TOWARDS A NEURAL CONTROL ARTIFICIAL PILOT

Qiuxia Liang, Patrick Ehlert, and Leon Rothkrantz
Department of Media and Knowledge Engineering
Faculty of Electrical Engineering, Mathematics, and Computer Science
Delft University of Technology
Mekelweg 4, 2628 CD Delft, the Netherlands
E-mail: {Q.Liang, P.A.M.Ehlert, L.J.M.Rothkrantz} @ ewi.tudelft.nl

ABSTRACT

In this paper we will address our efforts to design a neural control system that can control a simulated aircraft, which ultimately should lead to a realistic artificial pilot. The system we designed consists of a flight plan module, the actual neural controller module, and a graphic user interface. The goal of the flight plan module is to manage the global control of the whole system. For the neural controller we chose to use a Forward Modeling and Inverse Controller. The Jordan Network was used to construct the pre-trained identifier and the online learning controller. Our first experiments showed that improvements were necessary to make the aircraft fly more smoothly. Although the aircraft "wobbles" a bit at the start of a new flight procedure, the controller is able to adapt to changing circumstances during flight.

1. INTRODUCTION

The Intelligent Cockpit Environment (ICE) project is a project of the Department of Media and Knowledge Engineering of Delft University of Technology. Originally, the main purpose of this project was to investigate techniques that can be used to create a situation-aware crew assistance system [Ehlert and Rothkrantz 2003][1]. Basically, a crew assistance system functions as an electronic co-pilot looking over the shoulder of the crew of an aircraft and helping out when necessary.

A secondary objective of the ICE project has been to create a realistic artificial pilot, also called flightbot, that can be used for simulations. Such a pilot can increase the realism of flight simulators and enhance the training of real pilots as well as study the different ways different pilots are flying.

In this paper we will address our efforts to design a neural control system that can control a simulated aircraft. The control structure we used for this application is called Feed Forward and Inverse Control and consists of two neural networks. One is a pre-trained network and the other is an online learning network for inverse control.

[1] More information on the ICE project can also be found via http://www.kbs.twi.tudelft.nl/Research/Projects/ICE/

2. RELATED WORK

2.1. Simulating pilots

There are several projects that deal with the construction of artificial pilots. Here we will shortly mention two examples found in literature; the TacAir-Soar and TAC BRAWLER projects.

TacAir-Soar is a rule-based system that generates believable human-like pilot behaviour for fixed-wing aircraft in large-scale distributed military simulations [Jones et al 1999]. Each instance of TacAir-Soar is responsible for controlling one aircraft and consists of a Soar architecture [Laird, Newel and Rosenbloom 1987] linked to the ModSAF simulator [Ceranowicz 1994]. The advantage of using Soar is that the reasoning and decision-making of the system is similar to the way humans are generally believed to reason.

TAC BRAWLER is a simulation tool for air-to-air combat developed by the Linköping University in collaboration with Saab Military Aircraft AB in Sweden [Coradeschi, Karlsson, and Törne 1996]. The system is designed specifically for air-to-air combat experts and allows the experts to specify the behaviour and decision-making of the intelligent pilot agents, without the help of a system expert. The agents in TAC BRAWLER are modelled by decision trees. These trees contain production rules that describe the agent's dynamic task priorities. During one decision cycle, several branches of the tree can be processed in parallel after which all selected actions are evaluated for priority and compatibility.

Both TacAir-Soar and TACBRAWLER try to simulate realistic pilot flight behaviour and both focus primarily on decision-making during flight. However, both systems use a rule-based approach for aircraft control. No project was found that uses a neural control approach.

The possible advantage of neural control over a rule-based approach is that there is no need to specify rules for pilot behaviour. Although rule-based approaches are very suitable to model normative pilot behaviour, it is much more difficult to model the different styles that different pilots use for flying. Secondly, using neural networks automated learning can be used to avoid the difficult process of explicating flight rules and finetuning rules and parameters. Thirdly, neural networks allow automatic adjustment to changing circumstances, such as different weather conditions, different aircraft, and malfunctioning controls.

