Proceedings of the 8th IEEE Mediterranean Conference
on Control and Automation (MED 2000)
Rio, Patras, GREECE 17-19 July, 2000

Demonstration of Self-learning Fuzzy Logic Controller Performance in the Matlab+Simulink™ Environment

Z. Kovačić, Member, IEEE, S. Bogdan, Member, IEEE, T. Reichenbach

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, CROATIA
e-mail: zdenko.kovacic@fer.hr, url: http//www.rasip.fer.hr/flrcg

Abstract In this paper, we demonstrate a performance of a PD type self-learning fuzzy logic controller (SLFLC), which has been implemented as a function block for the very popular Matlab+Simulink environment. The SLFLC described in detail in [1] utilizes a reference model and a sensitivity model for learning of SLFLC parameters. The effectiveness of the SLFLC function block has been demonstrated on the model of a closed-loop engine speed control system provided in Matlab for demo purposes. The results show very clearly how the SLFLC brings improved performance into the selected control system example.

1. INTRODUCTION

The usage of powerful software packages for modeling, simulation and optimization of control systems has become a commonplace activity. Regardless whether it is a word about standard control solutions or about starting new research, new control jobs usually begin with identification of the control problem and derivation of acceptable mathematical model. Then follows consideration of potentially efficacious solutions, and providing that all previous steps have been made successfully, follows final validation by intensive simulations and experimental work.

One of advanced features of modern simulation tools is that they allow users to generate a real-time executable code directly from the simulation model. This code can be further downloaded into dedicated hardware systems. This enables fast validation and shortens product development time. In addition, by introducing new control solutions in the form of ready-to-use function blocks prepared for popular software packages such as the Matlab+Simulink, they can be brought to a wide academic and industrial community in a very convincing and representative way.

A Matlab+Simulink function block presented in this paper is a PD type self-organizing fuzzy logic controller (SLFLC) whose learning algorithm utilizes a sensitivity model and a 2nd-order reference model. The theoretical background of the SLFLC has been described in detail in [1-5] where it has been effectively used for control of different static and astatic nonlinear systems.

The paper describes the SLFLC structure and basic expressions necessary for understanding the learning algorithm. The usage of the sensitivity model presumes a differentiable character of the fuzzy controller input-output mapping function, which becomes a feasible goal within the classes of fuzzy controllers available in the Matlab Fuzzy Logic Toolbox [6]. The emphasis in the paper has been put on the description of function block parameters and the way that it is used.

The SLFLC function block has been tested on the model of a nonlinear engine speed control system that is contained in the Matlab library of models for demo purposes. The results obtained have demonstrated the simplicity of using and high ability of the SLFLC function block to raise the overall system performance.

2. DESCRIPTION OF THE CONTROL PROBLEM

The SLFLC function block is intended to control an unknown time-varying nonlinear static high-order SISO control process described as follows:

$$y(t) = f\left[y^{(n)}(t),\ldots,y'(t),y,u^{(m)}(t),\ldots,u'(t),u(t),t\right]+v(t) \quad (1)$$

where ν(t) denotes a measurement noise.

The basic structure of the SLFLC described in [1] contains a nonintegral (PD-type) fuzzy controller, a feedforward control element and a P controller (Fig.1):

$$u(k) = \Gamma[e(k), dy(k), \lambda] + \xi k_f \cdot u_r(k) + (1-\xi) k_p e(k) = \quad (2)$$
$$= \sum_{i=1}^{r} A_i \cdot \phi_i[e(k), dy(k)] + \xi k_f \cdot u_r(k) + (1-\xi) k_p e(k)$$

where: e(k)=u_r(k)-y(k) - system error, dy(k)=y(k)-y(k-1) - change of system output, A_i - centroid of the fuzzy controller output subset activated by the i-th fuzzy rule, k_f - feedforward gain coefficient, k_p - P controller gain, > - type of system (1- static, 0 - astatic), 8 - fuzzy controller parameter vector and N_i - fuzzy basis function describing the degree of contribution of the i-th fuzzy rule [1]:

$$\phi_j[e(k), dy(k)] = \frac{\mu_a^e[e(k)] \cdot \mu_b^{dy}[dy(k)]}{\sum_{m=1}^{p}\sum_{n=1}^{q} \mu_e^m[e(k)] \cdot \mu_{dy}^n[dy(k)]} = \frac{\mu_j[e(k), dy(k)]}{\sum_{i=1}^{r} \mu_i[e(k), dy(k)]} \quad (3)$$

where: p - number of fuzzy subsets of e(k), q - number of fuzzy subsets of dy(k), r = pq - number of rules, μ_i - membership function of a fuzzy implication.

