Flight Implementation of Dual Neural Network Control Technique for a Low-cost UAV

Sreenatha G. Anavatti[a,*] and Vishwas Puttige
[a]School of Aerospace, Civil and Mechanical Engineering, UNSW@ADFA, Canberra, Australia

Abstract

In this paper a novel indirect adaptive control technique based on neural networks for Unmanned Aerial Vehicles (UAV) is described. The technique demonstrates the use of two interconnected neural networks to provide faster tracking of the commanded reference. A pre-trained internal model network and an online trained controller network form the core of the dual neural network (DNN) architecture. A Telemaster UAV equipped with various sensors and an onboard data-logger and control (ODC) unit forms the experimental platform. The plant model and the controller have been designed in numerical simulations based on data obtained from experimental flights of the UAV. To validate the applicability of the DNN controller it has been flight tested on the UAV to demonstrate autonomous operation. Results from the numerical simulations and flight test are provided.

Key words: Modelling, adaptive control, UAVs, nonlinear system, neural networks

1. INTRODUCTION

Unmanned Aerial Vehicles (UAVs) have gained importance due to their reduced operating costs and repeated operations without human intervention. Most of the available research is generally simulation based [1,2,3]. Thus there is limited research available on low-cost, small-sized fixed wing UAVs for commercial applications. Currently, there are a few universities working towards the development and implementation of different algorithms on UAVs in realtime [4,5]. Similar fixed-wing platforms are under development at the School of Aerospace, Civil and Mechanical Engineering in UNSW@ADFA.

The UAV is a six degree of freedom (DOF), nonlinear, complex system. UAVs display inertial coupling and aerodynamic coupling effects and hence can be modelled as a MIMO system with six outputs and a minimum of four inputs. The flight control system (FCS) for the UAV performs tasks similar to a pilot for a manned aircraft. Control techniques, capable of adapting themselves to the changes in dynamics of the platform are necessary for the autonomous flight. Such adaptive controllers can be developed with the aid of suitable system identification (ID) techniques. Numerous system identification techniques have been proposed for the modelling of nonlinear systems. Fuzzy identification [6], statespace identification [7], frequency domain analysis [8], artificial neural networks [9] are some of the prominent ones. The ability of the neural networks to learn continuously and to approximate complex processes with sufficient ease make them suitable for modelling nonlinear systems. This paper proposes an indirect adaptive control technique based on Dual Neural Network (DNN).

The DNN controller is adapted from the self tuning regulator (STR) concept proposed in [10,9]. In self tuning regulators a neural network model provides predictions of the plant behaviour. These predictions are used to tune the gains or weights of a controller. However, when a neural network is used as the controller, updating the weights by validating the outputs against the external plant model

Email addresses: vishwas.puttige@gmail.com (Vishwas Puttige),
s.anavatti@adfa.edu.au (Sreenatha G. Anavatti)
URL: http://www.unsw.adfa.edu.au (Vishwas Puttige)
*Corresponding author

may not be possible in real-time. This is overcome by using two networks in the DNN control technique. Here, along with the online trained controller network, another offline trained model network of the plant is included internally as part of the controller structure. This model network allows the controller outputs to be validated at every iteration of training. Similar to the STR technique, the controller outputs are verified against an external model of the plant.

To collect data during flight, the Telemaster UAV is equipped with various sensors, actuators and an onboard data-logger and control (ODC) unit. Sensors are mounted on the Telemaster UAV to measure parameters such as, roll rate, pitch rate, yaw rate, angle of attack, side-slip angle and forward velocity. The actuators on the UAV are, ailerons, elevator, rudder and throttle. Data from the sensors and actuators are collected and processed in the ODC unit. The UAV platform is flown in the remotely piloted mode to collect flight data in various flight regimes. The flight data collected is used to develop black-box models of the UAV and in the design of controllers. To demonstrate the application of DNN control technique on the UAV, the angle of attack subsystem is considered in this paper.

The description of the UAV platform is presented in section 2. The proposed DNN controller is presented in section 3. The adaptive controller architecture and online training scheme are explained here. The numerical simulation results and the flight implementation results are provided in section 4. Some concluding remarks are presented in the section 5.

2. UAV PLATFORM

Many of the current UAV platforms in the market are available with built-in avionics unit and a few also include controllers for autonomous flight. However, their astronomically high costs make them unsuitable for simple civilian applications. On the contrary, most of the low-cost commercial platforms and avionic units have limited use due to the employment of non-reconfigurable conventional controllers with limited flight envelopes. Practical applications of UAV require advanced control techniques with the ability to cater for varying mission profiles and payloads.

The aim of the fixed wing UAV group at the School of Aerospace, Civil and Mechanical Engineering (ACME) of ADFA is to develop low-cost, small sized autonomously controlled UAVs for different applications using advanced control techniques. The purpose is not only to cater to defence applications but also to commercial civilian ones. There is also a necessity to strike a balance between the cost of the instrumentation and the quality of data obtained from it.

