SELF-TUNING FUZZY PID: REAL TIME CONTROL APPLICATIONS

S. Ahmed, A. Taneva, M. Petrov, I. Ganchev

Technical University of Sofia, Branch Plovdiv, Control Systems Department, Plovdiv, Bulgaria

{sevil.ahmed, altaneva, mpetrov, ganchev}@tu-plovdiv.bg

Abstract: This paper presents a neuro-fuzzy structure of a Fuzzy PID controller with self-tuning parameters. The main advantage here is that the equation of classical PID control low is used as a Sugeno function into the fuzzy rules. Hence the designed fuzzy PID controller can be viewed as a natural similarity to the conventional PID controller. Previously achieved a good simulation performance of the presented Fuzzy PID controller is tested now in real time applications. Two implementations of the controller for real time temperature and level control processes are examined. Real time applications are carried out via the Real-Time Workshop of Matlab.

Key words: Fuzzy PID controller, Self-Tuning PID, Real Time Application, Rapid Prototyping

INTRODUCTION

Nowadays, the better understanding of the potential of adaptive control strategies as self-tuning Fuzzy PID (FPID) controllers in many areas of science and engineering have led to the importance of studying aspects such as the analysis, design, tuning and implementation of these controllers. Therefore, two real time applications of process control via FPID controller are presented in this paper.

Fuzzy logic controllers (FLC) have emerged as one of the most active and useful research areas in fuzzy control theory. That is why fuzzy logic controllers have been successfully applied for control of various physical processes. On the other hand the best-known industrial process controller is the Proportional-Integral-Derivative (PID) controller because of its simple structure and robust performance in a wide range of operating conditions. This paper is devoted to study the performance of previously achieved good simulation performances [1,2,3] of the presented FPID controller in real time applications.

A SELF-TUNING FUZZY PID CONTROLLER

The structure of the control system with the studied fuzzy PID (FPID) controller, introduced in [1], is shown in Fig.1. The controller based on the Sugeno's fuzzy technique is implemented as a fuzzy - neural network [4].

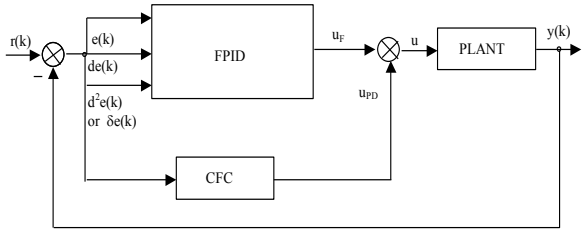

Fig.1. The structure of the control system with the proposed fuzzy PID controller

An additional conventional PD controller works in parallel with the main FPID controller. This is because the learning method for a neural network invested by Gomi and Kawato [5], which is used here. They have proposed learning schemes using feedback error learning for a neural network model applied to an adaptive nonlinear feedback controller. In these learning schemes, a conventional feedback controller (CFC) is used both as an ordinary feedback controller to guarantee global asymptotic stability in a compact space and as an inverse reference model of the response of the controlled object. This approach is used here in order to tune the fuzzy - neural network implementation of the FPID controller.

The traditional FLC works usually with input signals of the system error *e(k)* and the change rate *de(k)* in the error. The system error is defined as a difference between the set point *r(k)* and the plant output *y(k)* at the moment *k*, or

$$e(k) = r(k) - y(k) \tag{1}$$

Hence the change rate in the error *de* at the moment *k* will be:

$$de(k) = e(k) - e(k-1) \tag{2}$$

The FPID controller can use as a third input signal the sum of the system errors $\delta e(k)$ or the acceleration error $d^2e(k)$, which are calculated using the equations:

$$\delta e(k) = \sum_{i=1}^{k} e(i) \tag{3}$$

$$d^2e(k) = e(k) - 2e(k-1) + e(k-2) \tag{4}$$

The first one (3) is connected to the positioning type FPID controller and the second one (4) is connected to the velocity type FPID controller as it is mentioned below.

It is known from digital control theory, that the most frequently used digital PID control algorithm can be described with the well-known discrete equations as follows [6]:

- positioning type PID controller:

$$u(k) = k_p e(k) + k_i \delta e(k) + k_d de(k) \tag{5}$$

- velocity type PID controller:

$$\Delta u(k) = k_p de(k) + k_i e(k) + k_d d^2 e(k) \tag{6}$$

where $k_i = k_p \frac{T_k}{T_i}$, $k_d = k_p \frac{T_d}{T_k}$, T_k is the sample time of the discrete system, T_i is the integral time constant of the conventional controller, T_d is the differential time constant, k_p is the proportional gain, $u(k)$ is the output control action and $\Delta u(k)$ is the incremental control action.

