FLIGHT CONTROL SYSTEM DESIGN FOR COMMERCIAL AIRCRAFT USING NEURAL NETWORKS

Madjid HAOUANI*, Maarouf SAAD* and Ouassima AKHRIF*

* *Electrical Engineering*

Abstract: This paper describes a neural network based approach for flight control design of commercial aircraft. A neural flight control system is used to actuate the gains of a linear controller for gain scheduling purposes without any explicit system identification. The neural network system uses reference models to specify desired handling qualities. Simulation results demonstrate the good tracking performance of the proposed control approach and the learning capability of the proposed neural network controller by showing rapid convergence of the network weights. *Copyright © 2002 IFAC*

Keywords: Flight control, Neural network, Optimization, Gain Scheduling, Handling Quality

1. INTRODUCTION

Over the last decades, a wide variety of control approaches have been used to achieve desired maneuvers (Honeywell, 1996). Avionics field has consequently evolved remarkably from basic electrical systems to highly advanced engines. With the advent of fly-by-wire flight control technology, it is now possible to shape the handling qualities of an aircraft to desired specifications. However, while this approach has proven to be successful, the development process is generally very expensive and often results in aircraft specific implementations.

In this paper, a flight control system based on artificial neural networks is proposed. Artificial neural networks are characterized by their learning capability. The learning process consists in adjusting connection weights between neurons (interconnected processing units) according to some criteria that the network should satisfy. Once the learning phase is completed the network is able to map the input-output relationship without any information about the system internal structure. It has been proven that a two-layer feedforward neural network is able to approximate any continuous function (Cybenko, 1989).

The neural network proposed approach for flight control design realizes two main objectives. First, a multilayer neural network is used to optimize the gains of a linear controller. This network allows the gains of a linear controller to adapt according to the nominal configuration of the controlled system. In order to reduce the learning time, which can be very long, the Uniform Design method (Fang, 1994) has been adopted. This method aims to reduce the learning space while conserving its characteristics. Second, while the system is always confronted to disturbance and thus its configuration may be affected, a self-organizing map (Kohonen, 1987) based controller is used to compensate for errors and adapt the linear controller gains to these changes. Thus, desired handling qualities can be achieved across flight conditions and for different aircraft configurations. Simulation evaluations have been performed on several aircraft configurations to ensure the accuracy of the proposed approach.

This paper is organized as follows. A brief description of the handling qualities concept and the flight control system is provided. Then the neural network based flight control architecture is proposed. A description of the neural network design, including the learning and testing phases and the reduction of the learning space is detailed. For validation purpose, a simulation example is considered followed by some conclusions about the proposed approach.

2. HANDLING QUALITIES

Handling or flying qualities defined by Cooper and Harper (1969) are "*those qualities or characteristics of an aircraft that govern the ease and precision with which a pilot is able to perform tasks required in support of an aircraft role*". They are expressed in a combination of time-domain and frequency-domain constraints on the aircraft dynamic behavior. The handling quality criteria considered in this paper are pitch attitude bandwidth ωBW, phase delay τ_p, short period mode damping ratio ξ_{sp} and Gibson DropBack *DB*. The fixed boundaries about these criteria are given by MIL-STD-1797A (1990). Table (1) summarizes the boundaries of the handling qualities used in our design.

Table 1. Handling qualities boundaries

Handling Qualities	*Boundaries*
Attitude Bandwidth ωBW	> 1.5 (rad. / sec.)
Phase Delay τ_p	< 0.2 (sec)
Short Period Mode Damping Ratio ξ_{sp}	$0.35 < \xi_{sp} < 1.35$
Gibson Dropback DB	$-0.2 < DB/q_{ss} < 0.5$ (sec)

3. FLIGHT CONTROL SYSTEM DESIGN

This paper is interested in the longitudinal pitch response (figure 1).