The avionics architecture for the fixed-wing UAV, Telemaster is shown in Fig. 1. The shaded region in the block diagram indicates the components onboard the UAV platform. The pilot uses a remote controlled (RC) transmitter to control the aircraft during the remotely piloted mode. Signals received by the RC receiver on-board the UAV are fed to actuators through a hand-over take-over (HOTO) signal conditioning unit. The HOTO unit is designed to perform signal conditioning operations and to switch actuator control between the pilot and an onboard data-logger and control (ODC) unit. Sensors mounted on the UAV provide flight information to the ODC unit in real-time. For the Telemaster UAV, the ODC unit is a PC104 based onboard computer with additional cards

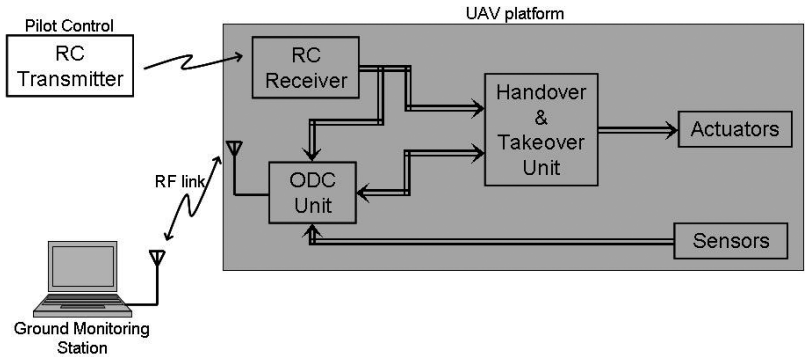

Figure 1. Avionics architecture for the fixed-wing UAVs

Figure 2. The Telemaster UAV ready to take-off during a test flight

mounted as a stack. The main processor card runs on a real-time operating system. An analog to digital converter card digitizes the sensor data. A pulse width modulation (PWM) counter/generator card is used to measure the PWM signals duty cycles given to the actuators by the pilot during remotely piloted mode. The Telemaster UAV prepared to take-off during one of its test flights is shown in figure 2.

The ODC unit is also used for control operations. The controller logic is programed into the main processor card of the ODC unit. The PWM card also generates PWM signals as commanded by the controller program. A wind vane attached to a potentiometer forms the angle of attack sensor. The sensor is calibrated in the wind tunnel to obtain accurate angle measurements. The entire system identification and control logic is programmed in Matlab-Simulink. The xPC target option and real-time window target from Matlab is used to develop the kernel program for real-time operation.

3. UAV CONTROLLERS

Adaptive control requires the adjustment of controller parameters to changes in the dynamics of the system. As compared to non-adaptive controllers, adaptive controllers provide wider operating range. However, traditional adaptive controllers may lack the ability to handle uncertainties in nonlinear dynamic systems. Intelligent indirect adaptive control techniques such as neural networks provide effective control of complex processes with the ability to achieve higher levels of autonomy [11]. In these controllers, both the identifier model and the controller are neural networks and online adaptations are expected to occur in both the networks.

It is necessary to update the adaptive control parameters for a nonlinear system continuously in accordance with the dynamic behaviour of the system. In the proposed control architecture, a pre-trained offline model network is concatenated with a controller network internally. The internal model network is trained with actuator inputs and steady state outputs of the nonlinear plant. A schematic block diagram of the proposed DNN architecture is shown in Fig. 3. Here, TF is the

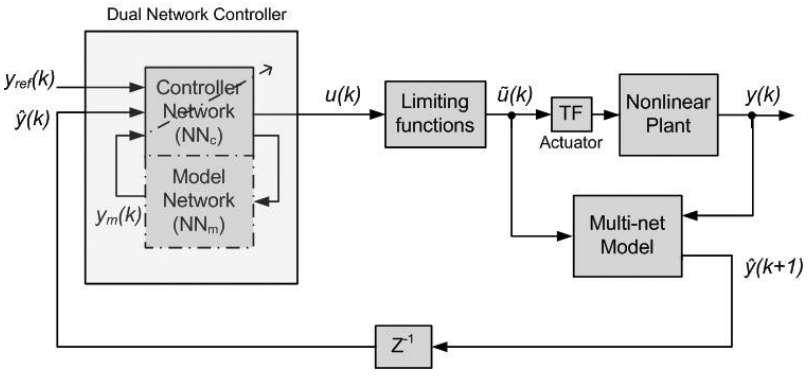

Figure 3. Block diagram of the overall adaptive system

actuator transfer function. As compared to the self tuning regulators where the controller outputs are validated against an external model at every instant of time, the DNN architecture has the internal model network which provides corrections to the controller outputs at every training iteration. Controller network outputs upon completion of training are validated against an external plant identifier model that is trained online. The online trained model predicts the plant behavior corresponding to the inputs from the DNN controller. This predicted output is compared with the commanded reference input and suitable modifications to the weights are performed to attain the desired plant outputs at every instant of time. This effectively provides two feedback loops for the controller network, one by the internal model network at every iteration and another by the external online trained model at every sample time.

3.1. Controller Architecture

The model network (NN_m) and the controller network (NN_c) have autoregressive recurrent neural network structures. The entire online controller training process occurs within a single sample period. The reference input ($y_{ref}(k)$), online plant model output ($\hat{y}(k)$) and the autoregressive terms form the external inputs to the controller network. The model network output (y_m) forms the internal input to the controller network. Saturation limits on the absolute values of the controller outputs and limiting functions on the rate of change of these outputs are imposed to provide suitable bounds on the controller outputs. The outputs from the controller ($u(k)$) are checked for compliance with the limiting functions at every instant of time before feeding as inputs to the plant. The plant inputs ($\tilde{u}(k)$) and its past outputs ($y(k)$) are used by the multi-net model to provide further predictions of the plant behaviour. The controller utilizes these predictions to train and calculate the next set of plant inputs.

