ARCHIVES OF ELECTRICAL ENGINEERING VOL. 69(3), pp. 609–624 (2020)

DOI 10.24425/aee.2020.133921

# Reproduction of the control plane as a method of selection of settings for an adaptive fuzzy controller with Petri layer

PIOTR DERUGO, MATEUSZ ŻYCHLEWICZ

*Wroclaw University of Science and Technology*
*The Department of Electrical Machines, Drives and Measurements*
*Smoluchowskiego 19 str, 50-370 Wroclaw, Poland*
*e-mail: {piotr.derugo/mateusz.zychlewicz}@pwr.edu.pl*

(Received: 08.11.2019, revised: 26.03.2020)

**Abstract:** The purpose of this paper is to show possibility and advantages of initial control plane reproduction for an adaptive fuzzy controller. Usually the fuzzy control is used when the object is not very well known. Yet the truth is, however, that some, at least general information about the object, is available. Usually, in such a case, optimization algorithms are used to tune the control structure. The purpose of this article is to show how to find a starting point that is closer to optimum than a statistically random point, and this way to obtain better results in a shorter time.

**Key words:** control plane reproduction, DC drive, fuzzy controller tuning, Petri layer

## 1. Introduction

Since the 60s when Lofti Zadeh [1] introduced fuzzy sets to control theory, many researchers have intensively exploited this branch of knowledge [2–6]. Fuzzy sets allow one to control objects which are controlled by humans yet their analytical description is uncertain or hard to define, and therefore they are not easy to control. The fuzzy approach allows one to implement knowledge of human experts into computer systems using linguistic description. On the other hand, all kinds of optimization algorithms allow tuning of control systems of unknown objects, causing many to choose this tool for controlled tuning. At the same time the knowledge about the systems grows rapidly and mathematical descriptions of more and more complicated phenomena and objects are almost common knowledge. Because of that we find it is useful to utilize as many information about the object as possible to improve a starting point of various algorithms.

For that purpose we modeled a two-mass system containing two DC motors connected with an elastic shaft described in the next section. We assume that we possess knowledge about the motors, which nowadays is obvious, but have very poor information about the shaft. For the purpose of controlling such an object we described a controller in the section *Adaptive neuro-fuzzy controller with transition Petri layer*. Normally, having limited data about the object and using an adaptive system, we would start by zeroing the weight coefficient matrix. In this case however, we start with a controller tuned as a classical IP controller for the system with stiff connection. In line with logic, a 1-mass system is closer to a 2-mass system than nothing.

The reproduction of a control function from an IP controller on a fuzzy controller is described in section 3.2. Some of numerical issues are also mentioned. Section 4 describes the Particle Swarm Optimization of adaptation parameters for an adaptive controller. Sections 6 and 7 present simulation and experimental results. Section 8 explains the purpose of Petri layer usage. The whole work is summed up in the last section.

## 2. Two mass system

For the purpose of this research a classical, separately excited DC motor with a cascade control structure has been used as the motor, and the same motor in a generation mode as the load. The excitation ($\Psi_f$) is constant. The machine is described by the undermentioned equations in p.u.

$$T_{em}\frac{\mathrm{d}i_a}{\mathrm{d}t} = -i_a + K_t\left(u_a - \Psi_f\omega_1\right), \tag{1}$$

$$T_1\frac{\mathrm{d}\omega_1}{\mathrm{d}t} = \Psi_f i_a - T_L, \tag{2}$$

where: $T_{em}$ is the electromagnetic time constant, $T_1$ is the mechanical time constant of the machine, $T_L$ is the load torque, $i_a$ is the armature current, $K_t$ is the gain factor of the motor, $u_a$ is the armature voltage, $\Psi_f$ is the excitation flux.

The cascade control system shown in Fig. 1 contains two control loops, each inner loop contains a current controller and current measurement. Its purpose is to compensate the electromagnetic time ($T_{em}$) constant. A classical IP controller tuned with a symmetry criterion is used as

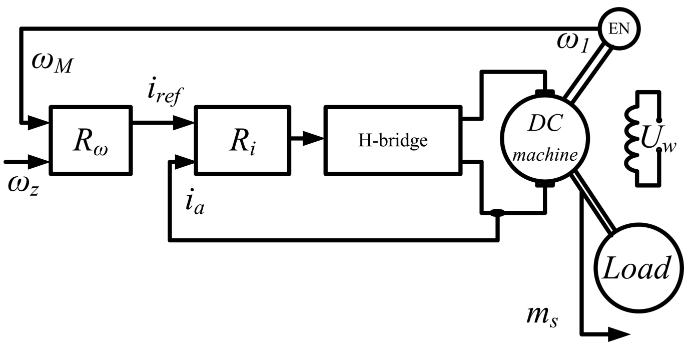


Fig. 1. Control system scheme

a current controller. The outer loop – speed control loop – contains a speed controller and a speed measurement system. Its purpose is to compensate the mechanical time constant ($T_1$) – the biggest time constant in the system. During the research, in the first step, the classical IP speed controller tuned with a modulus criterion has been used. Afterwards, this speed controller is substituted for the proposed fuzzy adaptive controller.

The system in the mechanical part contains two DC motors. One is a controlled drive, and the other works as a load machine, where the resistor is powered with generated energy. The machines are connected via a long shaft wherein elasticity of the shaft is non-negligible. The block diagram is presented in Fig. 2. The mathematical description includes Equations (3)–(5) [7, 8].