Due to its PD character, such a controller will not compensate a steady-state error in the presence of external disturbance, which has not been present during learning. In order to handle this problem, an integral term u_i(k) may be added, as elaborated in [7].

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

$$y_M(k) = a_{M1} \cdot y_M(k-1) + a_{M2} \cdot y_M(k-2) + b_{M1} \cdot u_r(k-1) + b_{M2} \cdot u_r(k-2) \quad (4)$$

The synthesis of the SLFLC described by (2) has a goal to find a set of control rules which would keep a reference model tracking error as small as possible:

$$e_M(k) = y_M(k) - y(k) \quad (5)$$

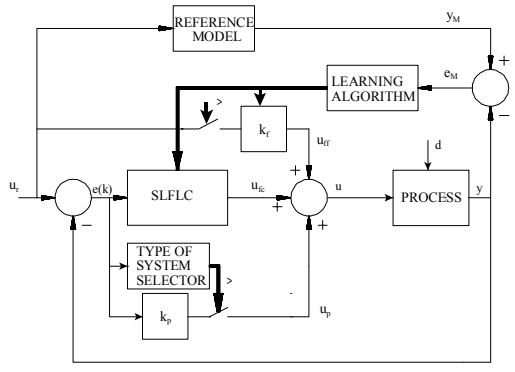

Fig. 1. The structure of the SLFLC.

3. SYNTHESIS OF A SENSITIVITY MODEL

A sensitivity model of a fuzzy controlled system can be built only if fuzzification and defuzzification operations lead to the differentiable form of a fuzzy input-output mapping function. In this sense, the inference engine of the SLFLC will utilize *a product operator* and input membership functions of the *differentiable gaussian* form:

$$\mu^{e}{}_{k} = e^{-\frac{(x-c_k^e)^2}{2(w_k^e)^2}} \qquad \mu^{dy}{}_{k} = e^{-\frac{(x-c_k^{dy})^2}{2(w_k^{dy})^2}} \quad (6)$$

where c_i^j is a centroid of a membership function μ_i^j, and w_i^j is a width of a membership function μ_i^j.

A total differential of the system output y(k) w.r.t. small variations of controller parameters can be determined as:

$$\Delta y(k) = \sum_{i=1}^{N} \frac{\partial y(k)}{\partial \lambda_i} \cdot \Delta\lambda_i + \xi \frac{\partial y(k)}{\partial k_f} \cdot \Delta k_f \quad (7)$$
$$= \sum_{i=1}^{N} \eta_{\lambda i} \cdot \Delta\lambda_i + \xi \eta_{kf} \cdot \Delta k_f$$

where > is a boolean type parameter (>=1 denotes a static type of system). Sensitivity functions related to the parameters of the fuzzy controller and feedforward term, respectively, have a form

$$\eta_{\lambda_i}(k) = \frac{\partial y(k)}{\partial \lambda_i} = \frac{\partial y(k)}{\partial u(k)} \cdot \frac{\partial u(k)}{\partial \lambda_i} \quad (8)$$

$$\eta_{kf}(k) = \frac{\partial y(k)}{\partial k_f} = \frac{\partial y(k)}{\partial u(k)} \cdot \frac{\partial u(k)}{\partial k_f} \quad (9)$$

It may be seen that transient behavior of all sensitivity functions depends on the dynamic characteristics of the control process, G_p=My(k)/Mu(k). Since the exact model of the control process (1) is unknown, among many possible process approximations, a transfer function proportional to the reference model (4), $G_{sm}(z) = K_s G_m(z)$, could be assumed for a reasonable process approximation.

