: *Journal of Electrical Engineering and Information Technologies,* Vol. **6,** No. 2, pp. 101–115 (2021)
Article 189
In print: ISSN 2545–4250
Received: December 10, 2021
On line: ISSN 2545–4269
Accepted: December 25, 2021
UDC: 681.515.8.082.52:621.311.243
DOI: https://doi.org/10.51466/JEEIT2162189101v

Original scientific paper

EXTENDED COMPARATIVE ANALYSIS OF DIFFERENT HELIOSTAT FIELD CONTROL ALGORITHMS

Ivan Andonov, Vesna Ojleska Latkoska, Mile Stankovski

Faculty of Electrical Engineering and Information Technologies,
"Ss. Cyril and Methodius" University in Skopje,
Rugjer Boškovic bb, P.O. box 574, 1000 Skopje, North Macedonia
vojleska@feit.ukim.edu.mk

A b s t r a c t: This study presents the use of various algorithms for control of a field of heliostats, through which a thermal power plant with concentrated solar energy is controlled. The design of the control algorithms consists of several steps. First, the Sun tracking alghorithm is presented for a specific location with an accuracy of ± 0.0003°. In order to obtain the mathematical model of the system, the real system is identified according to the gray box and the least-square method. The data used to identify the system is generated by step excitation on the real system, for a specific sampling period. The resulting mathematical model is used to design and simulate a continuous and discrete Proportional-Integral-Derivative (PID) controller, Mamdani and Sugeno fuzzy logic controllers, as well as ANFIS based fuzzy logic controller. The results of the applied controllers are analyzed and compared, based on the output overshoot, the rise and settling time. It can be concluded that we got best results (least settling time and the least overshoot) when fuzzy logic controller with ANFIS was used, while in terms of speed and rise time, the best results were obtained when discrete PID control algorithm was used. The study is extension of the work given in [23].

Key words; heliostat; Sun tracking algorithm; least square; PID; fuzzy logic conrol; adaptive neuro-fuzzy inference system (ANFIS)

КОМПАРАТИВНА АНАЛИЗА НА РАЗЛИЧНИ АЛГОРИТМИ ЗА УПРАВУВАЊЕ НА ПОЛЕ ОД ХЕЛИОСТАТИ

А п с т р а к т: Во овој труд се презентирани различни алгоритми за управување на поле од хелиостати преку кое се управува термоелектрана со концентрирана соларна енергија. Дизајнот на управувачките алгортми се состои од неколку чекори. Најпрво е претставен алгоритмот за следење на Сонцето за конкретна локација со точност од ±0.0003°. Со цел да се добие математички модел на системот, направена е идентификација на реалниот систем според методот на сива кутија и најмали квадрати. Податоците кои се користат за идентификација на системот се генерирани со отскочна возбуда врз реалниот систем и одреден периода на семплирање. Добиениот математички модел се користи за дизајн и симулација на континуиран и дискретен PID управувач, фази логички управувач на Мамдани и Сугено, како и на фази логички управувач базиран на ANFIS. Резултатите од применетите управувачи се компаративно анализирани врз основа на прескокот, времето на пораст и времето на смирување. Може да се заклучи најдобри резултати (најкратко време на смирување и најмал прескок) дека се добиени кога се користи фази-логички управувач со ANFIS, додека во поглед на брзината и времето на пораст најдобри резултати има при користење на дискретен PID управувач. Трудот е проширување на истражувањата изложени во [23].

Клучни зборови: хелиостат; алгоритам за следење на Сонцето; најмали квадрати; PID; фази логичко управување; адаптивен неуро-фази механизам на заклучување (ANFIS)

1. INTRODUCTION

Heliostat is a motorized mirror which is used to reflect the solar radiation into a receiver mounted on a tower. Heliostats are a basic component of a thermal power plant with a tower and even 40–50% of the cost of the entire plant. Recently, more and more investments are being made in research and analysis to reduce the cost of heliostats and thus the plant itself [1], [2].

The aim of heliostat is to reflect the sunlight on predefined target and therefore, for heliostat control, the orientation of the mirror needs to be known in order to determine deviations and precisely control the actuators to minimize the error of the overall system. The conventional approach uses open-loop calibration and control, while modern solutions use feedback sensors and closed-loop control [19]. A prerequisite for open loop control is very small statistical error or backlash and stable, observable system behavior. Calibration effort can be high with error systems. In addition, the heliostat geometry model must be appropriate to describe the real imperfections that can be changed [18].

Since the axis of each heliostat is driven by an electric motor, much of the heliostat control challenge comes down to control of the motor. An electric motor is a device that converts electrical energy into mechanical energy. The principle of working is the interaction between the magnetic fields generated by the stator or motor rotor magnets and the magnetic field created by the electric current in the windings, which generates a force in the form of torque on the motor axis. In applications such as the heliostat, DC motors are most commonly used for several reasons: higher starting power and torque required for the minimum heliostat rotation time for a certain angle, faster start-up response time, stopping or acceleration, which makes them more accurate and easier to control, simpler to install and cheaper.

The DC motor is controlled by changing the voltage with a PWM signal from the controller. The most commonly controlled variables are speed and position, and 95% of industry applications use a PID controller [7]. However, in processes where the dynamics change due to nonlinearity and interference, traditional PID controllers can not cope and system oscillations may occur due to precisely (crisp) adjusted controller parameters. The fuzzy logic controller is a good alternative to the PID controller, as it can handle nonlinear systems and can be designed using human operator knowledge without knowing the mathematical model of the system. Although the fuzzy logic controller does not have a better response in the time domain than the PID controller, it can still be applied to systems that have rapid changes, unlike PID which will need to adjust the values of the control parameters [20].

The main limitations of fuzzy logic controllers are the lack of a systematic design methodology and the difficulty in predicting the stability and robustness of a controlled system. Therefore, in many applications, fuzzy logic controllers are improved by fine tuning with the help of neural networks, i.e. the so-called hybrid fuzzy-neural controllers, which use a neural network to determine the rules and to make a conclusion [7].

This paper presents the open-loop control of heliostats as a systems, (Sun tracking algorithm and encoders) based on closed-loop position control of the DC motor explained in [19].

In order to make a comparative analysis of the above-mentioned different control algorithms in DC motors, i.e. indirectly in those used in the heliostat system, the paper performs design and selection of control algorithm and strategy for control, described in detail. Several types of control algorithms are used to control the heliostat position: PID, fuzzy-logic, and ANFIS (combination of fuzzy-logic control and neural networks), after which a comparative analysis is made.

In order to achieve the aforementioned control goal, several steps are used to design and select the control algorithm, i.e. (Figure 1):

1) identification of the real system and obtaining a mathematical model;
2) control algorithm design;
3) implementation on the mathematical model.

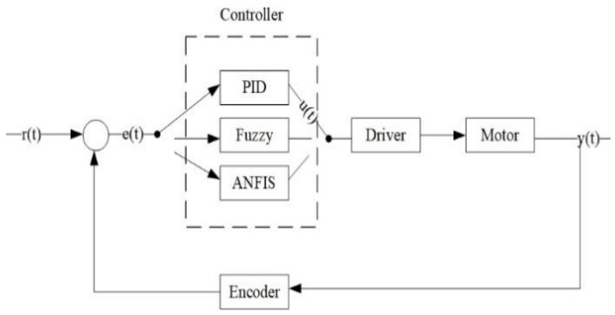

Fig. 1. Heliostat control algorithms

Analysis of the obtained results and selection according to the overshoot, settling time and rise time.

2. SUN TRACKING ALGORITHM

This paper presents the algorithm for tracking the Sun, i.e. the procedure for calculating the solar angles, azimuth and zenith, with an error of ±0.0003° in the period from -2000 to 6000. The algorithm was developed by the National Renewable Energy Laboratory (NREL) based on the book Astronomical Algorithms. The NREL report summarizes the complex elements of the algorithm contained in the book to calculate the solar position, while modifying the algorithm to accommodate solar applications [21]. For example, in Astronomical Algorithms the azimuth angle is measured west to south, but in solar applications it is measured east to north. Also, the longitude of the observer is considered positive west or negative east of Greenwich, while for solar applications it is the other way around (Figure 2).