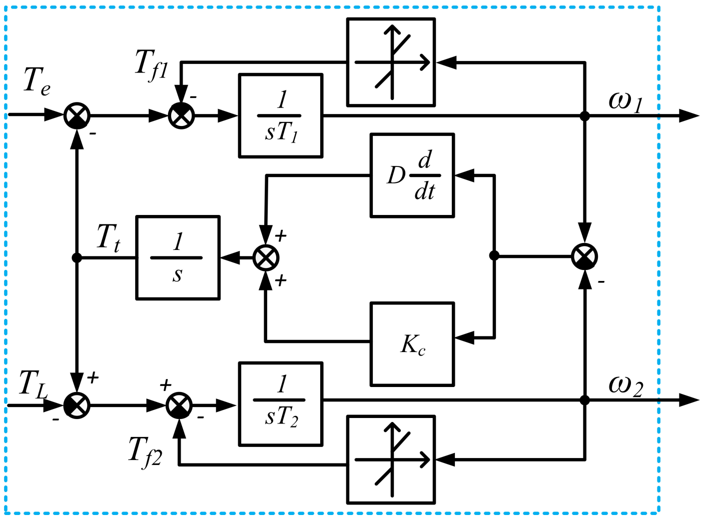


Fig. 2. Control system scheme

where: $T_e$ is the electromagnetic torque, $T_L$ is the load torque, $T_t$ is the torsional torque, $T_{f1}$, $T_{f2}$ represent the friction torque in the 1st and 2nd machine, $T_1$, $T_2$ are the time constants of the 1st and 2nd motor, $\omega_1$, $\omega_2$ represent the mechanical speed of the 1st and 2nd motor, $K_c$ is the elasticity coefficient of flexible joints, $D$ is the damping coefficient of flexible joints.

$$T_1\frac{\mathrm{d}\omega_1}{\mathrm{d}t} = T_e - T_t - T_{f1}, \tag{3}$$

$$T_2\frac{\mathrm{d}\omega_2}{\mathrm{d}t} = T_t - T_L - T_{f2}, \tag{4}$$

$$\frac{\mathrm{d}T_t}{\mathrm{d}t} = D\frac{\mathrm{d}\left(\omega_1 - \omega_2\right)}{\mathrm{d}t} + K_c\left(\omega_1 - \omega_2\right), \tag{5}$$

where: $K_c$ is the stiffness constant, $D$ is the damping factor of the shaft, $T_{f1}$, $T_{f2}$ stand for the friction torque of the motor and load.

## 3. Adaptive neuro-fuzzy controller with transition Petri layer

For the purpose of this research, an adaptive fuzzy (IP) controller with the transition Petri layer (TPL) has been used. The solution can easily be extended to a higher number of inputs [9]. Input variables can be selected in accordance with the circumstances. The neuro-fuzzy inference system consists of 6 basic layers, the TPL is implemented between the first two layers and an adaptation algorithm. A schematic diagram of the controller is presented in Fig. 3. The layers of the controller are described below. In order to obtain a non-adaptive neuro fuzzy controller, the adaptation parameters (Equation (10)) need to be zeroed.

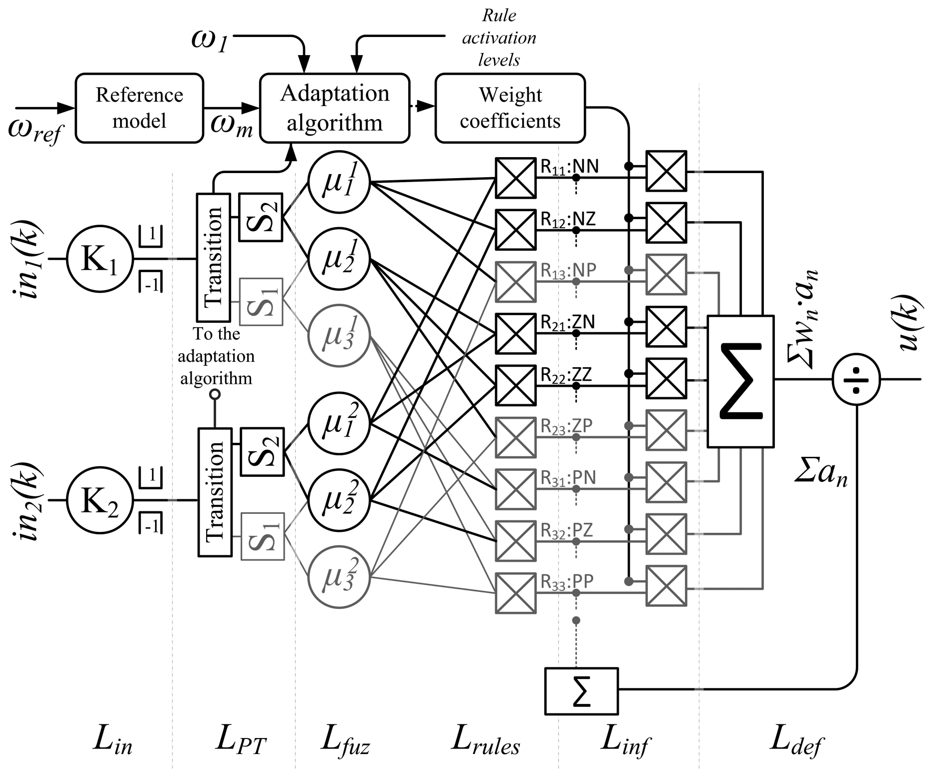


Fig. 3. Schematic diagram of the controller with Petri transition layers

### 3.1. Input layer

The input layer ($L_{in}$), normalizes input signals as in (6). The considered range of each of the variables is limited ($sat_{in\_k}$). Scaling factors ($K_{in\_k}$) cause the range of input values to fit the desired limited range of the expected value of the variable. Scaling factors for this research were chosen as nominal values, considering the model that we desire as an object to follow. This means, for $in_k = e(k)$, $K_{in} = 1$ since the tracking error equal to a nominal value or bigger is treated here as a maximal value. For $\Delta e(k)$, $\Sigma e(k)$ we analyzed the step response of the model and obtained maximal values of this parameters during normal operation of the system. Of course any lower value can be chosen, to obtain a more demanding system.

$$L_{\text{in}_{\text{out}}} = \max\left[\min\left[(K_{in} in_k),\ sat_{in_k}\right],\ -sat_{in_k}\right], \tag{6}$$

where: $in_k = [e(k),\ \Delta e(k),\ \Sigma e(k),\ u(k-1),\ x(k),\ \ldots]$.