It must be noted that parameters of the fuzzy controller and the feedforward term do not depend on each other. Therefore, a sensitivity of the controller output to the all SLFLC parameter variations has the following form

$$\frac{\partial u(k)}{\partial k_f} = \frac{\partial}{\partial k_f}\{k_f \cdot u_r(k)\} = u_r(k) \quad (10)$$

$$\frac{\partial u(k)}{\partial \lambda_i} = \frac{\partial}{\partial \lambda_i}\{\Gamma[e(k), dy(k), \lambda]\} \quad (11)$$

Insertion of (8)-(11) into (7) yields

$$\Delta y(k) \approx \sum_{i=1}^{N} G_{sm} \left\{ \frac{\partial}{\partial \lambda_i}\{\Gamma[e(k), dy(k), \lambda]\} \right\} \cdot \Delta\lambda_i + \xi G_{sm} u_r(k) \Delta k_f \quad (12)$$

4. SYNTHESIS OF THE LEARNING ALGORITHM

Equation (12) can be used for assessment of controller parameter variations that would provide a given change of the system output) y(k). In the model reference control) y(k) coincides with a tracking error $e_M(k)$.
In the SLFLC function block, system parameter variations will be compensated by adaptation of the centroids of the fuzzy output sets A_i and the feedforward gain coefficient k_f by using the learning mechanism. Referring from (2) and (3), it follows:

$$\frac{\partial u(k)}{\partial \lambda_i} = \frac{\partial u(k)}{\partial A_i} = \frac{\partial}{\partial A_i}\{\Gamma[e(k), dy(k), \lambda]\} = \phi_i[e(k), dy(k)] \tag{13}$$

By choosing the reference model (4) for the process approximation G_{sm} and referring from (13), the sensitivity functions (9) have a form (for a simplicity, $N_i[e(k), dy(k)]$ is replaced by $N_i(k)$):

$$\eta_{A_i}(k) = a_{M1}\cdot\eta_{A_i}(k-1) + a_{M2}\cdot\eta_{A_i}(k-2) + b_{M1}\cdot\phi_{A_i}(k-1) + b_{M1}\cdot\phi_{A_i}(k-2) \tag{14}$$

$$\eta_{kf}(k) = a_{M1}\cdot\eta_{kf}(k-1) + a_{M2}\cdot\eta_{kf}(k-2) + b_{M1}\cdot u_r(k-1) + b_{M1}\cdot u_r(k-2) \tag{15}$$

Let us make the following assumptions: 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, that must be persistent during the learning process. In this case, the learning algorithm has a form

$$\Delta A_j^{\kappa}(k) = \frac{e_M^{\kappa}(k) - \sum_{i=1, i\neq j}^{r}\eta_{A_i}^{\kappa}\cdot\Delta A_i^{\kappa} - \eta_{kf}^{\kappa}(k)\cdot\Delta k_f^{\kappa}}{\eta_{A_j}^{\kappa}(k)} \tag{16}$$

where 6 denotes a current learning iteration (i.e. a current run of the system). The meaning of this is that centroid A_j are changed only once during each run of the system. A new run of the system starts with a new change of $u_r(k)$.

Sensitivity functions (15) are calculated in each control interval, while centroid modifications are executed only once during a learning iteration. This is done in the moment when influence of a particular controller parameter is the highest, and that is in the maximum of the corresponding sensitivity function.

A modification of the feedforward gain coefficient k_f is always performed after modification of all centroid values in the current (6-th) learning interval is completed:

$$\Delta k_f^{\kappa}(k) = \frac{e_M^{\kappa}(k) - \sum_{i=1}^{r}\eta_{A_i}^{\kappa}(k)\cdot\Delta A_i^{\kappa}}{\eta_{kf}^{\kappa}(k)} \tag{17}$$

During a learning interval, it may happen that some sensitivity function reaches a maximum value lower than a predetermined threshold value. This means that the corresponding centroid has a negligible influence on the process behavior, and so remains unchanged.

5. THE SLFLC FUNCTION BLOCK

The SLFLC function block is written in C++ language as a CMEX S-function that is stored in final form as the slflc.dll file. The block has a mask in the form of standard dialog box (Fig. 2) which enables user to set several parameters that influence block operation.

