การประชุมวิชาการเครือข่ายวิศวกรรมเครื่องกลแห่งประเทศไทยครั้งที่ 21
17-19 ตุลาคม 2550 จังหวัดชลบุรี

Development and Control of 6-DOF Fully Autonomous Flying Robot

Sukon Puntunan[1], and Manukid Parnichkun[2]

[1] Mechanical Engineering Department, Engineering Division, Royal Thai Air Force Academy,
Bangkok, 10220, Thailand
Tel: 0-2534-3674, E-mail: sukon@aerotronix.net
[2] School of Engineering and Technology, Asian Institute of Technology,
Pathumthani 12120, Thailand
Tel: 0-2524-5229, E-mail: manukid@ait.ac.th

Abstract

Control of 6-DOF fully autonomous helicopter-type flying robot is very difficult because of the non-linear unstable nature of the flying robot. Many researchers in this field verified their control algorithms only on simulation. There are very few successful experiments on fully control of the flying robot. In order to make the robot fly autonomously, attitude and position controls are required. In this paper, neuro-fuzzy control (NFC) is applied to control roll, pitch and yaw of the flying robot, while hybrid adaptive neuro-fuzzy model reference control (Hybrid-ANFMRC) is proposed to control the robot position. The attitude controllers are trained offline to reduce roll, pitch and yaw errors. The position control learns online to track a velocity reference model to get short response time, small oscillation, and no steady state error. Parameter robustness of the proposed control algorithm is addressed by testing in the experiments under various ranges of control gains. The experimental results confirm the feasible performance of the proposed control algorithm for the flying robot.

.

Keywords: flying robot, adaptive control, hybrid control, model reference control.

1. Introduction

It is expected that flying robots will be used in many applications in the future, particularly in various hazardous areas. For examples, the robot can hover and transmit top view video image of hostage situations, enemy locations, or areas contaminated by toxic chemicals or biological agents for appropriate further actions. They can be used in geological survey and map generation purposes with less expense compared with using real airplanes or helicopters. In order to make use of flying robots in these various applications effectively, the robots are to have the ability to fly automatically. A flying robot is modified from X-Cell 60 radio-controlled helicopter. It is developed to support autonomous flight control covering wide-mode missions of operation from taking off, hovering, flying in forward, backward, leftward, rightward, upward, and downward directions to specified locations, until landing. The flying robot has six degrees of freedom in its motion. The problem of this kind of the flying robot is its inherent unstability. The robot dynamics is nonlinear and varies with environment. The system is always disturbed by noise and disturbance; wind turbulence, ground effect, for instances. As the result, control of the flying robot is very difficult and very challenging at the same time.

Currently, there are some researches focusing on control of autonomous flying robots by different control techniques [1]. However, there are very few successful experiments on fully control of this kind of robot. The researches are separated into 2 directions. The first direction applies model-based approach. The other direction is based on model-free approach. Model-based approach can not be implemented efficiently in real world, because of the difficulty in obtaining an acceptable and accurate dynamics model of the flying robot. As the system increases its complexity, complete and accurate identification of the robot mathematical model becomes difficult. Consequently, the applicable models are just only the approximations. To overcome the problem, some researches apply model-free technique. Neural network and fuzzy logic are widely used. Wyeth, et.al, applied flight data to train a neural network controller offline [2]. They obtained direct mapping of sensor inputs to actuator outputs. The control used a “cause” and “effect” approach. It was found from the experimental results that they failed to control the robot by this approach. Montgomery, et.al developed a “teaching by showing” method to train a fuzzy-neural controller [3]. The controller was developed and tuned by using training data gathered while the operator manually controlled the flying robot. The method was successfully applied in simulation but failed to control the flying robot in real operation. Sugeno could successfully control the flying robot by applying a fuzzy logic control [4]. He used the knowledge from experienced pilot to design his fuzzy logic controller. He also compared the performance of fuzzy logic control with conventional linear control under a windy environment. Fuzzy controller showed its robustness against wind turbulent better than in linear

controller. However, the design process took time and required experimental skill from the expert pilot.

