Scientific Bulletin of the Petru Maior University of Tirgu Mures
Vol. 6 (XXIII), 2009
ISSN 1841-9267

Rule-bases construction through self-learning for a table-based Sugeno-Takagi fuzzy logic control system

C. Boldişor, V. Comnac, Member IEEE, S. Coman, A. Acreală
Transilvania University of Braşov, Automation Department
E-mail: cristian.boldisor@unitbv.ro

Abstract

A self-learning based methodology for building the rule-base of a fuzzy logic controller (FLC) is presented and verified, aiming to engage intelligent characteristics to a fuzzy logic control systems. The methodology is a simplified version of those presented in today literature. Some aspects are intentionally ignored since it rarely appears in control system engineering and a SISO process is considered here. The fuzzy inference system obtained is a table-based Sugeno-Takagi type. System's desired performance is defined by a reference model and rules are extracted from recorded data, after the correct control actions are learned.

The presented algorithm is tested in constructing the rule-base of a fuzzy controller for a DC drive application. System's performances and method's viability are analyzed.

1. Introduction

As often mentioned in literature today, there are at least four main sources for finding and/or fine tuning control rules of a fuzzy logic controller in a control application ([1], [2], [3], [6], [8]):

- Based on experience and/or control engineering knowledge;
- Based on an operator's control actions (which are recorded and properly processed);
- Based on a fuzzy model of the plant (if available) or fuzzy identification;
- Based on complex intelligent techniques such as self-learning algorithms and neural networks.

Practical implementations often use more than one of these sources in order to exploit their benefits and avoid usual difficulties. As an example, a self-learning system can be used for designing the rule-base of the FLC, which might be further improved by designer.

Intelligent techniques (term derived from *artificial intelligence*) are meant to extract fuzzy rules from an automatic process of recording and processing data that somehow imitates human reasoning. One strategy for building a rule-base is by using a self-learning algorithm. Shortly, the concept of *self-learning* control system design can be described as follows. By introducing a reference model and employing an iterative learning scheme, the desired control actions are progressively learned by operating the system repeatedly. At the same time, the rule-base is formed by observing, recording and properly processing the learned actions which are used subsequently ([3], [4]). No expert or process model would be necessary and multiple sources of errors are avoided (such as model identification errors).

It is important to note here that the self-learning process is similar to the learning process possessed by a human being [4].

2. Modified Sugeno-Takagi fuzzy reasoning

A self-learning system needs to record a number of variables in a finite interval of time and process the recorded values in order to extract knowledge.

It is considered very difficult to extract fuzzy rules from numerical data. Usually, this is because there is no clear relation between numbers (*quantitative data*) and the linguistic terms used by an expert (*qualitative data*). However, from an engineering control point of view, it is possible to use a rule-base with rules having less linguistic meaning, but having the *if-then* statement form. This leads to a little change in the usual fuzzy reasoning scheme ([7]): crisp values (which will be further named *target values*) instead of fuzzy sets are initially chosen over the universe of discourse for each variable, and a fuzzy set for *close to* linguistic term is used to engage a fuzzy inference mechanism (further described). The inference engine computes the output by comparing actual inputs with some already known values, for which have the correct

Keywords: *fuzzy logic controller, Sugeno-Takagi fuzzy reasoning, DC drive fuzzy control application*

outputs. This action is termed *pattern matching*, where patterns are the already known cases or target values.

Let us consider a simple case: an input variable x and an output variable y, with crisp target values x_j for input. We assume that for every input target value a correct, useful value y_j of output is known. Hence, for every input value x_j, we can enounce a *non-fuzzy* rule with the *if-then* form:

if $x = x_j$ *then* $y = y_j$

The robust control objective requires that a system would not be extremely sensitive to small changes of some parameters or measured variables. Hence, it is reasonable to enounce the following *fuzzy* rule:

if x *is close to* x_j *then* y *is close to* y_j

The *close to* term is usually defined by a fuzzy set usually defined by a triangular membership function around the target value. The parameters of the *close to* fuzzy set are chosen in order to fulfill completeness condition for a fuzzy rule-base. The *close to* fuzzy set actually produces the fuzziness and defines a norm to describe the distance between an actual value x^* and the targets x_j for which we know the correct action y_j. The action y^*, taken for x^*, will be influenced by this norm.