These parameters are:

FISMATRIX (type: <name>)
User defines a name of the fismatrix. The fismatrix has a form of a standard Fuzzy Logic Toolbox v.2.0 fismatrix.

Number of Iterations (type: <real>)
User defines a maximum number of learning iterations. Value = 0 means that learning is disabled.

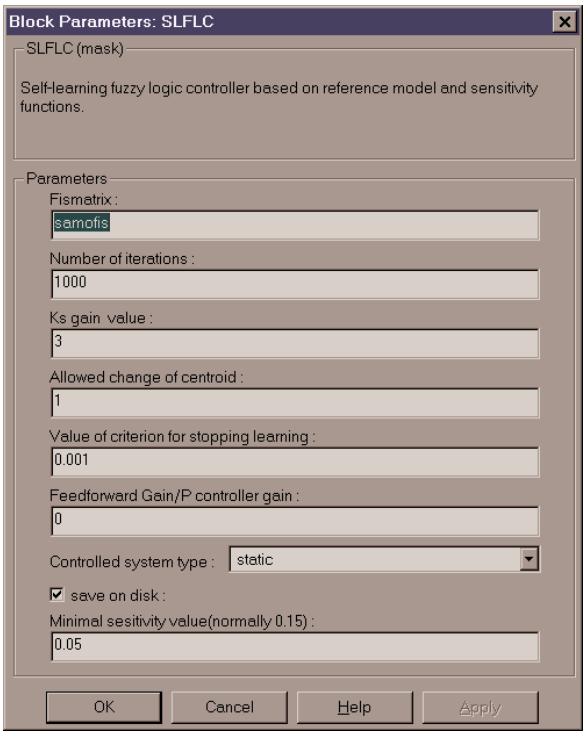

Fig. 2. Mask of the SLFLC function block for Matlab.

K_s gain value (type: <real>)
User defines a gain determining the relation between the reference model and the process approximation. Larger K_s will make learning slower, but smoother and more stable.

Allowed change of centroid (type: <real>=lim)
By defining the value of lim according to the following law:

$$\alpha = e^{(\lim\cdot|e_M|)} \tag{18}$$

user defines constraints on the centroid changes in one learning iteration where " denotes the value which divides calculated changes of the centroid. Larger "

makes learning slower. This parameter is partly supplementary to the parameter K_s.

User defines the threshold value of the IAE criterion

$$E = \sum_{i}^{i+1} |e| \qquad (19)$$

where i denotes start of learning instant, while i+1 denotes the end of learning instant. This criterion is recalculated in each learning iteration.

Controlled system type (Type: <options dialog>)
User may choose between two options: (i) static or (ii) astatic. This reflects the interpretation of the preceding parameter (K_f or K_p).

Save on disk (Type: <check-box>)
By checking out this option, user enables creation of files on disk which store particularly interesting results of simulation for subsequent analysis.

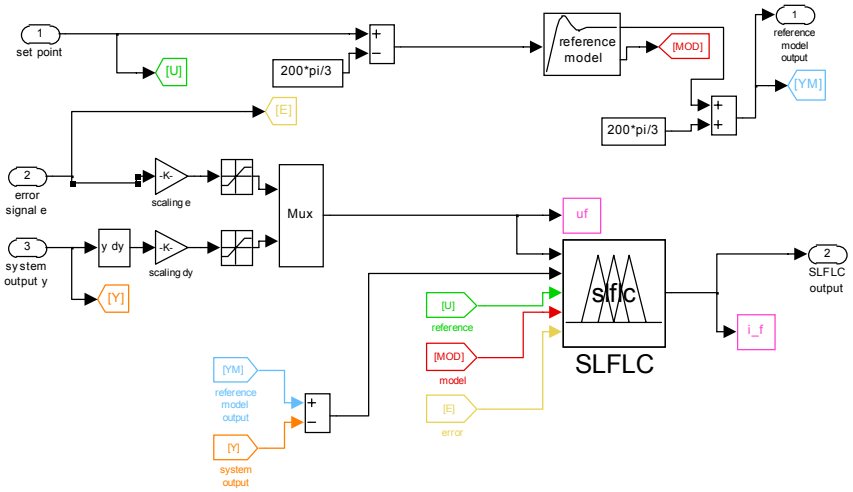

Fig. 3. Structure of the super-block SLFLC.

