in $\Re^N$ is:

$$-\frac{\partial E_m}{\partial W} = e_m(k)\frac{\partial O(k)}{\partial W} \tag{15}$$

The weights can be adjusted following a gradient method, i.e., the update rule of the weights becomes

$$W(n+1) = W(n) + \eta(-\frac{\partial E_m}{\partial W}) \tag{16}$$

Where η is a learning rate. The equations (9)-(16) define the dynamic backpropagation algorithm (DBP) for DRNI.

Dynamic Backpropagation For DRNC

In the case of DRNC, from (7), the negative gradient of the error with respect to a weight vector in $\Re N$ is

$$-\frac{\partial E_c}{\partial W} = e_c(k) y_u(k) \frac{\partial O(k)}{\partial W} \tag{17}$$

Since the plant is normally unknown, the sensitivity term $y_u(k)$ is unknown. This unknown value can be estimated by using the DRNI. When the DRNI is trained, the dynamic behavior of the DRNI is close to the unknown plant, i.e., $y(k) \approx y_m(k)$, where $y_m(k)$ is the output of the DRNI.

Once the training process is done, we assume the sensitivity can be approximated as

$$y_u(k) \equiv \frac{\partial y(k)}{\partial u(k)} \approx \frac{\partial y_m(k)}{\partial u(k)} \tag{18}$$

where u(k) is an input to the DRNI.
Applying the chain rule to (18), and noting that $y_m(k)=O(k)$ of (4),

$$\begin{aligned} \frac{\partial y_m}{\partial u(k)} &= \frac{\partial O(k)}{\partial u(k)} = \sum_j \frac{\partial O(k)}{\partial X_j} \frac{\partial X_j(k)}{\partial u(k)} \\ &= \sum_j W_j^O \frac{\partial X_j(k)}{\partial u(k)} \end{aligned} \tag{19}$$

Also from (2)-(4):

$$\frac{\partial X_j(k)}{\partial u(k)} = f'(S_j(k)) \frac{\partial S_j(k)}{\partial u(k)} \tag{20}$$

Since inputs to the DRNI are u(k) and y(k-1) (Fig.1), eq.(2) becomes:

$$S_j(k) = W_j^D X_j(k-1) + W_{1j}^I u(k) + W_{2j}^I y(k-1) + W_{3j}^I b_I \tag{21}$$

where b_I is the bias for DRNI. Thus:

$$\frac{\partial S_j(k)}{\partial u(k)} = W_{1j}^I \tag{22}$$

From (19), (20), and (22),

$$y_u(k) \approx \frac{\partial y_m(k)}{\partial u(k)} = \sum_j W_j^O f'(S_j(k)) W_{1j}^I \tag{23}$$

Where the variables and weights are those found in DRNI.

SIMULATION AND RESULTS

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.