The final control action for the second controller can be calculated according to the previous value of the control output $u(k\text{-}1)$ as follows:

$$u(k) = u(k-1) + \Delta u(k) \tag{6a}$$

The Sugeno's fuzzy rules into the FPID controller can be composed in the generalized form of '*if-then*' composition with a premise and an antecedent part to describe the control policy. The rule base comprises a collection of N rules, where the upper index (n) represents the rule number:

- positioning type FPID :

$$R^{(n)}: if\ e\ is\ E_i^{(n)}\ and\ de\ is\ dE_i^{(n)}\ and\ \delta e\ is\ \delta E_i^{(n)}$$
$$then\ f_u^{(n)} = k_p^{(n)} e(k) + k_i^{(n)} \delta e(k) + k_d^{(n)} de(k) + k_0^{(n)} \tag{7}$$

- velocity type FPID :

$$R^{(n)}: if\ e\ is\ E_i^{(n)}\ and\ de\ is\ dE_i^{(n)}\ and\ d^2e\ is\ d^2E_i^{(n)}$$
$$then\ f_u^{(n)} = k_p^{(n)} de(k) + k_i^{(n)} e(k) + k_d^{(n)} d^2e(k) + k_0^{(n)} \tag{8}$$

where e, de, d^2e , δe are the input variables. This way a similarity between the equations of the conventional digital PID controller (5), (6) and the Sugeno's output functions f_u (7) and (8) could be found. In this case, the FPID controller can be considered as a collection of many local PID controllers, which are represented by the Sugeno's functions into the different fuzzy rules and this way it is possible to approximate the nonlinear characteristic of the controlled plant.

The fuzzy implication, connected to the rules, is realized by means *product* composition [7]:

$$\mu_u^{(n)} = \mu_e^{(n)} * \mu_{de}^{(n)} * \mu_{\delta e}^{(n)} \tag{9}$$
$$\mu_u^{(n)} = \mu_e^{(n)} * \mu_{de}^{(n)} * \mu_{d^2e}^{(n)} \tag{10}$$

where μ_e, μ_{de}, $\mu_{\delta e}$ and μ_{d^2e} specify the membership degrees upon the fired fuzzy sets of the input signals into the $(n)^{th}$ fuzzy rule.

For a discrete universe with q quantisation levels in the fuzzy output, the control action u_F is expressed as a weight average of the Sugeno's output functions f_u and their membership degrees μ_u of the quantisation levels.

$$u_F = \frac{\sum_{i=1}^{q} f_u^{(i)} \mu_u^{(i)}}{\sum_{i=1}^{q} \mu_u^{(i)}} \quad \text{or} \quad u_F = \sum_{i=1}^{q} f_{ui} \bar{\mu}_{ui} \tag{11}$$

For simplicity in the second equation, the upper index (i), which represents the number of the corresponded rule, is used down.

Two case studies are examined in order to show the real time performance of the fuzzy PID controller, which efficiency is confirmed by numerous simulations. Both studies are carried out via rapid prototyping methodology of creating real time applications.

RAPID PROTOTYPING OF REAL TIME APPLICATIONS IN MATLAB ENVIRONMENT

The key to rapid prototyping is automatic code generation. It reduces algorithm coding to an automated process; this includes coding, compiling, linking, and downloading to target hardware. This automation allows design changes to be made directly to the block diagram. The traditional approach to real-time design and implementation typically involves multiple teams of engineers, including an algorithm design team, software design team, hardware design team, and an implementation team. When the algorithm design team has completed its specifications, the software design team implements the algorithm in a simulation environment and then specifies the hardware requirements. The hardware design team then creates the production hardware. Finally, the implementation team integrates the hardware into the larger overall system. This approach leads to long development processes, because the algorithm design engineers do not work with the actual hardware.

The rapid prototyping process combines the algorithm, software, and hardware design phases. The process allows engineers to see the results and rapidly iterate on the design before expensive hardware is developed. It begins in Simulink. First, the model is developed in Simulink®. In control engineering, this involves modeling plant dynamics and including additional dynamic components that constitute a controller and/or an observer. In digital signal processing, the model is typically an exploration of the signal-to-noise ratio and other characteristics of the input signal. Then the model is simulated in Matlab/Simulink® and toolboxes can be used to aid in the development of algorithms and analysis of the results.