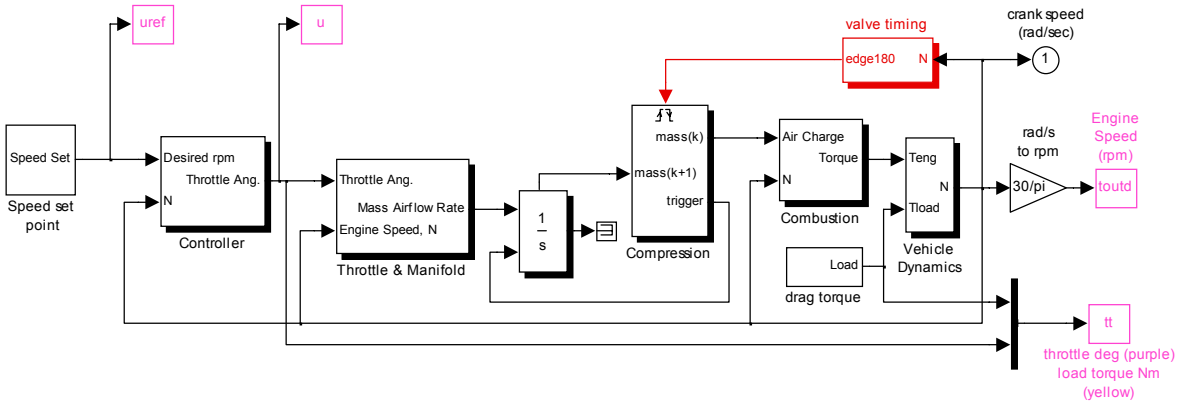

Fig. 4. The Matlab simulation scheme of the closed-loop engine speed control (directory: *\matlab\toolbox\simulink\simdemos\engine.mdl*).

Feedforward gain/P controller gain (type: <real>)
Depending on the type of controlled system (static or astatic) user defines feedforward gain K_f or proportional gain K_p, respectively. If K_f=0, then K_f will change according to the learning law (17) and contribute to the total output of the SLFLC. If K_p=0, then only fuzzy logic control is active.

Minimal sensitivity value (Type: <real>)
User defines a threshold value of sensitivity functions that will still contribute to changes of centroids.

The internal structure of the SLFLC super-block is shown in Fig 3. It may be seen that the super-block contains besides the SLFLC function block also blocks

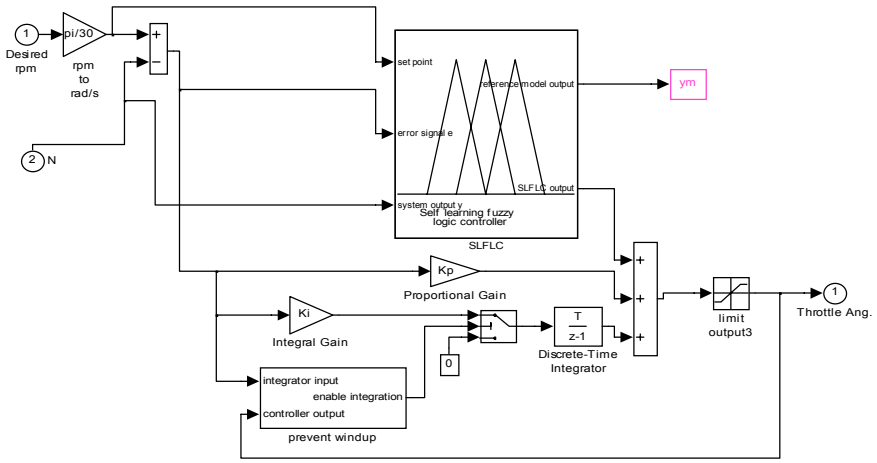

Fig. 5. The block of a hybrid controller (PI+SLFLC).

for scaling of FLC inputs and a reference model block.

The reference model is a second-order system that has been implemented according to the reference model equation (4) with the unity gain. The mask of the reference model block is made in a way that user can define the time of first maximum t_m, the overshoot in response F_m and the value of control interval. After ZOH discretization, resulting coefficients of the reference model equation (4) are available for user´s convenience at the reference model block output.