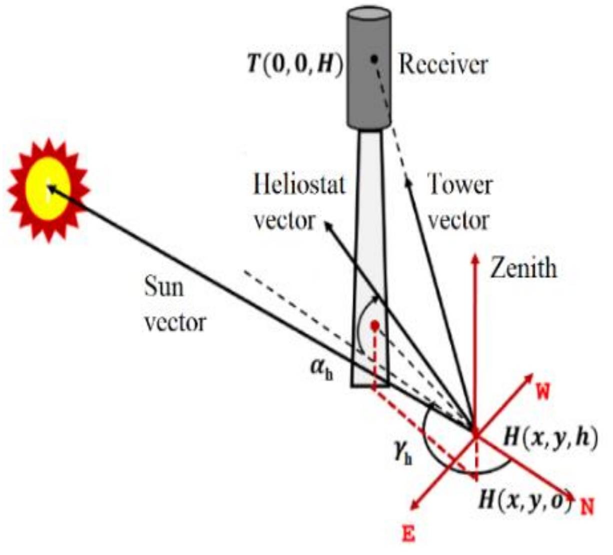

Fig. 2. Different vectors used in Sun tracking algorithm

The algorithm presented by NREL is extensive and contains many parameters for the correction of time and anomalies of the Earth, the Sun, the atmosphere and other influencing factors, but to obtain the angles of the solar vector $\vec{S}$ (γ_s,α_s) the general equations are used:

$$\delta = 23.45 \times \sin\left[\left(\frac{360}{365}\right) \times (284 + n)\right] \quad (1)$$

$$\omega = 15 \times (t - 12) \quad (2)$$

$$\alpha_s = \sin^{-1}[\cos(\varphi)\cos(\delta)\cos(\omega) + \sin(\varphi)\sin(\delta)] \quad (3)$$

$$\gamma_s = \cos^{-1}\left[\frac{\sin(\delta)\cos(\varphi) - \cos(\delta)\cos(\omega)\sin(\varphi)}{\cos(\alpha_s)}\right] \quad (4)$$

If $\sin(\omega) > 0$, then $\gamma_s = 360 - \gamma_s$, otherwise it remains the same.

Calculating the heliostat rotation angles requires the heliostat vector $\vec{H}$ (γ_h,α_h) which is the normal vector of the heliostat mirror and is obtained using the solar vector and the receiver vector $\vec{T}$ (γ_t,α_t). If we assume that the field of heliostats together with the receiver is a coordinate system with center in the receiver (0, 0, H) so that the positive part of the *x* axis is east and the positive part of the *y* axis is north, then each heliostat has certain coordinates with respect to the receiver . The height of the receiver *H* and the heliostats *h* do not change and are 35 m and 1.8 m, respectively. The angles of the heliostat vectors are constant for each heliostat and they change only by changing its coordinates, i.e. the location of the heliostat field. For a heliostat at a distance of 40 m east and 40 m north, the receiver angles are obtained:

$$d = \sqrt{x^2 + y^2} = 56.57 \text{ m} \quad (5)$$

$$\alpha_t = \tan^{-1}\left(\frac{H - h}{d}\right) = 30.41\,° \quad (6)$$

$$\gamma_t = 180\,° + \tan^{-1}\left(\frac{x}{y}\right) = 225\,° \quad (7)$$

The heliostat vector $\vec{H}$ is calculated by converting the angles of the solar vector and the receiver vector into three-dimensional coordinates (*x*, *y*, *z*). So the two angles of the solar vector get the coordinates [22]:

$$\begin{cases} x_1 = \cos(\alpha_s) \times \cos(\gamma_s \times -1) \\ y_1 = \cos(\alpha_s) \times \sin(\gamma_s \times -1) \\ z_1 = \sin(\alpha_s) \end{cases} \quad (8)$$

The two angles of the receiver vector get the following coordinates:

$$\begin{cases} x_2 = \cos(\alpha_t) \times \cos(\gamma_t \times -1) \\ y_2 = \cos(\alpha_t) \times \sin(\gamma_t \times -1) \\ z_2 = \sin(\alpha_t) \end{cases} \quad (9)$$

The angles of the heliostat vector are calculated using the coordinates (*x*, *y*, *z*):

$$\begin{cases} x = \dfrac{x_1 - x_2}{2} + x_2 \\ y = \dfrac{y_1 - y_2}{2} + y_2 \\ z = \dfrac{z_1 - z_2}{2} + z_2 \end{cases} \quad (10)$$

$$\alpha_h = \sin^{-1}\left(\frac{z}{\sqrt{x^2 + y^2 + z^2}}\right) \quad (11)$$

$$\gamma_h = \tan2^{-1}(y \times -1, x) \quad (12)$$

Since the function $\tan2^{-1}$ limits the angle γ_h for values from –180° to 180°, the negative values are shifted by 360°, thus obtaining the angles shown in Figure 3.

The angles α_h and γ_h can be further adjusted so that the angle γ_h instead of the zero position is the north axis, will be the south axis which reduces the movement (figure 4). In addition to this, the elevation angle α_h can be moved by 90° so that the zero position will be the zenith axis (also called the heliostat parking position used in strong winds).

Of particular interest is the change in angles at intervals of 5 s when updating the heliostat position. From Table 1 it can be seen that there is a period when the change in one of the angles (in this case α_h) is very small and the actuator can not move due to the encoder resolution, i.e. the minimum rotation angle. In this case, the microcontroller does not move the actuator until the difference in angle change is equal or greater than the minimum encoder resolution. Also, when the encoder pulses are a decimal number, they are rounded to an integer so that the difference is stored in the microcontroller's memory for later compensation

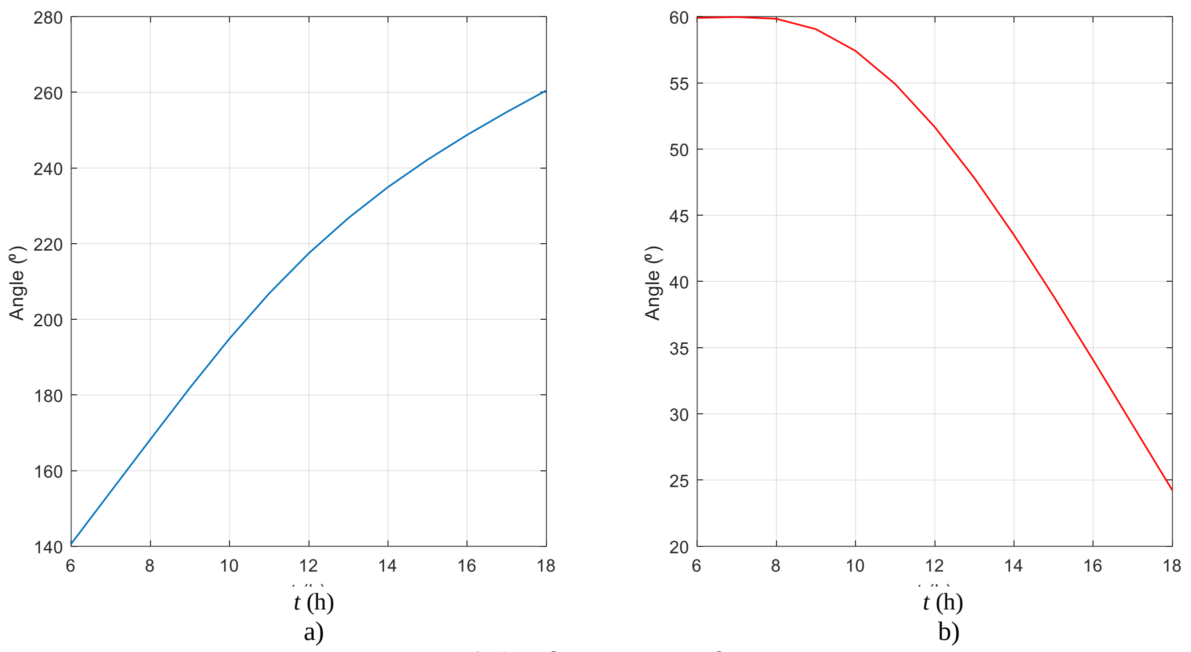

Fig 3. Heliostat vector angles
a) Azimuth heliostat angle. **b**) Elevation heliostat angle

