Journal of Electrical Engineering, Electronics, Control and Computer Science – JEEECCS, Volume 8, Issue 29, pages 49-60, 2022

Kinematic Control of A DC Motor by A Comparative Approach Between a Digital *PID* Controller and Two Soft Computing Controllers (Fuzzy and Neuro-Fuzzy)

Yannick Kenfah W. Tiawoun
Research Laboratory of Computer Science Engineering and Automation
ENSET, University of Douala
PO box. 8580, Douala, Cameroon
kenf.yann@gmail.com

Léandre Nneme Nneme
Research Laboratory of Computer Science Engineering and Automation
ENSET, University of Douala
PO box. 8580, Douala, Cameroon
leandren@gmail.com

Abstract – **The rise of mobile line-following robots in various industrial and civil fields of activity suggests the advent of increasingly robust autonomous mobile robots. However, this development is still marked by several major challenges such as navigation with obstacle avoidance in unknown or known environments and, more particularly, the fluidity of their mobility during trajectory turns due to their kinematics. The objective of this paper is to provide one of the appropriate solutions to this problem. Our contribution will focus on the estimation of the model of our DC motor via data from the acquisition of speed measurements of the latter during the experiment carried out under Arduino and Matlab, then on the design of speed controllers for the control of the DC motor of a mobile car-type robot and finally on a comparison of the responses between the classic *PID* controllers and those of Soft Computing (fuzzy controller and fuzzy neuro). The implementation is carried out with an Arduino microcontroller for the control of the DC motor (*FIT0521*) via a L298N double H-bridge motor driver, and on the other hand by implementing the different controllers mentioned above in the Matlab - Simulink environment. The simulation results of the three control models showed the robustness of the smart controllers compared to the classical one for non-linear systems. These results were confirmed experimentally on our DC motor "*FIT0521*" used for the circumstance in place of the motors of the manufacturer "*Elegoo*" in the mobile robots of car type and the analysis of the curves thus obtained.**

Keywords-PID **controller; fuzzy controller; neuro-fuzzy controller; DC motor; Soft Computing**

I. INTRODUCTION

Robotic vehicles have been increasingly used in industrial, civil and public environments in recent years [1]. This is the case of line-following robots, which have become indispensable in industry to transport multiple loads from one workstation to another. The automation of the mobility of these robots is largely due to the use of its increasingly powerful exteroceptive and proprioceptive sensors on the one hand, and on the other hand thanks to the speed control of its DC motor, the result of the use of increasingly robust controllers. Nowadays, more than 95% of control application designs use the classical Proportional-Integral-Derivative (*PID*) controller due to its simplicity and applicability [2],[3],[4]. Despite the popularity of this type of controller (*PID*), it is clear that they have shortcomings when dealing with more complex systems [5]. The advent of soft computing methods (fuzzy logic, neural networks, neuro-fuzzy networks) has opened up a new field of investigation for the development of new control systems that are more robust and efficient than those of conventional controllers. The works of [6],[7],[8] demonstrate the growing and unavoidable interest of soft computing.

The problem of our study is related to the speed control of the DC motor of our robot by the implementation of controllers adequate to our specifications. This mobile robot (*Elegoo*) of the car type is indeed confronted with a problem of fluidity of its mobility at the time of the follow-up of line particularly at the level of the turns due to its kinematics.

The objective of this article will be to estimate the model of the DC motor (*FIT0521*) via measurements of the speed of rotation of the latter carried out under Arduino and Matlab, on the other hand, on the design of three controllers: *PID*, fuzzy and a hybrid controller of type (*Anfis*). Then, on the implementation of the said controllers within the Matlab/Simulink platform, then on the simulation of a control system of an DC motor in open loop, and finally, on the comparison of the response of the behavior of the various above-mentioned controllers within the platform of the control system (DC motor of the *Elegoo* robot) via the experimental results obtained in order to make the choice of the most powerful and robust controller to answer our problematic as well as possible.

This paper is structured as follows: Section 2 is devoted to the design methodology developed, section 3 highlights the results and discussion from the test phases and finally section

4 is devoted to a conclusion and perspectives that will take stock of our entire study and future work.

II. MEANS AND TOOLS OF INVESTIGATION

In order to carry out this work, we developed and simulated, in Matlab/Simulink, the models corresponding to the elements of the system under study, namely: the model associated with the speed control of an DC motor, those of the "PID", "Fuzzy" and hybrid neuro-fuzzy "Anfis" controllers. This section is devoted to the presentation of the different means and tools used during our work.

A. Description of the block diagram of the working platform

The block diagram for the overall system description is shown in Figure 2.1. In this block, three main hardware components are implemented for this research. The Arduino Mega2560 microcontroller controls the kinematics of the DC motor by controlling the input speed to the motor. Both classical (PID) and intelligent (fuzzy and neuro-fuzzy) control techniques are implemented in the microcontroller via Matlab / Simulink to execute the PWM signal for driving the DC motor. A DC motor gearbox with magnetic encoder, planetary gearbox and 34:1 gear ratio, which delivers 210RPM with a nominal voltage of 6VDC. An L298N double H-bridge motor driver that controls the direction and speed of the DC motor.