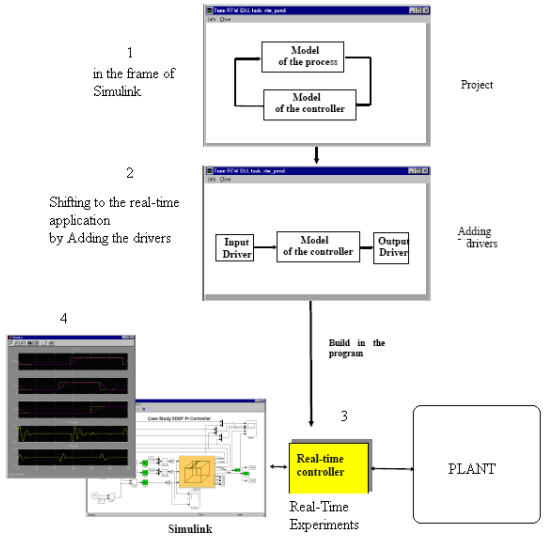

Fig.2. General scheme for RTW control in Matlab/Simulink®

If the results are not satisfactory, it is possible to iterate the modeling process until results are acceptable [8]. Once the desired results have been achieved, Real-Time Workshop generates downloadable C code (for the appropriate portions of the model). Using Simulink in external mode, allows tuning parameters and further refine the model, again rapidly iterating to achieve required results. At this stage, the rapid prototyping process is complete. The final implementation can begin for production with confidence that the underlying algorithms

work properly in designed real-time production system [8]. Rapid prototyping with RT-DAC/USB and RT-CON resources can be described by following scheme in Fig.3. It shows the general concept that comes within the scope of RTW and RT-CON. As it is described above MATLAB/ Simulink® is used to design, test and analyse a model of control or data acquisition system. Once the project is completed, appropriate input/output device drivers can be added and an executable code directly from the Simulink block diagram can be generated. When the real-time program is running communication with Matlab/Simulink® to capture and analyse data or to change parameters of the currently running real-time program is possible. This is the one of advantages of the used software package. Users dispose of all features and recourses of the intuitive Matlab/ Simulink® environment. They can observe signals, modify and tune block parameters in the real-time without a recompilation of the model [9]. It allows creating and executing a real-time program using Simulink models. This approach reduces the cost and time of development of new investigation applications.

The Simulink model for real time implementation purposes consist the device driver of used data acquisition RT-DAC/USB board. RT-DAC/USB device driver contains the all I/O signals of the board (Fig.3). It must be reconfigured depending of experiment purposes and caries out the connection between the Simulink model and the application target.

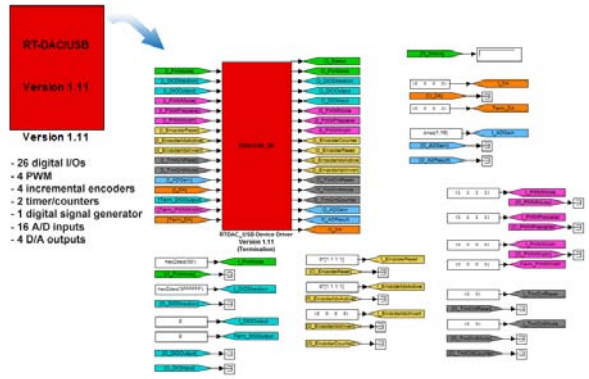

Fig.3. The RT-DAC/USB device driver

The controller is implemented via Matlab Embedded Function Block (Fig.4), which makes possible real time execution of the FPID source code.

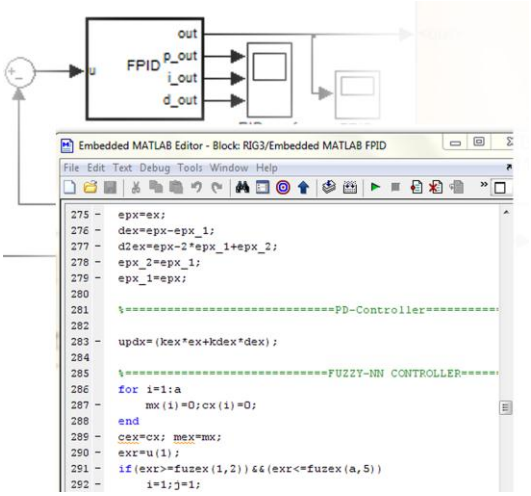

Fig.4. Implementation of the FPID in rapid prototyping scheme

CASE STUDIES

This section of the paper presents the results of performed two case studies of process control. The FPID controller with self-tuning FPID is applied to temperature and level control schemes. Fig.5 shows the design of the Simulink diagram of the studies, where the plant is presented by data acquisition board's driver.

