Journal of Cybernetics and Informatics

published by

Slovak Society for Cybernetics and Informatics

Volume 4, 2004

http://www.sski.sk/casopis/index.php (home page)

ISSN: 1336-4774

AUTOMATIC ADAPTATION OF FUZZY CONTROLLERS

Ján Vaščák

Center for Intelligent Technologies, Department of Cybernetics and Artificial Intelligence, Faculty of Electrical Engineering and Informatics, Technical University of Košice
Letná 9, 041 20 Košice, Slovak Republic
Tel: +421-55-602 42 18, FAX: +421-55-625 3574
E-mail: Jan.Vascak@tuke.sk

Abstract

The main drawback of ‘conventional’ fuzzy systems is the inability to automatically design and maintain their knowledge base. To overcome this disadvantage many types of extensions adding the adaptivity property to those systems were designed. This paper deals with the so-called direct adaptation methods of fuzzy control design and especially, with two ones of them: an improved, the so-called, self-organizing fuzzy logic controller designed by Procyk and Mamdani as well as a new hybrid adaptation structure, called gradient-incremental adaptive fuzzy controller connecting gradient-descent methods with the first type. Both types of adaptive fuzzy controllers are shown on design of an automatic pilot and control of LEGO robots. The results and comparisons to a 'conventional' (non-adaptive) fuzzy controller designed by a human operator are shown, too.

Keywords: adaptive fuzzy controller, gradient-descent methods, Procyk-Mamdani adaptation, incremental models

1 INTRODUCTION

Fuzzy logic has found many successful applications, especially in the area of control, but there are some limits of its use that are connected with the inability of knowledge acquisition and adaptation to changed external conditions or parameters of the controlled system. To overcome this problem there were published lots of papers, e.g. [1, 5, 7, 12], which deal with structures of **Adaptive Fuzzy Controllers** (AFC) using mostly approaches based on many variations of gradient-descent methods, the least square method [11], linear and non-linear regression [14] or linguistically based rule extraction.

For the sake of completeness it is necessary also to mention other very perspective means of computational intelligence as neural networks [10], genetic algorithms [4] or migration algorithms [15] (both as parts of evolutionary computing) whose hybridisation leads to neuro-fuzzy and genetic fuzzy systems. However, their computational efforts do not yet enable their use in practical applications.

Further, we will focus our attention only on direct AFC. The main reason why to deal with this type of AFC is that they are in their nature and calculus the most similar systems to the non-adaptive (conventional) FC. The properties of FC are well known, more than in the case of neural networks or genetic algorithms, in general. Fuzzy logic is able to simulate the human vague thinking very efficiently and therefore it seems to be very advantageous only to add the ability of knowledge acquisition to 'conventional' fuzzy systems and to preserve their properties, too.

In this paper we will describe and analyse the so-called **Self-Organizing Fuzzy Logic Controller** (SOFLC) proposed by Procyk and Mamdani [12], which was modified in many papers, e.g. [9, 11]. Further, its modification and implementation will be shown on two examples: automatic pilot and control of LEGO robots. Results of experiments with these systems are summarized in the concluding part of this paper.

2 STRUCTURE OF SOFLC

SOFLC (see fig. 1) belongs to the so-called *performance-adaptive controllers,* which evaluate the control quality by a criterion (or more criteria) like transition time, energy consumption, overshoots, etc. Such a quality measure is called **performance measure** *p(k).*