Drawback of neural network is the difficulty in re-tuning the network after the training process was once accomplished. Likewise drawback of fuzzy logic control is the requirement of knowledge about the plant under control. Parameters of fuzzy logic controller are determined manually. Neuro-fuzzy control takes the advantages from fuzzy logic control and neural network. Learning ability of neural network and tuning ability of fuzzy logic control are integrated in the neuro-fuzzy control.

In this paper, a model free approach, neuro-fuzzy control, is applied to control roll, pitch and yaw of the flying robot. The neuro-fuzzy controllers are trained offline from the flight data. Hybrid adaptive neuro-fuzzy model reference control (Hybrid-ANFMRC) is proposed to control position of the flying robot. The position control combines neuro-fuzzy with proportional control. The proportional control acts as the basis control, while the adaptive neuro-fuzzy model reference control learns to track a velocity reference model. The reference model is defined as the function of position error. It can be linear or non linear function of the position error. The position control learns from the flight data without using any expert knowledge. Experiments are undertaken to evaluate performances of the proposed control algorithm. Robustness of the position controls is addressed by testing in experiments under various ranges of the proportional gains.

2. Six-DOF Control of Flying Robot

2.1 Neuro-fuzzy control

Neuro-fuzzy controller is applied to control roll, pitch and yaw of the flying robot. The neuro-fuzzy control is a class of hybrid controls that fuses fuzzy logic with neural network. It combines the advantages of neural network in learning ability, optimization abilities and connectionist structure with the advantages of fuzzy logic control in human-like structure, ease of incorporating expert knowledge [5]. The structure of the neuro-fuzzy attitude control is shown in Figure 1.

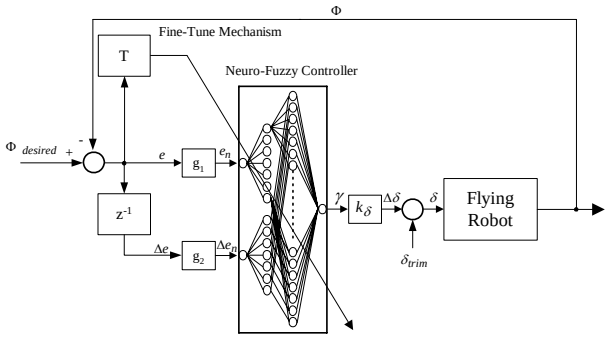

Figure 1. Neuro-fuzzy attitude control

From Figure 1, there are two inputs and one output of the neuro-fuzzy controller. The first input is attitude error, $e(k)$. The second input is change of attitude error, $\Delta e(k)$. The output of the neuro-fuzzy controller is change of actuator command, $\Delta\delta(k)$. The attitude error, $e(k)$ and the change of attitude error, $\Delta e(k)$, are determined as followings.

$$e(k) = \Phi_{desired}(k) - \Phi(k) \quad (1)$$

$$\Delta e(k) = e(k) - e(k-1) \quad (2)$$

where $\Phi_{desired}(k)$ is the desired attitude, and $\Phi(k)$ is the actual attitude of the flying robot.

The input parameters of the neuro-fuzzy controller have to be normalized to fit with the hardware constraints. The normalized attitude error, $e_n(k)$ and the normalized change of attitude error, $\Delta e_n(k)$ are calculated as followings.

$$e_n(k) = g_1(e(k)) \quad (3)$$

$$\Delta e_n(k) = g_2(\Delta e(k)) \quad (4)$$

where $g_1(\bullet)$ and $g_2(\bullet)$ are the normalization functions of the attitude error, $e(k)$ and the change of attitude error, $\Delta e(k)$, respectively.

The normalization functions are defined as followings.

$$\begin{aligned} g_1(e(k)) &= k_e g_{1neg} e(k) && \text{if } e(k) \le 0 \\ &= k_e g_{1pos} e(k) && \text{if } e(k) > 0 \\ g_2(\Delta e(k)) &= k_{\Delta e} g_{2neg} \Delta e(k) && \text{if } \Delta e(k) \le 0 \\ &= k_{\Delta e} g_{2pos} \Delta e(k) && \text{if } \Delta e(k) > 0 \end{aligned} \quad (5)$$

where k_e and $k_{\Delta e}$ are attitude error and change of attitude error gains, respectively. The constant values; g_{1neg}, g_{1pos}, g_{2neg} and g_{2pos}, are the normalization factors for each input parameters.