The small change in the fuzzy reasoning is not on the fuzzy inference itself, but on the way the input variables are treated: crisp values are chosen over the range, instead of fuzzy sets. Finally, the *close to* term produces fuzziness, but fuzzy sets are not clearly highlighted from the start (Figure 1): notice that no fuzzy sets are depicted.

Both the universe of discourse and the target values for every variable can be chosen based on the recorded data from a *learning* stage. This will increase the system's *learning* characteristic, which makes it more intelligent. In this case, a variable's range can be chosen depending on the maximum recorded value for that variable. Afterwards, the adequate target values are chosen over the range. It is worth mentioned here that the simplest uniform distribution is satisfactory, yet could have no relevance in some control applications. Better performances in steady-state conditions require a more detailed analysis of error values close to zero. Hence, we consider that target values for the input variables must cover the range but should be denser in the close to *zero* regions.

In conclusion, non-fuzzy relations between inputs and outputs, represented by recorded data, can be the basis for building fuzzy rules, and so for the modified reasoning scheme.

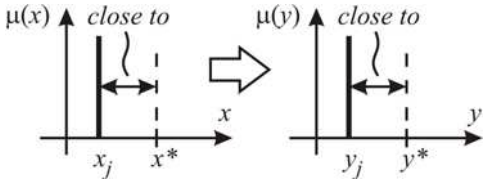

Fig. 1. A symbolic representation of the fuzzy inference mechanism.

3. Rule-base construction by self-learning

When it comes to real implementations, the main advantages of fuzzy control should be considered, along with performances, even during design stage ([1], [8]). An important and attractive characteristic of fuzzy control is that only little explicit knowledge regarding the process is needed while designing the controller. In other words, design should focus on building the rule-base and not on model identification. Hence, the main guidelines to be followed while designing the FLC would be ([3], [8]): (i) the control system should satisfy the desired performance and (ii) the required knowledge about the process should be kept as little as possible.

The block diagram of the self-learning system used in this paper is shown in Figure 2. The overall system is composed of four functional modules: the reference model, the learning algorithm, the rule-base formation mechanism and the controlled process

3.1. Learning algorithm

The concept of iterative self-learning was initially introduced in [4] and further treated from the theoretical viewpoint in a considerable amount of research papers. As its name implies, the correct control actions are learned and desired performance is progressively achieved by repeated trial in such a way that the modification of the present control is based on the error information obtained during previous trial.

A reference model $G_{ref}(s)$ is used to designate the desired performance, defined by time domain indices such as overshoot (σ), settling time (t_s) and steady-state error (e_{st}), or alternatively by desired pole position in the *s*-plane. The model can be a low-order linear one, with its parameters obtained from given performance indices. The output of the model, y_{ref}, represent the desired process output.

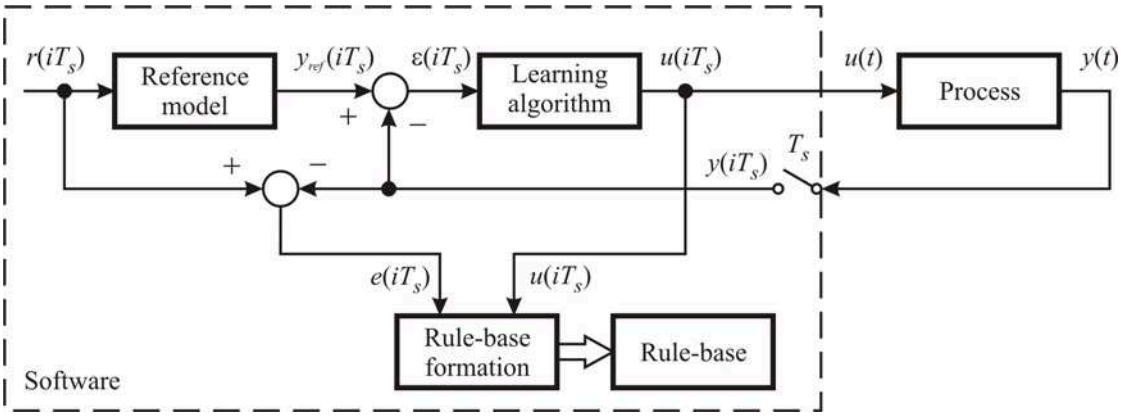

Fig. 2. Block diagram of the self-learning system.