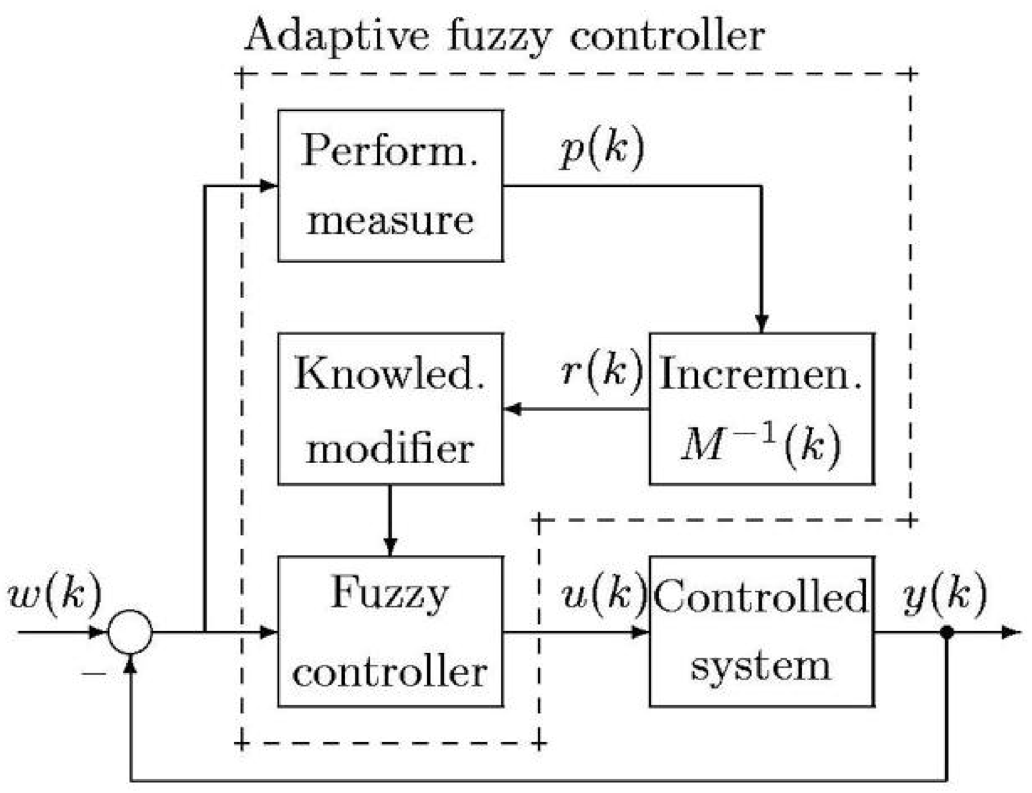


Figure 1: Structure of a self-organizing fuzzy logic controller.

Control criteria are contented in the *performance measure block,* where the quality is evaluated by *p(k),* which expresses the magnitude and direction of changes to be performed in the knowledge base of the controller. The basic design problem of AFC consists in the design of *M*, where for each time sample $t=K.T$ ($K=0,1,\ldots$) a simplified **incremental model** of the controlled system $M=J.T$ (*J* – Jacobean, *T* – sampling period) is computed. It represents a supplement to the original model and is analogous to the linear approximation of the first order differential equation or in other words to gradients, too. As Jacobean (1) is a determinant of all first derivatives of a system with *n* equations $f_1, \ldots, f_n$ of *n* input variables $x_1, \ldots, x_n$ it means *J* is equal to the determinant of the dynamics matrix, i.e. it is a numerical value describing all *n* gradients in the sense of a characteristic value:

$$J = \begin{vmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \mathbf{L} & \frac{\partial f_1}{\partial x_n} \\ & & \mathbf{L} & \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \mathbf{L} & \frac{\partial f_n}{\partial x_n} \end{vmatrix} \tag{1}$$

Now we need to transform this incremental description of a controlled system to the description of a controlling system, i.e. a controller. Considering the properties of the feed back connection we can see that *y(k)* » *e(k)* (*w(k)* is known). As inputs and outputs of a controlled system change to outputs and inputs of a controller, respectively we can get the controller description like an inverse function of $y(k) = f_M(u(k))$, i.e. the model of the controller is in the form $u(k) = f^{-1}_M(y(k))$. Because *J* is a number, then M^{-1} is a reverse value of *J.T*. The **reinforcement value** *r(k)* is computed as $r(k) = M^{-1}.p(k)$ and represents the correction of the knowledge base.

Let the knowledge base in the time step *k* of such AFC be *R(k)* and let its modification in the next time step be *R(k+1)*. The general adaptation rule can be described like:

$$R(k+1) = (R(k) \cap \overline{R_{bad}}(k)) \cup R_{new}(k)\,. \tag{2}$$

We see, firstly, the part of knowledge $R_{bad}(k)$ that caused the low quality control is removed from *R(k)* and then it is completed by new knowledge R_{new}, which is corrected by *r(k)*. $R_{bad}(k)$ and R_{new} are computed as follows:

$$R_{bad}(k) = fuzz(x_1^*(k))\, x\, \mathbf{K}\, x\; fuzz(x_n^*(k))\, x\; fuzz(u^*(k))\,, \tag{3}$$

$$R_{new}(k) = fuzz(x_1^*(k))\, x\, \mathbf{K}\, x\; fuzz(x_n^*(k))\, x\; fuzz(u^*(k) + r(k))\,, \tag{4}$$

where $x_1, \ldots, x_n$ are states of the controller, *u(k)* is its output and * denotes these values are crisp (to protect possible misunderstanding). The only difference between $R_{bad}(k)$ and $R_{new}(k)$ is in consequent parts of IF-THEN rules, i.e. in the output, which confirms the role of *r(k)* as a correction value. The implementation of the knowledge base adaptation can be either ***rule-based*** or ***relation-based*** and will be explained in next sections. In the following section some properties of SOFLC will be described.