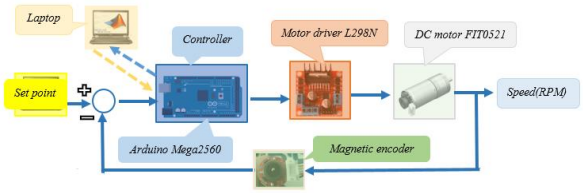

Figure 2.1: Block diagram for DC motor speed control

B. Hardware tools

1) Arduino Mega 2560 microcontroller

Arduino Mega is a microcontroller board based on the ATmega 2560 shown in Fig. 2.2. It has 54 digital input/output pins (15 of which can be used as PWM outputs), 16 analogue inputs, a clock rate of 16 MHz, a USB connection, a power supply socket, an ICSP socket and a reset button. It contains everything needed to support the microcontroller; it can be simply connected to a computer with a USB cable or powered with an AC-DC adapter or battery for powering up. In this article, the Arduino Mega microcontroller is very well suited to drive the PWM signal of the DC motor to improve the output response of the DC motor speed control system.

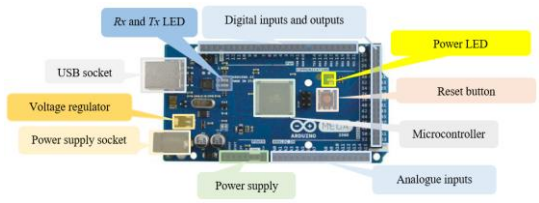

Figure 2.2: Arduino Mega2560 Microcontroller [9]

2) L298N Dual H-Bridge Controller

The L298N H-bridge IC, shown in Fig. 2.3, can control the speed and direction of two DC motors. This module can be used with motors that have a voltage between 5 and 35V DC with a peak current up to 2A. The module has two screw terminals for motor A and B, and another screw terminal for the ground pin, the VCC for the motor and a 5V pin which can be an input or output. The pin assignment for the L298N dual H-bridge module is shown in Table 1. The digital pin assignment of HIGH to LOW or LOW to HIGH is used IN1 and IN2 on the L298N board to control the direction. And the PWM output signal from the controller is sent to ENA or ENB to control the speed. The control of the speed or the forward and reverse position of the motor was done using the PWM signal. Then, using the analogWrite() function and sending the PWM signal to the Enable pin of the L298N board, actually drives the motor.[10]

TABLE 2.1: PIN ASSIGNMENTS FOR THE L298N

Out 1	*Motor A lead out*
Out 2	*Motor A led out*
Out 3	*Motor B lead out*
Out 4	*Motor B lead out*
GND	*Ground*
Vcc	*5V input*
ENA	*Enables PWM signal for motor A*
IN1	*Enable motor A*
IN2	*Enable motor A*
IN3	*Enable motor B*
IN4	*Enable motor B*
ENB	*Enables PWM signal for motor B*

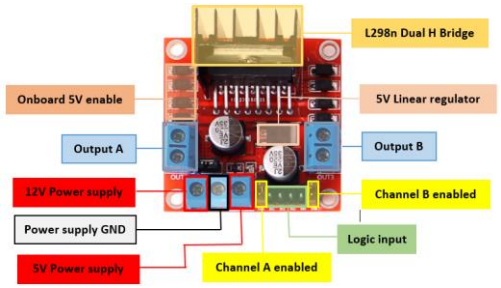

Figure 2.3: L298N motor control board [11]

3) Motor with DC magnetic encoder

The FIT0521 DC geared motor with encoder shown in Fig. 2.4 is a suitable motor to drive the speed control system. It comes with a magnetic encoder output and a reduced planetary gear ratio of 34:1. It can deliver 210RPM with a nominal voltage of 6VDC. To read the count values from the encoder, the user would check the rotational condition of channels A and B based on the signals obtained during the experiment. For the values of the number of rotor shafts per revolution, it is very important to multiply the gear ratio by the count values. The motor characteristics are shown in Table 2.2.

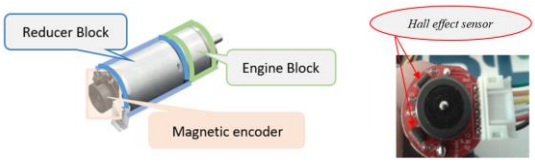

Figure 2.4: FIT0521 geared motor with magnetic encoder [12],[13]

TABLE 2.2: TECHNICAL SPECIFICATIONS OF THE FIT0521 GEARED MOTOR [13]

Manufacturer	**DFROBOT**
Motor type	DC
Type of motor	With encoder, with transmission
Motor Rated Voltage	6V
Encoder Rated Voltage	3,3 / 5V
Reducer Reduction Ratio	1/34
No load Speed	210RPM@0,13A
Maximum Efficiency Point	load 2,0kg.cm /170RPM /2,0W /0,6A
Maximum Power point	load 5,2kg.cm /110RPM /3,1W /1,1A
Stall Torque	10kg.cm
Stall Current	3,2A
Hall Resolution	Hall Resolution 11× Precision reduction Ratio 34,02 = 341,2PPR
Dimension	52× Φ 24,4mm/2,05× Φ 0,96inches
Weight	96g

4) Reading the magnetic encoder

The encoder wheels allow the direction and speed of rotation of the motor to be known. A magnetic disc passes in front of two sensors (hall effect) which detect it in turn. Each time the magnet passes, a pulse, also called a tick, is sent to the Arduino board. There is a rising edge (H) and a falling edge (L) for each sensor, i.e. four different possible positions (HH, LL, HL, LH). In one period, it is therefore possible to pass through four positions, each 90° apart, 360° being equivalent to one period. This is why we speak of a quadrature shift of the signal. Furthermore, depending on the sequence of edges, it is easy to determine the direction of rotation. For example, if HH is recorded and then HL comes from, we know that the motor is rotating clockwise. If, on the other hand, LH comes in, it is the other way round.