The error information used to control the algorithm is the *learning error*, defined as:

$$\varepsilon_k(iT_s) = y_{ref}(iT_s) - y_k(iT_s), \quad (1)$$

where k specifies the current iteration number of the algorithm and $i = \overline{0, I}$ is the sample number of all signals recorded with the sampling time T_s. Notice that we have $I+1$ values for every recorded variable.

The learning algorithm is called *PID-type update law* or *error correction algorithm*, and is defined by:

$$u_k(iT_s) = u_{k-1}(iT_s) + g_k \varepsilon_{k-1}(iT_s) \quad (2)$$

where g_k is a learning gain for current iteration. The control output is adjusted at every iteration, such that the learning error asymptotically tends to zero, or a pre-specified small value, $\varepsilon_{\max}$. The algorithm stops (at iteration k) if:

$$\left\|\varepsilon_k(iT_s)\right\| = \int_{i=0}^{I} \left|\varepsilon_k(iT_s)\right| = \sum_{i=0}^{I} \left|\varepsilon_k(iT_s)\right| \le \varepsilon_{\max} \quad (3)$$

The convergence of the algorithm is proven in [3]. In the most simple case, learning gain is constant for every iteration, $g_k = g$. A varying gain is possible for a better convergence speed control.

3.2. Rule-base construction

In this paper, the proposed method is a simplified version for a SISO process. Some aspects were intentionally ignored since they can be easily avoided or rarely appear in control system engineering. A detailed version is presented in [3] where a modified fuzzy reasoning scheme introduced in [7] is used, as it better accommodates the numerical data set recorded.

Suppose that, at the K-th learning iteration, the correct control action $u_K(iT_s)$ is learned so that the desired output response specified by the reference model is achieved. At the same time, the *measured error* or *control error*, defined as

$$e_K(iT_s) = r(iT_s) - y_K(iT_s), \quad (4)$$

is recorded and we have two sets of data, one for control action and one for measured error, having $I+1$ values. From $e_K(iT_s)$ data set we can obtain the values for the *change-in-error*:

$$ce_K(iT_s) = e_K(iT_s) - e_K(iT_s - T_s) \quad (5)$$

These values correspond to derivative error, needed for a PI or PD-like fuzzy controller.

The three data vectors are organized into pairs:

$$\{e_K(iT_s); ce_K(iT_s)\} \sim \{u_K(iT_s)\}, \; i = \overline{0, I}. \quad (6)$$

where $\sim$ means *corresponding to*. Notice that the iteration number is no longer needed:

$$\{e_i; ce_i\} \sim \{u_i\} \quad (7a)$$

The present $I+1$ groups are derived from a positive step reference, or positive command action. If the process' output is symmetrical around zero when command action sign is reversed, expressed as:

$$u(iT_s) \to y(iT_s) \Rightarrow -u(iT_s) \to -y(iT_s)$$

(which is most likely to be true), then another I data pairs having the same absolute values but with opposite signs will be obtained:

$$\{-e_i; -ce_i\} \sim \{-u_i\} \quad (7b)$$

The $2I+1$ pairs will be rearranged and processed according to the values $\{e_i; ce_i\}$ in order to obtain J groups as follows.

Let us consider the range of error variable as

$$e \in \left[-E_{max}; +E_{max}\right] \; E_{\max} = GE \cdot \max\{e_i\},$$

where GE is an optional scaling factor, and a set of *target* values conveniently distributed over the range, with the step Δe, as:

$$\left\{e_{-J_e} = -E_{max}; \ldots; e_0 = 0; \ldots; e_{J_e} = +E_{max}\right\}, \quad (8a)$$

with $j_e = \overline{-J_e, J_e}$ ($2J_e + 1$ target values).

The same definitions are made for *change-in-error* variable:

$ce \in [-CE_{limit}; +CE_{limit}], CE_{\max} = GCE \cdot \max\{ce_i\}$,

where GCE is the optional scaling factor, and

$$ce \in \{ce_{-J_{ce}}; ...; ce_0 = 0; ...; ce_{J_{ce}}\}, \quad (8b)$$

with Δce step and $j_{ce} = \overline{-J_{ce}, J_{ce}}$ ($2J_{ce}+1$ target values). We consider all possible combinations of target values and we get $J = (2J_e + 1)(2J_{ce} + 1)$ *target* pairs $\{e_j; ce_j\}$.