2.1 Advantages and drawbacks of SOFLC

It seems to be reasonable to look for methods that would minimize deviations from the optimal state as quickly as possible. Therefore gradient-based methods should be the most convenient for the knowledge modification. In such a sense SOFLC is also a special form embedding this calculus since it utilizes Jacobean, as seen in (1). The only fundamental difference between SOFLC and 'conventional' gradient-descent methods (GDA) can be described as follows. SOFLC represents *gradient of behaviour* and 'conventional' gradient methods can be related to the *gradient of the knowledge base*. In other words, SOFLC directly calculates the derivative of the system behaviour, i.e. its change and 'conventional' gradient methods compute the change of the control error in dependence on the knowledge base parameters. From this reason there is a close relation between *gradient of behaviour* and *gradient of the knowledge base* but no equivalence, rather resemblance or similarity ($\approx$).

Although GDA should be the fastest adaptation but two basic problems are related to it. Firstly, the error function *E(k)* is unknown in advance and it may be of a complex shape with a number of local minima. It is very difficult in advance to estimate their number and possible place of the global minimum, i.e. optimal solution. Further, the absence of such estimation disables the determination of the learning factor value, too. If it is too small the convergence will be too slow and if it is too great there will be a risk the global minimum will be 'jumped over'. Secondly, there is possibility to minimize only one criterion – error function *E(k)* but in the practice there are also other control criteria. SOFLC overcomes these problems partially and is more practice-oriented because it is able to involve other criteria, too. However, it is sensitive to external signals such as disturbances, noises and set-point changes [9, 11] because it is not able to distinguish whether the parameters of the controlled system are changed or an external signal entered the system. A negative effect will occur if the adaptation proceeds although it is not more necessary. In such a way some wrong changes in the knowledge base may be performed (e.g. an external error occurs and AFC will evaluate it as a parameter change). In [11] it is shown that adding some supervisory rules may solve this problem. A modification of SOFLC in the form of the so-called sliding mode control is made in [9] where not only the positions of the control error *e(t)* but also its change (the first derivative) are taken into consideration. This method needs complete knowledge about the states of the controlled system and proper design of the sliding hyper-plane. However, how to design a 'good' hyper-plane it is not solved in this approach. Further, the only criterion for the controller design is the control error. It is of course important that its value converges to zero but in many applications also another criteria may be still more important. From this reason we proposed a modification of this method, which is discussed in section 4. Further, we proposed a hybrid structure merging both SOFLC as well as GDA to balance their properties. This structure is described in the section 3.

3 RULE-BASED IMPLEMENTATION OF SOFLC

As already mentioned in the section 2, the knowledge base *R(k)* can be described in two fundamental ways: ***rule-based*** and ***relation-based***. In the first case there is a set of fuzzy IF-THEN rules and a set of defined linguistic terms in the form of membership functions. Let us denote such a set of rules in this section *R(k)*, too. *R(k)* is the set of N_r fuzzy rules r_p ($p=1, \dots, N_r$) of *n* inputs and one output. Such a rule r_p represents the Cartesian product of these input/output variables and is also a fuzzy relation $R_p=A_{1,p}\ x \dots A_{n,p}\ x\ B_p$, where $A_{1,p}, \dots, A_{n,p}$ are linguistic values of $x_1, \dots, x_n$ for the *p*-th rule and B_p represents its output. The knowledge base *R(k)* is then a union of such rules (fuzzy relations) and after substituting into (2) it will be changed to (5):

$$\begin{aligned}R(k+1) = &\left[\bigcup_{p=1}^{N_r}\left(A_{1,p} \cap \overline{A_1^{bad}}\right) x \dots x\ A_{n,p}\ x\ B_p\right] \cup \dots \cup \left[\bigcup_{p=1}^{N_r} A_{1,p}\ x \dots x \left(A_{n,p} \cap \overline{A_n^{bad}}\right) x\ B_p\right] \cup \\ &\cup \left[\bigcup_{p=1}^{N_r} A_{1,p}\ x \dots x\ A_{n,p}\ x \left(B_p \cap \overline{B^{bad}}\right)\right] \cup \left(\underbrace{A_1^{bad}\ x \dots x\ A_n^{bad}\ x\ B^{new}}_{R_{new}}\right)\end{aligned} \tag{5}$$

$R_{bad}(k)$ can be a union of all previously fired rules, too. However, for the sake of simplicity we will consider only one rule with the greatest strength α and therefore $A_1^{bad}\ x \dots x\ A_n^{bad}$ is its premise. The reinforcement value *r(k)* corrects only the consequent of such a rule and B^{new} is the fuzzified result of *u(k)+r(k)*, i.e. *fuzz(u(k)+r(k))*. The simplest fuzzification is in the form of singletons but in general, other forms are possible, too.