The magnetic encoder uses magnets instead of the traditional black lines, and these must not be too close together to avoid interference. However, attached to the motor shaft, the number of pulses per revolution of the output shaft is multiplied by 34 and is therefore largely sufficient in our case.

C. Software tools

Matlab R2020a and Arduino were used to model and acquire the rotational speeds of the DC motor and to implement the various controllers used and the resulting model.

D. Acquisition of the DC motor rotation speed

The acquisition of the kinematics of our DC motor can be carried out in three ways depending on the equipment available, namely: By the use of a digital tachometer, or by the use of an optical sensor and finally via a magnetic sensor with hall effect embedded on our DC motor. We have opted to use the magnetic incremental encoder of the DC motor and the use of an Arduino Mega microcontroller as shown in figure 3.1 below. Two options are possible : the first approach is to make the assembly shown in Figure 3.1 and develop a sketch implemented via the Arduino IDE (Integrated Development Environment). The second approach, the one chosen in our study, consists of creating a block diagram for acquiring the speed of the DC motor (see the "Rotation Speed Motor (RPM)" block in figure 3.3 or 3.4) using Simulink and the synoptic diagram in figure 3.1, after which we recorded the different values of the DC motor 's rotation speed and also the variation of this speed over a period of 13s.

E. Estimation of the DC motor parameters

To determine the parameters (R; L; J; f; Ke) of the DC motor, we adjusted the parameters of the acausal model from the initialization parameters (see Table 2.3) and the velocity acquisition measurements of the DC motor. The procedure can be simplified as follows:

- ✓ Creation of the input signal (voltage at the DC motor input) and output signal (DC motor speed) of the acausal model,
- ✓ Then, implementation of this model via Simulink (see figure 2.5) and
- ✓ Finally, search for the parameters via the Matlab toolbox (Apps →Parameter Estimator)

F. Modelling of our DC motor

According to [15], DC motors are widely used in industrial applications due to their speed control techniques. For the modelling of our FIT0521 gearmotor, we opted for an acausal model approach proposed by Matlab/Simulink whose characteristics of the initialization parameters are mentioned in Table 2.3.

TABLE 2.3: INITIALIZATION PARAMETERS OF THE ACAUSAL DC MOTOR MODEL [16]

Symbols	**Description**	**Values**
R	Armature resistance [Ω]	4
L	Armature inductance [H]	0,5
F	Viscous friction [N.m/(rad/s)]	10^{-4}
J	Moment of inertia [kg.m^2]	3.10^{-5}
K_e	Fem constant [V/(rad/s)]	5.10^{-5}

According to [17], the acausal model of a possible DC motor is represented as in Figure 2.5.

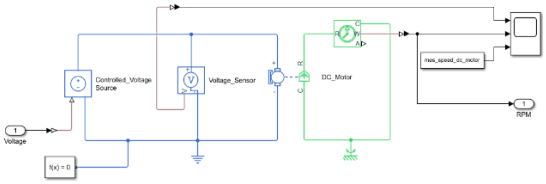

Figure 2.5: Acausal model of the studied DC motor *(made in Matlab R2020a)*

From the acausal model (in which the voltage sensor, the experimental measurements and the oscilloscope are removed), a subsystem is created in Simulink in order to obtain a causal model of our DC motor. This step is performed once the new estimated parameters of the DC motor are obtained (Table 3.1)

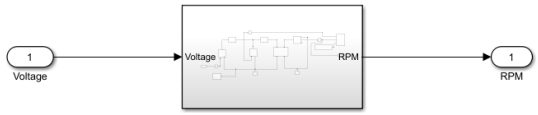

Figure 2.6: Block diagram of the acausal open-loop DC motor model (made in Matlab Simulink)

G. Design of PID, fuzzy and neuro-fuzzy controllers (Anfis type)

1) Design of the digital PID controller

The general transfer function for a PID controller in the Laplace domain can be written as shown in (2.1) where KP represents the proportional gain, *K*D the derivative gain and *K*I the integral gain. Considering the effect of each term in the PID controller, the PID parameters were determined by automatically setting the " PID Controller" block of Matlab/Simulink. This block was initialized with the following parameters (see Table 2.1). The "Tune" function allowed the automatic setting of the PID controller and its transfer function has the expression (2.2) where N represents filter coefficient.

$$PID(s) = K_P + \frac{K_I}{s} + K_D s \qquad (2.1)$$

$$PID(s) = K_P + \frac{K_I}{s} + K_D \frac{N}{1 + N\frac{1}{s}} \qquad (2.2)$$

TABLE 2.4: SYSTEM PID CONTROLLER PARAMETERS

Control parameters	Block parameters (before adjustment)	Block parameters (after adjustment)
K_P	0,6844	7,161e-05
K_I	0,5975	0,00032912
K_D	0,0119	0,000
N	59	100

2) Takagi-Sugeno (TS) fuzzy controller design

The work of [18] shows that Takagi-Sugeno fuzzy models are not only well known but also effective for non-linear systems. The design of the speed controller using fuzzy logic requires only a simple description of the behaviour of the system to be controlled. It consists of three parts (Figure 2.7) : fuzzification, inference rules and defuzzification. The controller used in our case consists of two input variables (the speed error "err" and the speed error variation "derr") and one output variable (the DC motor voltage command "cde")