Consider the plant model with nonlinear autoregressive structure and exogenous inputs (ARX) introduced in [12]. Let the weights matrices and the model network regressor function be represented by θ_m and ϕ_m respectively. The ARX model predicted output can be written as

$$\begin{aligned} y_m(k|\theta_m) &= f(a_1 y_m(k-1) + ..a_{na} y_m(k-na) \\ &+ b_1 u(k-1) + ..b_{nb} u(k-nb)) = f(\theta_m \cdot \phi_m(k)) \end{aligned} \tag{1}$$

where the predicted output ($y_m(k|\theta_m)$) depends on the past inputs ($u(k-1)...$) and the past outputs ($y_m(k-1)...$). The nonlinear mapping between the inputs and the outputs are represented by the function $f(.)$. A neural network model provides an approximate solution to the above equation without the need to analytically deduce the values of the coefficients [13].

The neural network approximation of this ARX model is given by

$$\begin{aligned} y_m(k|\theta_m) &= F_{1s}\{\sum_{j=1}^{l_{11}}\{W_{11js} G_{1j}\{\sum_{i=1}^{l_{12}}\{W_{12ij} \cdot x_{1i}(k) \\ &\quad + W_{120}\}\}\} + W_{110}\} \\ \theta_m(k) &= [W_{11js}, W_{110}, W_{12ij}, W_{120}] \\ \phi_m(k) &= x_{1i}(k) \end{aligned} \tag{2}$$

where y_m is the network predicted output, W_{11js} and W_{12ij} are the output layer and hidden layer weights matrices, F_{1s} and G_{1j} are the activation functions of the neurons in the output and the hidden layers respectively. l_{11} and l_{12} are the number of neurons in the two layers, W_{110} and W_{120} are the bias to the two layers and x_{1i} is the network input vector. i, j and s are the indices to the input vector, hidden nodes and output nodes in the network respectively.

Let the internal model network (NN_m) in the DNN controller be represented by (2). The weights matrices (θ_m) for this offline trained network remain constant with respect to time. This model network derives its inputs from the controller network outputs. The initial weights for the controller network are chosen such that the initial predictions are in the neighborhood of the equilibrium state. One possible method to achieve this is to use previously trained weights as the starting points for online training.

The controller network is an online trained ARX based recurrent neural network. Let the weights for the controller network (NN_c) be represented as a vector θ_c and the regressor matrix by ϕ_c. The predicted output from the controller network is given by

$$\begin{aligned} y_{2i}(k|\theta_c) &= F_{2i}\{\sum_{j=1}^{l_{21}}\{W_{21ji}G_{2j}\{\sum_{s=1}^{l_{22}}\{W_{22sj}\cdot y_{in}(k) \\ &\quad + W_{220}\}\}\} + W_{210}\} \\ \theta_c(k) &= [W_{21ji}, W_{210}, W_{22sj}, W_{220}] \\ \phi_c(k) &= y_{in}(k) \end{aligned} \tag{3}$$

where W_{21ji} and W_{22sj} are the weights of the hidden and output layers, F_2 and G_2 are the activation functions of the neurons in the output and the hidden layers respectively. l_{21} and l_{22} are the number of neurons in the two layers, W_{210} and W_{220} are the bias to the two layers and y_{in} is the controller network input vector. The controller input vector, y_{in} is obtained from a combination of the commanded reference input and the predicted plant outputs from the multi-network model. This combination can be called as the effective input vector of the controller network given by

$$y_{in}(k) = Fcn\,[\hat{y}(k), \hat{y}\,(k-1), \ldots, y_{ref}(k), y_{ref}(k-1)..]. \tag{4}$$

From (1) and (3) the model network output can be expressed as the function of the effective input vector given by

$$y_m(k|\theta_m) = f(\theta_m \cdot g(\theta_c(k)\cdot y_{in}(k))) \tag{5}$$

where, f and g are the neural network approximations of the model and the controller networks respectively. It can be seen that response from the internal model network relies on the effective input vector to the DNN controller. The training algorithm minimises the error difference between the model output and this input vector. The operation of the DNN controller is presented in algorithm. 1.

The adaptive controller is turned on when the system is in its equilibrium state or in the neighbourhood of this equilibrium. The equilibrium values for each of the control element (actuator inputs) is recorded and corrections to these values are provided to achieve the desired response. For

Algorithm 1 Dual network adaptive controller algorithm

1. Initialize iteration ($n = 0$)
2. Initialize controller Weights $\theta_{c_n}(k)$
3. Construct controller regressor $\phi_c(k)$
4. Evaluate controller output $u_n(k) = f(\theta_{c_n}(k) * \phi_c(k))$
5. Construct model regressor $\phi_{m_n}(k)$
6. Evaluate $y_{m_n}(k) = g(\theta_m * \phi_{m_n}(k))$
7. Evaluate $PI_n(k) = fn||y_{m_n}(k) - y_{in}(k)]$
8. **while** $PI_n(k) < PI_{th}$ or $n < I_{max}$ or $\lambda_{min} < \lambda_{c_n} < \lambda_{max}$ or $Gradient < ||G_{\max}||$ **do**
9. Evaluate *Gradient, Hessian* and update $\theta_{c_{n+1}}$
10. Reevaluate controller output $u_n(k+1)$
11. Reevaluate model output $y_{m_n}(k+1)$
12. Update $PI_n(k+1)$
13. Update λ_{c_n}
14. Increment n
15. **end while**
16. Provide u as plant input
17. Update memory stack with θ_c

example, the trimmed flight condition can be considered as the equilibrium position for an aircraft in flight. The aircraft can be trimmed at different flight conditions and at each time the trimmed values are utilised in the control process. The equilibrium values are used with the reference inputs to obtain the effective input vector into the DNN controller.