## 3.2. Transition Petri layer

Transition Petri layers (TPLs) identify the sector of the input variable, in which the variable is currently located. This layer selects appropriate membership functions and rules based on the identified area of the control plane. The reason is the desire to reduce the numerical effort of the controller with a large number of fuzzy sets. The TPL has the same mathematical influence on the algorithm output value as the concurrent Petri layer [10, 11, 12] but an additional advantage of lower computational cost is introduced [9, 13].

In Fig. 4 an example of a control plane and the consequences of TPL application in the neuro-fuzzy system with two inputs and three type-I triangular membership functions for each input is shown, it is easy to imagine such a solution for any other type or shape of function or higher dimension, yet the figure would be blurred. Since only two membership functions (for each input) are active, instead of four quadrants of the control plane, only one needs to be analyzed to obtain all information. This means that during each iteration only 4 ($2^2$) out of 9 ($3^2$) rules need to be calculated. Increasing the number of membership functions (for each input) to $l_{mf}$, and the number of inputs to $l_{in}$, results in the need of determination of $2^{l_{in}}$ instead of $lmf^{l_{in}}$ rules and the adaptation of the same number of weight coefficients. For 3 inputs and 5 membership functions

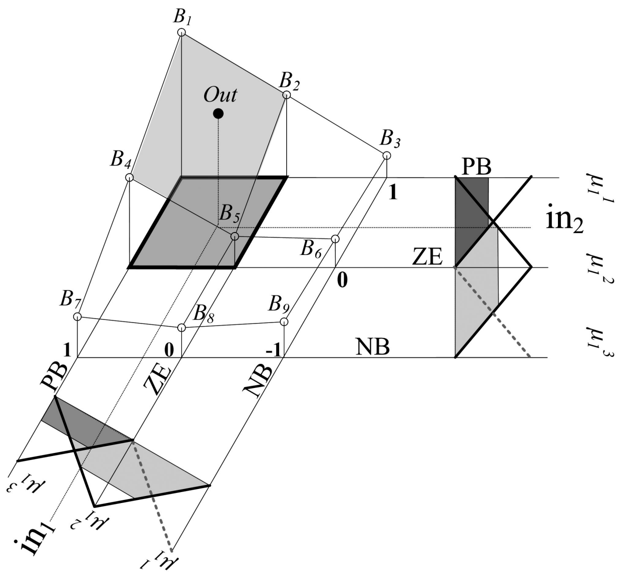


Fig. 4. Control plane for 2 input controller with triangular membership functions with TPL

per input, using a transition layer that activates only 2 functions per input, we have $2^3$ instead of $5^3$ rules that are calculated in each iteration.

Triangular membership functions have a finite medium, for any input value only two adjacent membership functions per each input are active in the non-zero level. The application of the TPL accepting these active functions will cause no change in an output control signal, yet smaller computational effort. This way more membership functions or dimensions may be implemented at the same computational cost. On the other hand, a more complicated control plane may be implemented at lower computational effort, so that the task time in a real life object may be shorter. This is important especially for complex control systems and for the control of rapidly changeable signals such as currents.

### 3.3. Fuzzyfication layer

In the fuzzyfication layer ($L_{fuz}$) according to Formula (7) the membership function activation level is calculated for the current input values. The undermentioned formula describes a triangular membership function (Fig. 5).

$$\boldsymbol{L}_{fuz_{out}} = \max\left[\min\left[\frac{\boldsymbol{L}^{k_{in}}_{fuz_{in}} - a^{k_{in}}_{j_{set}}}{b^{k_{in}}_{j_{set}} - a^{k_{in}}_{j_{set}}}, \frac{c^{k_{in}}_{j_{set}} - \boldsymbol{L}^{k_{in}}_{fuz_{in}}}{c^{k_{in}}_{j_{set}} - b^{k_{in}}_{j_{set}}}\right], 0\right]_{\substack{k_{in}=1\ldots l\\ j_{set}=1\ldots j}}. \tag{7}$$

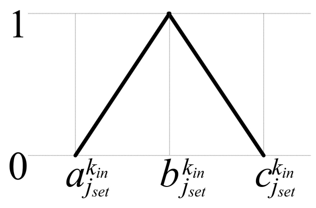


Fig. 5. Triangular membership function

The used triangular membership functions were distributed uniformly over the considered variable range. Markings in the formula: $k_{in}$ is the number of inputs, $j_{set}$ is the number of membership functions (for the TPL only the rules identified by the layer as the active ones are calculated), $l$ is the number of inputs, $j$ is the number of membership functions in set corresponding with the considered input, $a$, $b$, $c$ are the triangular membership function parameters as in Fig. 5.

### 3.4. Inference layer

The inference layer determines the firing levels of individual rules multiplied by the corresponding weight coefficient. The rule base consists of expressions as follows [14, 15]:

$$\begin{aligned} R_{k_{in_1},k_{in_2}} : \text{if } & \left(R_{in_1} \text{ is } \mu^{in_1}_{set_1}(in_1)\right) \text{ and } \left(R_{in_2} \text{ is } \mu^{in_2}_{set_2}(in_2)\right) \\ \text{than } & \left(y = f\left(\mu^{in_1}_{set_1}(in_1),\ \mu^{in_2}_{set_2}(in_2)\right)\right). \end{aligned} \tag{8}$$

The $t$-norm prod was used as an output function in this work.