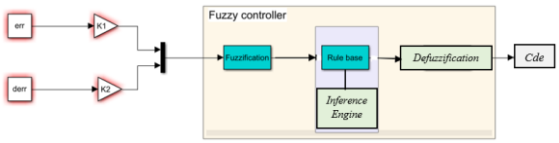

Figure. 2.7: Block diagram of the fuzzy controller [18]

The two most common types of fuzzy controllers are Mamdani and Sugeno. In all our application, we used the second controller (Sugeno) [6]. In this case the fuzzy operators are realized as follows:

- ✓ And method: by the « *min* » function
- ✓ Or method: by the « *max* » function
- ✓ Implication: by the « *prod* » function
- ✓ Aggregation: by the « *sum* » function
- ✓ Defuzzification: by the « *Wtaver* » function

The main advantage of the Takagi-Sugeno (TS) technique is that it simplifies the aggregation calculation, so that a net solution can be obtained more quickly and the mechanism of the overall calculation can be lightened significantly. According to [19], the final output in TS modelling is equal to the weighted average of the output of each rule. This average is given by expression (15):

$$CG_{sugeno} = \frac{\sum_{i=n}^{n} Z_i W_i}{\sum_{i=1}^{n} W_i} \qquad (15)$$

Where Zi is the output level of each rule Ri and Wi the membership degrees calculated by equation:

$$W_i = \mu_{Ai}(E)\mu_{Bi}(\Delta E) \qquad (16)$$

The writing of the inference rules is based on the description of the dynamic behavior of the system to be controlled. The interference rules of the fuzzy controller used for the speed control of the DC motor are summarized in Table 2.5. These rules have a local effect on the behavior of the system.

TABLE 2.5: INFERENCE RULES

Command (Cde)		*derr*				
		NB	***NS***	***Z***	***PS***	***PB***
err	***N***	*PB*	*PM*	*PS*	*PS*	*Z*
	Z	*PM*	*Z*	*Z*	*Z*	*NS*
	P	*Z*	*NS*	*NM*	*NM*	*NB*

The membership functions used in our work are of Gaussian, trapezoidal, triangular type for the inputs, while they are of singleton type for the output (see figures: 2.8, 2.9 and 2.10). Their distribution is defined as follows [20]:

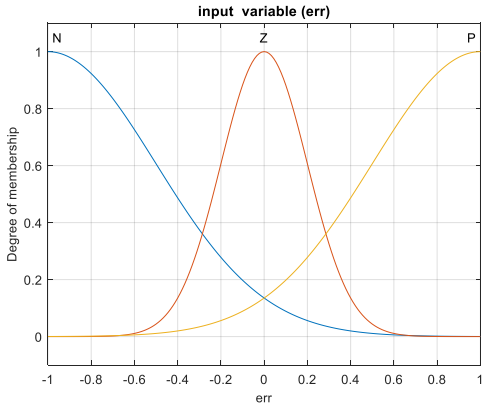

Figure 2.8: Fuzzification of the velocity error (err)

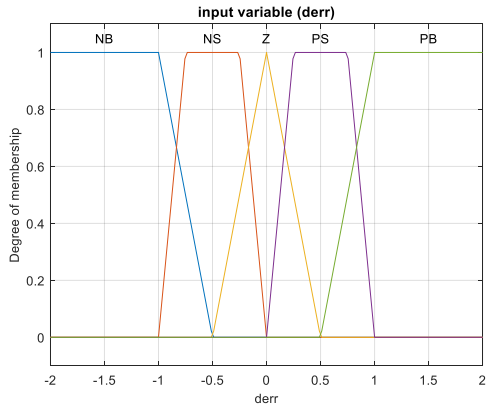

Figure 2.9: Fuzzification of the velocity error variation (derr)

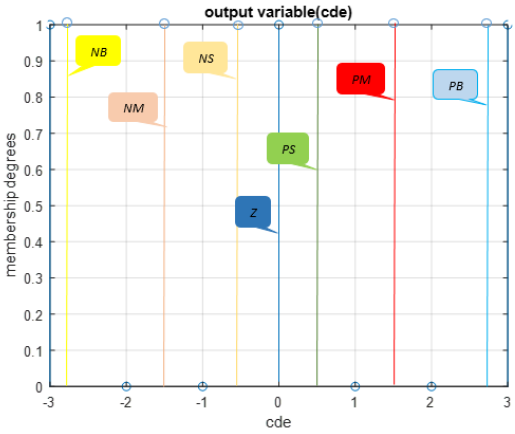

Figure 2.10: Fuzzification of the output command (cde)

The acronym of the linguistic values is defined as follows:

Nota Bene: NB: Negative Large; NM: Negative Medium; NS: Negative Small; Z: Zero; PS: Positive Small; PM: Positive Medium; PB: Positive Large or Big.