The DNN architecture shown in Fig. 3 has different sources for indication of the plant's performance. Other than the plant itself, the multi-net model provides online trained predictions and the internal offline model network allows validation of the controller during training. The right choice of inputs in combination is vital in achieving desired performance by the plant. It is assumed that predictions from the multi-net model comply to certain bounds imposed on them. The effective input vector is given by

$$y_{in}(k) = y_{ref}(k) - y_{eq}(k) + \hat{y}(k) \tag{6}$$

where $y_{eq}(k)$ is the equilibrium value and $\hat{y}(k)$ is the predicted output from the multi-network model. The controller network (NN_c) outputs $y_{2i}(k|\theta_c)$ from (3) is validated through the model network (NN_m) at every iteration of training. Upon completion of training this network output is provided as plant input,

$$u_{NN}(k) = y_{2i}(k|\theta_c). \tag{7}$$

The augmentation to the plant input corresponding to that of the controller input is given by

$$u(k) = u_{NN}(k) - u(k-1) + u_{eq}(k) \tag{8}$$

where $u(k)$ is the current plant input, $u_{NN}(k)$ is the online controller network output and $u_{eq}(k)$ is the equilibrium value of the plant input corresponding to $y_{eq}(k)$ value.

The internal model network provides steady state response of the process under control. This allows the controller to train faster and respond faster to a change in the reference inputs.

3.2. Online Training and Constraints

The logic behind DNN technique is to adapt to changes in the dynamics of the plant at every instant of time and train to minimise the error in achieving the commanded input at every iteration. Iterative training is performed to minimize a cost function using the mini-batchwise Levenberg Marquardt (LM) technique. The goal of training is to obtain the most suitable values of weights (W_{21} and W_{22} in NN2) to attain the desired performance through repetitive iterations. Here not only the mean square error (MSE) between the effective input y_{in}, and the validation output from the internal model network y_m as a function of the controller network weights, but also the instantaneous error between the reference input to the controller and the predicted output from the external identifier model are minimised.

The regressor matrix for the controller is calculated before initiation of the training at every time step. The regressor for the controller, as given in (3) is equated to the effective input vector. Initial weights of the controller networks are chosen close to their equilibrium values.

The model network regressor is updated at every iteration with the adapted controller network outputs. The fixed weights for the internal model network are obtained from offline training with different sets of input-output plant data.

The MSE as a function of the controller network weights θ_c, to be minimized is given by

$$MSE(\theta_c, k) = \frac{1}{N}\sum_{i=1}^{N}(y_{in}(k) - y_m)^2 \tag{9}$$

where N is the size of the mini-batch of data. Instantaneous error of the overall adaptive system is calculated from the controller reference input and the multi-network identifier model output,

$$E_{inst}(\theta_c, k) = (y_{ref}(k) - \hat{y}(k)) \tag{10}$$

where, $\hat{y}(k)$ is prediction from the multi-network model. The instantaneous error function initiates training in the controller network when the plant deviates from the reference input. It is to be noted that the output of the internal model y_m changes every iteration during training of the controller, whereas $y(k)$ remains constant during training. The PI is given by;

$$V_N(\theta_c, k) = MSE\ (\theta_c, k) + a * Penalty * E_{inst}(\theta_c, k). \tag{11}$$

The network training is initiated if all the conditions in the stopping criteria are not satisfied as given in Algo 1. A lower limit on the performance index (PI_{th}) is considered as the threshold during training. The value for this threshold is selected empirically to obtain fast adaptations with a desired level of accuracy. The LM training factor λ_{c_n} is updated every iteration such that it is maintained within a bound.

The updated value of θ for LM method is given by

$$\theta_{n+1} = \theta_n - \mathbf{R}_n^{-1}\mathbf{G}_n \tag{12}$$

where θ_n is the weights vector at the n^{th} iteration, $\mathbf{R}_n$ is a matrix responsible for the search direction and $\mathbf{G}_n$ is the gradient of the performance index with respect to the weights.

$$\begin{aligned} G_n = \nabla V_N(\theta_n) &= -\frac{2}{N}\left[\sum_{k=1}^{N}(y_{in}(k) - y_{in})\frac{\partial y_m}{\partial \theta_n}\right] \\ &\quad - a * Penalty * \frac{\partial \hat{y}(k)}{\partial \theta_n} \\ &= -\Psi(k,\theta_n)e_c(k) - a * Penalty * \frac{\partial \hat{y}(k)}{\partial \theta_n} \end{aligned}$$

where,

$$\Psi(k,\theta_n) = \frac{\partial y_m}{\partial \theta_n} \tag{13}$$

$$e_c(k) = y_{in}(k) - y_m \tag{14}$$

and $\psi(k, \theta_n)$ is the derivative of the predicted output from the internal network model with respect to weights θ_n. The multi-network model prediction output ($\hat{y}(k)$), does not depend on the controller weights directly. However, changes in outputs of the controller influenced by change in weights has an effect on the model response. But during the controller training $\hat{y}(k)$ remains constant. Hence, the PI gradient function can be written as

$$\mathbf{G}_n = -\Psi(k,\theta_n)e_c(k). \tag{15}$$

The search direction is given by $\mathbf{R}_n^{-1}\mathbf{G}_n$ which is used to update the weights. The $\mathbf{R}_n$ function has second order derivative terms accounting to the Hessian matrix which fastens the training the process. For the LM method $\mathbf{R}_n$ is given by

$$\begin{aligned} \mathbf{R}_n &\cong \left(\psi(k,\theta_n)\ \cdot\ \psi^T(k,\theta_n) + \lambda_{c_n}\mathbf{I}\right) \\ \mathbf{R}_n &\cong \left[\frac{\partial y_m}{\partial \theta_n}\right]\left[\frac{\partial y_m}{\partial \theta_n}\right]^T + \lambda_{c_n}\mathbf{I} \end{aligned} \tag{16}$$

where, λ_{c_n} is adapted during training to alter the step size and hence provides a reduction of the cost function.

