Sensitivity-based self-learning fuzzy logic control for a servo system

Citation for published version (APA):

DOI:
10.1109/37.687619

Document status and date:
Published: 01/01/1998

Document Version:
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

- A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher’s website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license above, please follow below link for the End User Agreement:
www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:
openaccess@tue.nl
providing details and we will investigate your claim.
This article describes an experimental verification of a self-learning fuzzy logic controller (SLFLC). The SLFLC contains a learning algorithm that utilizes a second-order reference model and a sensitivity model related to the fuzzy controller parameters. The effectiveness of the proposed controller has been tested by experiment in the position control loop of a chopper-fed dc servo system in the presence of a gravity-dependent shaft load and fairly high static friction. The experimental results prove that the SLFLC provides closed-loop behavior as desired and eliminates a steady-state position error.

Introduction

Fuzzy control has lately emerged as one of the most successful nonlinear control techniques. Numerous successful applications of fuzzy control in industry have given this branch of intelligent control so much credit and respect in the control com-

Zdenko Kovacic, Mario Balenovic, Stjepan Bogdan
munity that nowadays a control engineer cannot neglect fuzzy control as a potential solution of his problem. As with any other control technique, fuzzy control has some drawbacks too. For example, a heuristic design of a fuzzy logic controller (FLC), despite its advantages, may be seen as a burden on the designer, requiring difficult and error-prone controller optimization. Specialized development tools can make the designer’s life easier, but the controller optimization still remains a demanding job.

The idea of automated FLC optimization or self-organization was first announced and elaborated by the group of Prof. E.H. Mamdani [1]. Since then, many researchers have attempted to improve performance of self-organizing mechanisms and establish a more systematic method for designing and tuning the fuzzy controller. Some of them have used neural nets [2], [3], some have preferred usage of a reference model [4], [5]. Self-organizing fuzzy logic controllers have been successfully applied to various control processes such as a pH-neutralization, inverted pendulum, robot drives, induction motor and many others [6]–[9].

The fuzzy model reference control is inspired by the concept of conventional model reference adaptive control [10], [11]. In fuzzy model reference adaptive control (FMRAC) schemes responses of controlled system variables are compared with responses of accompanying reference model variables and then the consequent model tracking errors are used as inputs to an adaptation mechanism which either may further perform a parameter [12], [13], or signal adaptation [14], [15]. The design of both types of FMRAC scheme requires a certain knowledge of system dynamics.

In fuzzy model reference learning control (FMRLC) schemes the model tracking error is fed into a learning mechanism, which establishes an organized iterative way of changing the core of the FLC. To make a distinction between iterative learning and iterative adaptation control schemes it must be emphasized that the proper work of fuzzy learning schemes does not depend on a mathematical model of a control process, though any knowledge about the control process may be helpful, especially with the determination of the reference model function or with the estimation of normalizing scaling factors of the FLC (related to the estimated dimensions of the universes of discourse). A fuzzy learning concept has been successfully developed and tested by simulation for cargo ship steering [16], showing a superior performance of the FMRLC scheme compared to several conventional MRAC schemes. The learning mechanism in [16] and [17] consists of two parts: a fuzzy inverse model and a knowledge base modifier. The fuzzy inverse model performs the function of mapping the model tracking error and the changes in the control effort. The knowledge base modifier enforces the changes of the knowledge base which should provide a required control effort. The structure of the fuzzy inverse model resembles the structure of a standard PD-type FLC whose parameters are synthesized (optimized) in a heuristic way relying on the rough knowledge of the inverse model of the plant. In [18], a comparative study of several control strategies that enhance the performance of the FMRLC has been worked out showing that so called “dynamically focused learning” may contribute to more effective solving of demanding control problems such as control of a magnetic ball suspension system.