2.2. Neural networks and flight control

The first Neural Network (NN) controller was developed by Widrow and Smith in 1963. Since then many applications have shown that NNs can be applied successfully to control unknown nonlinear systems. There have been a number of studies that investigated neural networks for flight control, for example [Calise 1996],[Wyeth et al 2000],[Pesonen et al 2004]. However, all neural control studies try to improve (a particular part of) automated flight control and focus mainly on the control of Unmanned Aerial Vehicles (UAVs), helicopters, and missiles. Their goal is simply to create a controller that functions as good as possible. We did not find any studies that investigated neural networks for simulating realistic flight behaviour of real pilots.

3. System Design

3.1. General system scheme

The essential element of a powerful and flexible neural control system is of course the controller itself. However, to create a flightbot we also need some assistant parts, for example a flight-planning system.

Our system has been divided into three parts according to the different tasks and functions which are; the graphic user interface, the flight plan module, and the neural controller module. Figure 1 shows the general system scheme. The user interface sends orders from the user to the flight plan system. The flight plan system will analyse the order to determine whether it is reasonable or not. If it is reasonable, the planning system will create a flight plan, which consists of at least one flight procedure. Then, the planning system will send different data (the desired plant output) to the controller module corresponding to each flight procedure. The controller will produce the necessary control data that will finally be applied to the plant (aircraft).

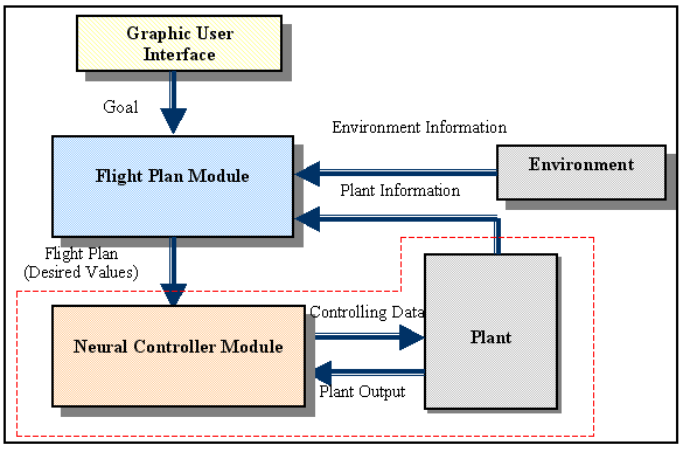

Figure 1: general scheme of the neural flight control autopilot system

3.2. Flight plan module

The goal of the flight plan module is to manage the global control of the whole system. The flight plan module will keep an eye on the flight process and update its flight records to provide the proper plant output data. To be precise the duties of the flight plan module are:

- Analyzing the reasonability of the current goal;
- Deciding on the flight plan;
- Providing the controller with the necessary data corresponding to each part of the flight plan;
- Checking the current flight situation.

Figure 2 shows a scheme of the flight plan system module.

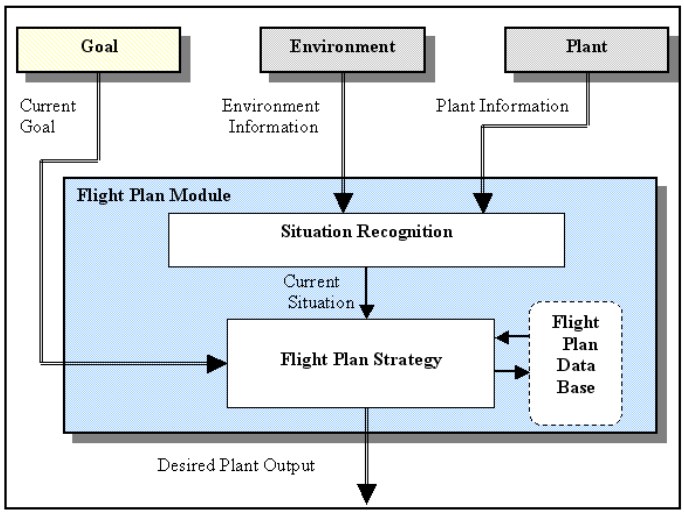

Figure 2: the flight plan system model

The flight plan module choses one of more procedures based on the goal that is currently set by the user as well as on the state of the aircraft and the environment. Figure 3 shows the (simplified) relationship between the goal of the flight plan module and the chosen flight procedures. With "default flying" we mean straight and level flight.