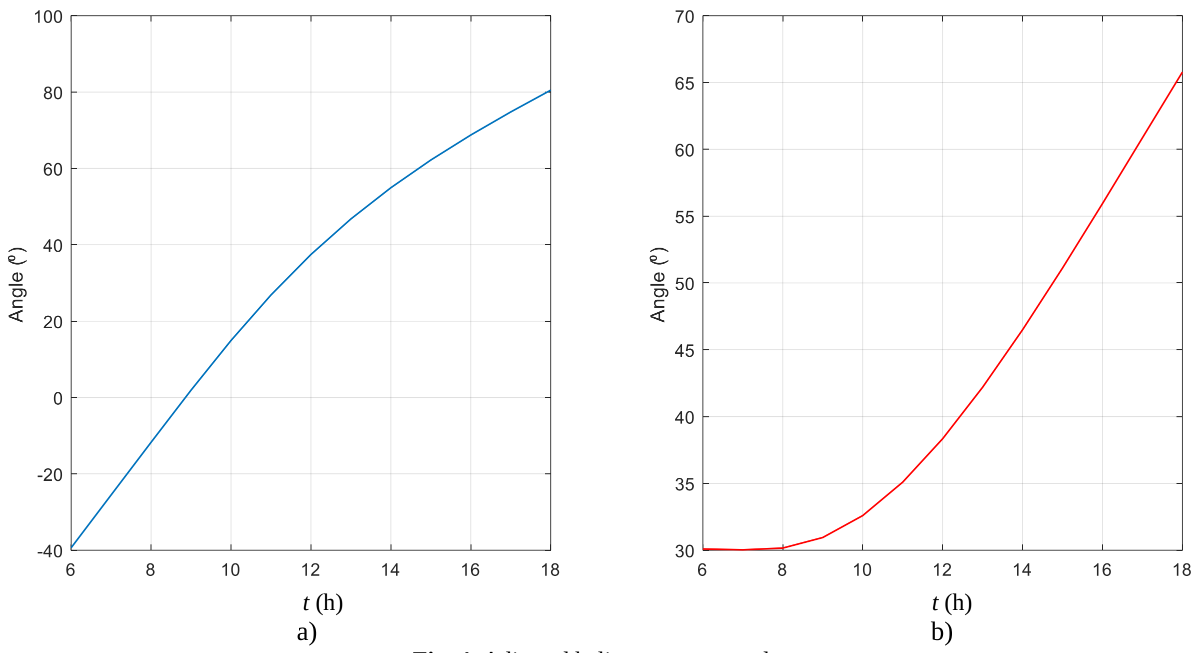

Fig. 4. Adjusted heliostat vector angles
a) Azimuth heliostat angle. **b)** Elevation heliostat angle

Table 1

Difference in the angles for period of 5 s

Time	γ_h	Difference	Min. angle	Encoder impulses
7:00:00	–25.52435	/	/	0
7:00:05	–25.50521	0.019132	0.0124	1.54 ~ 2
	α_h			
7:00:00	30.029853	/	/	0
7:00:05	30.029754	0.000099	0.0105	0.09~0

3. IDENTIFICATION

Mathematical models are commonly used to describe system behavior, and thus to simulate and design a controller. Depending on the knowledge (a priori information) about the system, obtaining the mathematical model can be done in the following ways [4]:

- Using white-box method – the mathematical model is obtained by applying the physical principles of modeling the system, while it remains to determine the most commonly given parameters.
- Using gray-box method – modeling occurs with the development of state space models with a known structure. For a given input/output system, there are infinite realizations with spatial variables that give the same connection for a given input/output. However, a particular structure may be desirable for identification. The limited optimization of the model parameters provides the necessary framework for the identification of this method.
- Using black-box method – the modeling uses input/output data without prior knowledge of system behavior. The mathematical model is obtained using neural networks and algorithms for their optimization.

Figure 5 shows the schematic and block diagram of a permanent magnet DC motor. By applying the laws of physics, the mathematical models is obtained, i.e. the transfer function of the system. The detailed parameters of the PMDC motor used are not known, so the gray box method is applied for their identification (although it can be said that there is not much difference with the black-box method except the limitation).

The Laplace transformation is applied to the obtained mathematical model, which gives the transfer function that represents the relationship between the voltage and the rotational speed of the PMDC motor [13]:

$$G_1(s) = \frac{\omega_m(s)}{v_a(s)} = \frac{1/k_b}{t_m t_e s^2 + t_m s + 1}, \qquad (13)$$

where $t_e = L_a/R_a$ is an electrical time constant, $t_m = RJ/k_t k_b$ is the mechanical time constant, k_t and k_b are torque constants and the back voltage EMF. Mathematically, velocity is a derivative of position with respect to time, which means that position is obtained by integrating velocity with respect to time [15]:

$$\theta_m(t) = \int \omega_m(t)dt \qquad (14)$$

which in s domain means multiplying the transfer function by s^{-1}:

$$G(s) = \frac{\theta_m(s)}{v_a(s)} = \frac{1}{s} \times G_1(s) = \frac{1/k_b}{t_m t_e s^3 + t_m s^2 + s} \qquad (15)$$

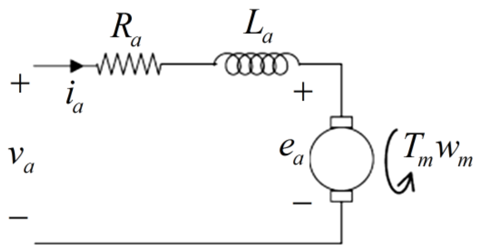

Fig. 5. Schematic drawing of PMDC motor

The same principle is used to obtain the mathematical model and the transfer function of the linear actuator. In [14] the obtaining of the second order portable f for the velocity/voltage ratio is shown, while in [16] the third order transfer function is obtained which later due to a simpler analysis is approximated to the first order function.

The identification of the parameters of the transfer function is performed by applying the least-squares method on the output data that are generated with step excitation. The purpose is to give N output data of the variables $y = \left[y[0], y[1] \cdots, y[N-1]\right]^T$ to get the best prediction (or approximation) of y using p descriptive variables (or regressors)) φ_i $[k]$, for $i = 1,\ldots, p$, so that the predictions $\hat{y} = \left[\hat{y}[0], \hat{y}[1] \cdots, \hat{y}[N-1]\right]^T$ are collectively at minimum (vector) distance from y [4]. Assume that the approximation of $y[k]$ is through a linear model:

$$\hat{y} = \sum_{i=0}^{p} \theta_i \varphi_i[k] = \varphi^T[k]\theta\,, \qquad (16)$$

where θ is the unknown set of free parameters that need to be optimized to achieve the goal of the smallest squares. We introduce:

$$\Phi = \left[\varphi[0], \varphi[1] \cdots, \varphi[N-1]\right]^T \tag{17}$$

$$Z = y\ U\ \Phi \tag{18}$$

since each $\varphi[k]$ is a $p \times 1$ vector, Φ is an $N \times p$ matrix. The Z matrix consists of known data. Then, the optimization problem can be written as:

$$\min_{\theta} J_N(Z, \theta) = \left|\left|y - \hat{y}\right|\right|_2^2 = (y - \hat{y})^T (y - \hat{y}), \tag{19}$$

where $\hat{y} = \Phi\theta$. Finding the minimum can be achieved by the method of descending gradient $\partial J / \partial\theta = 0$:

$$\frac{\partial J}{\partial \theta} = -2\Phi^T(y - \Phi\theta) = 0 \tag{20}$$

$$\hat{\theta} = (\Phi^T\Phi)^{-1}\Phi^T y \tag{21}$$

where $\hat{\theta}$ represents the minimum obtained from the optimization.

Step excitation of 15 % (3.6 V) of the rated voltage (24 V) is used to obtain the output data y at input due to nonlinear reduction, while displacement and velocity are measured with the microcontroller ESP32 and encoder with a sampling period of 50 ms (20 Hz) [3]. The heliostat is mounted without mirrors, so that the instantaneous mass that affects the movement of the DC motor is 85 kg (horizontal pipe and construction), which is almost half of the total mass of 180 kg (Figure 6).

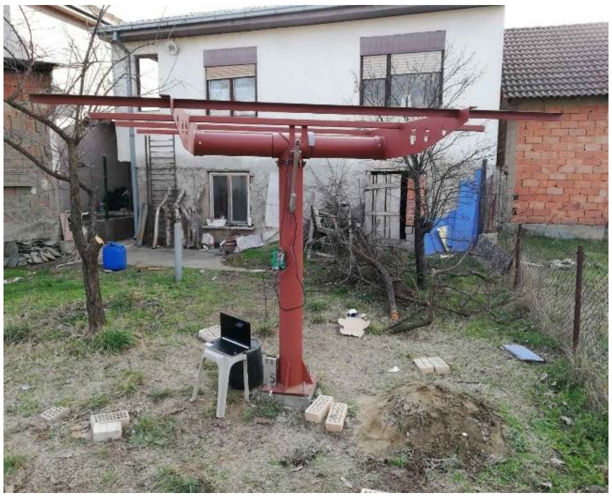
Fig. 6. Real system used for identification

The parameters of the tested PMDC motor and gears are given in Table 2.