The two error functions considered in (9) and (10) have different characteristics. The MSE function indicates error in the inner loop of the controller with respect to the internal model. The instantaneous error indicates the overall discrepancy in the adaptive system through the outer loop. It provides the difference between the online identifier model prediction and the commanded reference input. The instantaneous error is checked at every sample period and is retained for the duration of the training

cycle. With the two error functions in the performance index of training, overall performance of the dual network controller is enhanced.

4. EXPERIMENTS AND RESULTS

Controllers for the Telemaster UAV are designed in numerical simulations, validated on real-time hardware in loop (HIL) simulations and verified by flight implementations. The results from numerical simulations and the flight test results for one of the controllers is presented in this section 4.1. The flight test results are provided in 4.2.

4.1. Numerical Simulation Results

One of the outputs of interest for the UAV with elevator deflection is the angle of attack. A typical set of angle of attack measurements obtained from Telemaster UAV for pilot provided elevator deflections is shown in figure 4. A varied collection of such experimental flight data is used to obtain the neural network model of the angle of attack subsystem.

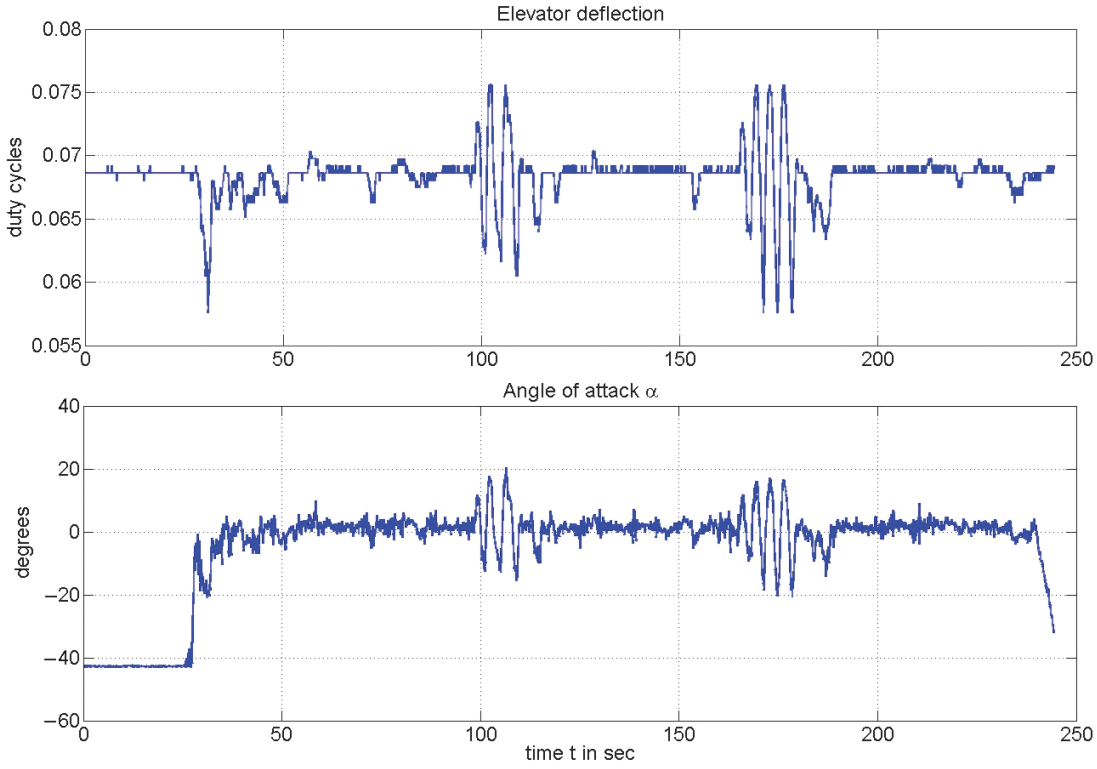

Figure 4. Flight data displays angle of attack dynamics for variation in elevator input used for UAV plant modelling

A DNN based angle of attack autopilot is designed for the Telemaster UAV with the aid of the above NN model as shown in figure 5. To test the controller in simulation for robustness, the sensor noise and effects of gust are included. The sensor noise is a band limited white noise, whereas the vertical gusts are included in the form of small bursts at discrete times. A typical result for tracking a given angle of attack in the presence of these two disturbances is shown in figure 6. It can be seen that controller maintains the plant output in the neighbourhood of the reference input even under the influence of these two disturbances. The sensor noise component and the gust are also shown in the figure. Upon completion of training the DNN controller is validated against the plant model and receives further predictions.