In (5) following drawbacks can be seen:

1. *Possibility to change only one rule in one step* - The adaptation process will be longer and it is possible the low control quality was not caused by the rule with the greatest strength but by another rules working as noise. In every case the convergence will be worse.
2. *Growth number of rules $N_r(k+1)$* - Let us consider *4* rules with two inputs and one output then there will be *13* rules in next step, which leads to an enormous growth of rules in steps *k+2*, *k+3*, etc. ($N_r(k+1)=N_r(k).(n+1)+1$).
3. *Need of garbage collection* - Previous two points show us such filtering or 'garbage collection', i.e. removing useless rules, is necessary to prevent the computational complexity and to improve the adaptation convergence.

Summarising the mentioned facts in the above we can see that GDA and relation-based implementation of SOFLC are with their properties more complementary than contradictory and therefore trying to utilise their advantages a new special hybrid connection of these two methods was proposed as seen in fig. 2.

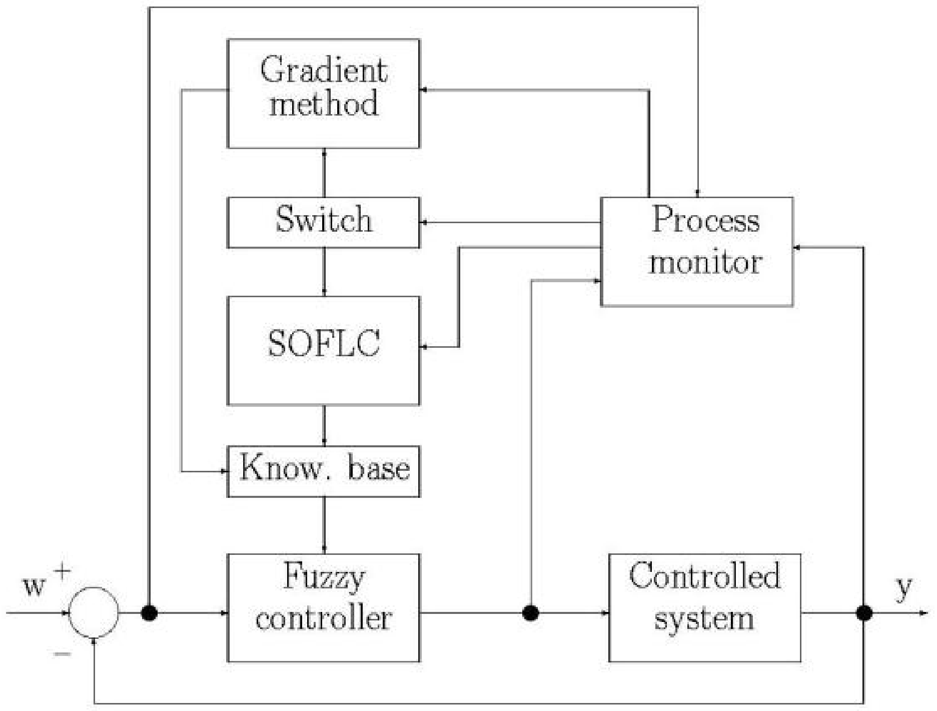


Figure 2: Hybrid control structure of a gradient-descent adaptation system and SOFLC.

The adaptation process can be described in following steps:

1. Defining input and output variables.
2. Defining term sets for variables in the step 1.
3. Designing initial membership functions (not necessary).
4. Processing GDA until the prescribed threshold of the control error *e(k)* is reached.
5. If *e(k)* is under such a threshold then processing SOFLC otherwise jump (switch) to the step 4.

The main idea is that GDA will be the fastest method if the threshold of the control error as the most important criterion is not too strict. In such a case we can choose a greater learning factor and speed up the adaptation. After this 'rough' adaptation we can switch the control to SOFLC to minimize the control error to be as small as possible and at this same time to include other criteria, too.