### 3.5. Defuzyfiaction layer

The deffuzyfication layer ($L_{def}$) evaluates the output of the fuzzy system. In the presented case a classical singleton deffuzyfication algorithm was used, the algorithm is described by the (9) [14, 16]. The undermentioned formula is designed for two input systems, for a larger number of inputs the number of dimensions will increase.

$$u(k) = \frac{\sum\limits_{\substack{in_1=1\ldots k_{in_1}\\ in_2=1\ldots k_{in_2}}} R_{k_{in_1},\, k_{in_2}} \omega_{k_{in_1},\, k_{in_2}}}{\sum\limits_{\substack{in_1=1\ldots k_{in_1}\\ in_2=1\ldots k_{in_2}}} R_{k_{in_1},\, k_{in_2}}}. \tag{9}$$

### 3.6. Adaptation algorithm

The value of the change of weight coefficients in each iteration is described by Equation (10):

$$w_j^i(k+1) = w_j^i(k) + L_{rules_{out}}{}_j^i(k) \cdot [k_{em} \cdot e_m(k) + k_{\Delta m} \cdot \Delta e_m(k)], \tag{10}$$

where $e_m$ (reference model tracking error) is calculated as the difference between the reference model signal and the measured motor speed ($e_m = \omega_{mod} - \omega_1$). An inertial second order object, (11), with parameters $\omega_r = 30$, $\xi = 1$ was used as a reference model. These values were chosen arbitrarily, as a consensus between good outcome dynamics of the system, and physical possibilities of the real object. This reference model represents desired dynamics of the whole system and is a design parameter under physical constraints.

$$G(s) = \frac{\omega_r^2}{s^2 + 2 \cdot \xi \cdot \omega_r \cdot s + \omega_r^2}. \tag{11}$$

In each iteration of the algorithm, each weighting coefficient is changed by the value dependent on the current tracking error of the model. It is important to limit the maximum values of the weighting coefficients and/or to incorporate a limit on the controller output, for example by stopping the process of adaptation. Weight coefficients should be limited in such a way as to allow for reaching the maximum desired output signal value. A detailed description of the control structure with a reference model was also presented in [13]. In the case of PTL usage, only the coefficients corresponding with active functions are being adapted.

## 4. Methodology of settings selection – reproduction of the control plane

The idea of control plane reproduction is based on the assumption of prior knowledge about the object, and knowledge about the fuzzy system. For the purpose of this research, where the authors want to show the reduction of computational effort due to Petri layer introduction, it is assumed that the plant can be controlled by a classical IP controller, thus this control function of the controllers is reproduced on the fuzzy controller. This method may be found particularly applicable for the objects, where adaptation algorithms are used, yet a designer has some general information about the object. Thus, the initial weight coefficient matrix, that defines the control plane, is not zeroed. This will certainly limit initial fluctuations caused by rapid adaptation in early stages of system operation [11].

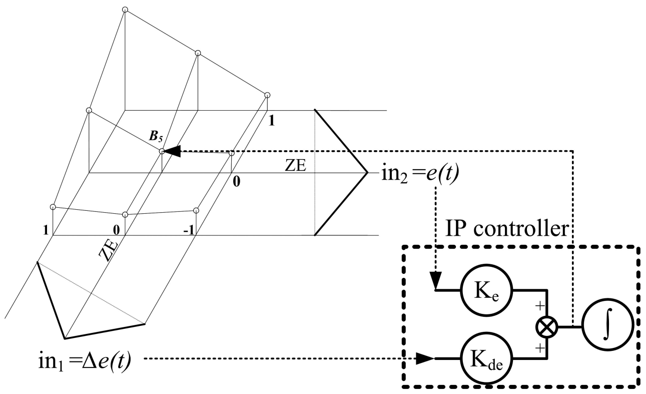


Fig. 6. Projection of the control plane of IP controller on fuzzy controller

This kind of research are planned in the near future. Obviously, the control function is not only limited to the plane or linear control, the source of information is arbitrary, yet not always it is possible to reproduce the function identically due to mathematical issues that will be discussed later.

The first things that need to be considered are the limitations for the system. Usually the output of controllers needs to be limited for the proper and safe operation of objects. For example, as used in this study, the speed controller in the outer loop of the cascade control structure for the DC motor generates reference torque/current that needs to be limited due to the safety reasons. In this study the $i_a$ limit is set as 1.5, the structure operates in the p.u. system. For the IP controller this means that the limit for the integrator as well as output limitation of the fuzzy system need to adopt this value. As the repercussion of this limitation we need to consider limitations of input signals. Since, when $e(k) = 1$ and $\Delta e(k) = -1$ the output of summation would be 0. Also, if $e(k) = 3$ and $\Delta e(k) = -1$ the summation block will give +2, yet due to the limitation the output would be +1.5. Therefore, the limitation of the inputs of the fuzzy controller should adopt a double value of the output limit.

$$R_{n,m} = \mu_n^1 \cdot \mu_m^2 \cdot w_{n,m}\,, \tag{12}$$

$$w_{n,m} = b_1^1 K_{de} + b_2^2 K_e\,, \tag{13}$$

where: $K_{de}$ and $K_e$ are the parameters of the base IP controller, $b_1^1$ and $b_2^2$ are the positions of membership function cores.

To reproduce a control plane of a classical controller the weight coefficient matrix must be calculated according to a source controller. The source controller is tuned according to prior knowledge. In this case the speed controller for a DC motor was tuned using a criterion of symmetry. For each rule in the rule base the output value of the source controller corresponding with the cores of component membership functions needs to be calculated. In triangular membership

Functions (7) the core is a point at input value equal to $b^{k_{in}}_{j_{set}}$, for two input controllers the output of the source controller needs to be calculated for combinations of $in_1 = b^1_1$ and $in_2 = b^2_2$. In general, since the fuzzy controller with triangular membership functions can be considered as a group of linear controllers with smooth transitions, we want to set each individual part (rule) to be as close to the source controller as possible.

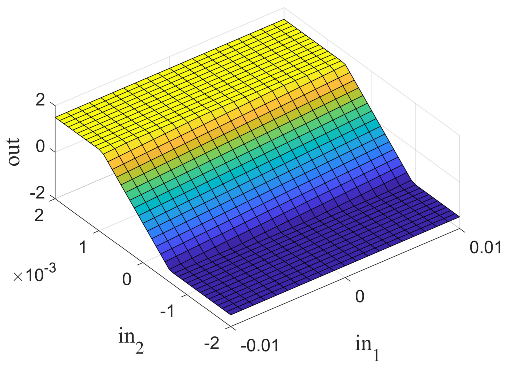


Fig. 7. Control surface of IP controller

It is very important to understand that not always perfect reproduction is possible due to computational restrictions. It is shown in Figs. 8 and 9. The mentioned computational restrictions origin in finite bit representation of fractions. An example for that might be the equation $\sin(\pi/4) - \cos(\pi/4)$ that analytically should be 0, yet for MATLAB it is –1.1102e–16 due to the finite expansion of a sine function into series. In this research parameters for the source controller are analytically computed, the number of decimal places results from the calculation possibilities. Nevertheless, as it is shown in the abovementioned figures the difference between the source controller and final fuzzy controller with a reproducted control plane is small, and not significant, faced with the fact that it is an initial plane for an adaptive controller.