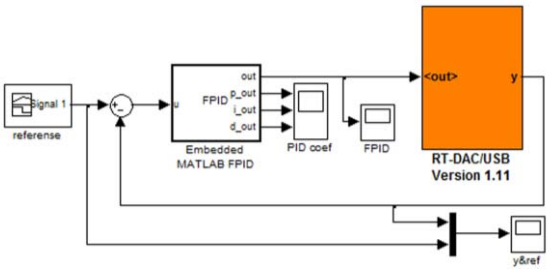

Fig.5. Simulink diagram of the applications

Both applications relies on normalized in [-1;1] signals to ensure the proper work of the FPID controllers. They are fuzzified by five triangular membership functions [2].

Level Control. The experimental platform Basic Process Rig 38-100 Feedback Unit has been used to test the self-tuning FPID controller. The platform consists on a low pressure flowing water circuit which is bench mounted and completely self contained. The water circuit is arranged in front of a vertical panel, as can be seen in Fig.6.

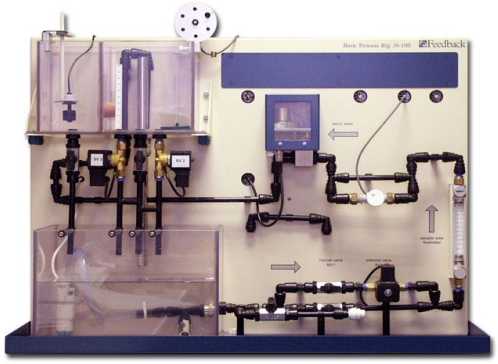

Fig.6. Basic Process Rig 38-100 Feedback Unit

The rig has three tanks on the level and flow rig constructed of welded polycarbonate. The three tanks are: a sump tank and a 3L and 2L dual compartment process tank. Both rigs, the level and flow rig is fitted with a submersible 10 L/m 12V dc pump. The system uses different orifice solenoid valves allowing variable drain rates. Control objective here is to achieve the desired level in the upper tank by acting on servo valve. One of manual valves is partly opened to ensure a difference in water drain rate. Level is controlled using a Float Sensor. Fig.7 presents the results for level set point of 85 mm. As it can be seen the output follows the reference in very good manner.

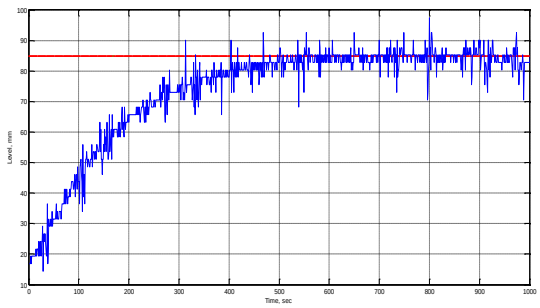

Fig.7. Transient response of real time FPID level control

Fig.8 shows the FPID output during the experiment. It is result of the work of the controller which updates its coefficients k_p, k_i, k_d according to process demands. The trend of the parameters is presented in Fig.9.

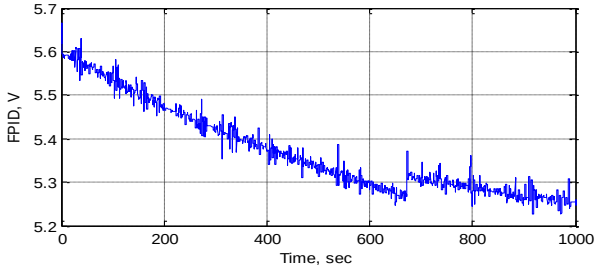

Fig.8. Control action of the self-tuning FPID controller applied to the servo valve

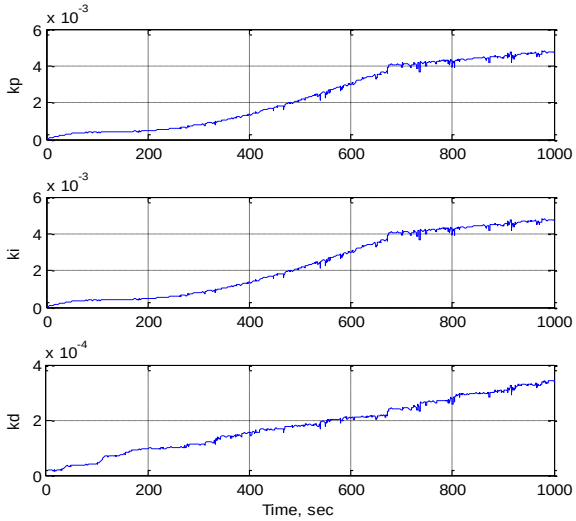

Fig.9. Coefficients of the implemented velocity type FPID

Temperature Control. Following the RT-CON rapid prototyping scheme, a Simulink model of the temperature control scheme consists of fuzzy PID controller and data acquisition board driver. The temperature process plant is represented by a 230V lamp controlled by a solid state relay (SSR).