The output of the neuro-fuzzy controller is the result of mapping from the normalized attitude error, $e_n(k)$ and the normalized change of attitude error, $\Delta e_n(k)$ to the output, $\gamma(k)$. The change of actuator command, $\Delta\delta(k)$, is obtained by multiplying the output, $\gamma(k)$ with the output gain k_δ.

$$\Delta\delta(k) = k_\delta \gamma(k) \quad (6)$$

The actuator command, $\delta(k)$, is the summation of the change of actuator command, $\Delta\delta(k)$ with the control trim, δ_{trim}.

$$\delta(k) = \delta_{trim} + \Delta\delta(k) \quad (7)$$

Performance of the neuro-fuzzy control is affected by changing the attitude error gain, k_e, the change of attitude error gain, $k_{\Delta e}$ and the output gain k_δ.

In this paper, the neuro-fuzzy controller is trained to reduce the attitude errors. The flight data is used as the training set. The offline learning of the neuro-fuzzy controller applies back propagation algorithm. Figure 2 shows symmetrical triangle membership functions of the neuro-fuzzy controller with fuzzy singleton rule.

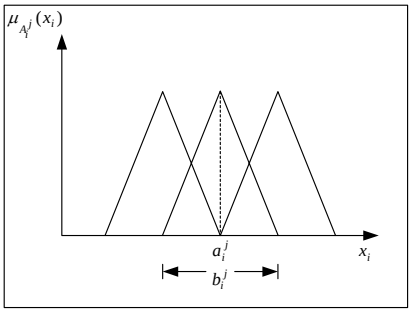

Figure 2. Symmetrical triangle membership function

The symmetrical triangle membership function is expressed by the following equation.

$$\mu_{A_i^j}(x_i) = 1 - \frac{2\left|x_i - a_i^j\right|}{b_i^j},\ i = 1,2,...n\ \ j = 1,2,...,m \quad (8)$$

where x_i is the input value, a_i^j is the center of triangle and b_i^j is the width of triangle. The fuzzy rules, also called fuzzy singleton, are in the following form [5].

Rule j: If x_i *is* A_1^j *and* x_2 *is* A_2^j and … and x_n is A_n^j *then* γ *is* w_j.

where A_i^j is a linguistic term with the membership function, $\mu_{A_i^j}^i(x_i)$, w_j is a real number of weight in the neural network part. By the singleton rule, control output, $\gamma(k)$ from the neuro-fuzzy controller is calculated by the following equation.

$$\gamma(k) = \frac{\sum_{j=1}^{m} \mu_j(k) w_j(k)}{\sum_{j=1}^{m} \mu_j(k)} \quad (9)$$

where

$$\mu_j = \mu_{A_j^1}(x_i)\mu_{A_2^1}(x_i) \ldots \quad \mu_{A_n^1}(x_i) \quad (10)$$

The weights of the neuro-fuzzy controller are modified by steepest gradient method to minimize a cost function. The cost function is defined as half of the square of the difference between the command attitude and the actual attitude.

$$E = \frac{1}{2}(\Phi_{desired} - \Phi)^2 \quad (11)$$

The weights of the neuro-fuzzy controller are modified by steepest gradient method as following.

$$w_j(k+1) = w_j(k) - \eta \frac{\partial E}{\partial w_j} \quad (12)$$

where $\eta \geq 0$ is the learning rate.

By applying chain rule, the adjusted weights can be determined from

$$\frac{\partial E}{\partial w_j} = \frac{\mu_j(k)}{\sum_{j=1}^{m} \mu_j(k)} (\Phi_{desired}(k) - \Phi(k)) \quad (13)$$

2.2 Hybrid adaptive neuro-fuzzy model reference control

Hybrid-ANFMRC is proposed to control position of the flying robot. The control is a hybrid of a proportional control with an adaptive neuro-fuzzy model reference control. In the proposed control algorithm, the proportional controller generates the output proportional to position error. The adaptive neuro-fuzzy model reference controller generates the output by learning to track a velocity reference model. Structure of the Hybrid-ANFMRC is shown in Figure 3.