For every target pair, we propose that the correct assumed control action value should be a medium of N_j values for command action $\{u_i\}$, as:

$$u_j = \frac{1}{N_j} \sum_{i=1}^{N_j} w_{ij} u_i \quad (9)$$

where

$$w_{ij} = \begin{cases} 1, & |e_i - e_j| \le \Delta e \text{ and } |ce_i - ce_j| \le \Delta ce \\ 0, & |e_i - e_j| > \Delta e \text{ or } |ce_i - ce_j| > \Delta ce \end{cases} \quad (10)$$

and N_j being the number of not null w_{ij} values. The weighting factor is reasonable since the pair $\{e_i; ce_i\}$ has a corresponding value u_i relevant to the proposed u_j if it is close enough to the target $\{e_j; ce_j\}$. More, the pairs that are not *close to* the target are ignored.

At this point, the J groups can be described as:

$$\{e_j; ce_j\} \sim \{u_j\}, \ j = \overline{1, J}, \quad (11)$$

and expressed as *non-fuzzy* rules:

if $e = e_j$ *and* $ce = ce_j$ *then* $u = u_j$

if e *is* $\{e_j\}$ *and* ce *is* $\{ce_j\}$ *then* u *is* $\{u_j\}$

As already proposed in section 2 of this paper, it is reasonable to enounce the *fuzzy* rules:

if e *is close to* $\{e_j\}$ *and* ce *is close to* $\{ce_j\}$ *then* u *is close to* $\{u_j\}$

The *close to* term is usually defined by a fuzzy set having a triangular membership function around target value, which is introduced for e_j as:

$$E_j : \mu_j^E\{e; e_j, \Delta e\} = \begin{cases} 1 - \dfrac{|e - e_j|}{\Delta e}, & |e - e_j| \ge \Delta e \\ 0, & |e - e_j| < \Delta e \end{cases} \quad (12a)$$

where Δe is the distance between two consecutively values in the discrete universe of discourse chosen before. It is worth mentioned here that the width of these fuzzy sets is $2\Delta e$, conveniently chosen to best fulfill the completeness requirement for the rule-base.

The same reasoning is used for the other input variable:

$$CE_j : \mu_j^{CE}\{ce; ce_j, \Delta ce\} = \begin{cases} 1 - \dfrac{|ce - ce_j|}{\Delta ce}, & |ce - ce_j| \ge \Delta ce \\ 0, & |ce - ce_j| < \Delta ce \end{cases} \quad (12b)$$

For command action, we can build triangular fuzzy set as $U_j : \mu_j^U\{u; u_j, \Delta u\}$ for Mamdami fuzzy rules, or we can keep the crisp values for Sugeno-Takagi fuzzy rules as $U_j = \{u_j\}$.

The rules obtained would be:

R_j : *if* e *is* E_j *and* ce *is* CE_j *then* u *is* U_j

(Mamdami), or

R_j : *if* e *is* E_j *and* ce *is* CE_j *then* $u = u_j$

(Sugeno-Takagi).

Several important aspects must be mentioned here. First, a scaling process is theoretically optional but might be required by the actual implementation of the fuzzy controller. The universe of discourse should best use the resources of the numerical device used for implementation. Second, the number of fuzzy rules depends on the number of target pairs. A large number of fuzzy rules imply a more complex and slower implementation, which can result in an unstable control system. Hence, a larger number of target values does not lead to better results, although it appears so.

Finally, an important observation must be noted. The procedure is meant to find the fuzzy rules of the controller. Choosing the learning gains, the scaling gains, the target values distribution and the reference model are still designer's task and his experience is most relevant.

4. Constructing a rule-base by self-learning for a DC drive fuzzy control application

The process subjected to the presented self learning fuzzy control system in our experiment is a DC drive. The main reason is that this application is simple and wide spread, and so it is easy to trustfully verify the algorithm by comparing the results with some already known. The DC drive model is not relevant, as this is one of the reasons for self-learning design strategy. Hence, the experiment does not include identification or parameter estimation stage.

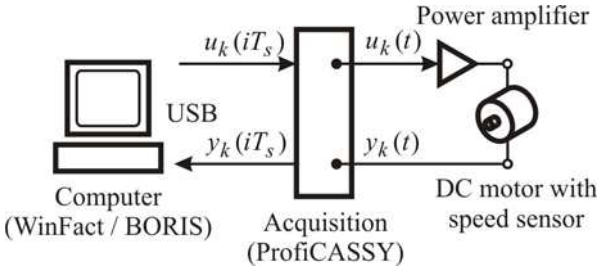

Fig. 3. Symbolic representation of the experiment.