Table 2

Parameters of PMDC motor

Voltage, V	24
Output power, W	40
No-load speed, rpm	3000
No-load current, A	1 max
Load, mNm	136
Load speed, rpm	2800
Load current, A	2.6
Gear ratio	1:180
Reduction speed, rpm	16
Max. load, Nm	10

By using MATLAB, the curves from the measured data and the identification of the system are calculated, and shown in Figure 7. The accuracy of the obtained transfer function of the identification system is 87.4 % and it is given by:

$$G_1(s) = \frac{\omega_m(s)}{v_a(s)} = \frac{0.4055}{0.0265s^2 + 0.4741s + 1} \tag{22}$$

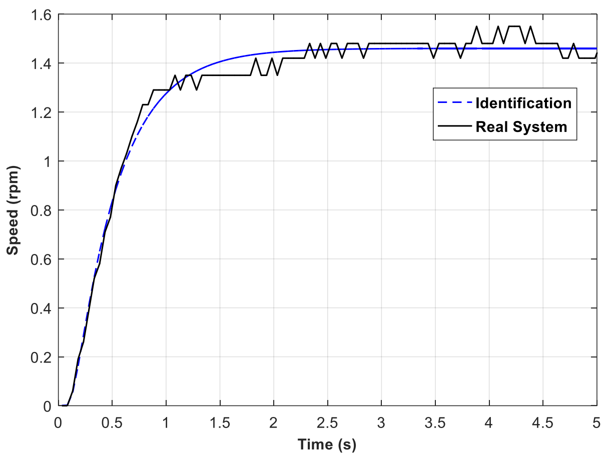

Fig. 7. Real and simulated system output with 15 % step input

4. PID CONTROL ALGORITHM

The PID controller is the most widely used control algorithm. Most feedback loops are controlled by this algorithm or small variations of it. It can be implemented in various forms, as a standalone controller, as part of a DDC (direct digital control) or hierarchically distributed process control system. The mathematical representation of the PID controller is [5]:

$$u(t) = K_p\left(e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d\frac{de(t)}{dt}\right), \quad (23)$$

where u is the control signal and e represents the control error ($e = r\text{–}y$). The control signal is a set of three terms: P-term is proportional to the error, *I* is proportional to the error integral, and D-term is proportional to the derivation (change) of the error. The control parameters are the proportional coefficient K_p, the integration time T_i, and the differentiation time T_d.

In cases where only proportional control is used, the control algorithm is represented only by $u(t) = K_pe(t)$, which means that the control signal is proportional to the error. The change of the coefficient K_p affects the change of the system error in the steady state and the occurrence of oscillations and overshoot. Thus, increasing K_p reduces the error in the steady state, but increases the response oscillations [5].

The main function of the integration term *I* is to ensure that the response of the system matches the reference value in the steady state. With proportional control, there is a steady error, while with the *I*-term, a small positive error will always lead to an increase in the control signal, and a negative error will give a decreasing control signal. In cases where the integration time $T_i = \infty$, the PI control combination switches to P control only. The steady state error is removed when T_i has finite values. For large values of integration time, the rise time is large, while for small values of T_i the rise time of the response is shorter, but oscillations and overshoot occur (increase of settling time). In the *I* term, the problem arises from the limitations of the physical systems (actuator length, limited speed, high latency) which leads to a situation where the controlled system reaches the limit, and the term continues to integrate the error and increases (windup). Then the error needs to have the opposite sign for a longer period of time for the control signal to return to normal. The consequence is that any controller with integrated action can cause major changes when the system is saturated (reaches the limit). This problem can be overcomed in several ways:

- limiting changes of the reference value;
- back-calculation – when the output is in saturation state, the control term is also recalculated so that its new value gives an output at the saturation limit. And the term is reset dynamically with time constant $\mathrm{T_t}$;
- tracking – another input is added to the controller which is a tracking signal and is followed by the control signal;
- conditional integration – the term is also excluded when the control is far from stady state and thus the term is used under certain conditions, otherwise it is constant.

The *D* term is used to improve the stability of a closed loop. Usually due to the dynamics of the process, it takes time to notice the change of the control variable in the response. This will cause the control system to be delayed in correcting the error. The action of the PD control can be described so that the control is proportional to the predicted response of the system, where the prediction is made by extrapolating the error from the tangent to the error curve. It can be said that the term *D* is used to predict the error in the future. The disadvantage of using the term *D* is that the ideal output has a very high coefficient for high frequency signals. This means that the high frequency measuring noise will generate large variations of the control signal. This problem is overcome by implementing a first-order filter with a time constant T_d/N. Thus, for small s the transfer function is approximately sK_pT, and for large s it is equal to K_pN. The approximation acts as a derivative for the low frequency components of the signal, and the high frequency coefficient is limited to K_pN. Thus, high frequency measurement noise is amplified mostly by the K_pN factor. The obtained transfer function for the PID controller is:

$$C(s) = K_p\left(1 + \frac{1}{sT_i} + \frac{sT_d}{1 + \frac{sT_d}{N}}\right) \quad (24)$$

In some cases, instead of filtering the *D* term, it is possible to filter the measured signal, which guarantees that the high frequency noise will not produce large control signals (high frequency roll-off).

The adjustment of the PID parameters is done with the frequency response method of Ziegler-Nichols. This method is the second of the two classical methods for determining the parameters of PID controllers presented by Ziegler and Nichols in 1942 and is used to adjust the parameters in a closed loop. These methods are still widely used, in their original form or with some modification. They are often the basis of adjustment procedures used by controller manufacturers and the processing industry. The methods are based on determining some characteristics of the process dynamics. The control parameters are then expressed in terms of features with simple formulas. These methods have a major impact on the practical adjustment of the PID controller even if they do not result in a good setup. Additional extensions of the method are presented in [5]. It is often necessary to supplement the design method with manual adjustment to obtain the desired behavior of the closed loop. The method consists of calculating the critical (limit) values of the parameters K_c and T_u with which the system is on the edge of stability, i.e. oscillates. K_c represents the critical amplification, while T_u represents the period of one oscillation. K_c is calculated using the Ruth-Hurz criterion for closed system stability at K_i, $K_d = 0$ ($T_i = \infty$, $T_d = 0$) [17].

$$G(s) = \frac{322.7}{s} G_1(s) \tag{25}$$

$$T(s) = \frac{K_p G(s)}{1 + K_p G(s)} \tag{26}$$

The critical value is K_c <0.1369. Substitution of K_c gives ω_{cr} = 6,167 rad/s.

$$T_u = \frac{2\pi}{\omega_{cr}} = 1.0183 \text{ s} \tag{27}$$

Then, by applying the obtained values according to Table 3, K_p = 0.08214, T_i = 0.5092 and T_d = 0.1273 are calculated, while $K_i = K_p/T_i$ and $K_d = K_p \cdot T_d$.

Table 3

PID tuning with Ziegler-Nichols closed-loop method

	K_p	T_i	T_d
P controller	0.5 K_c	∞	0
PI controller	0.45 K_c	$\frac{T_u}{1.2}$	0
PID controller	0.6 K_c	$\frac{T_u}{2}$	$\frac{T_u}{8}$

The response of the system with the calculated PID controller and step excitation is shown in Figure 8. It can be noticed that there is a significant overshoot of 74.2% and a settling time of 4.5 s. For this purpose, additional manual tuning of the parameters of the PID controller is used, so that by increasing K_p the overshoot is reduced and the response is faster, while by increasing the T_d the oscillations and the settling time are reduced. The newly obtained PID controller has values for K_p = 0.1232 and T_d = 0.2546, so the overshoot is 36.1 % and the settling time is 2.5 s.