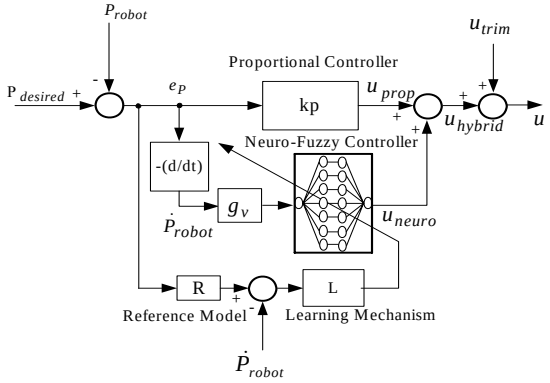

Figure 3. Structure of Hybrid-ANFMRC

From the figure, the Hybrid-ANFMRC consists of a proportional controller and a neuro-fuzzy controller. The position error $e_p(k)$ is the difference between the desired position, $P_{desired}(k)$ and the actual robot position, $P_{robot}(k)$.

$$e_p(k) = P_{desired}(k) - P_{robot}(k) \quad (14)$$

The proportional controller is used to generate the control output, $u_{prop}(k)$ proportional to the position error, $e_p(k)$. The output of the proportional controller is calculated as following.

$$u_{prop}(k) = u_{prop}(k-1) + k_P(e_p(k) - e_p(k-1)) \quad (15)$$

where k_P is the proportional gain.

Main function of the proportional controller is to reduce the overall error. However, the control output of the proportional controller becomes smaller when the error approaches zero. As the result, steady-state error always exists by a sole proportional controller. The adaptive neuro-fuzzy model reference controller is used to reduce steady state error remaining from the proportional controller, while still attenuates the oscillation. The output of the proportional controller, $u_{prop}(k)$, is summed with the output of the adaptive neuro-fuzzy model reference controller, $u_{neuro}(k)$, becoming the hybrid control output, $u_{hybrid}(k)$.

$$u_{hybrid}(k) = u_{prop}(k) + u_{neuro}(k) \quad (16)$$

The hybrid output, $u_{hybrid}(k)$, is then summed with the control trim, u_{trim}, to generate the control output, $u(k)$.

$$u(k) = u_{hybrid}(k) + u_{trim} \quad (17)$$

The input of the adaptive neuro-fuzzy model reference controller is velocity of the robot. The robot velocity is then normalized to the normalized velocity, $\dot{P}_{n,robot}(k)$. The normalized velocity is obtained by multiplying the velocity, $\dot{P}_{robot}(k)$ with a scaling factor, g_v.

$$\dot{P}_{n,robot}(k) = g_v \dot{P}_{robot}(k) \quad (18)$$

The output of the adaptive neuro-fuzzy model reference controller is the mapping result from the velocity, $\dot{P}_{robot}(k)$ to the adaptive output, $u_{neuro}(k)$, which is calculated by the weight average method. With the normalized inputs, $\dot{P}_{n,robot}(k)$, the output, $u_{neuro}(k)$, is determined from equation (19).

$$u_{neuro}(k) = \frac{\sum_{j=1}^{m} \mu_j(k) w_j(k)}{\sum_{j=1}^{m} \mu_j(k)} \quad (19)$$

where

$$\mu_j(k) = A_1^i(\dot{P}_{n,robot}(k)) \quad (20)$$

and $A_1^i(\dot{P}_{robot}(k))$ is a triangle membership function.

The adaptive neuro-fuzzy model reference controller learns to track the desired velocity reference model, $r(k)$, which is defined as the function of the position error as following.

$$r(k) = f(P_{robot}(k)) \quad (21)$$

where $f(\bullet)$ is a linear or nonlinear function.