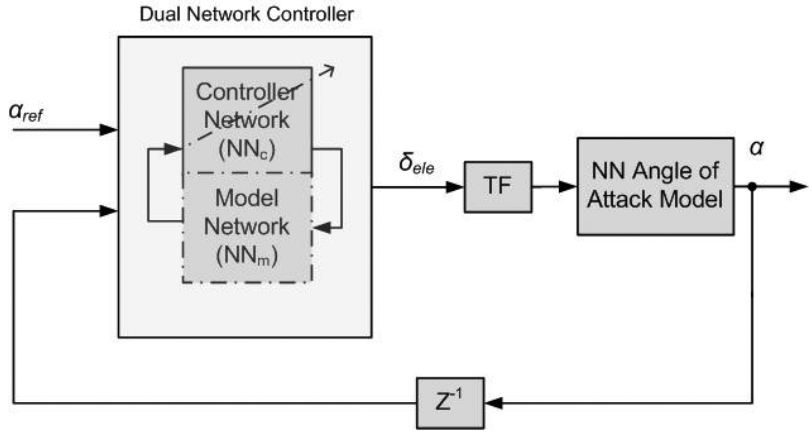

Figure 5. Adaptive DNN based angle of attack autopilot for Telemaster UAV

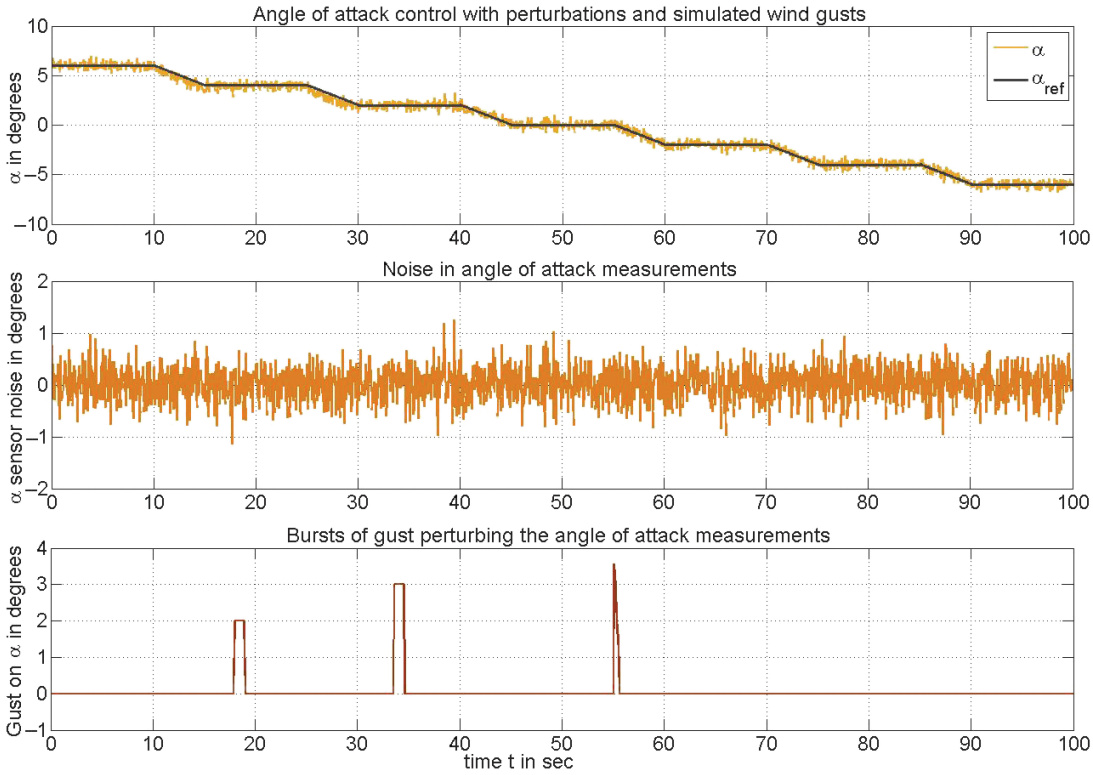

Figure 6. Numerical simulation of α control for Telemaster UAV influenced by perturbations in sensor measurements and by wind gusts

4.2. Flight Test Results

The implementation of controllers for aircraft is a venturous process due to risks involved and the influence of various external conditions. Prior to the flight of a UAV, rigorous ground tests including the HIL tests are performed to ensure safety of platform during implementation. A safe flying field and an experienced pilot is essential to avoid damage to the platform or personnel in case of erratic behaviour of the UAV due to autonomous control. The UAVs are flown by the pilot along approximate

paths and the control is handed over to the computer under level flight conditions. During pilot control mode, controller and identifier model are continuously trained.

The angle of attack controller for the Telemaster UAV is flight tested for different commanded reference inputs. Figure 7 shows the UAV response for a constant reference input for angle of attack (-2° with respect to the equilibrium value). The autopilot is turned on for durations in time as indicated (80-100 sec and 200-240 sec). The throttle variations provided by the pilot during flight is also shown. It can be seen that each time the controller is turned on, the desired angle of attack value is attained and is maintained very close to the commanded value in spite of different throttle settings provided as input by the pilot.