6. Simulation Results

The SLFLC function block has been tested in the closed-loop engine speed control system available as a demo example in the Simulink 2 (directory denoted in Fig. 4). Control of an engine speed is based on the control of a throttle-valve opening. The simulation scheme of the engine speed control system is shown in Fig. 4. It is not intention of this paper to discuss the properties of the model, but only to mention that the model is nonlinear and originally controlled only by a linear PI controller. A block of a hybrid engine speed controller is shown in Fig. 5. It may be seen that the SLFLC has been added in parallel to the existing PI controller. This means that the SLFLC acts as an adaptation mechanism, which should improve the overall system performance.

Five linearly distributed gaussian membership functions have been determined for both SLFLC inputs e(k) and dy(k) (LNE, SNE, ZE, SPE, LPE and LNDY, SNDY, ZDY, SPDY, LPDY, respectively). Simulations has been performed with the simulated change of the reference input) u_r=±1000 rpm. Fuzzy input scaling coefficients have been estimated at k_e=1/100, k_{dy}=1/300. Minimal value of the sensitivity function has been set to 0.05, a limit for an allowed change of centroid has been set to the unity value, while the gain coefficient k_s has been set to 3.0. The proportional gain k_p has been set to zero, because the PI controller is already in the loop. Desired performance indices of the reference model have been as follows: F_m=0.5% , t_m=0.5s, (sampling interval T_s is 0.01 s).

Figs. 6 and 7 show the reference input, system output, and reference model responses obtained after three and after thirteen iterations of learning, respectively. It may be seen that after completion of learning the system follows the reference model much better than with the PI controller itself. The original system with a PI controller reaches the steady state after four seconds, while the hybrid fuzzy controller enforces system to reach the steady state in less than a second. This proves that addition of the SLFLC in parallel to the PI controller has contributed to better quality of the system response. Figs. 8 and 9 show the hybrid controller outputs in the beginning (only PI is active) and the end of learning (both controllers are active), respectively. The hybrid fuzzy controller output has a very acceptable form, which clearly indicates that the target system is very nonlinear.

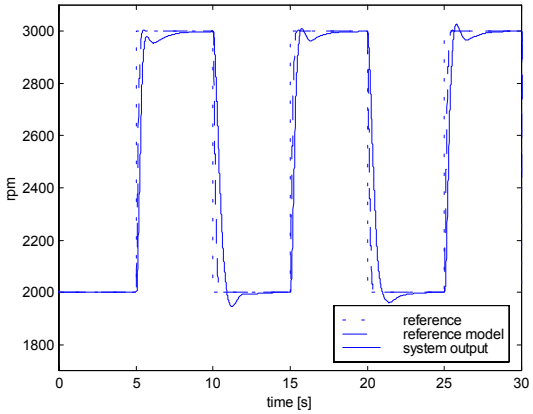

Fig. 6. Start of learning: the reference input, system output and reference model output responses of the engine speed control system.

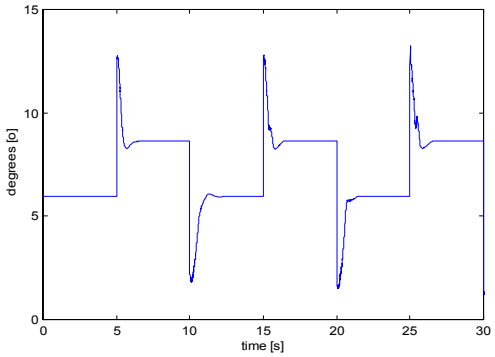

Fig.7. End of learning: the reference input, system output and reference model output responses of the engine speed control system.

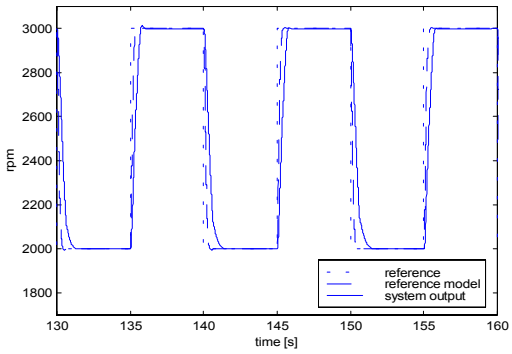

Fig.8. Start of learning: the hybrid controller output responses.