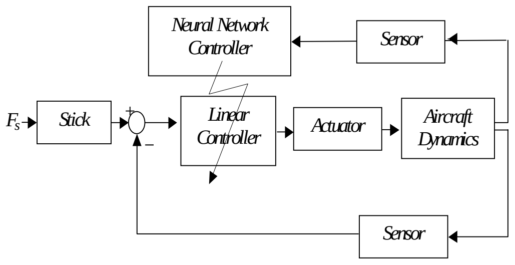

Fig. 1. Closed-Loop Flight Control System

The design of this flight control system consists of finding the appropriate gains for the linear controller such that the handling qualities can be satisfied. Obviously, this design can be modeled as parametric optimization problem.

3.1. Linear controller gains optimization

The general formulation of optimization problems is given by

$$\min_{x\in R^n}\{f(x)|x_l \le x \le x_u;\, h(x)=0;\, g(x)\le 0\} \qquad (1)$$

with

R^n	*n*-dimensional Euclidean space;
f	objective function;
x	vector of *n* design variables;
g	vector of *p* inequality constraints;
h	vector of *q* equality constraints; and
x_l, x_u	lower and upper bounds of design variables

The design variables and the constraints form the feasible space

$$\Omega \equiv \{x\in R^n | x_l \le x \le x_u;\, h(x)=0;\, g(x)\le 0\} \qquad (2)$$

which describes the design freedom.

If only one objective function $f(x)$ is taken into account, the problem is called scalar optimization. If several objectives are considered simultaneously, a vector or multiple objective optimization problem exists. In this case, the formulation (1) changes to

$$\min_{x\in R^n}\{F(x)|x_l \le x \le x_u;\, h(x)=0;\, g(x)\le 0\} \qquad (3)$$

where $F(x)=(f_1(x), f_2(x),\ldots, f_k(x))$ is now a vector of k objective functions. A multiple objective optimization problem can be solved by means of a substitute scalar optimization problem.

A standard technique for multiple objective optimization problem is to minimize a positively weighted sum of the objectives, that is,

$$\min_{x\in\Omega}\sum_{i=1}^{k}\alpha_i f_i(x), \quad 0\le\alpha_i<\infty \qquad (4)$$

In this study, a performance evaluation function is assigned to each handling quality criterion. This results in an unconstrained optimization problem. Since the system configuration changes each time the flight conditions are modified and a certain number of different flight configurations are considered, the unconstrained optimization should be done for all these different cases. In other words, a set of gains should be determined for each different system configuration. However, the use of a classical optimization routine requires a considerable amount of time. In order to reduce this optimization time a neural network based approach is proposed. The idea is to design a multilayer neural network for the approximation of the inverse optimization function. Then, to use this network to determine the optimal set of gains corresponding to the minimum of this function. Since the optimization problem is reduced to function approximation, a two-layer feedforward neural network is considered (figure 2).

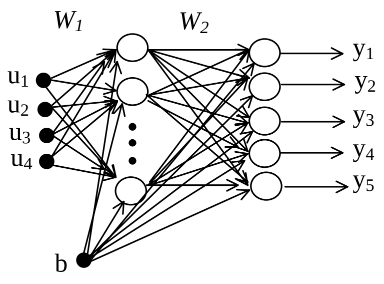

Fig. 2. Multilayer neural network

A schematic representation of a two-layer neural network is shown in figure (2). The basic mathematical expression representing the network is given by:

$$y(k) = N(u(k)] = \Gamma[W_2(\Gamma[W_1(u(k),b),b)] \quad (5)$$

where W_1 and W_2 are $[m_2 \times (m_1+1)]$ and $(n \times m_2)$ weight matrices, $u(k)$ and $y(k)$ are m_1 input and n output discrete vectors, and b is a bias term always equal to one. The nonlinear sigmoid function $\Gamma(X) = [(1-e^{-x})/(1+e^{-x})]$ is applied to each element of the vector X. Usually the learning stops when the weight matrices become stable. This is done by minimizing the quadratic error between the measured output $y(k)$ and the desired output $y_d(k)$ for a given input $u(k)$. In other words, the weight matrices are obtained by minimizing the function:

$$J(k) = \frac{1}{2}\sum_{l=1}^{L}\sum_{i=1}^{n}[y_{di}(l) - y_i(l)]^2 \quad (6)$$