The learning scheme (in Figure 2) is implemented using a software application (WinFact/BORIS) that enables both data acquisition and real-time processing (Figure 3).

The data acquisition is realized with a device connected to a computer through the USB port (ProfiCASSY). The drive is powered at 24[V] DC voltage, and it produces 3000[rpm] for a 10[V] DC command voltage applied on power amplifier. The speed sensor generates 1[V] signal for 1000[rpm] speed. An additional scaling factor of 10/3 is used to adjust sensor's voltage to the [-10; +10][V] reference range, so the maximum speed will correspond to the maximum command signal.

The following settings were chosen:

- reference signal is: $r(t) = 5 \cdot 1_+(t)$;
- reference model is a first order element with no time delay $G_{ref}(s) = K_{ref}/(T_{ref}s + 1)$, having $K_{ref} = 1$ and $T_{ref} = 1$;
- learning gain is constant and arbitrarily chosen $g_k = g = 1$ (further investigations about its effect on the learning speed and performance are to be done; this value was experimentally verified);
- learning error value to stop iterative learning algorithm is $\varepsilon_{\max} = 0.5$ (which can be replaced by a maximum iteration number k - [3]);
- scaling factors for the range of each variable, that multiplies the maximum recorded values are neglected, or GC, GCE, $GU = 1$, since recorded values are already scaled to [-10; +10][V] range;
- there are 7 target values for *error* variable and 3 for *change-in-error* variable, uniformly chosen over the universe of discourse;
- the sampling time for acquisition $T_s = 0.1[s]$.

With these settings, a self-learning stage was performed. The IAE criterion exponentially tends to zero, as expected (see Figure 4). It reaches the stop condition (3) at the 10th iteration ($K = 10$):

$$\left\|\varepsilon_k(iT_s)\right\|_{k=K=10} = 0.483.$$

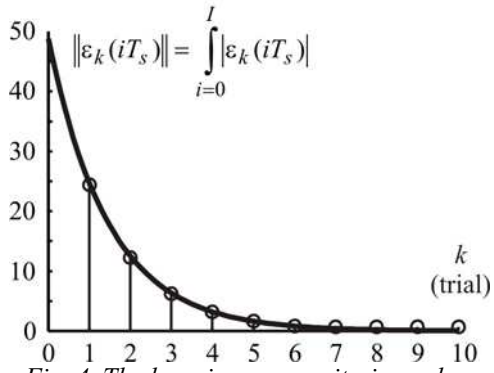

Fig. 4. The learning error criterion value.

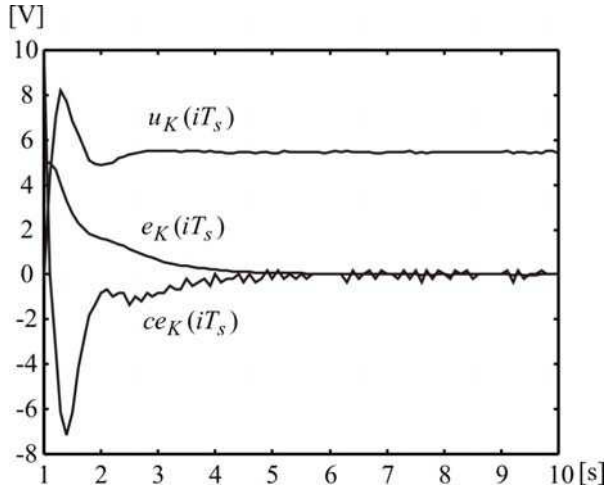

Fig. 5. Values for error, change in error and command when self-learning stops.

The now available values for error and command (Figure 5) are used to extract fuzzy rules, by running a custom made Matlab program. Notice in Figure 5 that only partial information is available since it is impossible to reach all possible combinations of target values (8a, 8b). With these data, an incomplete rule-base would be formed, that is not satisfactory. To avoid that, the rule-base construction stage has two steps. First it extracts fuzzy rules from available data (7a) and from the inverse values of them (7b). Second,

Table 1. The table of extracted fuzzy rules.

e \ *ce*	-10	0	10
-4.98	-8.1823	-6.9409	-6.5448
-3.32	-5.4411 *	-5.4411	-5.4411
-1.66	-0.0204	-2.3783	-5.3949
0	-0.0150	4.7808	0.0150
1.66	5.3949	5.2157	5.4628
3.32	5.4411	5.4411	5.4411 *
4.98	6.5448	6.9409	8.1823

it considers supplementary fuzzy rules so that the rules table will be symmetrical around the *zero* values of each input variable (or around the middle cell in the table). Rules are presented in Table 1, where the marked cells are filled in the second step (notice there are only two cells).