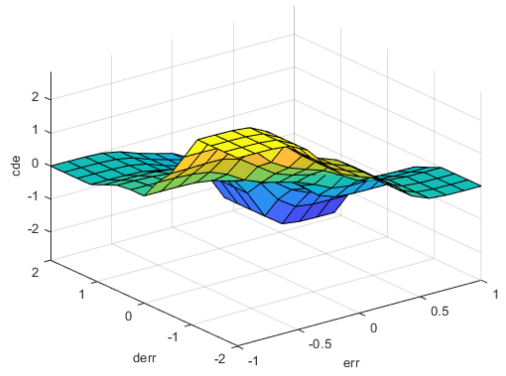

Figure 2.11: Characteristic surface of inference rules (defuzzification)

3) Design of the Anfis neuro-fuzzy controller

Developed by [8], the Anfis model, also known as an adaptive network based on a fuzzy inference system, is a universal approximator. The neuro-fuzzy system used for this purpose is of the Takagi-Sugeno (TS) type.

a. Description and structure of the neuro-fuzzy controller (*Anfis*)

Our developed Anfis controller consists of two input variables (the velocity error "err" and the velocity error variation "derr" of the DC motor) and one output variable (the voltage command "cde"). This controller allows the automatic generation of fuzzy rules based on Sugeno's inference model.

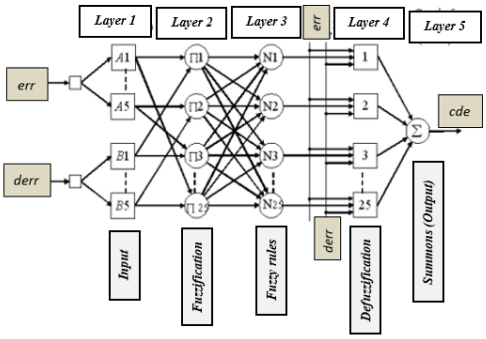

Figure 2.12: Architecture of the proposed ANFIS model [21]

The equivalent neural structure proposed in Matlab is shown in Figure 2.13.

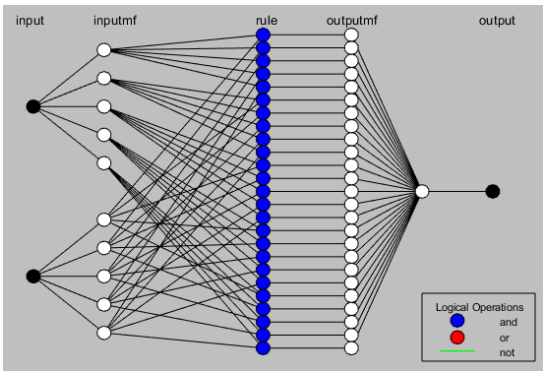

Figure 2.13: Neural structure of the proposed model in Matlab

b. Training of the "*ANFIS* network training" controller

Our ANFIS speed controller uses for its learning a combination of two algorithms: the backpropagation algorithm for learning the membership functions and the least squares estimation algorithm for determining the linear combination of the rule conclusions. Therefore its learning is done in two phases, a direct and an inverse one, during the direct phase the signals are propagated to the 5th layer, the parameters of the consequences of the rules are adjusted following the least squares algorithm. In the reverse phase the error is propagated in the opposite direction and the parameters of the premises are adjusted according to the back-propagation algorithm [22].

The database is obtained via the "To Workspace" block of Simulink from the fuzzy controller and has been randomly split into two parts, one for learning the models (ANFIS) which represent (80%) of the total database size and the other for validation (20%). The result of the learning process is shown in Figure 2.14.

Figure 2.14: Neuro-fuzzy network training (with a number of iterations of 2500)

c. Testing the generalisation capacity after training

After training the adaptive neuro-fuzzy system (ANFIS), we tested its generalisation, by presenting validation data (Checking Data) that is 20% of the total size of the database.

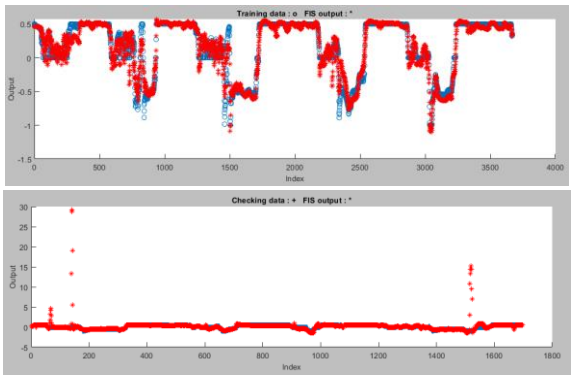

Figure 2.15: Testing generalisation capacity after learning (validation phase)

III. RESULTS AND DISCUSSION

The performance of the three designed controllers was simulated in Matlab/Simulink as a block diagram as shown in Figures 3.2, 3.3 and 3.4. The setpoint of the block diagram is a step and or square wave signal. The results show that the soft computing controllers perform better than the digital PID controller.

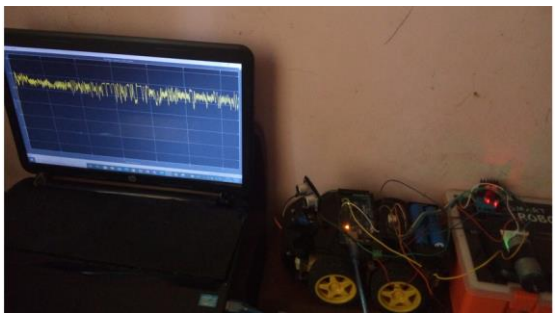

Figure 3.1: DC motor speed acquisition flow chart

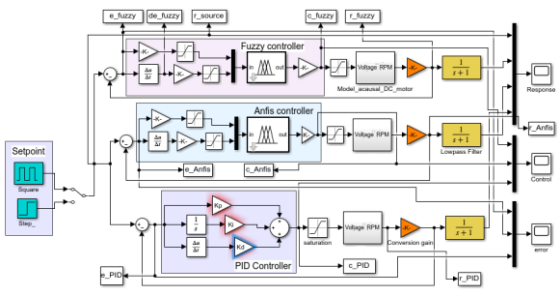

Figure 3.2: Block diagram of the velocity response of the DC motor with the estimated model (and the different controllers)