This hybrid control algorithm was implemented and tested on LEGO robots. Its results were compared also with a non-adaptive FC designed by a human operator. The control task was the so-called *parking problem*, i.e. to park a mobile robot at a given place and direction. It was solved with and without obstacles. The process monitor (see fig. 2) evaluates the parking process by two criteria: *parking error* E_P - more important corresponding with the control error and *trajectory error*

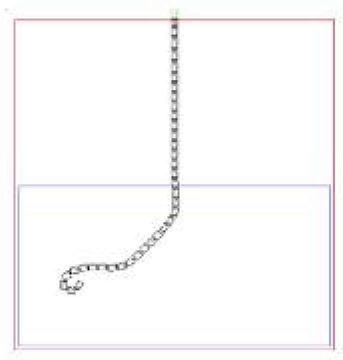
E_T - computed as division of the ***real*** trajectory length and ***optimal*** trajectory length. The optimal trajectory is the shortest distance between the robot and the goal. The first criterion is in the form:

$$E_P = \sqrt{(f_f - f)^2 + (x_f - x)^2 + (y_f - y)^2}\ , \tag{6}$$

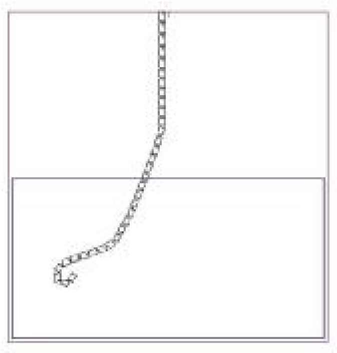
where ***(x, y)*** are coordinates of the robot, *f* is its position angle and ***(x_f, y_f, f_f)*** are position and direction of the goal (parking place). Similar description is used for starting (initial) points, too. In the case of obstacles one additional criterion comes still into consideration - ***number of impacts*** on the obstacle.

The criterion predetermines also the structure of IF-THEN rules, in our case three inputs ***(x, y, f)*** and one output - change of the wheels angle for such a robot. The parking problem was solved with the help of a non-adaptive controller by 35 rules. It has no sense to observe the number of rules in the case of SOFLC because this number was very varying and there is almost no upper limit for their number. We need to consider there are an unlimited number of combinations for intersections of fuzzy sets in (5). However, their number was considerably (several times) greater than for the non-adaptive controller. From this reason it was necessary to use a simple garbage collector, which minimizes the number of rules. It removes replaced and identical rules. If there are rules with identical premises but different consequents the older rule will be removed. Further, it is possible to improve its efficiency removing rules whose membership functions have small heights or merging rules with similar premises.

Results of several experiments for different starting points (in parentheses) are depicted in figures 3, 4 and 5.



(a)



(b)

Figure 3: Comparison of trajectories for a non-adaptive FC (20, 80, 260) (a) and a hybrid AFC (20, 80, 260) (b).

We can see that the first two criteria E_P and E_T are better fulfilled at a non-adaptive FC. There are two reasons. First, E_P and E_T are not totally independent. Both are quantitative and E_P influences E_T directly proportionally. If E_P increases then also the trajectory will be more different from the optimal length but the shape may be in spite of that of 'better quality' what is also this case. It can be seen especially at the obstacle avoidance fig. 4 and 5. This assertion is supported by a smaller number of impacts for GIFC than for non-adaptive FC. Secondly, reinforced rules are fired till in next steps after the error already occurred and in such a way delay influences the efficiency of GIFC negatively. Shortening the sampling period *T* can eliminate this problem. There are only hardware limitations.

4 RELATION-BASED IMPLEMENTATION OF SOFLC

This kind of implementation is considerably simpler than in the previous case. We will construct all three fuzzy relations *R(k)*, $R_{bad}(k)$ and $R_{new}(k)$ as described in (3) and (4). *R(k)* can be set up in the initialisation step as a zero matrix. In such a way we get *n*-dimensional cubes (in the case of two inputs and one output three-dimensional ones) where each element is characterized by the grade of membership to such a fuzzy relation as seen in fig. 6.

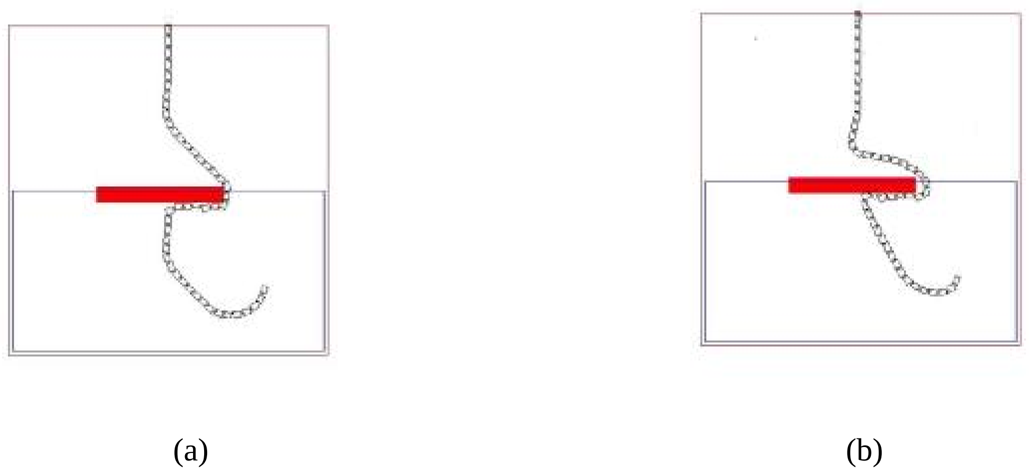


Figure 4: Comparison of trajectories with an obstacle for a non-adaptive FC (80, 80, 260) (a) and a hybrid AFC (80, 80, 260) (b).