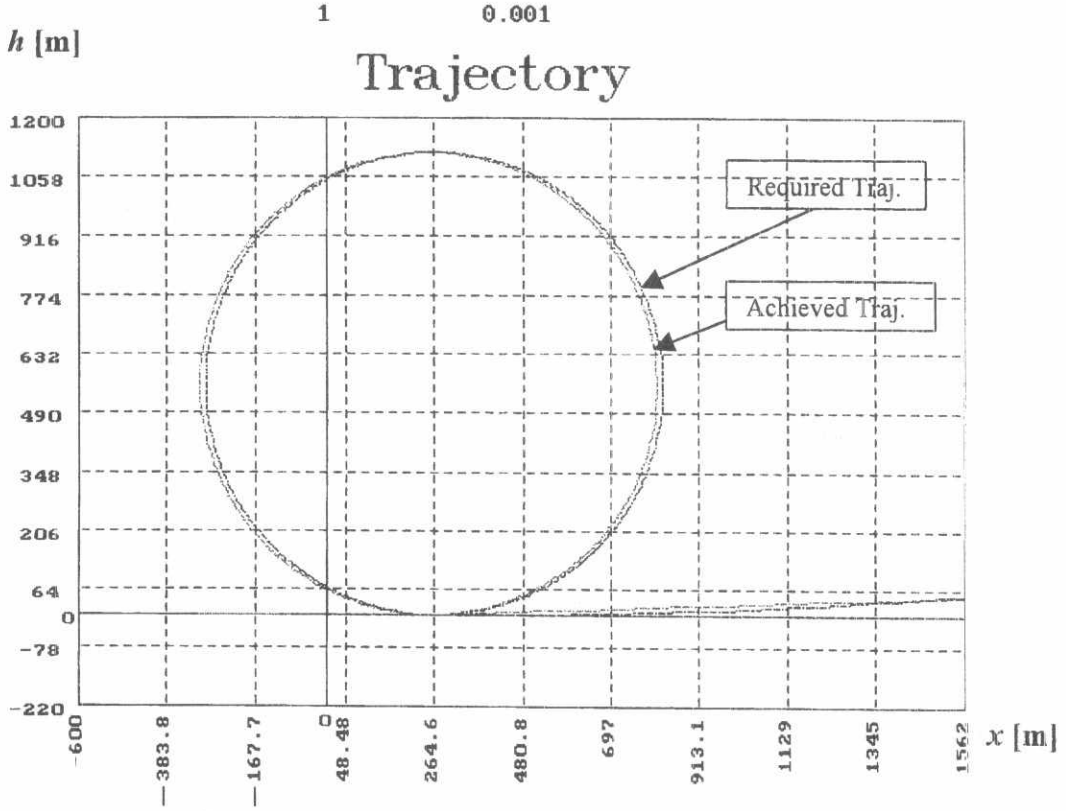

Fig.4. Circular loop Trajectory (required and achieved)

The simulation is performed on IBM PC-Pentium machine of 100 MHz speed.
The simulation is carried out at integration step T=0.01 sec.
The NNI structure is given as follows:

- **Input Layer:** consists of 2 neurons (y_{k-1} , u_k)
- **Hidden Layer**: consists of 7 neurons
- **Output Layer**: consist of 1 neuron (y_k)

The NNC structure is given as:

- **Input Layer:** consists of 3 neurons (y_d, y_{k-1} , u_k)
- **Hidden Layer:** consists of 9 neurons
- **Output Layer:** consist of 1 neuron (u_k)

Fig.4 shows the required and achieved trajectories.
The required and achieved pitch rate is shown in Fig.5.
The control inputs generated by the NN controller is shown in Fig.6.

The following remarks can be addressed from the computer simulation:

- The inverse modeling it is not as easy as the forward identification
- The problem of learning stability is generally highlighted.