Weights of the adaptive neuro-fuzzy model reference controller are modified with steepest gradient method to minimize a cost function. The cost function is defined as half of the square of the difference between the velocity reference model and the actual velocity.

$$E = \frac{1}{2}(r(k) - \dot{P}_{robot}(k))^2 \quad (22)$$

The weights are modified as following.

$$w_j(k+1) = w_j(k) - \eta \frac{\partial E}{\partial w_j} \quad (23)$$

where $\eta \geq 0$ is the learning rate.

By applying chain rule, the adjusted weights can be determined from

$$\frac{\partial E}{\partial w_j} = \frac{\mu_j(k)}{\sum_{j=1}^{m} \mu_j(k)} (r(k) - \dot{P}_{robot}(k)) \quad (24)$$

3. Experimental Results

3.1 Experiment of neuro-fuzzy attitude control

Only yaw control is considered here. The principle of row and pitch control is similar to yaw control and omitted in the explanation. The training data are obtained by applying series of input signal to the uncontrolled yaw axis, while maintaining the flying robot stable in the other axis. The signal causes the robot to oscillate about z-axis.

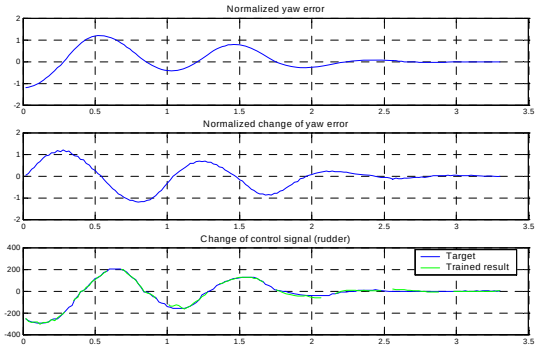

Figure 4. Training data for neuro-fuzzy yaw control

Figure 4 shows the training data and the offline training result of the neuro-fuzzy yaw control. In the control, there are 7 membership functions for each input. Each linguistic value is expressed by its mnemonic; for example, *NB* for "negative big", *NM* for "negative medium", *NS* for "negative small", *ZO* for "zero", and likewise for the positive (*P*) mnemonic.

In the experiment, the learning rate is selected at 0.02. The design parameters are shown in Table 1.

Table 1. Neuro-fuzzy yaw control parameters (* indicates the values after fine tuning)

g_1		g_2		k		
g_{1neg}	g_{1pos}	g_{2neg}	g_{2pos}	k_e	$k_{\Delta e}$	k_δ
				1.0	1.0	1.0
0.0529	0.0684	0.7619	0.4706	*1.0	*2.1	*1.39

Offline training helps to initialize the valid control parameters that can stabilize the yaw control of the flying robot. Without offline training, parameters of the neuro-fuzzy controller might fall in the unstable region. After offline training, manual coarse and automatic fine tunings are also required practically. Automatic online fine tuning is applied to reduce steady state error. Without manual coarse tuning, the parameters obtained by sole fine tuning might not be converged. The experimental result is shown in Figure 5. At the beginning, the gains of the neuro-fuzzy controller are manually tuned until an acceptable control performance is achieved. Then the neuro-fuzzy controller is online fine-tuned to reduce the steady state error.

The experimental result of yaw control under step inputs is shown in Figure 6. From the figure, the yaw response could follow the step inputs with small oscillation magnitude and steady state error.