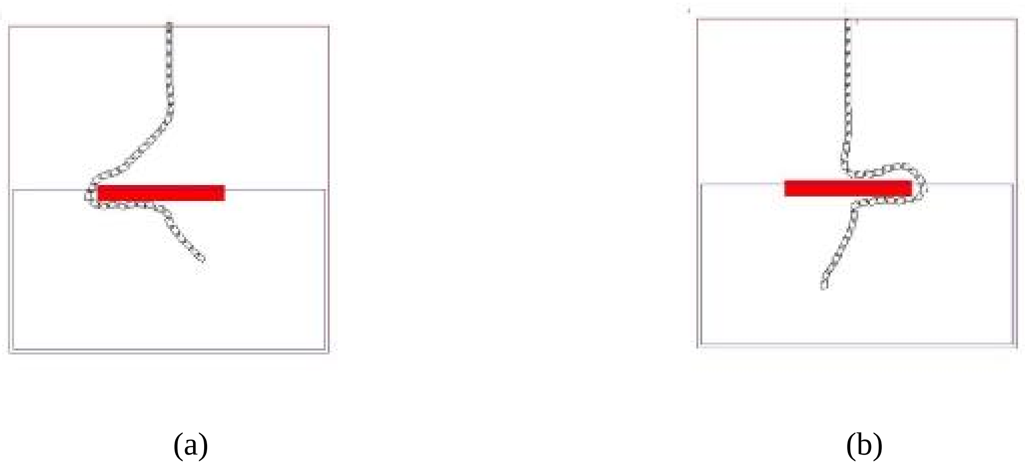


Figure 5: Comparison of trajectories with an obstacle for a non-adaptive FC (60, 70, 150) (a) and a hybrid AFC (40, 80, 110) (b).

However, there are also two basic drawbacks:

1. *Loss of IF-THEN rules* - Knowledge representation in the form of fuzzy relations is not 'user-friendly' and it is not possible to make an unambiguous transform from a fuzzy relation to a fuzzy set (reverse transform is possible).
2. *Higher computational complexity* - To calculate next *R(k+1)* it is needed to perform a complete set of all matrix calculations for each value from supports of input/output values. If the discretization of the support is dense (support has many values) then the size of such a matrix will grow enormously.

The relation-based implementation of SOFLC was used for the design of an automatic pilot. For the sake of simplicity we will take into account only the longitudinal flight (plain *X x Z*) and therefore we will control only the height of the aircraft as seen in fig. 7. The goal of control is to follow the prescribed rising trajectory, in other words, to keep the pitch angle equal to the angle of the prescribed trajectory. The basic description of an aircraft model resulting from motion equations by a fuzzy state model consists of four state quantities:

a - angle of attack (slope of the aircraft in the horizontal flight)

w - vertical velocity (projection of total aircraft velocity into the vertical plane)

q - pitch angle (angle between the longitudinal aircraft axis and earth)

q - pitch rate (derivative of *q*)

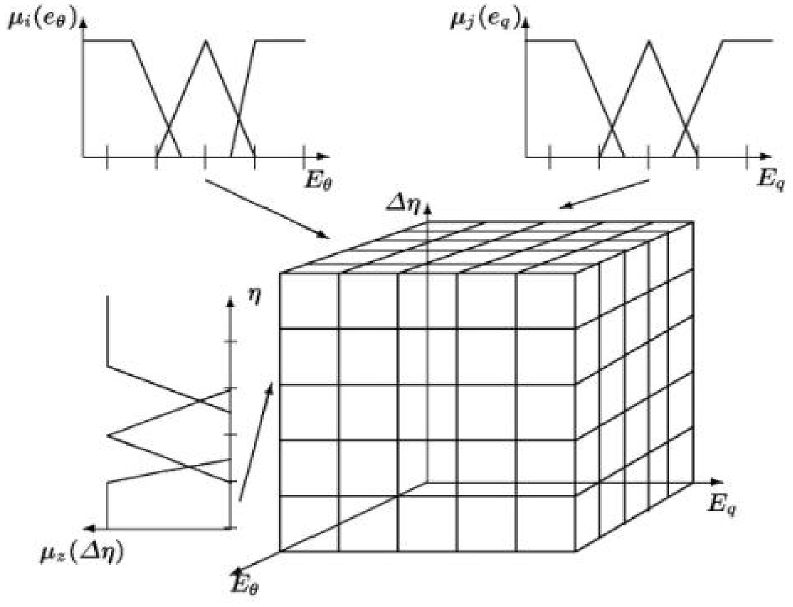


Figure 6: Structure of fuzzy relations for *R(k), R(k+1), Rbad(k)* and $R_{new}(k)$.