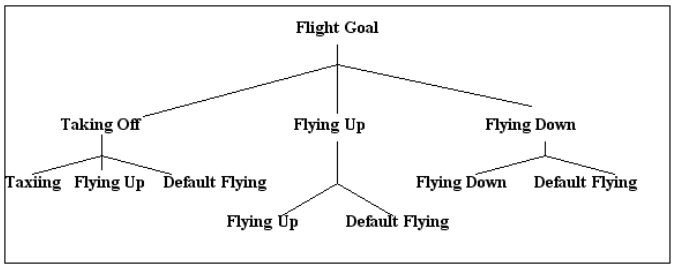

Figure 3: flight strategies

3.3. Neural control module

For the neural control module we have looked into several neural network control topologies. For our purposes we investigated three often-used neural network control topologies:. Direct Inverse Control, Neural Predictive Control and Forward Modeling and Inverse Control.

Topology comparisons

In the Direct Inverse Controller the structure will force the network to represent the inverse of the plant. However, there are drawbacks to this approach. First, if the nonlinear system mapping is not one-to-one then an incorrect inverse can be obtained. Second, inverse plant models are often instable, which may lead to the instability of the whole control-system.

The Neural Predictive Controller consists of four components, a plant to be controlled, a reference model that specifies the desired performance of the plant, a neural network modelling the plant, and an optimisation model used to produce the plant input vector. The object is to have an input vector for which the value of the cost function is lower than a defined value. Then the first element of the plant input for current time will be applied to the plant. The Newton-Rhapson algorithm has been widely used for the optimisation model to determine the best-input vector. The main disadvantages of Neural Predictive Control are that numerical minimization algorithms are usually very time consuming (especially if a minimum of a multivariable function has to be found), what may make them unsuitable for real-time applications. When sampling intervals are small, there may be no time to perform minimum searching between sampling. Additionally, the prediction controller requires an accurately trained neural network model to simulate the plant, since the result of the whole controller system depends on the correct prediction value.

The Forward Modeling and Inverse Control (see Figure 4) has an additional NN plant model, compared to the Direct Inverse Control, which is used in the inverse neural network training process. The error signal is propagated back through the forward model and to the inverse model. However, only the inverse network model is adapted during this procedure.

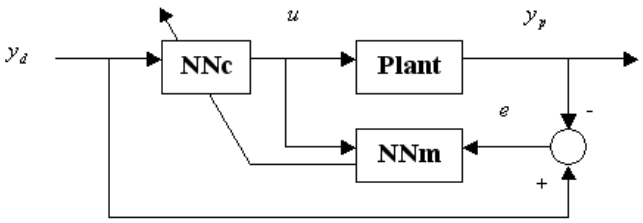

Figure 4: basic Forward Modeling and Inverse Control

The error signal for the training algorithm in this case is the difference between the training signal and the system output (it may also be the difference between the training signal and the forward model output in the case of noisy systems, which is adopted when the real system is not viable). Jordan and Rumelhart [1992] have showed that using the real system output it can produce an exact inverse controller even when the forward model is inexact, which will not happen when the forward model output is used. Another plus is that since the controller neural network gets trained assuming the correct plant input is equal to the backpropagated error from the forward model plus controller output, the training process will be stable.

All things considered, we have chosen the Forward Modeling and Inverse Control, mainly because we want our pilot controller to run in real-time alongside flight simulator software, so we do not have much CPU time available. In addition, as mentioned above Forward Modeling and Inverse Control is better in producing an inverse controller.

Airplane system modeling

Figure 5 shows the representation of the basic airplane model used for our application, which has four inputs (elevator, throttle, rudder and aileron control) and four outputs (airspeed, pitch, heading and bank).

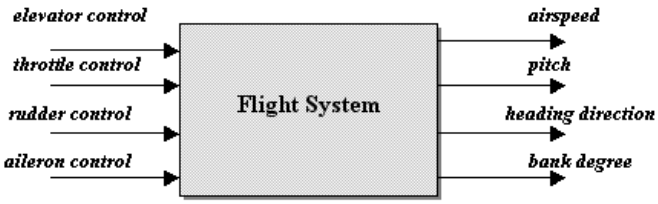

Figure 5: the basic airplane model

The elevator and throttle directly influence the airspeed and pitch of an aircraft, whereas the rudder and aileron directly influence heading and bank. The ailerons control is used to bank the airplane in the direction the pilot wants to turn, and the rudder control is used to keep the nose of the airplane pointing to the direction of turn.