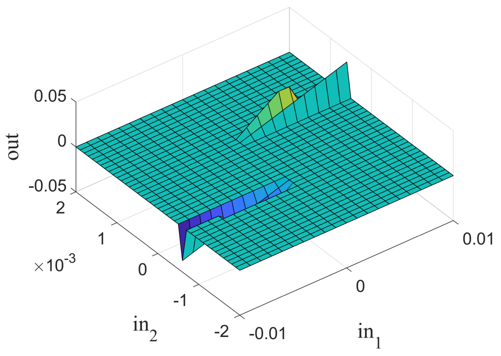


Fig. 8. Difference between control surface of IP controller and fuzzy controller with 7×7 fuzzy functions with reproducted controll surface

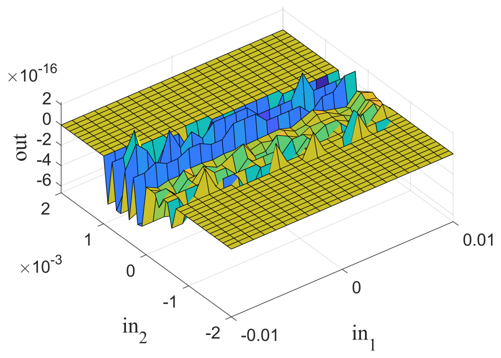


Fig. 9. Difference between control surface of IP controller and fuzzy controller with $13 \times 13$ fuzzy functions with reproducted controll surface

## 5. Optimization process of adaptation coefficients for certain initial weight coefficients matrix

Since input scaling factors are reproduced based on a source controller and the matrix of initial weight coefficients is zeroed or reproduced from the source controller, the remaining parameters to be determined are adaptation coefficients ($k_{em}$ and $k_{\Delta em}$ in Equation (10)).

For determination of these parameters MATLAB built in a Particle Swarm Optimization (PSO) [17, 18], algorithm was used. The swarm size was set to 20. Function tolerance was $1e-4$. Inertia range was set as [0.1, 1.5]. Initial swarm span as a default – 2000, max iterations – 200, were never reached. The maximum number of stall iterations – 5, combined with function tolerance was the reason to stop the algorithm. There were no time constraints. The minimum adaptive neighborhood size – 0.25. The self-adjustment as well social adjustment of weight were equal to a default – 1.49. For reproducibility the random number generation was set to the default in each iteration.

For both cases of initial weights many optimizations have been conducted, where boundaries of search space were variable. For both cases 10 different local minima have been found and presented in the table below. The cost function was chosen as ISE $(\omega_{mod} - \omega_1)$.

Table 1. Summary of adaptation parameters after PSO and resulting ISE values

| **IP** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|---|---|
| AD_P | 9.29 | 1000.00 | 98.11 | 2.47 | 736.87 | 2.47 | 1.86 | 1.44 | 95.72 | 0.98 |
| AD_D | 3.48 | 347.04 | 29.30 | 0.24 | 265.40 | 0.24 | 0.22 | 0.26 | 27.69 | 0.22 |
| ISE | 2.96 | **1.46** | 1.68 | **1.48** | 2.68 | **1.59** | **0.73** | 1.75 | **1.63** | **1.39** |
| **ZERO** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| AD_P | 95.72 | 1.41 | 1.46 | 976.03 | 707.28 | 2.05 | 2.33 | 1.37 | 87.78 | 0.96 |
| AD_D | 27.69 | 0.21 | 0.61 | 343.16 | 156.26 | 0.23 | 0.20 | 0.23 | 24.16 | 0.28 |
| ISE | **1.54** | **1.54** | 18.36 | 2.81 | 3.19 | 1.83 | **1.16** | 2.69 | **1.58** | 2.22 |

As it is seen globally, the best (bold) result was found for the case, where the initial matrix was reproduced from an IP controller, as well as the globally worst (italic) case was determined for the case with a zeroed initial matrix. What is more important, for each case (1, 2,. . . ) the boundaries were set the same for both cases of the initial matrix, however, in the 1st case the outcome for the reproduced plane was worse than for the zeroed one. 10 globally best outcomes were highlighted in bold font, and the worse ones with italic.

The conclusion, according to intuition (by inference), is, in general, that starting the adaptation with a reproduced initial control plane gives better outcomes than starting with zeroed coefficients. Nevertheless, a proper choice of search boundaries is also important. So, as it might be obvious, any kind of prior knowledge about the process might and should be used during the tuning process.

In Fig. 10, the adaptation process of the best obtained cases of a PSO algorithm for the case with the reproduced (a, c, e) and zeroed (b, d, f) matrix of initial weight coefficients, has been presented. In all the cases a fuzzy controller with 2 inputs and 7 fuzzy membership functions per input was used.