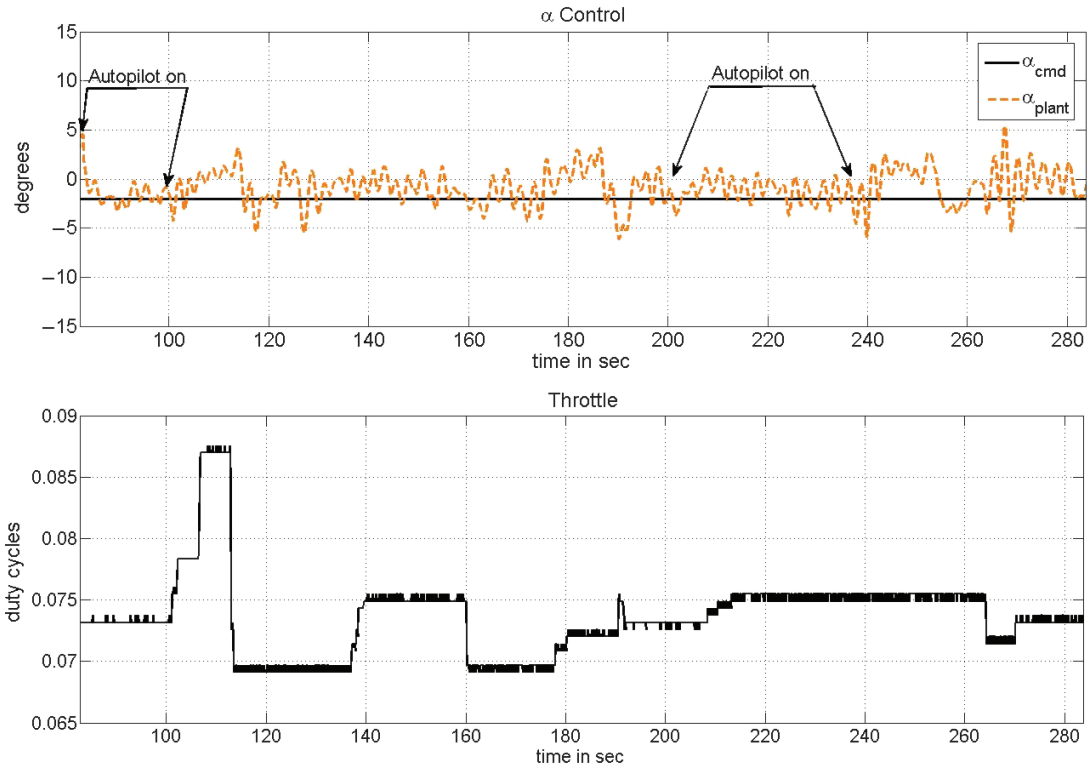

Figure 7. Flight implementation result of the angle of attack controller based on the DNN technique for the Telemaster UAV with constant reference input

The result from flight test for a sinusoidal reference input with the same controller is shown in figure 8. The internal model output (α_{mdl}) and the plant output (α) are also plotted with the commanded reference input (α_{cmd}). The hand-over take-over signal is also shown in the subplot to indicate when the computer control is turned on. Here, 1 indicates computer control turned on whereas 0 indicates the pilot control. It can be seen that the UAV output, even though noisy, follows a sinusoidal trajectory. The noise in angle of attack measurements is mainly due to the high sensitivity of the measuring vane.

A low pass software filter is designed to reduce noise in the measurements from the angle of attack sensor. The raw data, the filtered signal and the internal model predictions during another flight test are shown in figure 9. It is observed that the internal model predictions of angle of attack from flight matches the measured output closely.

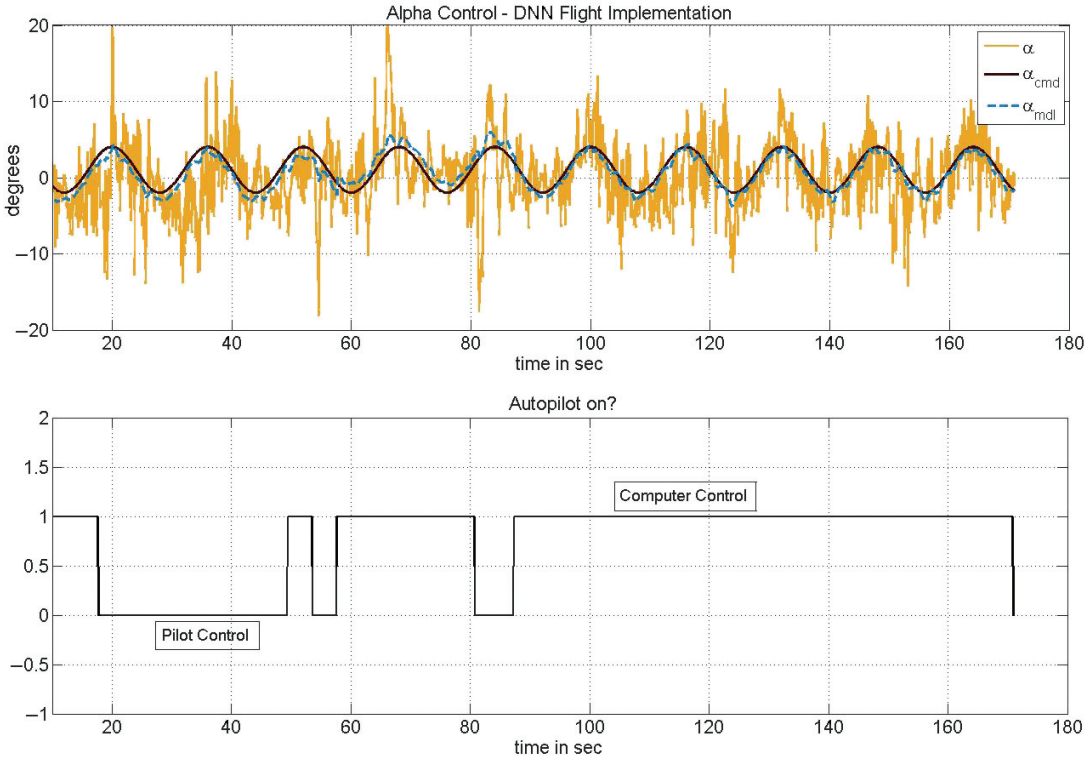

Figure 8. Flight implementation result of the angle of attack controller based on the DNN technique for the Telemaster UAV with varying reference input