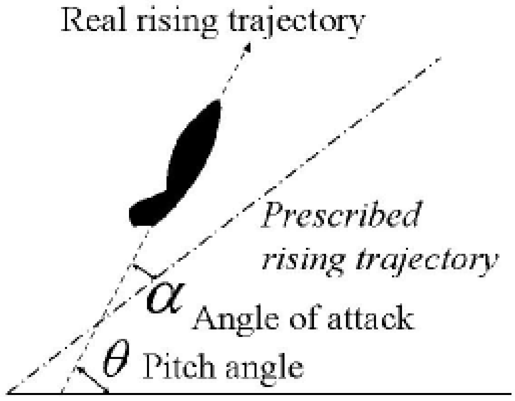
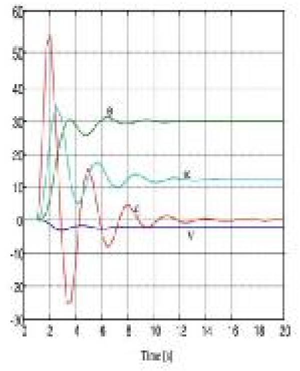


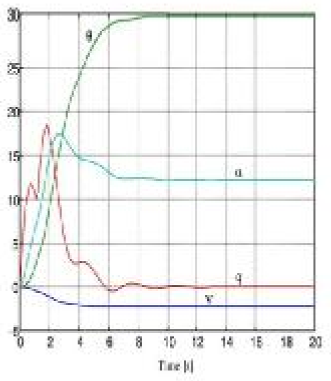
Figure 7: State values of an aircraft for its control in the longitudinal plane.

To overcome problems with the excessive sensitivity to external signals we will in contrast to [9] observe both the position of the performance measure *p(k)* and its trend, i.e. the first derivative $\dot{p}(k)$. The knowledge modifier (see fig. 1) of our AFC is completed still by an *adaptation* *supervisor* where these rules are included. They are mostly problem-oriented and their parameters are dependent on the application but we can find some general knowledge, which can be described as follows. If we have a totally empty knowledge base we will let the adaptation (more correctly learning) in full processing without any limitations until it reaches a proper state of the controlled system (by defined criteria). Then we will stop the adaptation and will observe *p(k)* and $\dot{p}(k)$. If *p(k)* is not very significant and its change ($\dot{p}(k)$) is moving in an appropriate direction then we will not start the adaptation. This state indicates probably only an external error, which can be eliminated by the controller without any change. It seems to be advantageous in many cases if the change of *p(k)* is calculated from a longer time interval.

The performance measure *p(k)* is defined in this case as the difference between the optimal dumping ratio l_{opt} and the current l. If l is high then the control will be slow and with small oscillations. If l is low then the control will be fast but with high oscillations. The goal of the control is to minimize the performance measure, which is identical to the physical point of view, i.e. to stabilize the pitch angle f at a certain value.

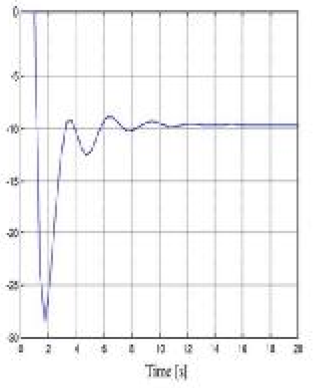
The results of some experiments are shown in fig. 8 and fig. 9. The main improvement is the minimization of oscillations so the dumping continuance form is almost aperiodic and the transition time remains the same. This fact enables a more comfortable flight as well as the increase of lifetime for mechanical parts of aircraft body (the most evidently in fig. 9).

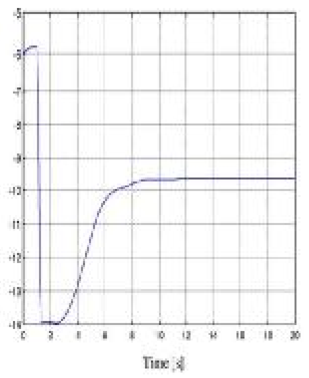




(a) (b)

Figure 8: Responses of state quantities for an aircraft with a PI controller (a) and with AFC (b): a - angle of attack; V - air velocity; q - pitch rate; f - pitch angle.