The self-learning fuzzy logic controller (SLFLC), which utilizes a sensitivity model and a 2nd-order reference model, has been effectively introduced for control of nonlinear control systems [19], [20]. In order to determine a sensitivity function, a function describing a dependence of two variables (or variables and parameters) must be differentiable. As shown in [19], [20], there is a class of fuzzy controllers widely used in practical applications. They are usually called the singleton fuzzy controllers and can be organized to assume an analytical and differentiable form.

This article presents a design and experimental verification of an SLFLC organized by means of a model reference-based and a sensitivity model-based learning mechanism. In the presence of a proportional controller in parallel to the SLFLC, learning starts from a blank fuzzy rule-table, and proceeds after each run of the system by adding centroid increments to fuzzy output subsets of the activated fuzzy control rules. Up to this point, the proposed concept is very similar to the learning concept presented in [16]–[18]. The main difference to be found is in implementing the needed heuristic determination of a knowledge base modifier with a completely automatic procedure which does not involve any effort or experience on the part of the designer.

In what follows, first a brief description of the structure of a nonlinear positioning (astatic) fuzzy control system is given. Then the synthesis of a self-learning fuzzy controller is explained. We next show experimental results obtained in the position control loop of a chopper-fed dc servo system affected by friction and gravitation-dependent load variations. After concluding remarks we give some proposals for future research activities and we set some research goals.

Description of the Control Problem
Let us consider control of an unknown time-varying nonlinear astatic high-order SISO control process described as follows:

$$\begin{align*}
x' &= x, \\
\ldots &= \ldots, \\
x' &= f(x,t) + b(t)u + d(t), \\
y' &= x.
\end{align*}$$

(1)

where:

- $x$ - state vector,
- $u$ - control input,
- $y$ - system output,
- $f$ - unknown nonlinear time-varying function,
- $b > 0$ - unknown process gain,
- $d(t)$ - a measurement noise.
The output of the fuzzy controller used in the SLFLC (Fig. 1) scheme is computed according to a center of gravity principle. If all controller output subsets have the same area, then the output subsets, which are normally found in the consequent part of the fuzzy rules, can be substituted by their centroid. Such a controller, which can be seen as a special form of a Takagi-Sugeno controller [21], is very often called a singleton FLC. Specifically, the Takagi-Sugeno type of the FLC has fuzzy rules which map the antecedent fuzzy relations $R_i$ with the consequent parts containing an explicitly determined control input $u$:

$$\text{IF } R_i \text{ THEN } u = p_i(x_1, x_2, \ldots, x_n),$$

(2)

where $p_i$ is an arbitrary function, and $x_1, x_2, \ldots, x_n$ are crisp values of input variables.

If the function $p_i$ is linear,

$$p_i = a_0 + a_1 x_1 + a_2 x_2 + \ldots + a_n x_n,$$

(3)

and all coefficients except $a_0$ are equal to zero, then the fuzzy control rules of the Takagi-Sugeno FLC resemble the rules of a FLC with singleton fuzzy output sets:

$$\text{IF } R_i \text{ THEN } U = a_{0i},$$

(4)

How much the $j$-th fuzzy control rule will contribute to the defuzzified controller output depends on a degree of contribution described with the fuzzy basis function [19], [20]:

$$\phi_j(e(k), dy(k)) = \frac{\mu_j(e(k), dy(k))}{\sum_{i=1}^{r} \mu_i(e(k), dy(k))}, \quad j = 1, 2, \ldots, r,$$

(5)

where

- $e(k) = y(k) - y(k)$ - system error,
- $dy(k) = y(k) - y(k-1)$ - change of system output,
- $p$ - number of fuzzy subsets of $e(k)$,
- $q$ - number of fuzzy subsets of $dy(k)$,
- $r = pq$ - number of rules,
- $\mu_i$ - membership function of a fuzzy implication.