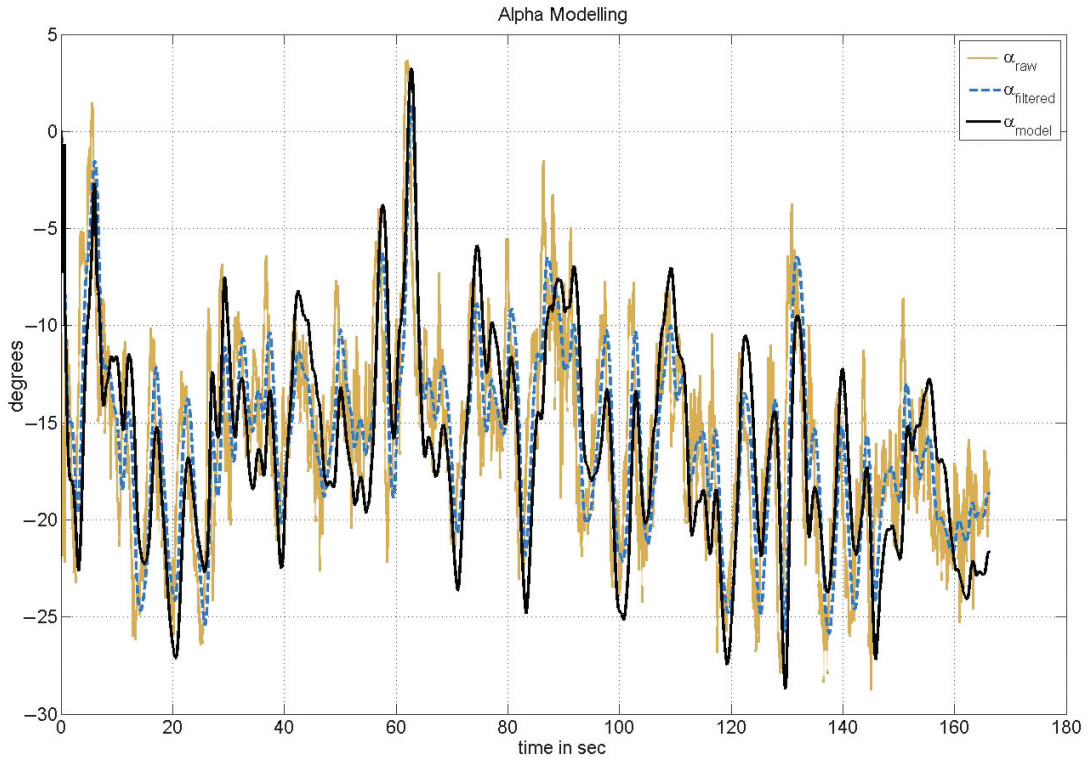

Figure 9. Flight validation results of the internal model in DNN controller for Telemaster UAV angle of attack system

5. CONCLUSION

A novel indirect adaptive control technique based on neural network for the Telemaster UAV is presented in this paper. The DNN provides faster tracking of the commanded reference with better accuracy. The controller is designed in numerical simulations and upon validation in real-time it is flight implemented for verification. Results from the numerical simulations show the noise tolerance capability of the DNN control technique. It can be seen from the flight test results that the DNN controller allows the desired command trajectory to be tracked very closely.

6. REFERENCES

[1] L. Doitsidis, K.P. Valavanis, N.C. Tsourveloudis, and M. Kontitsis. A framework for fuzzy logic based UAV navigation and control. *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, 4:4041-4046 Vol.4, 26-May 1, 2004.

[2] I. Schiller and J.S. Draper. Mission adaptable autonomous vehicles. In *IEEE Conference on Neural Networks for Ocean Engineering*, pages 143-150, Washington D.C, 1991. IEEE.

[3] F. Borrelli, T Keviczky, and G.J. Balas. Collision-free UAV formation flight using decentralized optimization and invariant sets. In *IEEE Conference on Decision and Control*. IEEE, 2004.

[4] S. Howitt, D. Richards, and A. White. Supervisory control of autonomous UAVs in networked environments. In *AIAA Infotech@Aerospace*, Arlington, Virginia, 2005. AIAA.

[5] Anthony J. Calise, Eric N. Johnson, Matthew D. Johnson, and J. Eric Corban. Applications of adaptive neural-network control to unmanned aerial vehicles. In *AIAA/ICAS International Air and Space Symposium and Exposition: The Next 100 Years*, Dayton, OH, 2003. AIAA.

[6] Li-Xin Wang. Design and analysis of fuzzy identifiers of nonlinear dynamic systems. *IEEE Transactions on Automatic Control*, 40(1), 1995.

[7] Shaaban Salman, Vishwas Puttige, and Sreenatha Anavatti. Real-time validation and comparison of fuzzy identification and state-space identification for a UAV platform. In *IEEE International Conference on Control Applications*, pages 2138-2143, Munich, 2006.

[8] R. Pintelon and J. Schoukens. *System Identification : A Frequency Domain Approach*. Wiley-IEEE Press, 1st edition, 2001.

[9] Kumpati S. Narendra and Kannan Parthasarathy. Identification and control of dynamical systems using neural networks. *IEEE transaction on Neural Networks*, 1(1), 1990.

[10] Kumpati S. Narendra and Anuradha M. Annaswamy. *Stable adaptive systems*. Prentice-Hall, Inc, Upper Saddle River, NJ, USA, 1989.

[11] P.J. Antsaklis. Defining intelligent control, report of the task force on intelligent control. *IEEE Control Systems Magazine*, 1994.

[12] Vishwas Puttige and Sreenatha Anavatti. Real-time system identification techniques based on neural networks for a low-cost UAV. *Journal of Engineering, Computing and Architecture*, 2, 2008.

[13] Lennart Ljung. *System Identification-Theory for the User*. Prentice Hall, 2nd edition, 1999.