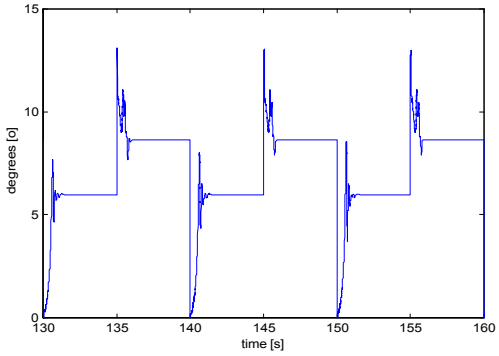

Fig.9. End of learning: the hybrid controller output responses.

7. CONCLUSIONS

The usage of simulation software packages for modeling, simulation and optimization of control systems has become a part of regular engineering practice both in the academic and industrial community. New features such as possibility to generate a real-time executable code directly from the simulation model and then download it into the target control hardware enable shorter development times and faster validation of new control solutions.

The paper describes a function block created for the Matlab+Simulink environment, which contains a PD type self-learning fuzzy logic controller (SLFLC) whose learning law is based on the usage of a reference model and a sensitivity model built with respect to the fuzzy controller parameters. The concept of the SLFLC allows control of unknown inherently stable static and astatic nonlinear systems if the desired closed-loop behavior may be represented with a linear second-order reference model. The effectiveness of the SLFLC has been demonstrated on the nonlinear simulation model of the engine speed control system, which may be found in the library of the Matlab+Simulink software package models for demo purposes. In the selected example of the control system the SLFLC has been added in parallel to the existing PI engine speed controller and the results obtained have confirmed a stable convergence of learning and significant improvement of the overall closed-loop system performance.

The SLFLC function block has been also successfully tested on other Matlab+Simulink models such as the model of the PMSM angular speed control system. More about the results obtained in these experiments can be found at the web site http://www.rasip.fer.hr/flrcg. The author's have also provided that the SLFLC function control block described in this paper is downloadable from the same web site for public testing in various control projects.

8. REFERENCES

[1] Z. Kovačić, M. Balenović, S. Bogdan, "Sensitivity-Based Self-Learning Fuzzy Logic Control for a Servo System", *IEEE Control Systems Magazine*, Vol 18, No. 3, pp. 41-51, 1998.

[2] Z. Kovačić, S. Bogdan, M. Balenović, "A Model Reference & Sensitivity Model-based Self-learning Fuzzy Logic Controller as a Solution for Control of Nonlinear Servo Systems", *The IEEE Transactions on Energy Conversion*, Vol. 13, No.4, pp. - , December 1999.

[3] Z. Kovačić, S. Bogdan, M. Balenović, "A Sensitivity-Based Self-Learning Fuzzy Logic Controller as a Solution for a Backlash Problem in a Servo System", *Proceedings of The 1997 IEEE Conference IEMDC'97*, pp. TC2-11.1 - TC2-11.3, Milwaukee, 1997.

[4] Z. Kovačić, M. Balenović, S. Bogdan, "An Experimental Verification of a Model Reference and Sensitivity Model-based Self-learning Fuzzy Logic Controller Applied to a Nonlinear Servosystem", *Proc. of the 12th IEEE ISIC,* pp. 263-268, Istanbul, 1997.

[5] S.Bogdan, Z. Kovačić, "On the Design of Self-Learning Fuzzy Controllers for Nonlinear Control Systems by Using a Reference Model and a Sensitivity Model", *Proceedings of the 4th IEEE Mediterranean Symposium on Control & Automation*, pp. 799-804, Chania (Crete), 1996.

[6] N. Gulley, J.S.R. Jang, "Fuzzy Logic Toolbox User's Guide", Math Works Inc., 1995.

[7] Z. Kovačić, S. Bogdan, M. Balenović, "Robustness Improvement of a Model Reference & Sensitivity Model-based Self-learning Fuzzy Logic Controller", *Proceedings of the 1998. IEEE Conference on Control Applications*, pp. 643-647, Trieste, 1998.