Bulletin of the *Transilvania* University of Braşov • Vol. 3 (52) - 2010
Series I: Engineering Sciences

A SELF-LEARNING BASED FUZZY CONTROLLER FOR DC DRIVE CONTROL

C. BOLDIŞOR[1] V. COMNAC[1] I. ŢOPA[1] S. COMAN[1]

Abstract: *A self-learning based methodology for building the rule-base of a fuzzy logic controller (FLC) is presented and verified in a practical experiment. The methodology is a simplified version of those presented in available research papers. Some aspects are intentionally ignored as they rarely appear 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. Presented algorithm is tested for a DC drive control application.*

Key words: *fuzzy control, self-learning, DC drive control.*

1. Introduction

As often mentioned in current literature, there are at least four main sources for finding control rules of a fuzzy logic controller (FLC) [6], [11]: *i*) based on experience; *ii*) based on an operator's control actions properly recorded; *iii*) based on a fuzzy model of the plant; *iv*) 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, the self-learning algorithm can be used to obtain a rough rule-base of the FLC, which might be further improved by designer [1], [3-5], [7], [9].

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. The concept of self-learning control [2] implies a reference model and an iterative learning scheme, through which the desired control actions are progressively learned by operating the system repeatedly [2]. At the same time, the fuzzy rule-base is formed by observing, recording and properly processing the learned actions [1], [4], [8]. No expert or process model would be necessary and model identification errors are avoided.

2. Modified Sugeno-Takagi Fuzzy Reasoning

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

[1] Centre "Systems for Process Control", *Transilvania* University of Braşov.

(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 [8]: 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. The inference engine computes the output by comparing actual inputs with some already known values, for which have the correct 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 corresponding output value y_j is known. Hence, for every x_j, we can enounce a non-fuzzy rule with the if-then form:

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

Further, in order to increase the robustness of the control system, 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, with its parameters chosen to fulfil completeness condition for the 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', is influenced by this norm.

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.

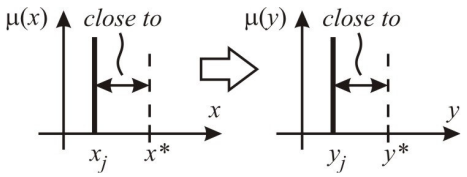

Fig. 1. *A symbolic representation of the fuzzy inference mechanism*

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.

3. Rule-Base Construction by Self-Learning

The 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: *i*) the control system should satisfy the desired performance and *ii*) the required knowledge about the process should be kept as little as possible [6].

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: reference model, learning algorithm, rule-base formation mechanism and the controlled process.

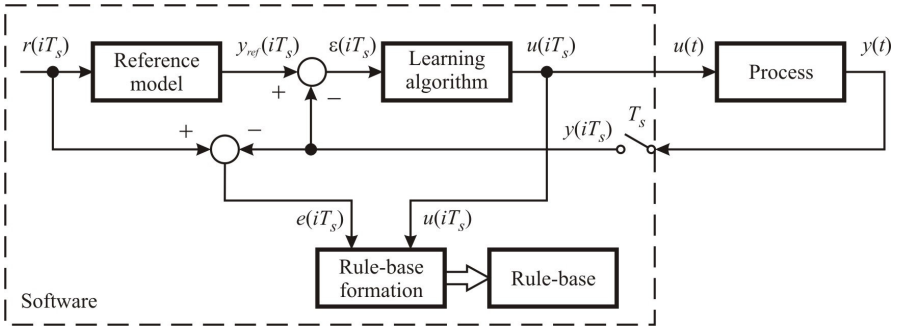

Fig. 2. *Block diagram of the self-learning system*

3.1. Learning Algorithm

The concept of iterative self-learning was initially introduced in [2]. As its name implies, the correct control actions are learned 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 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}, represents the desired process output.

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

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

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:

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

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, ε_{max}. The algorithm stops (at iteration k) if:

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

In the most simple case, learning gain is constant for every iteration, $g_k = g$.

3.2. Rule-Base Construction

The rule-base construction consists in a statistical data processing [1], [4], which is a simplified version for a SISO process. A detailed version for MIMO processes is presented in [7].

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:

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

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:

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

The three vectors are organized in pairs:

$$\{e_i; ce_i\} \sim \{u_i\}, \quad (6a)$$

where: ~ means "corresponding to". Notice that the iteration number is no longer needed.

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 then another I data pairs having the same absolute values but with opposite signs will be obtained:

$$\{-e_i; -ce_i\} \sim \{-u_i\}. \quad (6b)$$

For every input variable, target values are chosen uniformly distributed over symmetrical ranges, as presented in section 2. All possible J combinations of the target values will form target pairs $\{e_j; ce_j\}$, with $j = \overline{1, J}$. For every target pair, the corresponding value for the output variable is calculated from the recorded data pairs:

$$u_j = \frac{1}{N_j} \sum_{i=1}^{N_j} w_{ij} u_i, \quad (7)$$

$$w_{ij} = \begin{cases} 1, |e_i - e_j| \le \Delta e \text{ and} \\ \quad |ce_i - ce_j| \le \Delta ce, \\ 0, |e_i - e_j| > \Delta e \text{ or} \\ \quad |ce_i - ce_j| > \Delta ce, \end{cases} \quad (8)$$

and N_j being the number of not null w_{ij} values. Notice that only the recorded data pairs close enough to the target pairs are considered. With output values calculated in (7), non-fuzzy rules can be expressed:

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

and then it is possible to enounce the fuzzy rules:

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

The close to term is usually defined by a fuzzy set having a triangular membership function around target value and is preferably chosen.