The SLFLC (Fig. 1) contains a nonintegral (PD) fuzzy controller to be organized and a proportional (P) controller:

$$u(k) = \Gamma[e(k), dy(k), \lambda] + k_p * e(k)$$

$$= \sum_{i=1}^{r} A_i \phi_i(e(k), dy(k)) + k_p e(k),$$

(6)

where:

- $A_i$ - centroid of the fuzzy controller output subset activated by the $i$-th fuzzy rule which corresponds to the coefficient $a_{0i}$ in (4),
- $\phi_i$ - fuzzy basis function of the $i$-th fuzzy rule,
- $k_p$ - proportional gain coefficient,
- $\lambda$ - fuzzy controller parameter vector.

The controller parameter vector $\lambda$ contains the following parameters: output subset centroid $c_i^e$, input subset centroid of $e$ and $dy$, $c_i^dy$, and widths of input subsets of $e$ and $dy$, $w_i^e$ and $w_i^dy$.

Regarding a determination of desired closed-loop dynamic behavior, a second-order reference model is used:

$$y_m(k) = a_{0m} + a_{1m} y_m(k-1) + a_{2m} y_m(k-2) + b_{0m} u(k) + b_{1m} u(k-1) + b_{2m} u(k-2).$$

(7)

The synthesis of the SLFLC described by (6) has a goal of finding a set of control rules which would enforce a control system to have a reference model tracking error as small as possible:

$$e_m(k) = y_m(k) - y(k)$$

(8)

A sensitivity model of a fuzzy controlled system can be built only if fuzzification and defuzzification operations have a form that allows a fuzzy input-output mapping function to assume an analytical and differentiable form. In this sense, an inference engine will utilize a product operator:

$$\mu_j(e(k), dy(k)) = \mu_j[e(k)] \mu_j[dy(k)]$$

(9)

and membership functions will also have a differentiable form:

$$\phi_j^e(x) = e^{-\frac{x - c_j^e}{w_j^e}}, \quad \phi_j^{dy}(x) = e^{-\frac{x - c_j^{dy}}{w_j^{dy}}},$$

(10)

where $c_j^e$ is a centroid of a membership function $\mu_j^e$, and $w_j^e$ is a width of a membership function $\mu_j^e$.

**Design of a Sensitivity-Based Self-Learning Fuzzy Controller**

Sensitivity functions can be very useful to express how much influence some variable or parameter has on the focused variable. In this sense, sensitivity functions represent information about interactions between causes and consequences which may be very useful for planning interventions in the system.

Having this concept in mind, a total differential of the system output $y(k)$ with respect to small variations of controller parameters ($k_p$=const.) is determined by

$$\Delta y(k) = \sum_{i=1}^{r} \frac{\partial y(k)}{\partial \lambda_i} \Delta \lambda_i = \sum_{i=1}^{r} \eta_{yi} \Delta \lambda_i,$$

(11)

where $\eta_{yi}$ are sensitivity functions related to parameters of the FLC:

$$\eta_{yi}(k) = \frac{\partial y(k)}{\partial \lambda_i} = \frac{\partial y(k)}{\partial \lambda_i} + \frac{\partial y(k)}{\partial \lambda_i} = \frac{\partial y(k)}{\partial \lambda_i} + \frac{\partial y(k)}{\partial \lambda_i}.$$  

(12)

A sensitivity function of the controller output with respect to controller parameter variations has the form

$$\frac{\partial u(k)}{\partial \lambda_i} = \left[\Gamma[e(k), dy(k), \lambda]\right].$$

(13)

Insertion of (13) into (12) and further in (11) yields:

$$\Delta y(k) = \sum_{i=1}^{r} \left[\frac{\partial}{\partial \lambda_i} \left(\Gamma[e(k), dy(k), \lambda]\right)\right] \Delta \lambda_i.$$
Equation (14) can be used for assessment of controller parameter variations that would provide a given change of the system output. In the proposed concept, system parameter variations will be compensated for only by modifying the centroid of the fuzzy output sets $A_i$ (input membership functions remain predefined). Therefore,

$$\frac{\partial a(k)}{\partial A_i} = \frac{\partial a(k)}{\partial A_i} = \frac{\partial}{\partial A_i} \{G[e(k),d_y(k),\lambda]\}. \tag{15}$$