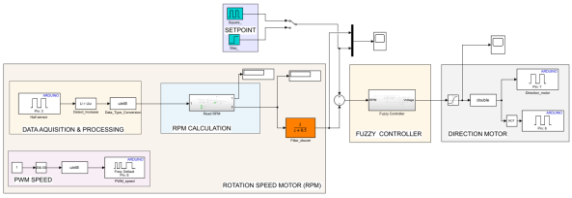

Figure 3.3: Block diagram of the measured velocity response model of the DC motor with the fuzzy controller

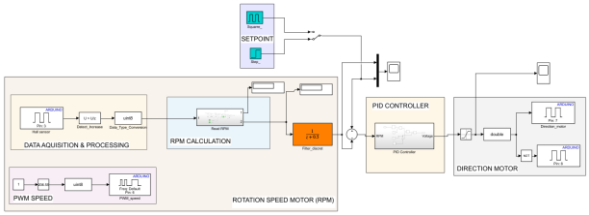

Figure 3.4: Block diagram of the measured speed response model of the DC motor with the PID controller

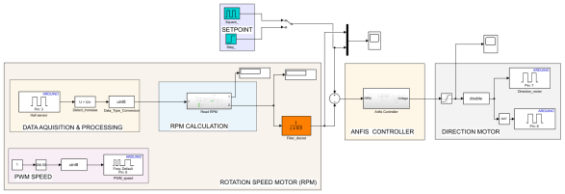

Figure 3.5: Block diagram of the measured speed response model of the DC motor with the Anfis controller

Results obtained with the estimated model of the considered DC motor (Figures 3.7 and 3.8).

The acquisition of the speed of the DC motor (see figure 3.6) is done via figure 3.1 and from Simulink. The velocity acquisition will act as a database for the setpoint when servo-controlling the DC motor.

Figure 3.6 shows the estimated or simulated model in red and the real or measured model in blue. The voltage at the input of the DC motor is also represented and has been fixed at 7,4volts corresponding to the power supply batteries (two accumulators of 3,7V each) used for the test case.

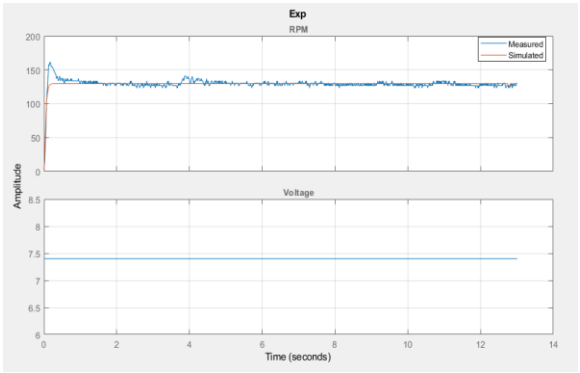

Figure 3.6: Velocity representation of the estimated and measured model of the DC motor and the setpoint voltage (7.4Volts)

Figure 3.6.1 shows the evolution of the estimation of the DC motor parameters (R, L, J, f, Ke) according to the different iterations performed by minimizing the square error of the sum of the estimation. It took 36 iterations to obtain the best precision, which is of the order of 0.8357.

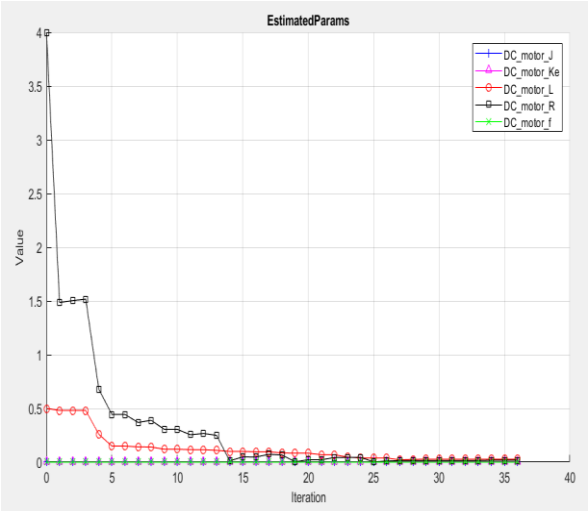

Figure 3.6.1: Estimated DC motor parameter curves

- New DC motor parameters after estimation of the real model

TABLE 3.1: ESTIMATED PARAMETERS OF THE ACAUSAL MODEL

Symbols	Description	Values
R	Armature resistance [Ω]	0.017657
L	Armature inductance [H]	0.027677
F	Viscous friction [N.m/(rad/s)]	9.5527e-05
J	Moment of inertia [kg.m^2]	2.4952e-06
Ke	Fem constant [V/(rad/s)]	0.0017572

Figures 3.7, 3.7.1, and 3.7.2 represent the step responses from the real model (from the block diagrams in Figures 3.3, 3.4 and 3.5). Figures 3.8, 3.8.1 and 3.8.2 represent the results from the simulated model and from the comparison of the index responses with the DC motor speed controllers respectively. The observation made by comparative approach between the real or experimental model and the estimated model shows similarities concerning the index response of the motor for a fixed setpoint (ω_m= 91,44RPM). The fuzzy (see figure 3.7.1) and Anfis (see figure 3.7.2) controllers are robust despite a relative inaccuracy in steady state (non-linear DC motor speed) recorded around the setpoint as minor overshoots of the setpoint are recorded. A lower transient inaccuracy is observed with the real model compared to the simulated model (Figure 3.8). The speed of the DC motor with the soft computing controllers is relatively stable and steady in both models (real and simulated). In contrast, the PID controller (Figure 3.7) is highly inaccurate despite a stability observed in steady state in the case of the experimental model. In the simulated model (Figure 3.8), the PID controller is highly inaccurate in the transient state and stable in the steady state.