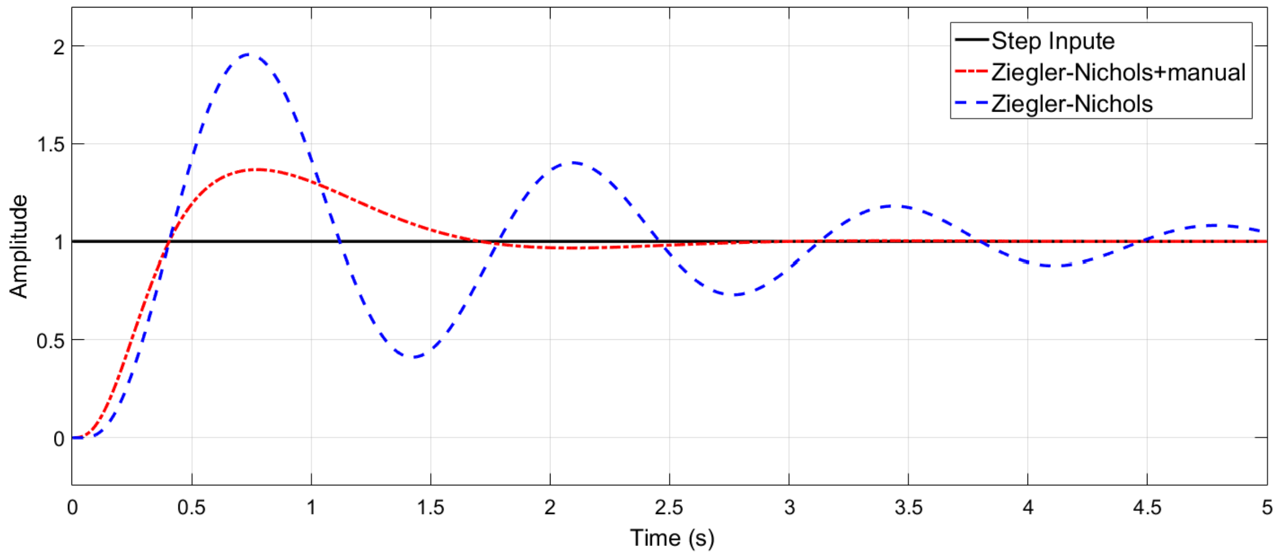

Fig. 8. Ziegler-Nichols and additional manual tuning

Due to the implementation of the microcontroller, the PID controller is transformed from a continuous to a discrete form, and then a differential equation is obtained [3]. When converting the PID controller, it is important to select the sampling period which should be at the least 10 times the system bandwidth [12]. The bandwidth of the closed system with PID controller is 14.5 rad/s, which means that the sampling period is T_0 <0.043 s. A shorter sampling period will adjust the system response faster based on the changes that have occurred. Therefore 10 ms (T_0 = 0.01 s) is used for the sampling period. The conversion from s-domain to z-domain is performed using the method of backward Euler calculation, so that for $s = \frac{z-1}{T_0 z}$ we get a controller of the form:

$$C(z) = K_p + \frac{K_i T_0 z}{z-1} + \frac{K_d(z-1)}{T_0 z} = \\ = \frac{3.263z^2 - 6.397z + 3.137}{z^2 - z} \quad (28)$$

From Figure 9 can be seen that the obtained discrete PID controller gives better results than the continuous controller, with an overshoot of 32.7 % and a settling time of 1 s.

Using a shorter sampling period (1 ms) gives better results, but reduces the performance of the microcontroller needed to perform calculations for other processes.

The discrete PID controller can be presented in the form:

$$C(z) = \frac{U(z)}{E(z)} = \frac{3.263 - 6.397z^{-1} + 3.137z^{-2}}{1 - z^{-1}}, \quad (29)$$

from which is obtained the differential equation:

$$u(n) = u(n-1) + 3.263\, e(n) - 6.397\, e(n-1) + \\ + 3.137\, e(n-2) \quad (30)$$

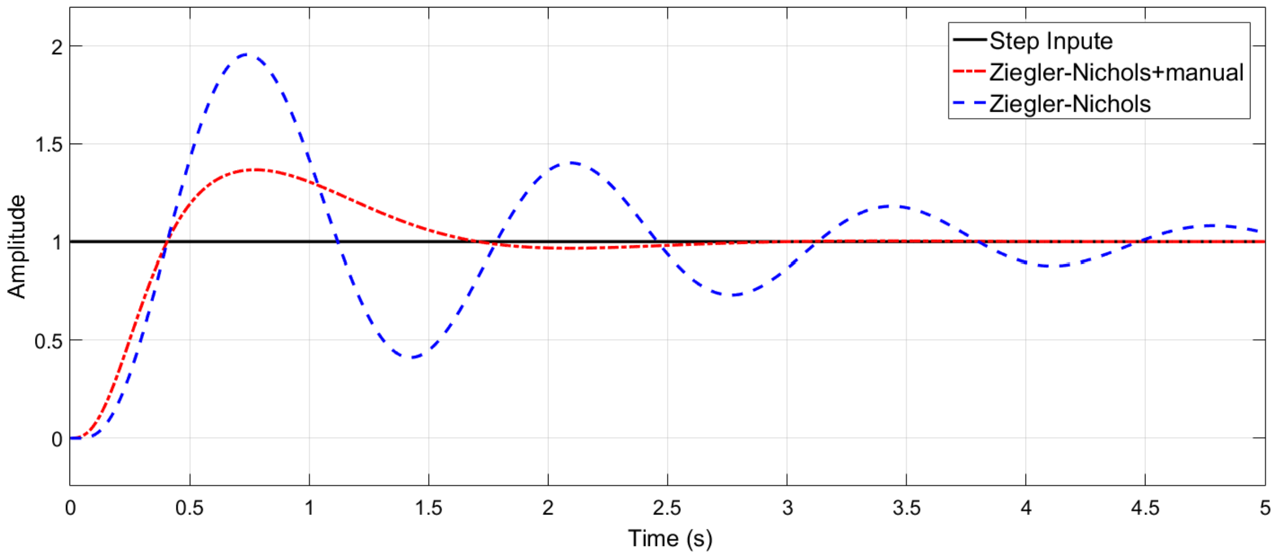

Fig. 9. System response with continuous and discrete PID controller

5. FUZZY LOGIC CONTROL ALGORITHM

In processes where the dynamics change as a result of nonlinearity and interference, conventional PID controllers can not cope and system oscillations may occur due to precisely adjusted control parameters. The fuzzy-logic controller is a good alternative to the PID controller, as it can handle nonlinear systems and can be designed using human operator knowledge without knowing the mathematical model of the system. Although the fuzzy logic controller usually does not have a better response in the time domain than the PID controller, it can still be applied to systems that have rapid changes, unlike PIDs that will need to adjust the values of the control parameters [6].

Fuzzy logic control is a control algorithm based on linguistic control, which derives from the expert knowledge applied in an automatic system control algorithm [7], [8]. The components of the fuzzy logic controller are: fuzzification, rule base, inference system and defuzzification (Figure 10).

- Fuzzification – converts all inputs to a membership function so that there is a degree of membership for each linguistic term referring to the input variable.
- Rule base — is a collection of rules that are usually in the format "if-Then" and formally the side "If" is called premise and the side "Then" is called a consequent. The computer is able to execute the rules and calculate the control signal depending on the inputs.

- Defuzzification – is the combination and conversion into a single output signal that is not fuzzy but crisp, which is the control signal of the system. The output signal depends on the rules of the system [9].
- Inference system – assesses which control rules should be ignited at a given moment and then decides what the control output signal will be. The most commonly used are Mamdani and Sugeno (Takagi-Sugeno) inference systems.
- The Mamdani inference system is based on Lotfi Zade's 1973 work on fuzzy algorithms for complex systems and decision processes that expects all output functions to be fuzzy sets. This inference system is intuitive, and widely accepted, better suited to human input, but the main limitation is that the calculation for the defuzzification process takes longer.
- Sugeno inference system is based on the Takagi-Sugeno-Kang fuzzy inference method, in their joint effort to formalize a systematic approach to generating fuzzy rules from a set of input-output data, which expects all affiliation functions to be singleton. This inference system is computer efficient, works well with linear techniques (PID control, etc.), works well with optimization and adaptation techniques, guarantees output surface continuity, and is more suitable for mathematical analysis. The results are very similar to the consequents from Mamdani's style.

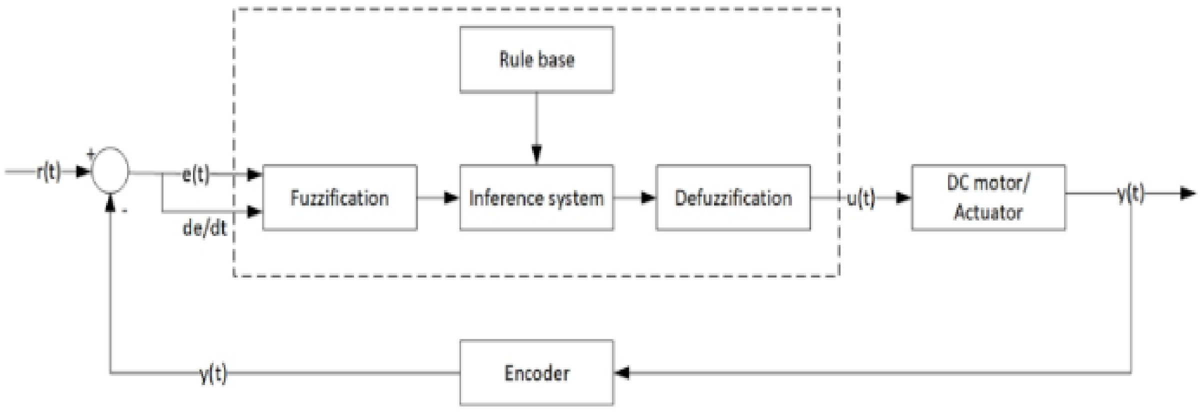

Fig. 10. Fuzzy logic controller

In this paper, the Mamdani inference system is used to design the fuzzy logic controller. The fuzzification of the two inputs is performed with 5 triangular and two Γ (trapezoidal) membership functions for each input respectively, which creates a base of 49 rules. The output is formed by five triangular and two Γ membership functions. The rule base is presented in Table 3, where N – negative, P – positive, B – big, S – small.