From (5) and (6) it follows:

$$\frac{\partial}{\partial A_i} \{G[e(k),d_y(k),\lambda]\} = \phi_i(e(k),d_y(k)) = \sum_{i=1}^{m} \mu_i(e(k),d_y(k)). \tag{16}$$

Accordingly, equation (12) assumes a form:

$$\eta_i(k) = G_p \phi_i(e(k),d_y(k)). \tag{17}$$

It may be seen that sensitivity functions related to the output centroid $A_i$ depend on the dynamic characteristics of the control process $G_p$. Since the exact model of the control process (1) is unknown (only a static process gain is directly identifiable), some approximation, denoted as $G_{pa}$, must be used. Among many possible linear process approximations, the one determined by the dynamics of the reference model (7) could be assumed to be a reasonable choice. If such an approximation is adopted, then the sensitivity functions described by (17) have a form (for simplicity, $\phi_i(e(k),d_y(k))$ is replaced by $\phi_i(k)$):

$$\eta_i(k) = a_{i0} \eta_i(k-1) + a_{i1} \eta_i(k-2) \tag{18}$$

In general, selected approximations of the control process (1) may have different dynamic characteristics, which would finally result in different dynamic characteristics of the sensitivity functions (18). If such sensitivity functions are being used in the SLFLC algorithm, then the process approximation $G_{pa}$ would act like a filter whose dynamic behavior determines a speed of learning and thus directly affects the final form of the FLC. In other words, after completion of the learning process, different variants of the FLC will be obtained for different process approximations.

Let us make the following assumptions: the static part of the control process is inherently stable, the boundary values (limits) of the system output $y(k)$ and the tracking error $e_m(k)$ are known, the reference input $u_r(k)$ is imposed as a sequence of alternating step changes. The given change of the system output $\Delta y(k)$ coincides in the model reference control concept with a tracking error $e_m$. Having in mind that parameter variations will be compensated for by modifying centroid $A_i$, relation (11) attains the following form:

$$\Delta \eta_i(k) = a_{i0} \Delta \eta_i(k-1) + a_{i1} \Delta \eta_i(k-2) \tag{19}$$
In this case, a learning algorithm has a form where \( \Delta A_i \) denotes a current learning iteration (i.e., a current run of the system). The meaning of this is that centroid \( A_i \) is changed only once during each run of the system. A new run of the system starts with a new change of \( u_i(k) \). It must be noted that approximate equality in (14) is intentionally replaced by normal equality in (19) and (20).

The sensitivity function equation (18) holds for every sampling interval, while the modification of the centroid \( A_i \) is executed only once in the learning iteration. Therefore, it is necessary to pick the moment when the influence of a particular controller parameter is highest, and that is in the maximum of the corresponding sensitivity function.

\[
\beta_i = \eta_i^s(v_i^s) = \max(\eta_i^s(k))
\]

(21)

On insertion of (21) into (20), the learning law assumes a form

\[
\Delta A_i^s = \frac{\sum_{i=1}^{N} \eta_i^s(v_i^s) \times \Delta A_i^s}{\beta_i^s},
\]

(20)

where \( \kappa \) denotes a current learning iteration (i.e., a current run of the system). The meaning of this is that centroid \( A_i \) is changed only once during each run of the system. A new run of the system starts with a new change of \( u_i(k) \). It must be noted that approximate equality in (14) is intentionally replaced by normal equality in (19) and (20).

The sensitivity function equation (18) holds for every sampling interval, while the modification of the centroid \( A_i \) is executed only once in the learning iteration. Therefore, it is necessary to pick the moment when the influence of a particular controller parameter is highest, and that is in the maximum of the corresponding sensitivity function.

where \( \kappa \) denotes a current learning iteration (i.e., a current run of the system). The meaning of this is that centroid \( A_i \) is changed only once during each run of the system. A new run of the system starts with a new change of \( u_i(k) \). It must be noted that approximate equality in (14) is intentionally replaced by normal equality in (19) and (20).