Subsequently, the rule-base was verified by using again WinFact/BORIS environment, which provides a powerful tool to run real-time tests and analyze fuzzy control systems. The results are satisfactory and very close to the reference model (see Figure 6):

- DC drive speed varies around reference (constant value corresponding to half the drive's maximum speed), within a reasonable ±1% stability range; steady-state error is zero;
- raising time is approximately equal to the value for the considered reference model (actually, we obtained a slightly lower value);
- as expected, no overshoot is recorded.

Conclusions

A simplified self-learning based methodology for building the rule-base of a fuzzy logic controller (FLC) was presented and verified, aiming to engage intelligent characteristics to a fuzzy logic control systems. The process subjected to control is a DC drive, a single input single output system that leads to the main simplification in the general algorithm. System's desired performance is defined by a reference model and rules are extracted from recorded data, after the correct control actions are learned. The DC drive is used because of the huge number of successful applications, which assures a reasonable and trustful verification of the presented algorithm.

A custom Matlab code was used to process recorded data and the fuzzy controller was tested in real-time by using again the WinFact/BORIS environment. The designed system has satisfactory behavior, that proves method's viability.

An important aspect to mention here is that design guidelines were followed and fuzzy control advantages were achieved. The control system has satisfactory performance and the controller was built without any information about the process (model, experience, parameters etc.).

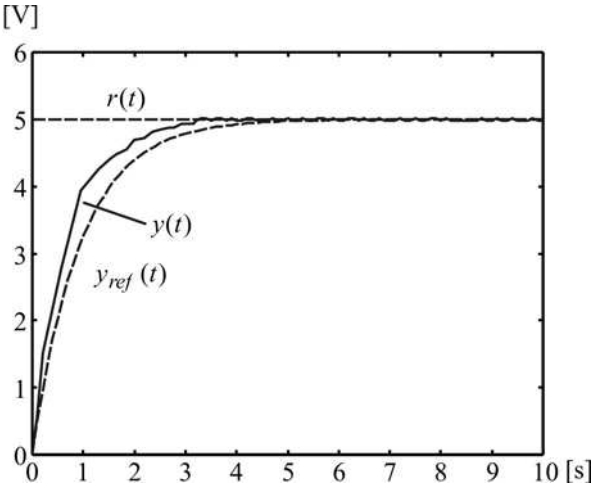

Fig. 6. The step response of the designed control system and the reference model.

References

[1] Jantzen, J., *Foundations of Fuzzy Control*, John Wiley & Sons, 2007.

[2] Jantzen, J., Østergaard, J., Verbruggen, H. "Fuzzy Control in the Process Industry: Common Practice and Challenging Perspectives", in Zimmermann, H.J. (ed.), *Practical Applications of Fuzzy Technologies*, Springer, 1999.

[3] Nie, J. and D. Linkens, *Fuzzy-Neural Control: Principles, Algorithms and Applications*, Prentice Hall, 1995.

[4] S. Arimoto; S. Kawamura; F. Miyazaki; S. Tamaki *Learning control theory for dynamical systems*, Proc. 24th IEEE Conf. on Decision and Control, 1985, pp 1375-1380.

[5] Moore, K.L. "Iterative learning control: An expository overview" in Datta, B.N. (ed.) *Applied and computational control, signals, and circuits*, Birkhauser, 1999.

[6] Sala, A., Guerrab, T.M., Babuška, R., Perspectives of fuzzy systems and control, *IEEE Fuzzy Sets and Systems* vol. 156 (3), 2005, pp. 432-444.

[7] Nie, J., "A class of new fuzzy control algorithms", *Proceedings of IEEE International Conference on Control and Applications*, Israel, April 3-6, 1999.

[8] Reznik, L. *Fuzzy Controllers*, Newnes, 1997.

[9] Tso, S.K., Fung, Y.H. Methodological development of fuzzy-logic controllers from multivariable linear control, *IEEE Transactions on Systems, Man, and Cybernetics*, vol.27 (3), 1997, 566–572.

[10] *** WinFACT 6 Manual, Ingenieurburo Dr. Kahlert.