The membership functions are shown in Figure 11, where the ranges of the linguistic variables are graphically represented. The range [–2000 2000] is used for the error, while for the change of the error [–27 27], taking into account that at a maximum speed for 10 ms a maximum of 27 units can be passed, and the voltage [–23.7 23.7]. The inference system is based on composition so that the fuzzy relations that represent the meaning of each individual rule are merged into one fuzzy relation that describes the meaning of the whole set of rules.

The inference is performed through an operation of the composition of the fuzzy input and the fuzzy relation which represents the meaning of the whole set of rules. The result is a fuzzy set that describes the fuzzy value of the total control output [3] [10]:

$$\mu_u(u) = \max_{x} \min_{e,\Delta e}\left(\mu_{ant}(e,\Delta e), \mu_R(e,\Delta e,u)\right) \quad (31)$$

Table 3

Rule base

		Error e(t)						
Control voltage $u(t)$		BN	N	SN	Z	SP	P	BP
Change of error Δe	BN	BN	BN	BN	BN	N	SN	Z
	N	BN	BN	BN	N	SN	Z	SP
	SN	BN	BN	N	SN	Z	SP	P
	Z	BN	N	SN	Z	SP	P	BP
	SP	N	SN	Z	SP	P	BP	BP
	P	SN	Z	SP	P	BP	BP	BP
	BP	Z	SP	P	BP	BP	BP	BP

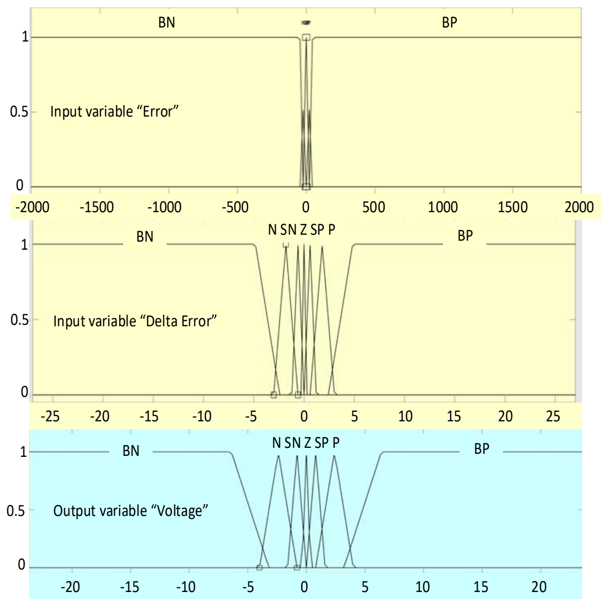

Fig. 11. Membership functions

The centroid method (center of gravity) is used for defuzzification. In continuous case the crisp value of the control signal is obtained with the following relation:

$$u = \frac{\int u * \mu(u)du}{\int \mu(u)du}, \tag{32}$$

While in discrete case with the relation:

$$u = \frac{\sum_{i=1}^{n} u_i * \mu(u_i)}{\sum_{i=1}^{n} \mu(u_i)}. \tag{33}$$

The resulting fuzzy logic controller with Mamdani inference system is used to generate fuzzy logic controller with Sugeno inference system which is more suitable for microcontroller implementation. Generation is performed using MATLAB (mam2sugeno or convertToSugeno) so that the newly obtained Sugeno inference system has constants as output membership functions. These constants are determined by the centroids of the output (consequential) membership functions of the original Mamdani mechanism while the input membership functions and the rules remain the same. The weighted-average method is used for defuzzufication, for product – implication, and for aggregation – sum. From Figure 12 can be seen that using the Mamdani inference system the system has a overshoot of 9.5%, and oscillates with an amplitude ± 0.05 around the reference value, while the oscillations decrease over time. This value is acceptable considering that the distance between two displacement units is 0.0176 mm, which in case of physical realization can occur oscillations due to the backlash of the reduction itself (gears) of the motor. Using the Sugeno inference system there are no oscillations and the response has a settling time of 1.1s and a overshoot of 15.9%.

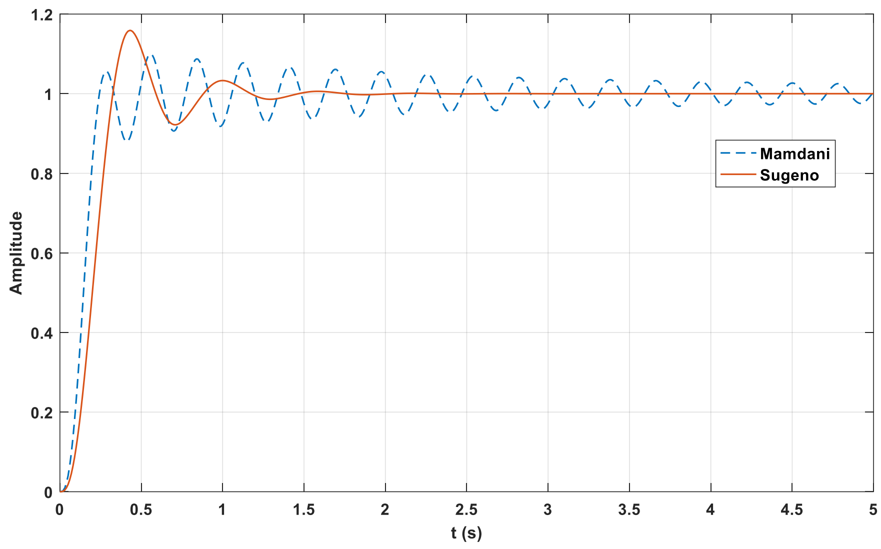

Fig. 12. System response with Mamdani and Sugeno inference systems

6. ADAPTIVE NEURAL FUZZY INFERENCE SYSTEM – ANFIS

The Adaptive Neural Fuzzy Inference System (ANFIS) is a combination of two computational methods, neural networks and fuzzy logic. Fuzzy logic has the ability to change the qualitative aspects of human knowledge and insights in the process of precise quantitative analysis. However, there is no defined method that can be used as a guide in the process of transforming human thought into a fuzzy inference system and also takes a long time to adjust to membership functions. Unlike fuzzy logic, neural networks have great capabilities in the process of adapting to their environment. Therefore, neural networks can be used to automatically adjust the membership functions and reduce the error rate in determining the rules in fuzzy logic [11].

A simple fuzzy inference system has limited learning (or adaptation) opportunities. If learning skills are required, it is convenient to place a fuzzy model within the supervised neural networks that can systematically calculate gradient vectors. Sugeno system is used for consequent and the typical fuzzy rule is:

IF *x* is *A* and *y* is *B*, then $z = f(x, y)$,

where *A* and *B* are fuzzy sets in the premise part and $z = f(x, y)$ is a sharp function in the consequent part. Typically, the *z* function is a first-order (moving single) or zero-order (constant single) fuzzy Sugeno model. An example of modeling a Sugeno first-order inference system is shown in Figure 13, which contains the following two rules:

Rule 1: IF *x* is A_1 and *y* is B_1 then $f_1=p_1x+q_1y+r_1$,

Rule 2: IF *x* is A_2 and *y* is B_2 then $f_2=p_2x+q_2y+r_2$.

the functionally equivalent supervised neural network in Figure 13 that follows the general design algorithm has one input layer, three hidden layers and one output layer, the meaning of which is:

Layer 1: Each adaptive node in this layer generates values for the membership of the input vectors A_i, for $i = 1, 2$. For example, the membership function of the i-node can be a generalized bell membership function:

$$O_i^1 = \mu_{A_i} = \frac{1}{\left[1 + \left|\frac{x - c_i}{a_i}\right|^{2b_i}\right]} \qquad (34)$$

where O_i^j denotes the output of the *i*-node in the *j*-layer, *x* is the input of node *i*, A_i are the input vectors connected to the *i*-node, and $\{a_i, b_i, c_i\}$ are the parameter set that changes the form of the membership function. The parameters in this layer are listed as the parameters of the premise part.