If we only look at the relation between elevator/throttle and airspeed/pitch, we can represent this dynamical system as in Figure 6, which is used for the input and output analysis.

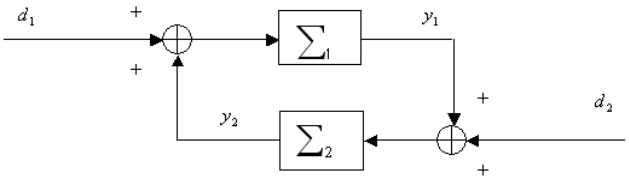

Figure 6: input-output relationship for the elevator/throttle and airspeed/pitch model

This interconnected dynamic system has (d_1, d_2) as input and (y_1, y_2) as output, in which d_1 denotes the elevator input, d_2 denotes the throttle input, y_1 denotes the pitch output and y_2 denotes the airspeed output. In the dynamic sub-system Σ_1, the input $d_1 + y_2$ produces the output y_1, which means the current elevator input and current airspeed value determine the pitch value of the next cycle. In the dynamic sub-system Σ_2, the input $d_2 + y_1$ produces the output y_2, which means the current throttle input and current pitch value determine the airspeed value of the next cycle.

Besides airspeed and pitch, there are also some other parameters influenced by the throttle and the elevator controls, like altitude and vertical speed. Compared with airspeed and pitch, those parameters are the indirect results of the throttle and the elevator controls. For example, if the airplane is in the air and pitches up, then the altitude will increase and the vertical speed will be a positive value. Therefore, we regard the altitude value and the vertical speed value as the references, instead of the parameters that should be used in the system modelling. For example, when the user sets a flight order for the airplane, besides the flight action he (she) will also be asked to set the altitude the airplane should fly to and this value is checked by the system during flight to analyse the situation.

For more details on how throttle and elevator influences airspeed and pitch, or on flying the Cessna aircraft in general, the interested reader is referred to the Microsoft Flight Simulator manual [2002].

4. IMPLEMENTATION

All software was written using Visual C++ 6.0 environment and in C language. For each module we tested the functions seperatly before I we did a full system's test. The neural networks were implemented using a program that is called the Stuttgart Neural Network Simulator (SNNS). SNNS is an open source program, which not only provides the interface to construct the neural network and simulate its running, it also offers a variety of kernel functions for the creation and manipulation of networks that can be combined in the user's own program [Zell et al 1995]. The simulator we used to test our controller is the Microsoft Flight Simulator 2000. The default Cessna 172 aircraft was chosen for all experiments described in this paper.

For simplicity reasons, in our first implementation we only looked at elevator/throttle and airspeed/pitch. We did not implement turning.

We first trained a neural network to model the airplane plant using SNNS. The topology we used to construct the identifier is the Jordan Network. From experiments we found that, due to its simplicity, the Jordan Network (see Figure 7) is better for on-line training than the other network we tried, a Non-linear AutoRegressive Moving Average (NARMA) network (see also [Liang 2004] for more details).

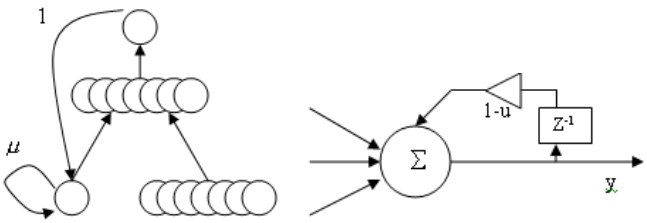

Figure 7: topology of the Jordan Network and the structure of its context PE

The difference between a common neural network and a recurrent neural network, such as the Jordan Network, is that in a recurrent neural network there is a context Processing Element (PE). In the right part of Figure 7 you can see, the one-step delay in the context PE. After this one-step delay, the output of the neuron is returned as an input. For the Jordan network, the context PEs only exists in the input layer, and there is no recurrency in the input-output path.