where n is the number of outputs nodes and L is the number of samples. To achieve the minimization, gradient methods of backpropagation errors are generally applied. Steepest descent method with a fixed step (Hertz, 1991) or Quasi-Newton (Fletcher, 1987) method can be used to solve (6). Using Quasi-Newton, the updating matrices are given by:

$$W(k+1) = W(k) - \eta H(k)\nabla(k) \quad (7)$$

where $\nabla(k)$ is the gradient, η is the learning coefficient adjusted at each iteration (Fletcher, 1987) and $H(k)$ is the estimation of the inverse Hessian matrix. Convergence is achieved only if $H(k)$ remains symmetric and positive definite.

At $k = 0$, the matrix $H(0)$ is usually fixed equal to the identity matrix. In this case, the first iteration is a classic steepest descent minimization. In general, the learning algorithms with a variable step converge more rapidly than the methods with a fixed step. However, the implementation in real time of variable step algorithm is difficult. In this study, the learning is realized off-line and the Brayden-Fletcher-Goldfarb-Shanno algorithm is used to accelerate the convergence process.

Since there is no explicit information about the location of the optimal solution for our multiobjective optimization problem, a considerable amount of learning data is needed to perform the training of the network. To reduce the optimization solution space, preliminary simulation of the multiobjective function is performed (see figure 3).

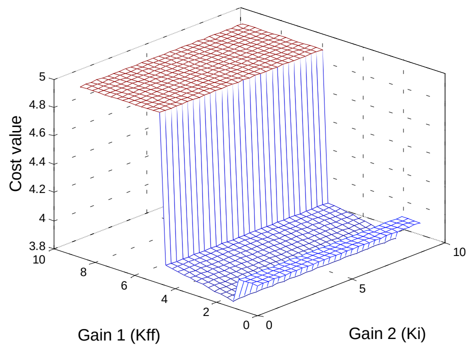

Fig. 3. First two dimensions of the solution space

The above figure shows the first two dimensions of the solution space in the case of our example (detailed in section 4). This simulation method shows that the optimization solution space may be reduced to the following:

Table 2. Example of solution space reduction

Linear Controller Gains	*Interval*
Gain 1*(Kff)*	$0 < Kff < 5$
Gain 2 *Ki*	$0 < Ki < 2$
Gain 3 *(Kprop)*	$0 < Kprop < 2$
Gain 4 *(Knz)*	$0 < Knz < 8$
Gain 5 *(Kfb)*	$0 < Kfb < 1$

The solution space proposed in Table 2 is still large. This results in a considerable amount of data and consequently a long time of simulation to explore the whole space. However, an experimental design method called Uniform Design (Fang, 1994) is very useful for this problem. The uniform design main objective is to sample a small set of points uniformly scattered from a given set of points. In the following, the main features of uniform design are given and the reader is referred to (Wang and Fang, 1981) for more details. Consider the first three gains of our linear controller and suppose that each gain may take ten possible values from the solution space. To find the optimal set of gains, one thousand combinations should be considered. When it is not possible to consider all these combinations, uniform design allows to select a small but representative sample of combinations. Let n be the number of gains considered in the optimization and q the number of possible values for each gain, the uniform design selects q combinations out of the q^n possible combinations, such that these q combinations are scattered uniformly over all possible combinations space. The selected combinations are expressed in terms of *uniform array* $G(n,q) = [G_{i,j}]_{q \times n}$, where $G_{i,j}$ is the value of jth gain in the ith combination. Uniform arrays are determined as follows. Consider a unit hypercube over an n-dimensional space, denoted by:

$$C = \{(c_1, c_2, \ldots, c_n) \mid 0 < c_i < 1 \text{ for } i = 1, 2, \ldots, n\} \quad (8)$$

Consider any point r = ($r_1, r_2, \ldots, r_n$) in C. A hyper-rectangle between 0 and r is formed and can be denoted by the set of points:

$$C(r) = \{(c_1, c_2, \ldots, c_n) \mid 0 < c_i < r_i \text{ for } i=1,2,\ldots,n\} \quad (9)$$