The sensitivity function equation (18) holds for every sampling interval, while the modification of the centroid \( A_i \) is executed only once in the learning iteration. Therefore, it is necessary to pick the moment when the influence of a particular controller parameter is highest, and that is in the maximum of the corresponding sensitivity function.

where \( \kappa \) denotes a current learning iteration (i.e., a current run of the system). The meaning of this is that centroid \( A_i \) is changed only once during each run of the system. A new run of the system starts with a new change of \( u_i(k) \). It must be noted that approximate equality in (14) is intentionally replaced by normal equality in (19) and (20).

The sensitivity function equation (18) holds for every sampling interval, while the modification of the centroid \( A_i \) is executed only once in the learning iteration. Therefore, it is necessary to pick the moment when the influence of a particular controller parameter is highest, and that is in the maximum of the corresponding sensitivity function.

where \( \kappa \) denotes a current learning iteration (i.e., a current run of the system). The meaning of this is that centroid \( A_i \) is changed only once during each run of the system. A new run of the system starts with a new change of \( u_i(k) \). It must be noted that approximate equality in (14) is intentionally replaced by normal equality in (19) and (20).

The sensitivity function equation (18) holds for every sampling interval, while the modification of the centroid \( A_i \) is executed only once in the learning iteration. Therefore, it is necessary to pick the moment when the influence of a particular controller parameter is highest, and that is in the maximum of the corresponding sensitivity function.

where \( \kappa \) denotes a current learning iteration (i.e., a current run of the system). The meaning of this is that centroid \( A_i \) is changed only once during each run of the system. A new run of the system starts with a new change of \( u_i(k) \). It must be noted that approximate equality in (14) is intentionally replaced by normal equality in (19) and (20).

The sensitivity function equation (18) holds for every sampling interval, while the modification of the centroid \( A_i \) is executed only once in the learning iteration. Therefore, it is necessary to pick the moment when the influence of a particular controller parameter is highest, and that is in the maximum of the corresponding sensitivity function.

where \( \kappa \) denotes a current learning iteration (i.e., a current run of the system). The meaning of this is that centroid \( A_i \) is changed only once during each run of the system. A new run of the system starts with a new change of \( u_i(k) \). It must be noted that approximate equality in (14) is intentionally replaced by normal equality in (19) and (20).

The sensitivity function equation (18) holds for every sampling interval, while the modification of the centroid \( A_i \) is executed only once in the learning iteration. Therefore, it is necessary to pick the moment when the influence of a particular controller parameter is highest, and that is in the maximum of the corresponding sensitivity function.

where \( \kappa \) denotes a current learning iteration (i.e., a current run of the system). The meaning of this is that centroid \( A_i \) is changed only once during each run of the system. A new run of the system starts with a new change of \( u_i(k) \). It must be noted that approximate equality in (14) is intentionally replaced by normal equality in (19) and (20).

The sensitivity function equation (18) holds for every sampling interval, while the modification of the centroid \( A_i \) is executed only once in the learning iteration. Therefore, it is necessary to pick the moment when the influence of a particular controller parameter is highest, and that is in the maximum of the corresponding sensitivity function.
Fig. 6. Start of learning; full load weight; $\theta_0=0^\circ$.

Fig. 7. End of learning; full load weight; $\theta_0=0^\circ$. 
Fig. 8. Start of learning; reduced load weight; $\theta_0=0^\circ$.

Fig. 9. End of learning; reduced load weight; $\theta_0=0^\circ$. 

June 1998
Fig. 10. Start of learning; full load weight; $\theta_0=90^\circ$.

Fig. 11. End of learning; full load weight; $\theta_0=90^\circ$. 
Fig. 12. Start of learning; full load weight; faster reference model.