Based on early experiments, we came to the conclusion that for an identifier whose input-output relationship is not so complex, one hidden layer with around 20 neurons is enough. Of course, one can construct a multi hidden layer neural network with each hidden layer having around 12 neurons. However, it will not improve results much and only waste time in training. Therefore, in our application we used only one hidden layer.

After training the identifier was fixed. The controller was also constructed using the Jordan Network and trained in real-time.

5. TESTING

During the implementation phase of each module, we already tested each function separately. When the implementation was finished we performed a full system's test. During the full system's test we encountered several problems in the current control system:

- The airplane wobbled a lot at the start of each flight procedure (visible in Figure 8);
- The airplane kept descending during the default flying procedure (see the lower picture in Figure 9);

- The airplane changed its behaviour dramatically when going from one flight procedure to another (see the upper picture in Figure 9);

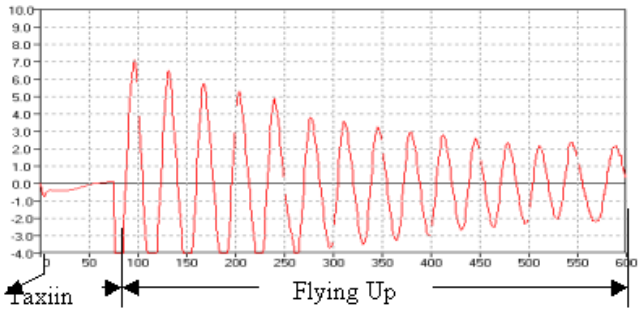

Figure 8: pitch error during the taxiing and flying up procedure

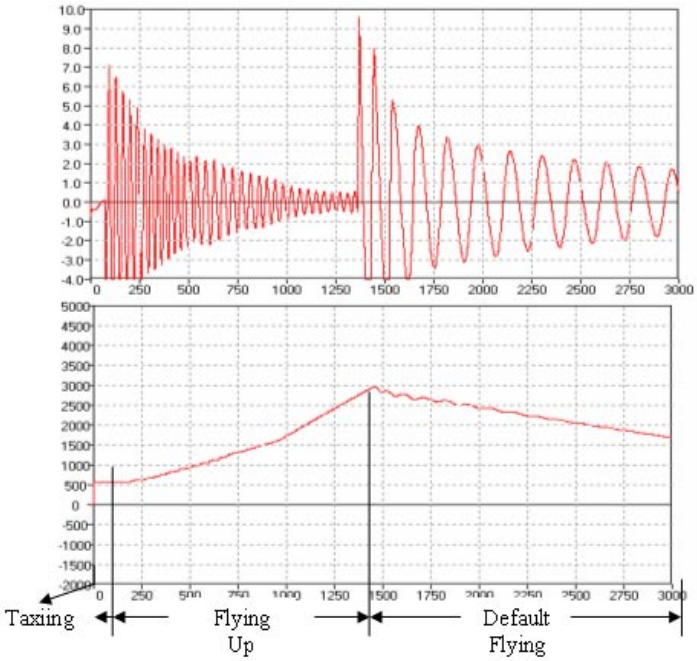

Figure 9: pitch error (top) and altitude (bottom) during take-off

6. IMPROVEMENTS AND EVALUATION

Trying to solve the above-mentioned problems, we came to the following solutions;

- Limit the controller's output range of the elevator;
- Change the desired pitch value for the default flying procedure, slightly above 0;
- Modify the reference table used by the controller to make the desired pitch output change gradually.

6.1. Results

Figure 10 shows the pitch error during the taxiing and the flying up procedure. The pitch error shown in the right plot is taken from the airplane controlled by the improved controller and the data shown in the left plot is from the airplane controlled by the previous controller. From the comparison, you may see the pitching magnitude has decreased considerably.