A set of q points is selected such that they are scattered uniformly in the hypercube. Suppose that $q(r)$ of these points pertain to the hypercube $C(r)$. The fraction of points in the hypercube would be $q(r)/q$. The volume of the unit hypercube is one and hence the fraction of this hyper-rectangle is $r_1 r_2 \ldots r_n$. The uniform design is to determine q points in C such that the following discrepancy is minimized:

$$\sup_{r \in C} \left| \frac{q(r)}{q} - r_1 r_2 \cdots r_n \right| \quad (10)$$

Then the obtained q points are mapped in the unit hypercube to the space with n gains and q values. When q is prime and $q > n$, $G_{i,j}$ can be determined using the following relation (Wang and Fang, 1981):

$$G_{i,j} = (i\sigma^{j-1} \bmod q) + 1 \quad (11)$$

where σ is a parameter given in Table 3. (Fang, 1994).

Table 3. Values of σ for different number of gains and different number of values per gain.

Number of values per gain	*Number of gains*	σ
5	2 - 4	2
7	2 - 6	3
11	2 - 10	7
13	2	5
	3	4
	4 – 12	6
17	2 - 16	10
19	2 - 3	8
	4 - 18	14
23	2, 13 – 14, 20 - 22	7
	8 - 12	15
	3 – 7, 15 - 19	17
29	2	12
	3	9
	4 - 7	16
	8 – 12, 16 - 24	8
	13 - 15	14
	25 - 28	18
31	2, 5 – 12, 20 - 30	12
	3 – 4, 13 -19	22

Unfortunately, only uniform arrays with at most 37 values have been tabulated (Fang, 1994). Since the solution space for the linear controller may be remarkably larger, much more points are needed for a better coverage. To bypass this difficulty, the solution space can be subdivided into multiple subspaces, and then the uniform design can be applied on each subspace. The obtained sets of gains are then used in the multiobjective cost function to calculate their corresponding sets of four cost values corresponding to the four objective functions. For the training of the two-layer network, the obtained cost values are used as inputs and the corresponding gain sets are used as outputs. Thus, the network will learn to approximate the inverse of the multiobjective cost function. Once the training is finished, the network should be able to respond to a given set of cost values by the appropriate set of gains. Since the optimal set of gains $G_d^{\,i}$ corresponding to i^{th} flight system configuration is the one that responds with a minimum cost value, this optimal set can be obtained using the following relation:

$$G_d^{\,i} = N^i(0) = \Gamma[W_2(\Gamma[W_1(0,b),b)] \quad (12)$$

For validation purposes, the system is simulated with the obtained set of gains. The handling qualities are verified and the obtained cost is compared to the one presented at the network's input. If the difference is significant the input value is increased slightly and the same operation is repeated until the difference becomes insignificant. The obtained set of gains can then be considered as the optimal solution for the considered flight case. This operation is repeated for all the different cases. The obtained results are then stored in a Table with the case number and the corresponding optimal set of gains. This Table will then be used by the neural network controller. The design of this controller and the way it uses this Table to control the system is detailed in the following sections.

3.2. Neural network controller design

The role of the neural network controller is to provide the linear controller with its five optimal gains such that the handling qualities are verified. This operation is made continuously to maintain the aircraft in this level of performance at all times. The neural controller should thus react to any change in the flight conditions and modify, if necessary, the linear controller gains. These flight conditions are characterized by variables such as the Mach number, the dynamic pressure, the gravity center, the weight and the stability angle. The neural network considered in this work is a self organized map of Kohonen (1989). For the training of the network, different nominal situations are considered. These configurations cover almost all the main situations that the system may have. In case a different situation is presented, the neural network should converge automatically to the closest flight condition within the nominal cases. Then, the corresponding gains should be loaded for this new situation. This procedure is called gain scheduling. The training of the neural network is made off-line. A detailed description of the network training is presented in the following subsection.