(a) (b)

Figure 9: Responses of actuator - elevator for an aircraft with a PI controller (a) and with AFC (b).

A very important quantity is the angle of attack *a* that influences the flight stability and also the control process. Relative improvements in minimizing oscillations were also obtained when the flight altitude was changed (as an external error). The not fully smooth control is the tax the adaptation is limited by our adaptation supervisor. Its partial improvement can be reached by increasing the sampling frequency but this way enhances the computational efforts very enormously. Therefore it is important to find the balance between computational complexity and control quality.

5 CONCLUSIONS

The principal advantage of both approaches is the substitution of a human expert in the establishment of the fundamental knowledge about the fuzzy controller, which is the most serious disadvantage of standard fuzzy systems. The designs presented enable fuzzy systems to be used for the control of systems characterized by great changes of parameters during their working. The second advantage is that we need to know only the dynamics of the controlled system and no input - output samples are necessary in advance. This fact enables the on-line adaptation and the set up of minimum parameters, which can lead to the decrease of the computational complexity. A hypothesis can be stated the rule-based implementation of SOFLC is more convenient for dynamical systems of higher order or with significant non-linearities but it has great demands on computational capacity.

References

[1] P. Busaba, A. Ohsato, “Proposal of convex cone method for nonlinear optimization problems with linear constraints”, Fuzzy workshop, Nagano, 25-26 October, 2001.

[2] W. L. Baker, J. A. Farrell, “An introduction to connectionist learning control system”, IN: D. A. White, D. A. Sofge (Eds.), Handbook of Intelligent Control-Neural, Fuzzy and Adaptive Approaches, van Nostrand Reinhold Inc., New York, 1992.

[3] H. Bersini, J.P. Nordvik, A. Bonarini, “A simple direct adaptive fuzzy controller derived from its neural equivalent”, see IEEE, 1993, pp. 345-350.

[4] O. Cordón, F. Gomide, F. Herrera, F. Hoffmann and L. Magdalena, “Ten years of genetic fuzzy systems: current framework and new trends”, Int. Journal Fuzzy Sets and Systems, Elsevier Publisher, Holland, N.1, Vol. 141, 2004, pp. 5-31.

[5] M. Gavalec, K. Mls, “Fuzzy Cognitive Maps and Decision Making Support”, Proc. of the 21th Int. Conf. Mathematical Methods in Economics, Prague, 2003, pp. 87-93.

[6] F. Guély, P. Siarry, “Gradient descent method for optimising various fuzzy rule bases”, see IEEE, 1993, pp. 1241-1246.

[7] C. J. Harris, C. G. Moore, “Intelligent identification and control for autonomous guided vehicles using adaptive fuzzy-based algorithms”, Engineering Applications of Artificial Intelligence 2, 1989, pp. 267-285.

[8] R. Jager, “Fuzzy Logic In Control” (PhD. thesis), Technical University of DELFT, Holland, 1995.

[9] Y. T. Kim, Z. Bien, “Robust self-learning fuzzy controller design for a class of nonlinear MIMO systems”, Int. Journal Fuzzy Sets and Systems, Elsevier Publisher, Holland, N. 2, Vol. 111, 2000, pp. 17-135.

[10] C.T. Lin, C.S.G. Lee, “Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent Systems”, Upper Saddle River, N.J.: PrenticeHall, 1996.

[11] M. Ma, Y. Zhang, G. Langholz, A. Kandel, „On direct construction of fuzzy systems”, Int. Journal Fuzzy Sets and Systems, Elsevier Publisher, Holland, N. 1, Vol. 112, 2000, pp. 165-171.

[12] T. J. Procyk, E. H. Mamdani, “A linguistic self-organizing process controller”, Automatica N. 15, 1979, pp. 15-30.

[13] Y. Shi, M. Mizumoto, "Some considerations on conventional neuro-fuzzy learning algorithms by gradient descent method", Int. Journal Fuzzy Sets and Systems, Elsevier Publisher, Holland, N. 1, Vol. 112, 2000, pp. 51-63.

[14] Várkonyi-Kóczy, A.R., T. Kovácsházy, O. Takács, and Cs. Benedecsik, "Anytime Evaluation of Regression-Type Algorithms", International Journal of Advanced Computational Intelligence, Vol. 5, No. 1, pp. 2-7, Feb. 2001.

[15] I. Zelinka, "Umělá inteligence v problémech globální optimalizace" (In Czech), Ben, Czech Republic, pp. 189, 2002.