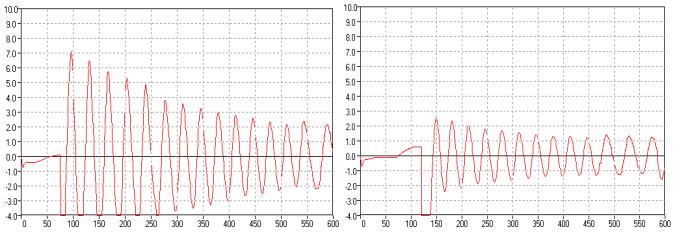

Figure 10: pitch error comparison between original (left) and improved controller (right)

From the pitch signal, which is the lower plot in Figure 11, you can see that at the start of the default flying procedure, the pitch is levelling off gradually, resulting in a slow increase of the altitude until it settles on a certain value. Compared to the altitude plot in Figure 9, it is clear that the improved reference model of the desired plant output makes the airplane fly much better.

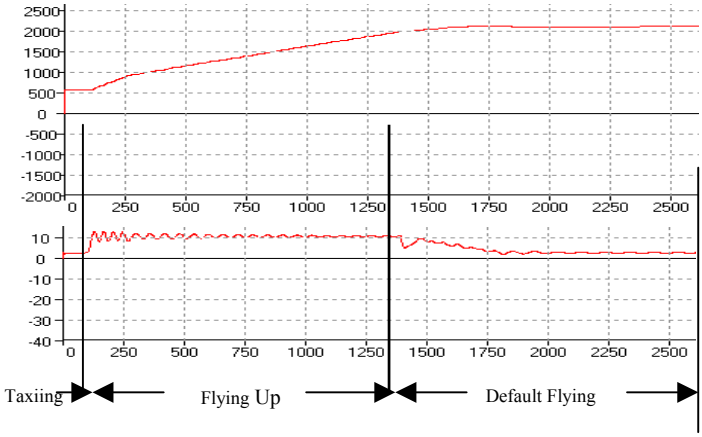

Figure 11: altitude (top) and pitch error (bottom) during the take-off and fly to 2000 feet procedures

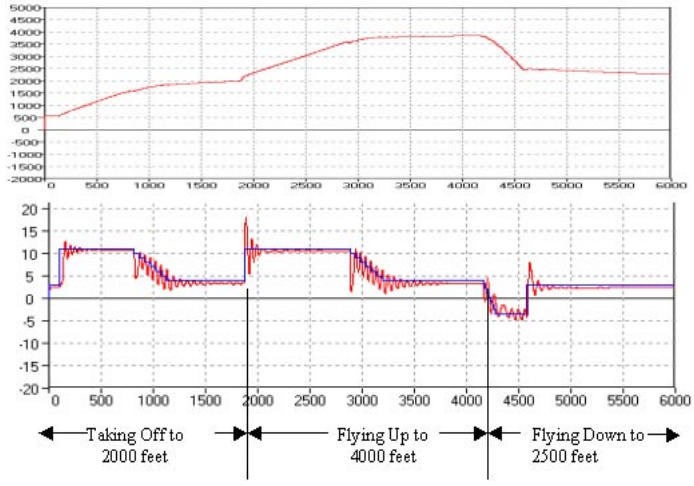

Figure 12: altitude (top) and pitch error (bottom) during take-off, flying up and flying down procedures

Figure 12 shows the altitude and pitch error during a flight were the aircraft flew up to 2000 feet, then up to 4000 feet, and again down to 2500 feet. The red line in the lower part of Figure 12 shows the actual pitch output value while the blue line is used to visualize the desired pitch output value. From the comparison we can see that the neural controller can respond to changes of the desired pitch value immediately. In another words, the controller has fast reaction ability.

6.2. Controller stability analysis

There are several ways to analyse the stability of the controller. For example, we may characterize stability from an input-output viewpoint, or we can characterize stability by studying the asymptotic behaviour of the state of the system near steady-state solutions, like equilibrium points. We prefer to use the steady-state stability analysis, so we studied if the current system is asymptotic stable and characterized the attraction region.