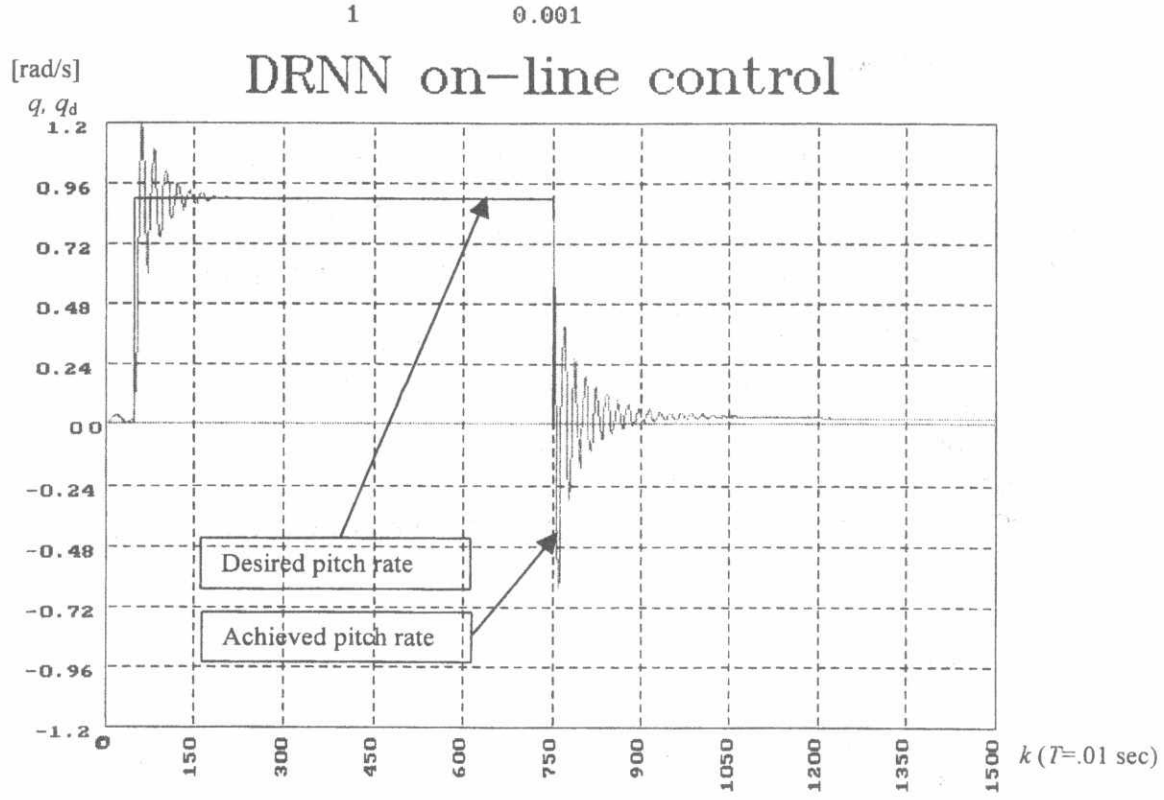

Fig.5. Desired and achieved pitch rate (the output) history

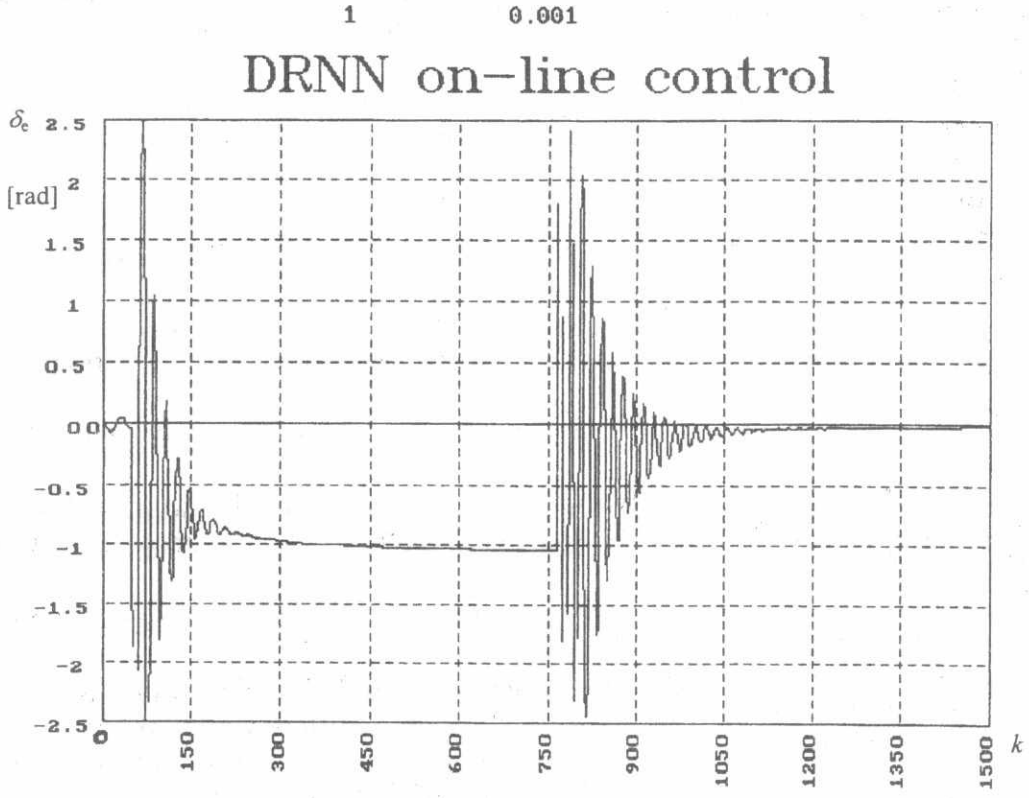

Fig.6. Commanded input (elev. angle) generated by NNC

CONCLUSION

In this work the NN is used as a forward controller, using inverse dynamics concept, to control the aircraft along predetermined trajectory. Two NN models were used: one for forward identification of aircraft dynamics and the other for inverse modeling of the aircraft. After the complete training of both models, a predetermined trajectory, circular loop trajectory is given to the inverse controller as a pitch rate history. The achieved trajectory shows good agreement with the prescribed one.
The study shows that the neural network is a very good tool in the field of identification and inverse dynamics control. It can even identify and control the aircraft while it runs without any time loss. Any changes in the plant dynamics could be captured and reclaimed online.
Dynamic or recurrent neural networks are much better than static neural networks in both identification and control.

REFERENCES

[1] Etkin, B., "Dynamics of Flight", John Wiley & Sons, New York, 1959, chap.11., pp. 341-352
[2] Meyer, G. and L. Cicolani (1981). "Application of nonlinear system inverses to automatic flight control designs-system concepts and flight evaluations. In Theory and Application of Optimal Control in Aerospace Systems", AGARD-AG251, pp. 10.1-10.29.
[3] Al-Bahi, A. M., and Abdel-Rahman, M. M., "Numerical Simulation Of Aircraft Prescribed Maneuvers," IASTED International Journal of Modeling and Simulation, Vol. 13, No. 4, pp. 141-145, 1993.
[4] Al-Bahi A. M. and Abdel-Rahman M. M., "A Generalized Technique For The Inverse Simulation Of Aircraft Motion Along Predetermined Trajectories", AIAA paper 94-3523-CP, 1994