a. Theoretical and experimental results obtained for an indexed or step setpoint

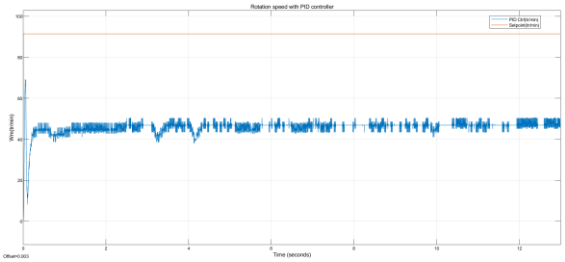

Figure 3.7: Step response of the DC motor speed with the PID controller in closed loop of the real model

Figure 3.7 shows the setpoint (step speed) in red fixed at (ω_m= 91,44RPM) and in blue the rotational speed of the DC motor with the PID controller. On the ordinate we have the amplitude or speed of rotation ω_m (RPM) and on the abscissa the time (duration of the experiment =13s). At start-up or in transient regime, we observe a peak in the speed of rotation of the DC motor (maximum speed), which translates into a rapidity of the speed of the DC motor with the PID controller. High inaccuracy (error between the setpoint and the obtained rotation speed) in steady state.

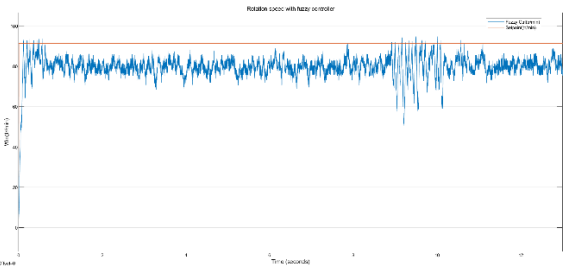

Figure 3.7.1: Step response of the DC motor speed with the fuzzy controller in the closed loop of the real model

Nota Bene: As mentioned before, the reference speed is in red in the figures (3.7, 3.7.1, 3.7.2) and the DC motor rotation speed is in blue. The ordinates represent the rotational speed ω_m (RPM) and the abscissa the time (seconds) or duration of the experiment considered.

Figure 3.7.1 shows a high accuracy at start-up, more precisely in the transient regime. Small overshoots of the DC motor speed are also observed, related to the inertia of the DC motor speed because, when the response is above the setpoint, the DC motor stops immediately until the response converges to the setpoint. The fuzzy controller remains robust. The speed of the DC motor is almost stable despite a relative inaccuracy also observed in steady state.

The fuzzy controller (Figure 3.7.1) offers better accuracy than the PID controller.

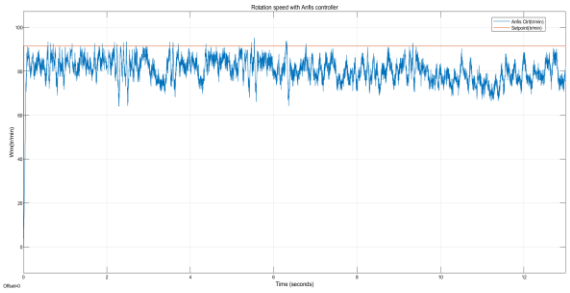

Figure 3.7.2: Step response of the DC motor speed with the Anfis controller in closed loop of the real model

The Anfis controller (Figure 3.7.2) provides better a priori accuracy than the fuzzy and PID controller in both transient and steady state. Stability is also observed in ω_m (RPM).

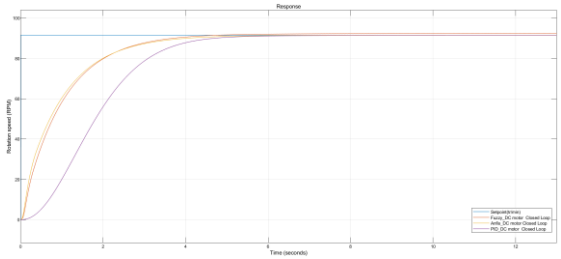

Figure 3.8: Comparison of the index response of the DC motor speed in closed loop of the estimated model (with PID, fuzzy and Anfis controllers)

Nota Bene: Figures 3.8 and 3.9 have the time (seconds) on the abscissa and the DC motor speed on the ordinate. The setpoint (ω_m= 91,44 rpm) is in blue, the rotation speed regulated by the PID controller is in purple, the one regulated by the fuzzy controller is in red and finally the one regulated by the neuro-fuzzy controller (Anfis) is in yellow.

Figure 3.8 shows a comparison between the step responses from the three controllers when servo-controlling the kinematics of the DC motor. The performance is summarized in Table 3.4.