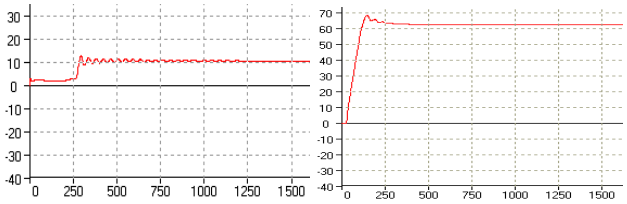

Figure 13: elevator (left) and throttle input (right) during the flying up procedure

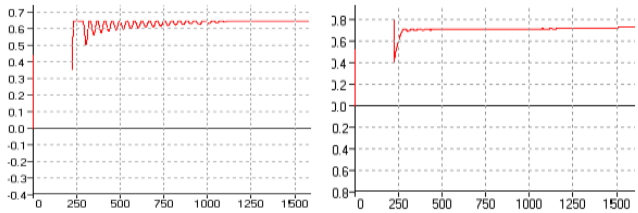

Figure 14: pitch (left) and the airspeed output (right) corresponding to Figure 13

Figure 13 shows that at the start of the control phase both the elevator input and the throttle input start with an arbitrary value. As the process goes on, both slowly settle on a certain value to achieve the desired pitch and airspeed output. Figure 14 includes the corresponding pitch value and the airspeed value taken from the elevator control input and the throttle control input.

With a smaller input the output will be smaller also, and the control inputs will finally settle on a certain value to achieve a certain desired output. These two characteristics indicate that this control system is asymptotic stable.

7. CONCLUSIONS AND DISCUSSION

7.1. Conclusions

The results presented in the previous chapter demonstrate that the current neural flight controller system can:

- Control the airplane to take off, fly up and fly down;
- Run alongside the Microsoft Flight Simulator, which is a large CPU time consuming application;
- Control the airplane so that it achieves a stable flight;
- Respond to the changes of the desired plant output immediately;
- Provide the current flight situation to the user and visualize the evaluation data in 2D coordinates in real time.

The test results also show that the training of the controller neural network is affected by the pre-defined desired plant output. Therefore, setting the proper desired plant output for each flight procedure is very crucial for a good controller system.

As mentioned before, one of our improvements was to limit the output range of the elevator control to decrease the "wobbling" pitching magnitude, but this phenomenon still occurs (to a much smaller extent) at the beginning of each flight procedure. It does disappear as training progresses, as can be seen in Figure 15.

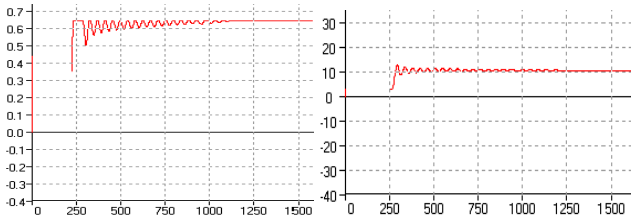

Figure 15: normalized elevator input (left) and pitch value (right) during flying up procedure

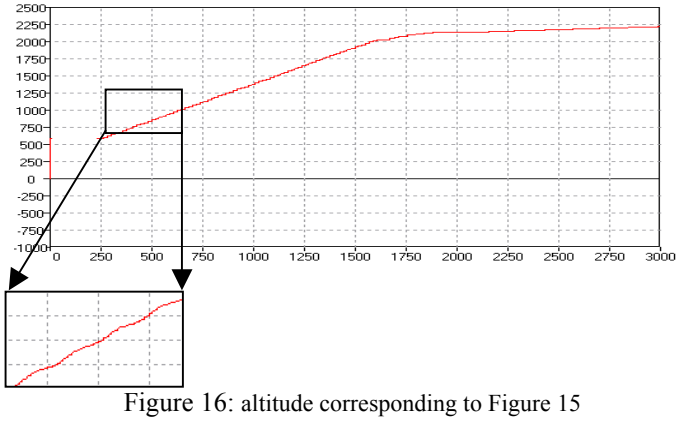

Figure 16: altitude corresponding to Figure 15

7.2. Discussion

Unfortunately, the "wobbling" pitch phenomena cannot be avoided if we stick to the continuous online training of the single neural controller system that we use. However, we feel that this is not a big problem, since this phenomenon can also be found in normal piloting behaviour, which we ultimately intend to simulate. Figure 16 shows that the overall change in altitude progresses as normal and small variations are only visible on closer inspection.