Layer 2: Each node in this layer is fixed and calculates the ignition power of a particular product rule. The output of each node represents the ignition power of the rule:

$$O_i^2 = w_i = \mu_{A_i}(x) \cdot \mu_{B_i}(y), \qquad i = 1, 2. \qquad (35)$$

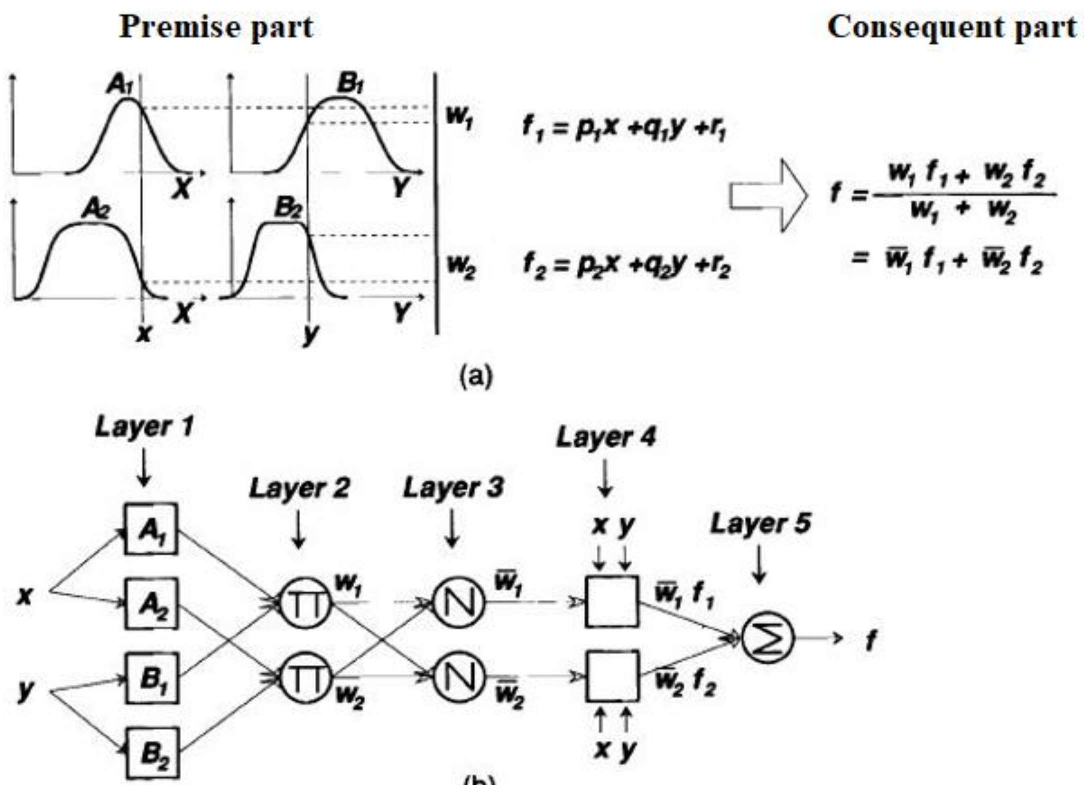

Fig. 13. a) Sugeno fuzzy logic inference system **b)** ANFIS

In fact, any other *T*-norm operator that performs fuzzy and operation can be used as a node function in this layer.

Layer 3: The fixed node i in this layer calculates the ratio between the ignition power of the i rule and the sum of the ignition powers of all the rules:

$$O_i^3 = \overline{w_i} = \frac{w_i}{w_1 + w_2}, \qquad i = 1, 2. \tag{36}$$

For simplicity, the outputs of this layer are also called normalized firing powers.

Layer 4: Adaptive node i in this layer calculates the contribution of i-rule to total output, with the following node function:

$$O_i^4 = \overline{w_i} f_i = \overline{w_i}(p_i x + q_i + r_i), \tag{37}$$

where $O_i^4$is the output of layer 4, and $\{p_i, q_i, r_i\}$ are the parametric set. The parameters in this layer are listed as parameters of the consequent part.

Layer 5: The only fixed node in this layer calculates the total output as the sum of the values of each rule:

$$O_i^5 = \sum_i \overline{w_i} f_i = \frac{\sum_i w_i f_i}{\sum_i f_i}. \tag{38}$$

The basic learning rule is a descending backpropagation gradient, which calculates the error signals (the change in the square error relative to the output of each node) recursively from the output layer back to the input nodes. This learning rule is exactly the same as the reverse propagation learning rule used in feedforward neural networks. The total output *f* can be expressed as a linear combination of the following parameters:

$$f = \overline{w_1} f_1 + \overline{w_2} f_2 = (\overline{w_1} x) p_1 + (\overline{w_1} y) q_1 + \\ + (\overline{w_1}) r_1 + (\overline{w_2} x) p_2 + (\overline{w_2} x) q_2 + (\overline{w_w}) r_2 \tag{39}$$

Based on Equation (27), the hybrid learning algorithm combines descending gradient methods and the least squares for optimal parameter search.

The steps used to obtain ANFIS are:

Draw the Simulink model with the fuzzy logic controller and simulate it with the given rule base.

The first step in designing ANFIS is to collect training and testing data while simulating the fuzzy logic controller.

The two inputs, i.e. $e(t)$ and Δe and the output signal u(t) provide the data for training and testing.

Use the anfisedit command in MATLAB to create the ANFIS .fis file or use the genfis1 and anfis commands.

The training data collected in step 2 is loaded and a inference mechanism is generated with selected membership functions (in this case triangular).

The collected data are trained with the generated inference system up to a certain number of epochs (iterations) and then tested [7].

In this paper, an ANFIS controller is designed based on the data obtained from the error signals, the error change and the control signal from a modified Sugeno inference system (the error change signal is multiplied by a coefficient of 1.5) from the previous section. Data were collected with a sampling time of 0.01s over a period of 5 s and divided into 80% for training and 20% for testing. 7 triangular membership functions are selected for both inputs, a hybrid learning algorithm and 100 epochs. Trial and error have shown that the best results for ANFIS are obtained by collecting data with excitement greater than about 25% of the single. By using a single step excitation the obtained ANFIS destabilizes the system with small changes in the data (errors). System response with Sugeno, modified Sugeno and ANFIS inference systems is given on Figure 14.

The data obtained from the system response simulations with the applied controllers on the DC motor are numerically presented in Table 4. It can be noticed that the least settling time and the least overshoot has the ANFIS fuzzy logic controller. In terms of speed and rise time, the discrete PID controller has the best results. This conclusions are also evident in Figure 15.

Table 4

Results of the applied controllers

	Rise time [s]	Settling time [s]	Overshoot [%]
Continuous PID + ZN	0.2	4.5	74.2
Continuous PID + ZN + manual tuning	0.3	2.5	36.1
Discrete PID	0.1	1.1	32.7
Fuzzy logic with Mamdani	0.1	5	9.5
Fuzzy logic with Sugeno	0.2	1.1	15.9
Fuzzy logic with modified Sugeno	0.3	0.8	0
Fuzzy logic with ANFIS	0.4	0.7	0

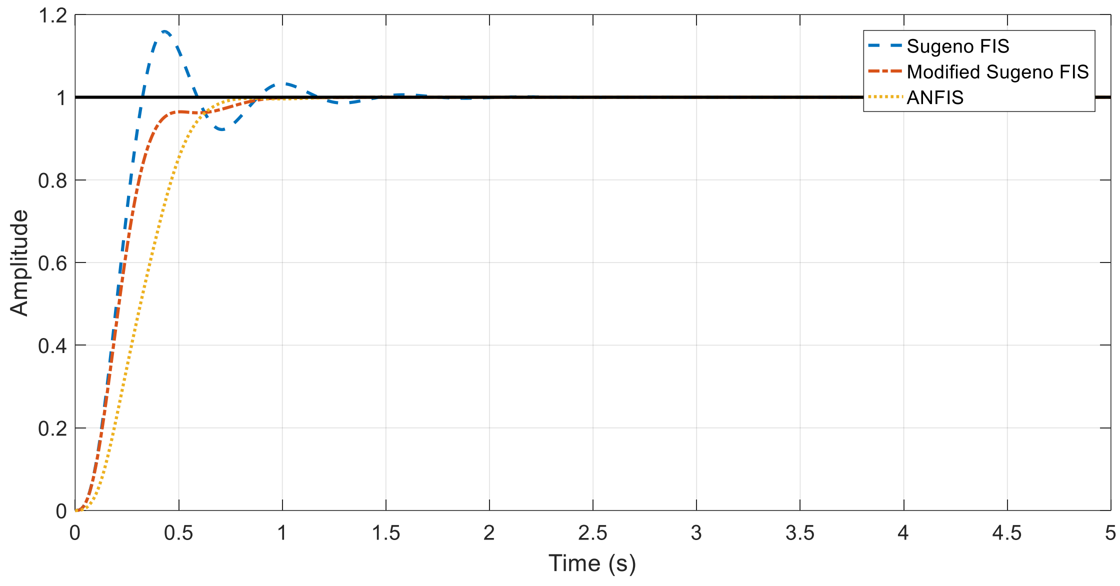

Fig. 14. System response with Sugeno, modified Sugeno and ANFIS inference systems