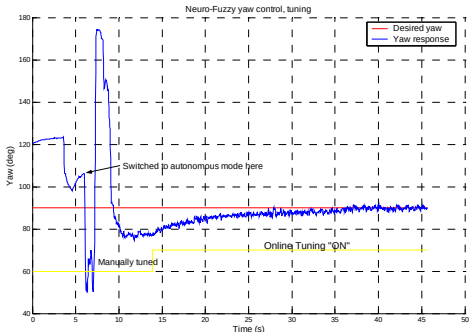

Figure 5. Tuning result of neuro-fuzzy yaw control

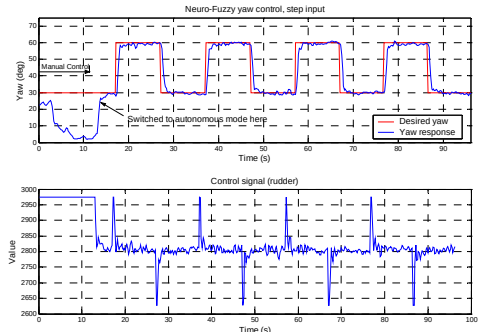

Figure 6. Step input response of neuro-fuzzy yaw control

3.2 Experiment of position control with Hybrid-ANFMRC

The outputs of the lateral, longitudinal position and altitude control are the desired roll, the desired pitch and the change of collective commands, respectively. The roll and pitch control are designed similar to the yaw control described in section 2.1. In the experiment, the proportional gains of the lateral and longitudinal position control are both 8.0. The proportional gain of the altitude control is 30.0. The lateral and longitudinal position commands are both 0 meter. The altitude command is 13.0 meter. The learning rates of the lateral, longitudinal position and altitude control are all 0.4. The velocity reference model is defined as a linear function of the position error. The robot velocity is normalized within the range between –1.2 and 1.2. There are 7 elements of the weight for each control axis, which are initialized to zero at the beginning.

In Figures 7 and 8, only the proportional control is applied at the beginning. The learning process of the Hybrid-ANFMRC is then activated. During learning, the controller adapts the control parameters and learns to control position of the robot. Finally, the robot can track the desired position with no steady state error.

Figure 9 shows the result of altitude control. The control is switched between manual pilot control and Hybrid-ANFMRC computer control. Every time the pilot takes control the robot, the learning process is stopped. The control performance is better with more learning which can be seen from the figure that the oscillation magnitude is small during every learning. The control learns to control the altitude of the robot effectively.

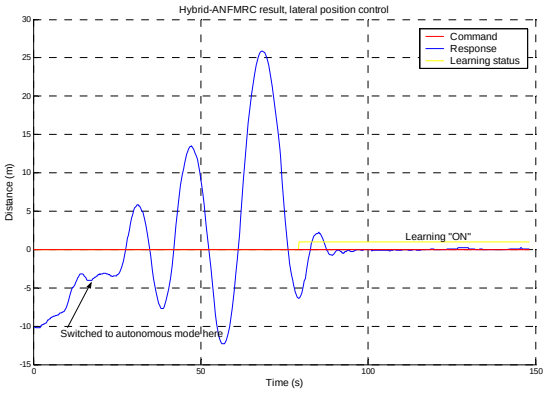

Figure 7. Lateral position control by Hybrid-ANFMRC,

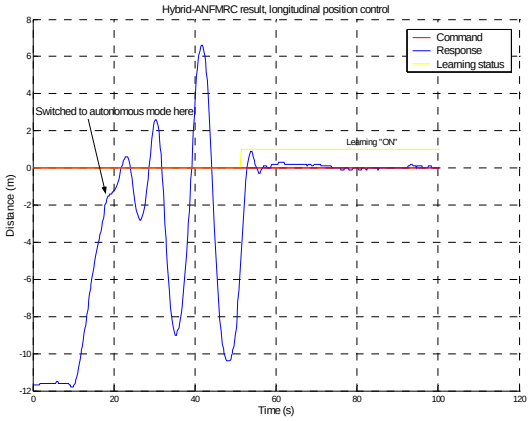

Figure 8. Longitudinal position control by Hybrid-ANFMRC

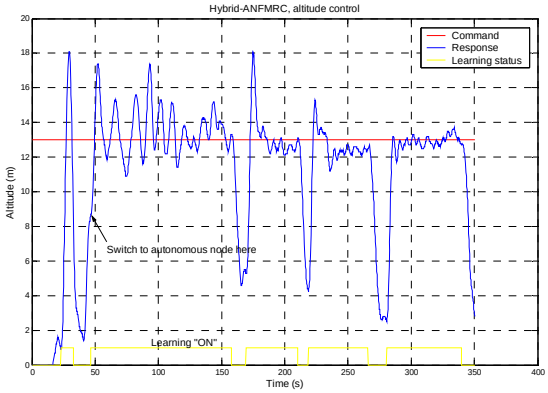

Figure 9. Altitude control by Hybrid-ANFMRC

To evaluate robust performance of the proposed control algorithm, the longitudinal position control is studied. The proportional gain of the longitudinal position control is varied. Figures 10-12 show responses of Hybrid-ANFMRC at the proportional gains of 2.0, 4.0 and 8.0 respectively.