Because of its online training ability, the developped neural controller system can adapt itself to new situations as they arise. This makes our neural network controller more flexible than rule-based control technique, like fuzzy control. With fuzzy control one would define the control rules beforehand, based on the experience of the expert. The advantage of fuzzy control is that for some problems one may have an intuitive idea about how to achieve high performance control, but the consequent problem is that a human expert cannot predict all situations that can occur. Even if the expert is able to predict everything and write it into rules, the rule base will become large and complex, and might not balance the stability criteria and the performance objectives.

Our neural flight control system is flexible and can be applied to different aircraft applications. The architecture will remain the same. Adapting to other aircraft only requires replacing the pre-trained neural network identifier by another suitable one and indicating the desired output of the airplane.

8. FUTURE WORK

As discussed in the last section, the neural control system will make the airplane fly not smoothly at the start of a flying procedure. Although it is not very troublesome we would like to be able to simulate pilots that do not try to correct the pitch. One possible solution we are thinking of is the following. The current neural network controller is trained at each time there is a training pattern available, which will make the controller quite "sensitive". Instead of training the controller every time a training pattern is available, we could train it when, for example, 10 training patterns are available. Then the airplane could keep the current pitch for a while.

Of course our current neural control autopilot system is still quite limited. It can only control the airplane to fly up and down in a straight line. For future's work, we plan to incorporate more functions such as turning and landing.

Once we have a fully functional controller we plan to compare the behaviour of the resulting controller with the flight behaviour of the pilot who delivered the training data. Additionally we plan to record flight data from different types of pilots, which would allow us to train different controllers with different flying behaviour.

REFERENCES

Calise, A.J (1996) "Neural networks in nonlinear aircraft flight control", *IEEE Aerospace and Electronic Systems Magazine*, Vol. 11, Issue 7 (July), pp. 5-10

Ceranowicz, A (1994) "Modular semi-automated forces", in *Proceedings of the 26th conference on Winter simulation*, Orlando, Florida, US, pp. 755-761

Coradeschi, S., Karlsson, L. and Törne, A. (1996) "Intelligent agents for aircraft combat simulation", in *Proceedings of the 6th Computer Generated Forces and Behavioral Representation Conference*, pp. 23-25, July 1996, Orlando, Florida, US.

Ehlert, P.A.M. and Rothkrantz, L.J.M. (2003) "*The Intelligent Cockpit Environment Project*", Research Report DKS03-04/ICE 04, Knowledge Based Systems Group, Delft University of Technology, The Netherlands.

Jones, R.M., Laird, J.E., Nielsen, P.E., Coulter, K.J., Kenny, P. and Koss, F.V. (1999) "Automated intelligent pilots for combat flight simulation", in *AI Magazine*, Vol. 30, No.1, pp. 27-41

Jordan, M.I. and Rumelhart, D.E. (1992) "Forward models: Supervised learning with a distal teacher", in *Cognitive Science,* Vol. 16, No.3, pp. 307-354

Microsoft (2002) "*Rod Machado's Ground School*", Microsoft Flight Simulator 2002 manual.

Laird, J.E., Newell, A. and Rosenbloom, P.S. (1987) "Soar: and architecture for general intelligence", in *Artificial Intelligence*, Vol. 33, No.1 pp.1-64

Liang, Q, (2004) "*Neural flight control autopilot system*", MSc thesis, Research Report DKS04-04 / ICE 09, Delft University of Technology, The Netherlands.

Pesonen, U.J., Steck, J.E. and Rokhsaz, K. (2004) "Adaptive neural network inverse controller for general aviation safety", in *AIAA Journal of Guidance, Control, and Dynamics*, Vol.27, No.3 (May–June), pp. 434-443

Wyeth G.F., Buskey G. and Roberts J. (2000) "Flight control using an artificial neural network", in *Proceedings of the Australian Conference on Robotics and Automation* (ACRA 2000), August 30 - September 1, Melbourne, pp. 65 -70.

Zell, A., Mamier, G., Vogt, M., Mach, N., Huebner, R., Herrmann, K.U., Soyez, T., Schmalzl, M., Sommer, T., Hatzigeogiou, A., Doering, S., Posselt, D (1995) "*SNNS Stuttgart Neural Network Simulator*", User Manual, version 4.1, Report No. 6/95, University of Stuttgart. See also http://www-ra.informatik.uni-tuebingen.de/SNNS/