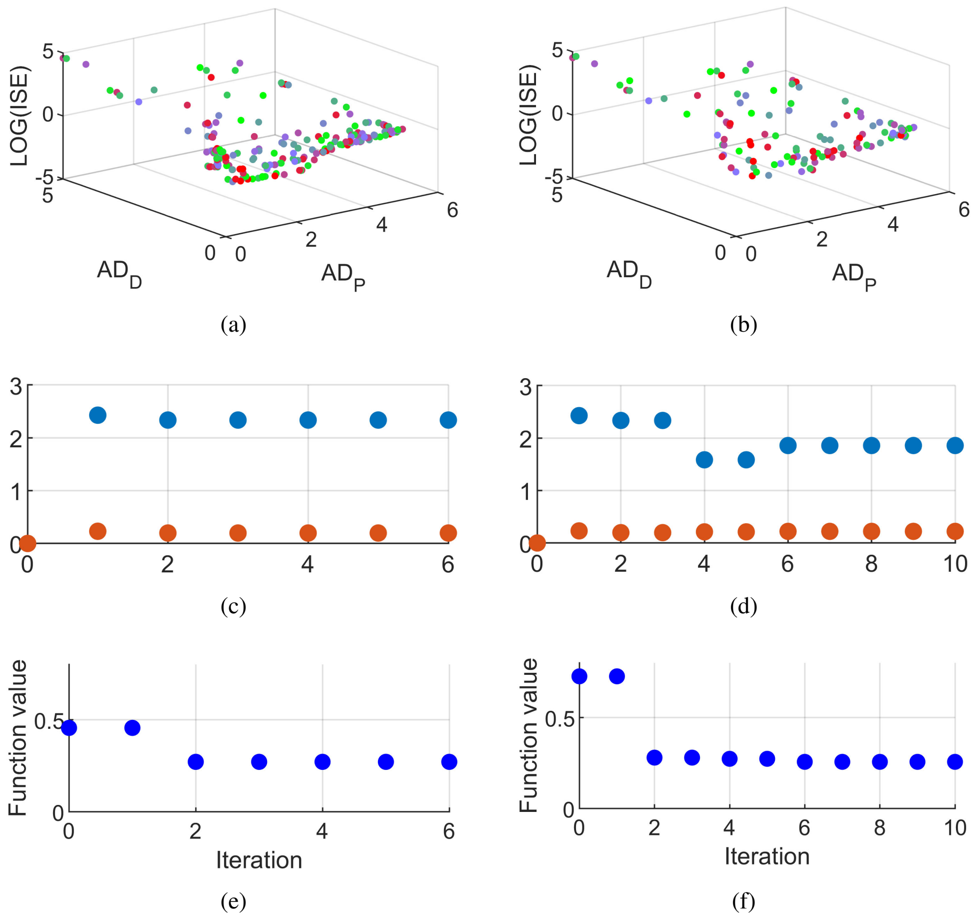


Fig. 10. Points in adaptation coefficients space (a, b); best values of adaptation coefficients in each iteration (c, d); best ISE value for each iteration (e, f). Cases (a, c, e) are the cases, where initial weight coefficient matrix is reproduced from IP controller, cases (b, d, f) are for the case where initial matrix was zeroed

## 6. Results

Underneath chosen simulation results are presented. Firstly, transients of a reference (IP) controller versus a fuzzy controller (with and without the Petri layer) with a reproduced control plane are presented in Fig. 11. Afterwards, a system with the above fuzzy controller and fuzzy adaptive controllers with and without the Petri layer are shown (Fig. 12). In final Fig. 13, transients of the motor using a classical IP controller, fuzzy adaptive controller with adaptation coefficients optimized for an initial weight coefficient matrix reproduced from the classical IP controller, and a fuzzy adaptive controller with adaptation coefficients optimized for a zeroed initial weight coefficient matrix, are presented. All fuzzy controllers used for the simulations are ones optimized in the above section (with 2 inputs and 7 fuzzy membership functions per input).

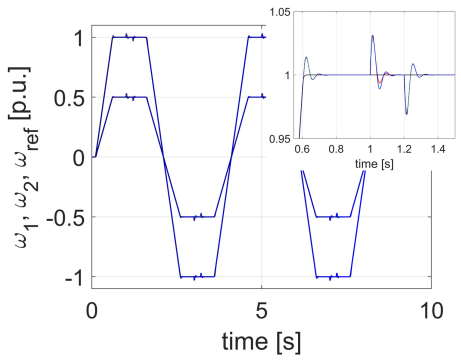


Fig. 11. Transients of reference model speed (black), motor speed using IP controller (red), fuzzy controller (green) and fuzzy controller with Petri layer (blue)

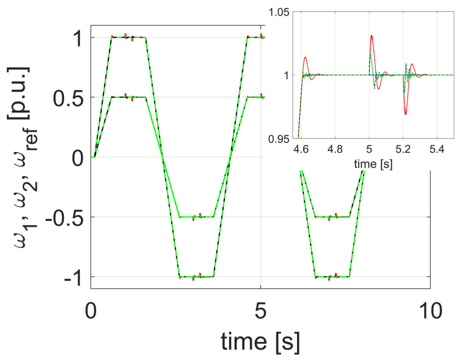


Fig. 12. Transients of reference model speed (black), motor speed using fuzzy controller (red), fuzzy adaptive controller (green) and fuzzy adaptive controller with Petri layer (blue)