The objective is to control the temperature of the plant. The output has to closely follow a specified reference. A platinum resistance temperature sensor Pt100 with temperature transmitter is used as a feedback to the controller. Fig. 10 presents the realization of the connection between the target and RT-DAC board.

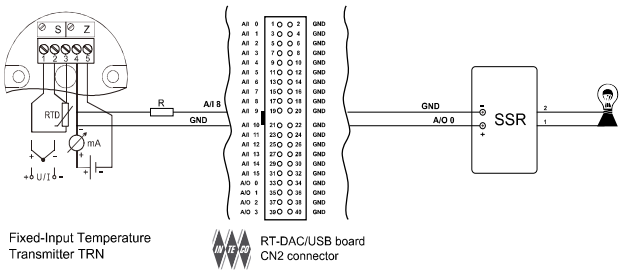

Fig.10. Scheme of the experiment realization

It is evident that the temperature closely follows the reference. The result (Fig.11) verifies the good performance of the studied in this paper self-tuning FPID controller.

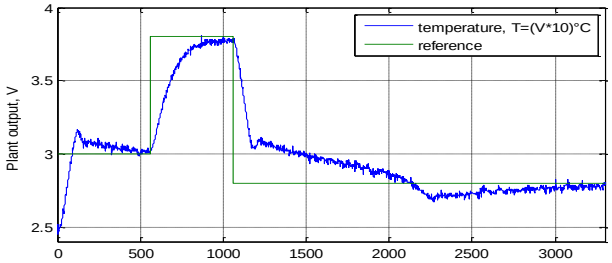

Fig.11. Transient response of real time FPID temperature control

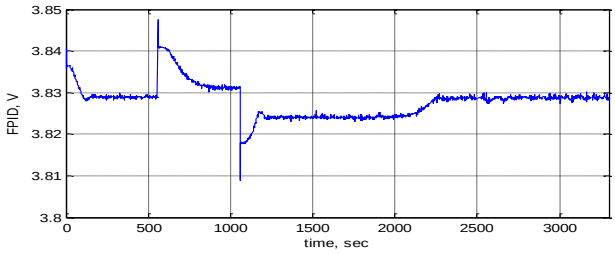

Fig.12. Control action of the self-tuning FPID controller applied to the SSR

CONCLUSION

The performed in this paper real time experiments proof the effective performance of the studied FPID controller with self-tuning parameters. It is evident from the obtained results, where the transient responses of the systems outputs (Fig.7 and Fig.11) have shown an improvement after several learning epochs with the implemented fuzzy-neural network. A future work will be focused on implementation of the controller to more complicated nonlinear processes.

REFERENCES

1. Petrov M., I. Ganchev, A. Taneva. Fuzzy PID Control of Nonlinear Plants, Preprints of the International Summer School on Automation'2002, Split, Croatia, 16 - 29 June, 2002.
2. Taneva A., M. Petrov, I. Ganchev. Modified Fuzzy PID Control Algorithms for Nonlinear Process Control. Proceedings of the International Conference 'ISAC 2006' on " Modern Trends in Control", Kosice, Slovak Republic, 15 - 30 June 2006. ISBN 80-969224-6-7, pp. 275 - 288.
3. Petrov M., I. Ganchev, K. Kutryanski, A Study of the Fuzzy PID Controller, Preprints of the International Summer School on Artificial Intelligence in Control and Measurement, Brno, Czech Republic, 21 - 31 August, 2000, ISBN 80-214-1641-6, pp.93 - 99.
4. Petrov M., T. Proychev and A. Topalov, An Adaptive Hybrid Fuzzy - Neural Controller. Prepr. of 2nd IFAC Workshop on New Trends in Design of Control Systems, Smolenice, Slovak Republic, 7-10 September 1997, pp. 326-331.
5. Gomi H. and M. Kawato, Neural Network Control for a Closed–Loop System Using Feedback–Error–Learning, Neural Networks, 1993, vol. 6.
6. Astrom K. and B. Witenmark, Computer controlled systems 3 ed., Prentice Hall, 1997.
7. Takagi H. and M. Sugeno, FUZZY Identification of Systems and its Applications to Modeling and Control, IEEE Trans. on Systems, Man., and Cybern., SMC-15(1), 1985, pp.116-132.
8. Dabney J. B. and T. L. Harman, Mastering Simulink, Pearson Prentice Hall, 2004
9. Inteco Ltd., RT-CON professional User's Manual, 2006