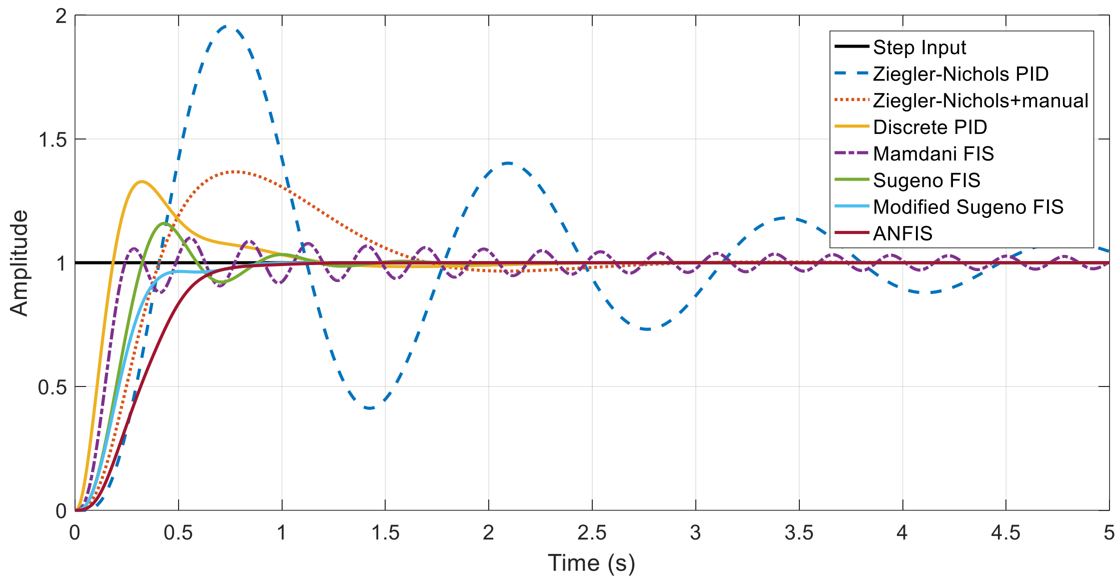

Fig. 15. System response with different type of controllers

7. CONCLUSION

This paper presented the design of a various control algorithms for the control of actuator as part of a heliostat. Real system was used for identification of the data and obtaining mathematical model. Furthermore, the mathematical model was used for design and simulation of continuous PID controller from which was obtained discrete PID controller. In addition to the PID, fuzzy logic controller was designed both with Mamdani and Sugeno inference systems. Fuzzy logic controller with Sugeno inference system was used for generating data for the ANFIS. Finally, the results from all controllers were summarized and analyzed. It can be concluded that with the application of modified fuzzy logic controller and ANFIS, satisfactory results have been achieved with minimal rise and settling time, without overshoot and steady state error.

REFERENCES

[1] Téllez, F., Burisch, M., Villasente, C., Sánchez, M., Sansom, C., Kirby, P., Turner, P., Caliot, C., Ferriere, A., Bonanos, C. A., Papanicolas, C., Montenon, A., Monterreal, R., Fernández, J.: *State of the Art in Heliostats and Definition of Specifications:*

Survey for a Low Cost Heliostat Development, STAGE-STE Project, 2014.

[2] Wang, Z.: *Design of Solar Thermal Power Plants*, Academic Press, 2019.

[3] Станковски, М., Колемишевска-Гугуловска, Т., *Компјутерско водење на процеси*, Електротехнички факултет, Скопје, 2006.

[4] Tangirala, A. K.: *Principles of System Iidentification – Theory and Practice*, CRC Press, 2015.

[5] Aström, K. J., Hägglund, T.: *Advanced PID Control*, ISA, 2006.

[6] Salgado-Plasencia, E., Carrillo-Serrano, R. V., Toledano-Ayala, M., Development of a DSP Microcontroller-Based Fuzzy Logic Controller for Heliostat Orientation Control, *Applied Sciences* **10** (5):1598, February 2020. https://doi.org/10.3390/app10051598

[7] Khuntia, S. R., Mohanty, K. B., Panda, S., Ardil, C.: A comparative study of P-I, I-P, fuzzy and neuro-fuzzy controllers for speed control of DC motor crive, *International Journal of Electrical Systems Science and Engineering*, **1**, 1 2009.

[8] Mustafa, G. Y., Ali, A. T., Bashier, E., Elrahman, M. F., Neuro-fuzzy controller design for a DC motor dDrive, *UofKE*J Vol. **3,** Issue 1, pp. 7–11 (February 2013).

[9] Zeghoudi, A., Chermitti, A: Speed Control of a DC Motor for the Orientation of a Heliostat in a Solar Tower Power Plant using Artificial Intelligence Systems (FLC and NC), *Research Journal of Applied Sciences, Engineering and Technology* **10** (5), 570–580 (2015).

[10] Grigore, O., Florescu, A., Vasile, A., Stoichescu, D. A.: Part A: Fuzzy Design For DC Motor Speed Control?, *International Workshop on Trends and Recent Achievements in Information Technology*, Cluj-Napoca, Romania, May 2002.

[11] Grigore, O., Florescu, A., Vasile, A., Stoichescu, D. A.: Part B: Neuro-Fuzzy Design For DC Motor Speed Control?, UPB *Scientific Bulletin, Series C: Electrical Engineering,* **6**1 (3-4), 225–235, May 2002.

[12] El-Sharif, I. A., Hareb, F. O., Zerek, A. R.: Design of discrete-time PID controller, International Conference on Control, Engineering & Information Technology (CEIT'14), 2014.

[13] Wu, W.: DC Motor Parameter Identification Using Speed Step Responses, *Modelling and Simulation in Engineering*, Article No. 30, January 2012.

[14] Ruiz-Rojas, E. D., Vazquez-Gonzalez, J. L., Alejos-Palomares, R., Escudero-Uribe, A. Z., Mendoza-Vázquez, J. R.: Mathematical Model of a Linear Electric Actuator with Prosthesis Applications, *The 18th International Conference on Electronics, Communications and Computers*, April 2008.

[15] Mohamed, M. E. A., Guo, Y.: Separately Excited DC Motor Speed Tracking Control Using Adaptive Neuro-Fuzzy Inference System Based on Genetic Algorithm Particle Swarm Optimization and Fuzzy Auto-Tuning PID, *IOP Conference Series Earth and Environmental Science* 300, 042114, August 2019.

[16] Mao, W.-L. Suprapto, S., Hung, C.-W.: Adaptive neural network-based synchronized control of dual-axis linear actuators, *Advances in Mechanical Engineering*, Vol. **8** (7), 1–17 (2016).

[17] Tymerski, R.: ECE317: Feedback and Control, Lecture: Routh-Hurwitz Stab*ility Criterion Examples*, Dept. of Electrical and Computer Engineering, Portland State University.

[18] Pfahl, A., Coventry, J., Röger, M., Wolfertstetter, F., Vásquez-Arango, J. F., Gross, F., Arjomandi, M., Schwarzbözl, P., Geiger, M., Liedke, P.: Progress in heliostat development, *Solar Energy*, Vol. **152**, pp. 3–37 (August 2017),

[19] Prinsloo, G. J., Dobson, R. T.: *Solar Tracking: High precision solar position algorithms, programs, software and source-code for computing the solar vector, solar coordinates & sun angles in Microprocessor, PLC, Arduino, PIC and PC-based sun tracking devices or dynamic sun following hardware*, Stellenbosch: SolarBooks, 2015.

[20] Salgado-Plasencia, E., Carrillo-Serrano, R. V., Toledano-Ayala, M.: Development of a DSP Microcontroller-Based Fuzzy Logic Controller for Heliostat Orientation Control, *Applied Sciences* **10** (5), 1598 (February 2020).

[21] Reda, I., Andreas, A.: *Solar Position Algorithm for Solar Radiation Applications*, NREL, January 2008

[22] Hamanah, W. M., Salem, A., Abido, M. A.: Heliostat Dual-Axis Sun Tracking System: A Case Study in KSA, *The 18th International Conference on Renewable Energies and Power Quality (ICREPQ'20), Granada (Spain)*, 1–2 April 2020

[23] Andonov, I., Ojleska Latkoska, V., Stankovski, M.: Comparative Analysis of Different Heliostat Field Control Algorithms, In: *Proceedings of the 15th International Conference - ETAI 2021, held virtually*, September 23-24, 2021 (ISSN 2545-488).