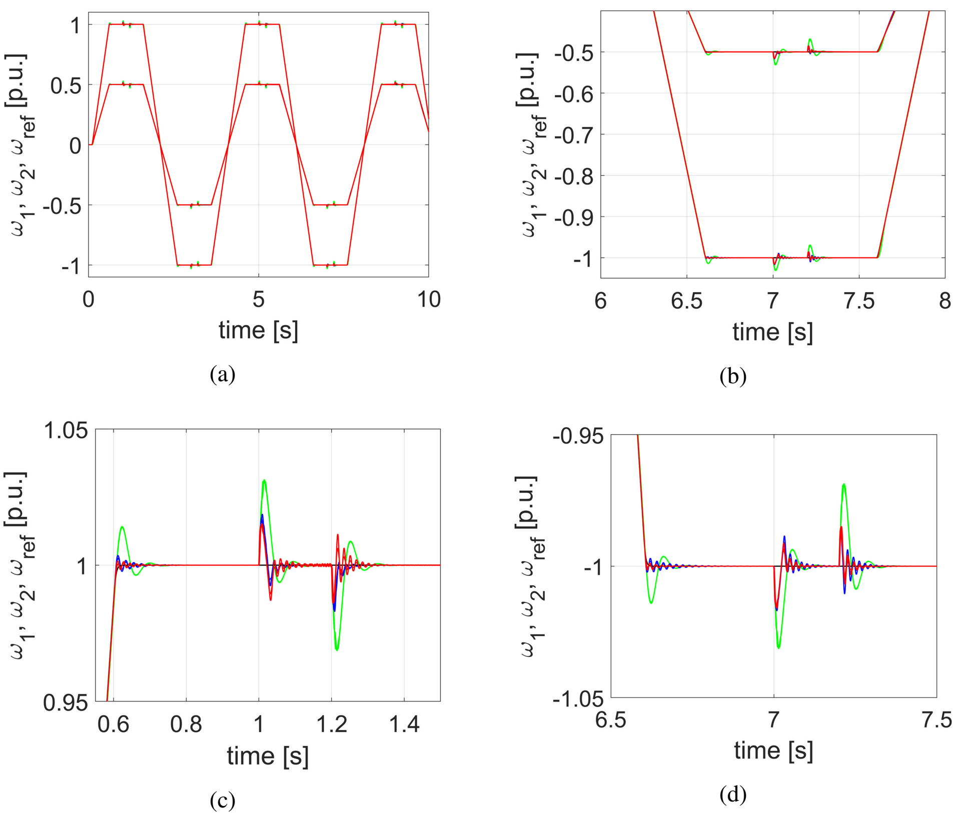


Fig. 13. Transients of reference model speed (black), motor speed using classical IP controller (green), fuzzy adaptive controller with adaptation coefficients optimized for initial weight coefficient matrix reproduced from classical IP controller (red) and fuzzy adaptive controller with adaptation coefficients optimized for zeroed initial weight coefficient matrix (blue)

As it is shown in Fig. 11, both fuzzy controllers with and without the Petri layer coincide perfectly and show slightly worse performance than the source IP controller. That is due to the inaccuracy described in the section *Methodology of settings selection*. This issue might be resolved by a bigger size base of rules or more fortunate composition of source and considered controller parameters.

Fig. 12 shows the transients of motor speed in the system with a fuzzy controller versus an adaptive fuzzy controller with and without the Petri layer. Again, the adaptive fuzzy controller with and without the Petri layer coincide perfectly, which confirms that the Petri layer does not affect an output control function. What is clearly seen, the adaptive controllers show better performance than the base controller. Only a parameter that might be considered as worse is a bigger number of oscillations when approaching reference speed. Nevertheless, the settlement time and amplitude of this oscillations are visibly smaller.

Fig. 13 shows the performance of a classical IP controller, fuzzy adaptive controller with adaptation coefficients optimized for an initial weight coefficient matrix reproduced from the classical IP controller and a fuzzy adaptive controller with adaptation coefficients optimized for an initial weight coefficient matrix set at zero.

As it is visible, adaptive controllers with optimized adaptation parameters show better performance than basic IP controllers. What is more, the controller optimized for the initial weight coefficient matrix reproduced from the IP controller shows better performance than the one with a matrix set at zero.

## 7. Experimental results

The experimental setup has been presented in Fig. 14. The shunt DC motor is fed with a Mentor DC drive, which is controlled by the STM32 Discovery board. The generic code for a microcontroller is created using the MATLAB Simulink environment.

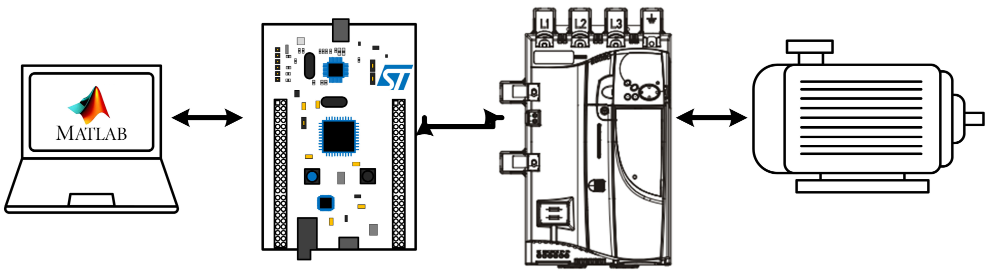
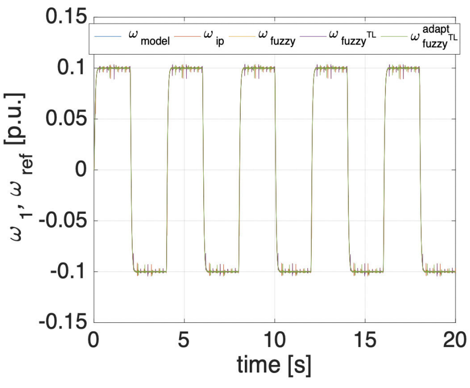


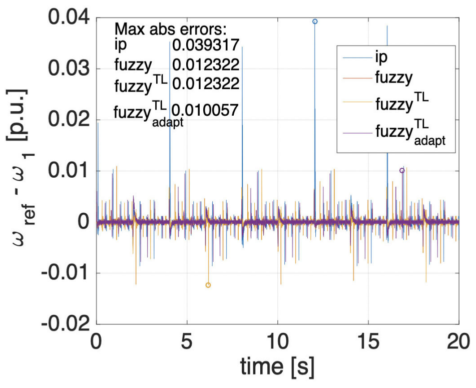
Fig. 14. Experimental setup

For experimental purposes the system presented in the experimental setup section was used. Fig. 15 shows the experimental results, as it is seen, the weight coefficients (c) change during the operation to obtain better fitting towards optimal function, minimizing the ISE quality indicator, yet since their initial values are reproduced from a basic controller which was properly tuned, the changes are small. The mentioned controller has 11 membership functions per each of

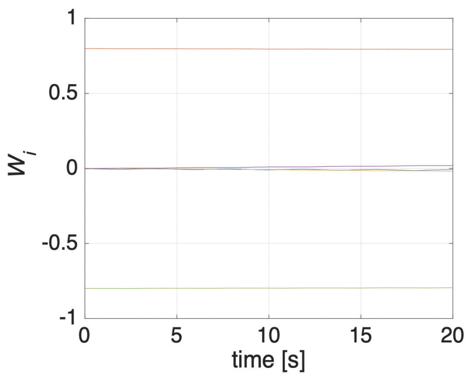
2 inputs, only few chosen weight coefficients are shown in the figure. The non-adaptive fuzzy controllers with and without a transition layer work identically. The adaptive controller offers best performance, and the original IP controller is the worst.