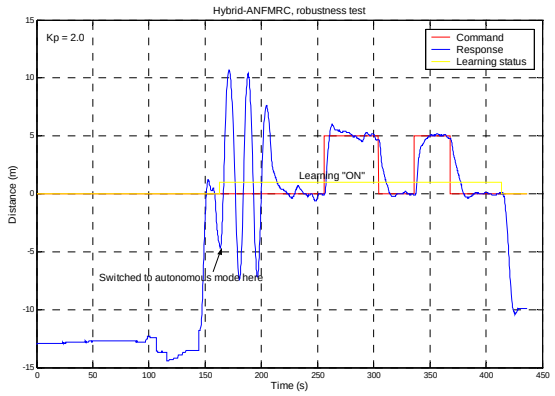

Figure 10. Response of Hybrid-ANFMRC with k_P = 2.0

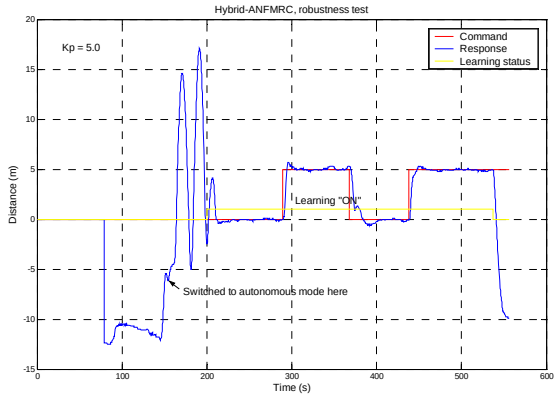

Figure 11. Response of Hybrid-ANFMRC with k_P = 4.0

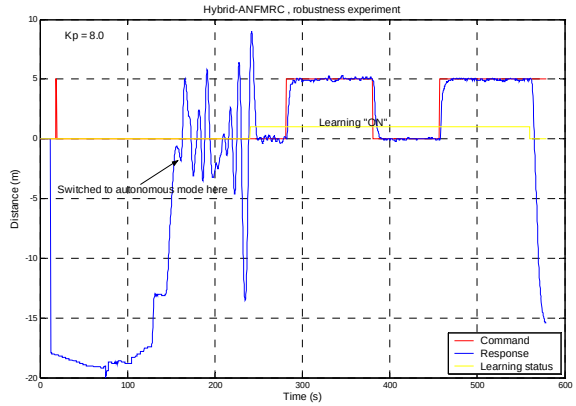

Figure 12. Response of Hybrid-ANFMRC with k_P = 8.0

5. Conclusion

In this paper, neuro-fuzzy control and the Hybrid-ANFMRC were proposed to control fully autonomous flying robot. The neuro-fuzzy controller was applied to control the roll, pitch and yaw of the flying robot. The controller was trained using the flight data and fine tuned to achieve the desired response. The Hybrid-ANFMRC was proposed to control the lateral, longitudinal positions, and altitude of the flying robot. The control performance of the Hybrid-ANFMRC was evaluated from many experiments. The desired performance could be achieved by the proposed control algorithm even with the variation of control gain. The experiments showed that the proposed control algorithms were able to successfully control the flying robot.

Acknowledgments

This research is financially supported by Thailand Research Fund.

References

[1] S. Saripalli, J. M. Roberts, P. I. Corke, G. Buskey. A Tale of Two Helicopters. Robotics Research Lab, University of Southern California, 2002.

[2] Gordon Wyeth, Gregg Buskey, Jonathan Roberts. Flight Control Using an Artificial Neural Network. University of Queensland, 2002.

[3] Jame F. Montgomery and George A. Bekey. Learning Helicopter Control Through "Teaching by Showing".

[4] M. Sugeno. Development of an Intelligent Unmanned Helicopter. Fuzzy Modeling and Control, Selected Works of M. Sugeno. P.13-43.

[5] C. Teng, C.S. George. Neural Fuzzy Systems. Prentice Hall Inc; 1999.