3.3. Neural Network training

One of the most important issues in pattern recognition is feature extraction. For such a crucial objective, self-organizing networks are very accurate. These networks can learn correlations in their inputs and adapt their future responses according to that input. Self-organizing feature maps are one layer nets with linear neurons using a competitive learning rule. In such nets, there is one and only one winning neuron for every input pattern (i.e. the neuron whose weights are closest to the input pattern). In competitive nets, only the weights of the winning node get updated. Kohonen proposed a slight modification of this principle with tremendous implications. Instead of updating only the winning neuron, the neighboring neuron weights are also updated with a smaller step size. This means that in the learning process (topological) neighborhood relationships are created in which the spatial locations correspond to features of the input data.

The self-organizing feature map algorithm is resumed to the following steps. First, the weights of the network are initialized with small different random values for symmetry breaking. Each row of the weight matrix is set to a random vector normalized to a length of one. Then, for each input data *x(n)*, the winning neuron *i(x)* is obtained using a minimum distance rule, i.e.

$$i(x) = \arg_j \min \| x(n) - w_j \| \tag{13}$$

For the winning neuron, its weights and those in its neighborhood are updated by

$$w_j(n+1) = w_j(n) + \eta(n)\lfloor x(n) - w_j(n) \rfloor \tag{14}$$

The network should be trained for several iterations until it stretches itself over the input vectors. The network weights are adjusted so that each neuron responds strongly to a region of the input space occupied by input vectors. When the training phase is completed, the network is tested, for validation purpose, with a couple of cases, some of them are taken from the learning data and the others are completely different. The training and testing phases being completed, the network becomes operational and ready to be implemented in the flight control system. In the following, a simulation example is presented in order to illustrate the proposed approach.

4. SIMULATION EXAMPLE

Results from simulations using the proposed neural network approach are presented in this section. The system to be controlled is the longitudinal flight control system of figure (4). This classical control architecture is adapted from (Tisher, 1997). The adjustable gains are K_{ff}, K_i, K_{prop}, K_{nz} and K_{fb}. The aircraft is represented by a 5th–order linearized model. The available measurements are pitch rate q and normal acceleration n_z.

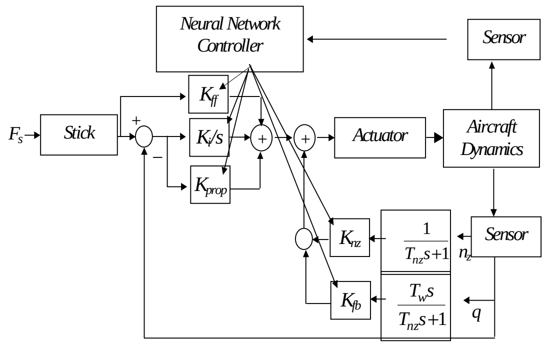

Fig. 4. Flight control system example

For the training of the neural controller, 160 different flight cases are considered. For each flight configuration, the proposed neural network based optimization approach is used to determine a set of five gains to be implemented on the linear controller.
To determine the optimal set of gains, a solution space of 992 values, divided into 32 subspaces with 31 values per gain for each subspace is considered. The uniform design method is then applied on each subspace to determine the 31 values representative of the 31^5 initial possible combinations. The obtained 992 sets of five gains are then used in the multiobjective cost function to calculate their corresponding 992 sets of four cost values corresponding to the four objective functions. Then, the obtained cost values are used as inputs and the corresponding gain sets are used as outputs for the training of the two-layer neural network. At the end of the training, the network has learned to approximate the inverse of the cost function and then it can be used to determine the optimal gains for the corresponding flight case. The results obtained for one case among the different flight configurations are shown on Table (4) below.

Table 4. Handling qualities for a case example

Handling Quality	*Value*
Attitude Bandwidth ω_{BW}	2.37
Phase Delay τ_p	0.11
Short Period Mode Damping Ratio ξ_{sp}	0.82
Gibson Dropback DB	-0.01

All the handling qualities are respected (Table 4). Figure (5) shows a good tracking performance.