[5] Sadhukhan, D. and Feteih S. , "F8 Neurocontroller Based on Dynamic Inversion," Journal of Guidance, Control, and Dynamics, Vol. 19, No. 1, Jan-Feb 1996, pp. 150-156.
[6] Stevens B. L.and Lewis F. L., "Aircraft Control And Simulation", John Willey & Sons, 1992.
[7] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," in Proc. Nat. Academy of Sciences, vol. 79,1982, pp. 2554-2558.
[8] F. J. Pineda, "Generalization of backpropagation to recurrent networks," Phys. Rev. Lett., vol. 59, no. 19, pp. 2229-2232, Nov. 1987.
[9] F. J. Pineda, "Recurrent backpropagation and the dynamical approach to adaptive neural computation," Neural Computation, vol. 1, pp. 161-172, 1989.
[10] B. A. Pearlmutter, "Learning state space trajectories in recurrent neural networks," Neural Computation, vol. 1, pp. 263-269, 1989.
[11] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," Neural Computation, vol. 1, pp. 170-180, 1989.

[12] C. C. Ku and K. Y. Lee, "System identification and control using diagonal recurrent neural networks," in Proc. the 1992 Anerican Control Conference, Chicago, IL, June 24-26, 1992, pp. 545-549.

[13] C. C. Ku and K. Y. Lee, "Diagonal recurrent neural networks based control using adaptive learning rates," in Proc. the 31st IEEE Conference on Decision and Control, Tuscon, AZ, Dec. 16-18, 1992, pp. 3485-3490.

[14] C. C. Ku and K. Y. Lee, "Diagonal recurrent neural network-based control: Convergence and stability," in Proc. the 1994 American Control Conference, Baltimore, MD, June 29- July 1, 1994, pp. 3340-3345.

[15] C. C. Ku and K. Y. Lee, "Diagonal recurrent neural networks for dynamic system control," IEEE Trans. on Neural Networks, vol. 6, no. 1, pp. 144-156, Jan. 1995.

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