The rules obtained would be:

if e is E_j and ce is CE_j then $u = u_j$.

Several important aspects must be mentioned here. First, 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. Second, the procedure is meant to find the fuzzy rules of the controller. Choosing learning gains, scaling gains, target values and reference model is still designer's task and his experience is most relevant.

4. Constructing a Rule-Base 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. This application is wide spread, and so it is easy to 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.

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

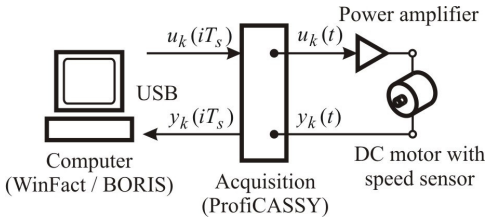

Fig. 3. *The experiment*

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 –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, having K_{ref} = 1 and T_{ref} = 1;
- learning gain is constant and arbitrarily chosen $g_k = g = 1$;
- learning error value to stop iterative learning algorithm is $\varepsilon_{max} = 0.5$;
- scaling factors for the range of each variable, that multiplies the maximum recorded values are neglected since recorded values are already scaled to –10…+10 V;
- there are 7 target values for error variable and 3 for change-in-error variable, uniformly chosen over their ranges;
- the sampling time is T_s = 0.1 s.

With these settings, an iterative self-learning stage was performed. It reaches the stop condition (3) at the 10th iteration (K = 10):

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

The now available values for error and command are used to extract fuzzy rules, by running a custom made Matlab program.

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 (6a) and from the inverse values of them (6b). Second, 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.

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

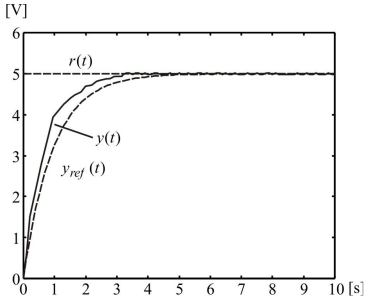

Fig. 4. *The step response of the designed control system and the reference model*

The results are satisfactory and very close to the reference model (see Figure 4): *i*) DC drive speed varies around reference (constant value corresponding to half the drive's maximum speed), within a reasonable $\pm 1\%$ stability range; steady-state error is zero; *ii*) raising time is approximately equal to the value for the considered reference model; *iii*) 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. 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. Fuzzy controller 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.).

References

1. Angelov, P.: *An Approach for Fuzzy Rule-Base Adaptation Using On-Line Clustering*. In: Int. J. of Approximate Reasoning **35** (2004) No. 3, p. 275-289.
2. Arimoto, S. Kawamura, S., Miyazaki, F., Tamaki, S.: *Learning Control Theory for Dynamical Systems.* In: Proc. 24th IEEE Conf. on Decision and Control, 1985, p. 1375-1380.
3. Ashrafa, S., et al.: *Self Learning Fuzzy Controllers Using Iterative Learning Tuner.* In: Digital Signal Processing **20** (2010) No. 1, p. 289-300.
4. Boldisor, C., Comnac, V., Coman, S.: *Rule-Bases Construction through Self-Learning for a Table-Based Sugeno-Takagi Fuzzy Logic Control System.* In: Proc. of the 4th Int. Conf. on Interdisciplinary in Engineering InterEng'09 Târgu Mureş, România, 12-13 November 2009, p. 167-172.
5. Dai, X.-F., et al.: *Some Key Issues in the Design of Self-Organizing Fuzzy Control Systems.* In: Lecture Notes in Computer Science **3972** (2006), p. 991-997.
6. Jantzen, J.: *Foundations of Fuzzy Control.* John Wiley & Sons, 2007.
7. Kim, Y.-T., Bien, Z.: *Robust Self-Learning Fuzzy Controller Design for a Class of Nonlinear MIMO Systems.* In: Fuzzy Sets and Systems **111** (2000) No. 2, p. 117-135.
8. Nie, J.: *A Class of New Fuzzy Control Algorithms*. In: Proceedings of IEEE International Conference on Control and Applications, Israel, April 3-6, 1999.
9. Phan, P.A., Gale, T.J.: *Direct Adaptive Fuzzy Control with a Self-Structuring Algorithm.* In: Fuzzy Sets and Systems **159** (2008) No. 8, p. 871-899.
10. Preitl, Ş., Precup, R.E., Korondi, P.: *Aspects Concerning the Development of Fuzzy Controllers for Servo Systems.* In: Proc. of 4th Int. Symp. of Hungarian Researchers on Comput. Intelligence, Budapest, Hungary, 2005, p. 89-100.
11. Sala, A., Guerrab, T.M., Babuška, R.: *Perspectives of Fuzzy Systems and Control*. In: IEEE Fuzzy Sets and Systems **156** (2005) No. 3, p. 432-444.
12. *** *WinFACT 6 Manual*. Ingenieurburo Dr. Kahlert.