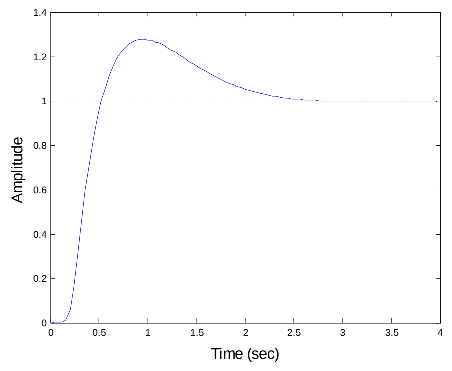

Fig. 5. Pitch rate step response

This operation is repeated 160 times to obtain the 160 different sets of five gains for the training of neural network controller.
Before starting the training phase the Matlab function *randnr* is used to generate random normalized vectors for our (5 by 32) layer neural network corresponding to the 160 different cases.

One hundred and twenty epochs were required to obtain satisfactory training (see figure 6).

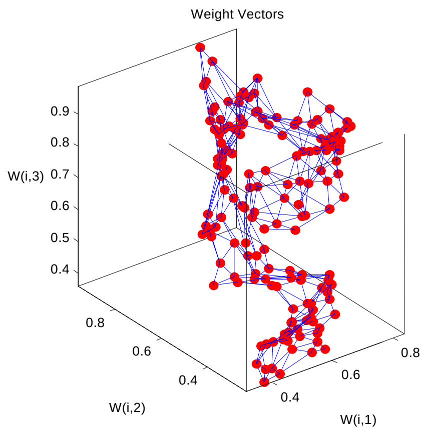

Fig. 6. Controller training results

The above figure shows the first three dimensions of the weight matrix. After one hundred and twenty training cycles, the network has adjusted its weights so that each neuron responds strongly to a region of the input space occupied by input vectors. Then it has been tested for several cases and has responded correctly. The trained controller is then implemented on the system of figure (4). To test the controller on the system, two situations were considered. The system was first loaded with a given flight case and after 4 seconds the flight conditions were changed intentionally to observe the reaction of the neural controller. For both situations, the neural network was able to determine the adequate gains for the linear controller in order to stabilize the system. The handling qualities were tested for both situations and the results were satisfactory.

5. CONCLUSION

A neural network approach for flight control system design has been presented in this paper. An obvious interest of the proposed approach is the combination of classical flight control system architecture with modern control theory without sacrificing the system architecture presently in use. This method provides guaranteed properties that are handling qualities. The proposed method can be addressed to obtain good performance for a large flight envelop using the neural network scheduled gain controller. This approach can be improved by considering the neighborhood of each single flight case. Instead of taking a unique set of gains per case, a space solution will be provided and the neural scheduled gain controller will select the closest one with small gains variations.

REFERENCES

Cooper, G. E., Harper Jr R. P., (1969). *The use of Pilot Rating in the Evaluation of Aircraft.* Technical Report NASA-TN-D5153, Washington D. C.: National Aeronautics and Space Administration.

Cybenko, G. (1989). *Approximation by superposition of sigmoidal functions.* Mathematical Control Signals and Systems, Vol. 2, pp. 303-314.

Fang K. T., 1994. *Uniform Design and Design Tables,* Beijing, China: Science. In Chinese.

Fletcher R., (1987). *Practical Methods of Optimization.* New York: John Wiley and Sons.

Hertz J., Krogh A., and Palmer R. G., (1991). *Introduction to the theory of Neural Computation.* Reading, MA: Addison-Wesley.

Honeywell, (1996). *Markets Report,* Tech. Rep. NASA Contract NAS2-114279, Final Repot for AATT Contract.

Kohonen, T. (1989). *Self-Organization and Associative Memory*. Springer-Verlag, 3rd edition , Berlin.

MIL-STD-1797A, (1990). *Fling Qualities of Piloted Aircraft,* Military Standard 1797A Washington D.C.: Government Printing Office.

Tisher M. B., Colbourne J. D., Morel M. R., Biezad D. J., Levine W. S. and Moldoveanu M., 1997. *CONDUIT – A New Multidisiplinary Integration Environment for Flight Control Development.* NASA Technical Memorandum 112203, USAATCOM Technical Report 97-A009.

Wang Y. and Fang K. T., 1981. *A note on uniform distribution and experimental design,* KEXUE TONGBAO, **Vol. 26**, pp. 485-489. In Chinese.