(a)



(b)



(c)

Fig. 15. Measured and reference speed (a); difference of real and measured speed (b) for different systems and chosen weight coefficients during operation of adaptive fuzzy system with transition layer (c)

The improvement of computational effort caused by the Petri transition layer has been considered in [9, 12]. In that paper authors proved that Petri layer implementation limits computational complexity of a control algorithm, and thus shortens the time of calculations.

## 8. Summary

For the purpose of this research a 2-mass system with an elastic shaft connecting two DC motors in MATLAB Simulink SimPowerSystem has been modeled. The classical cascade control structure has been used. The investigated controller worked in the outer control loop as a speed controller. Firstly, the classical IP controller has been implemented and treated as the base. It showed poor quality of control but ensured stable work of the system. Than the control function of the above-mentioned controller was reproduced on a fuzzy controller, and later on an adaptive fuzzy controller. We proved that it is possible to copy the control function, with some accuracy, from the base controller to the fuzzy controller.

Later on, the fuzzy controller was expanded with an adaptation algorithm. It was shown that the adaptive controller gives better results than the classical controller when full information about the process is unavailable. The adaptive controller was optimized using a PSO algorithm for two cases, the reproduced weight coefficient matrix and a the weight coefficient matrix set at zero. It turned out, which is in line with logic, that the system with the pre-tuned control function provides a better outcome.

The use of the PTL was caused by another fact in line with logic, the more rules the better possible fitting of the control function to desired one, but also more computational complexity. Incorporation of the PTL allows one to limit the rapid increase in computational complexity along with a higher number of rules.

## References

[1] Zadeh L.A., *Fuzzy sets*, Information and Control, vol. 8 no. 3, pp. 338–353 (1965).

[2] Demidova G.L., Lukichev D.V., Denisov K., *Implementation of Type-2 Fuzzy Control of PMSM Position Drive with Flexible Coupling*, International Conference on Control, Decision and Information Technologies, pp. 1917–1922 (2019).

[3] Chen Y.-C., Tu C.-H., Lin C.-L., *Integrated electromagnetic braking/driving control of electric vehicles using fuzzy inference*, IET Electric Power Applications, vol. 13, no. 7, pp. 1014–1021 (2019).

[4] Akcayol M.A., *Application of adaptive neuro-fuzzy controller for SRM*, Advances in Engineering software, vol. 35, no. 3–4, pp. 129–137 (2004).

[5] Civelek Z., Lüy M., Çam E., Barışçı N., *Control of pitch angle of wind turbine by fuzzy PID controller*, Intelligent Automation and Soft Computing, vol. 22, no. 3, pp. 463–471 (2016).

[6] Daode Z., Wei L., Hu X., Zhang C., Li X., *Research on torque ripple suppression of brushless DC motor based on PWM modulation*, Archives of Electrical Engineering, vol. 68, no. 4, pp. 843–858 (2019).

[7] Pradeep M., Sharmila B., Devasena D., Srinivasan K., *PID and PI$\lambda$D$\mu$ Controller Implementation for Speed Control Analysis of Two Mass Drive System*, 2018 International Conference on Communication and Signal Processing (ICCSP) (2018).

[8] Demidova G.L., Lukichev D.V., Brock S., *Fuzzy adaptive PID control for two-mass servo-drive system with elasticity and friction*, 2015 IEEE 2nd International Conference on Cybernetics (CYBCONF) (2015).

[9] Derugo P., Szabat K., *Implementation of the low computational cost fuzzy PID controller for two-mass drive system*, 16th IEEE International Power Electronics and Motion Control Conference and Exposition, pp. 564–568 (2014).

[10] Wai R.J., Liu C.M., *Design of dynamic petri recurrent fuzzy neural network and its application to path-tracking control of nonholonomic mobile robot*, IEEE Transactions on Industrial Electronics, vol. 56, no. 7, pp. 2667–2683 (2009).

[11] Derugo P., Szabat K., *Adaptive neuro-fuzzy PID controller for nonlinear drive system*, COMPEL – The International Journal for Computation and Mathematics in Electrical and Electronic Engineering, vol. 34, no. 3, pp. 792–807 (2015).

[12] Wai R.J., Chu C.C., *Robust petri fuzzy-neural-network control for linear induction motor drive*, IEEE Transactions on Industrial Electronics, vol. 54, no. 1, pp. 177–189 (2007).

[13] Derugo P., *Application of competitive and transition petri layers in adaptive neuro-fuzzy controller*, Power Electronics and Drives, vol. 1, no. 1, pp. 103–115 (2016).

[14] Derugo P., Szabat K., *Analysis of adaptive neuro-fuzzy PD controller with competitive Petri layers in speed control system for DC motor*, Computer Applications in Electrical Engineering, vol. 11, pp. 267–280 (2013).

[15] Li H., Wang J., Du H., Karimi H.R., *Adaptive sliding mode control for Takagi–Sugeno fuzzy systems and its applications*, IEEE Transactions on Fuzzy Systems, vol. 26, no. 2, pp. 531–542 (2017).

[16] Derugo P., Kacerka J., Jastrzębski M., Szabat K., *Linear motor control using an adaptive structure of fuzzy logic control PID*, Electrical Review (in Polish), vol. 91, no. 7, pp. 93–96 (2015).

[17] Reda T.M., HassanY.K., Elarabawy I.F., Abdelhamid T.H., *Comparison Between Optimization of PI Parameters for Speed Controller of PMSM by Using Particle Swarm and Cuttlefish Optimization*, IEEE Twentieth International Middle East Power Systems Conference, pp. 986–991 (2018).

[18] Kabziński J., Kacerka J., *Optimization of Polytopic System Eigenvalues by Swarm of Particles*, International Conference on Artificial Intelligence: Methodology, Systems, and Applications, pp. 178–185 (2014).