Fig. 13. End of learning; full load weight; faster reference model.
conditions in the operating point \( \theta_0 = 0^\circ \). The steady state error and varying dynamics in different directions, caused by friction and load, are clearly seen, and the tracking error has exceeded 25\% of the imposed change of the reference input.

Fig. 9 shows the same group of responses obtained after twelve learning iterations (positive runs) of the system. The system follows the reference model very closely and the maximum tracking error value is kept below 10\%, while the steady-state system error is kept at zero as required. As may be seen, the controller output has also a very acceptable nonoscillatory form.

Fig. 10 shows the same group of responses obtained after first two runs of the system under full load weight conditions in the operating point \( \theta_0 = 90^\circ \). This is the most difficult starting operating point which results in the highest tracking error (over 75\% due to the presence of active load which cannot be compensated only by means of a P controller).

Fig. 11 shows the same group of responses obtained after completion of learning (slightly longer, i.e. after fourteen iterations). The system follows the reference model very closely and the maximum tracking error value is kept below 5\%, while the steady-state system error is kept at zero as required. The controller output has retained a very acceptable nonoscillatory form.

More experiments with changed reference model parameters \( t_m = 0.4s; 0.6s; 0.8s \) have indicated that the choice of reference model parameters is not a critical issue, since it has not greatly influenced the convergence of the learning process (as shown in Figs. 12 and 13 for the case \( t_m = 0.4s \)). We have also tried to increase robustness by adding an integral term in parallel to the P controller and FLC, but responses obtained in the presence of the integral term did not differ much from those obtained without it. Specifically, the integral term was activated conditionally (i.e., only if it was necessary) after the major part of the learning process was completed.

Since the selection of the process approximation \( G_p \) affects the dynamics of the sensitivity functions, we have investigated the influence of zero-order, first-order, second-order and third-order linear process approximations on the performance of the learning mechanism, while keeping the second-order reference model unchanged. It must be emphasized that all experiments finished successfully indicating a stable convergence of the learning process and providing satisfactory results of model tracking control.

Conclusions

We have presented a sensitivity-based self-learning fuzzy logic control (SLFLC) scheme suitable for control of astatic systems. In the presence of a P controller in parallel to the FLC, learning starts from a blank fuzzy rule-table, and continues after each run of the system by adding new centroid increments to fuzzy output subsets of the activated fuzzy control rules. A learning law is based on the use of a reference model and a sensitivity model built with respect to the fuzzy controller parameters. Thanks to the selection of analytical and differentiable fuzzification and defuzzification operators, the learning law assumes an analytical form. This enables a completely automated modification of the fuzzy control surface reducing the need for the designer's engagement in the setup of learning mechanism. Learning iterations coincide with successive runs of the closed-loop system, and adjustment of the fuzzy control surface is performed only at the end of the learning iteration. A similar structure of the SLFLC with feedforward and integral elements in parallel to the FLC can be used for control of static systems, as described in [19].

Experimental results obtained in the nonlinear position control loop of a dc servo motor drive have confirmed a fairly fast convergence of learning and effective control without steady-state errors and with very close following of the reference model response. Due to a marked simplicity, the proposed SLFLC scheme could be recommended for many practical nonlinear control applications.

Although experimental results have confirmed the closed-loop stability, this issue remains as one of the most important to be solved in the further research in order to establish stability criteria which would enable the synthesis of the SLFLC.

The proposed SLFLC covers the constrained operating range around the selected operating point. These constraints are directly determined by the quantitative dimensions of the FLC input universes of discourse. Therefore, one of the future research goals would be a systematic investigation of strategies for covering the whole operating range.

The obtained experimental results demonstrated the ability of the SLFLC to track the reference model and simultaneously compensate for operating point-dependent variations of the shaft load. On the other hand, it would be interesting to investigate a strategy of learning for systems exposed to external disturbances which do not allow any change of the reference input (usually called stabilization systems). If such a strategy were successfully found, it could be combined with the proposed learning concept to cover a wider spectrum of potential applications.

References