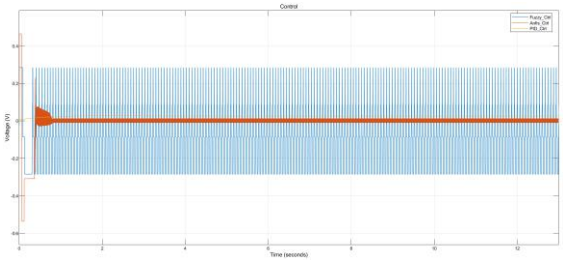

Figure 3.8.1: Comparison of the voltage variation at the output of each controller (PID, fuzzy, Anfis)

Figure 3.8.1 shows the variation of the voltage at the input of each controller in the case of the estimated or simulated model. On the ordinate the voltage in volts and on the abscissa the time in seconds, in blue the output of the fuzzy controller, in red that of the Anfis controller and finally in yellow that of the PID controller.

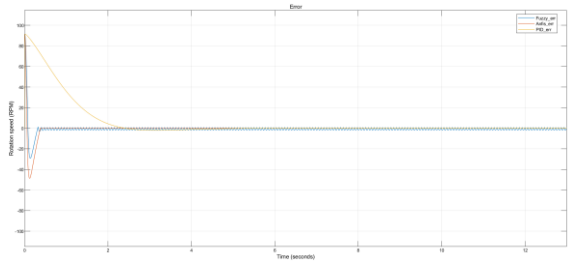

Figure 3.8.2: Comparison of the DC motor velocity error at the input of each controller (PID, fuzzy, Anfis)

Figure 3.8.2 shows the closed loop DC motor velocity errors (closed loop) of each controller studied for the estimated and simulated model. Its errors are recorded after 13s. In blue the curve of variation of the DC motor speed error at the input of the fuzzy controller, in red that at the input of the Anfis controller and finally in yellow that at the input of the PID controller.

b. Theoretical and experimental results obtained for a square setpoint

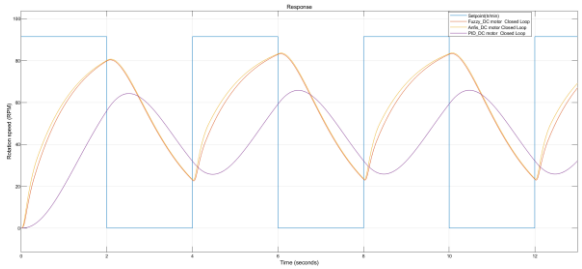

Figure 3.9: Comparison of the square response of the DC motor velocity in closed loop of the estimated model (with PID, fuzzy and Anfis controllers)

Figure 3.9 shows a comparison of the square velocity responses of the DC motor with each controller studied. From our observations, a low accuracy of the transient and steady state velocities (rising edge or maximum set point, falling edge or minimum set point) of the simulated model, although stable, is evident. A better analysis is made via table 3.1 concerning the performances of the various controllers.

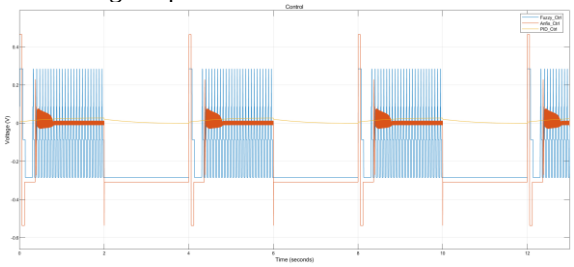

Figure 3.9.1: Comparison of the voltage variation at the output of each controller (PID, fuzzy, Anfis)

Figure 3.9.1 shows the variation of the voltage at the input of each controller in the case of the estimated model. As before, we have: on the ordinate the voltage in volts and on the abscissa the time in seconds, in blue the output of the fuzzy controller, in red that of the Anfis controller and finally in yellow that of the PID controller.

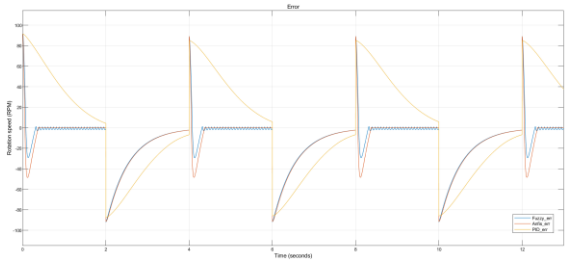

Figure 3.9.2: Comparison of the DC motor speed error at the input of each controller (PID, fuzzy, Anfis)

Figure 3.9.2 shows the evolution of the error of the DC motor rotation speed in closed loop with each studied controller with each controller studied. Its errors are recorded after 13s. The y-axis represents the rotational speed (RPM).

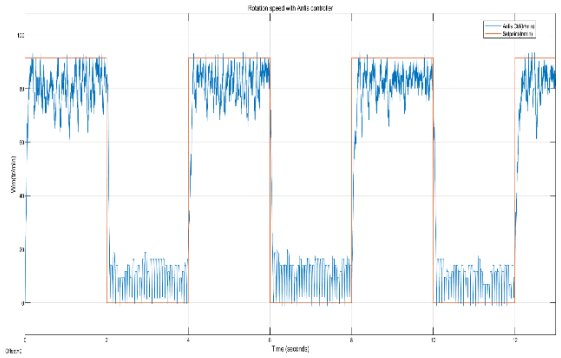

Figure 3.10: Square response of the DC motor speed with the Anfis controller in closed loop of the real model

Figure 3.10 shows the speed of the DC motor regulated by the Anfis controller with a square setpoint. In red the maximum amplitude speed setpoint (ω_m= 91,44RPM) and in blue the evolution of the DC motor speed. We have two transition phases (rising and falling edges) with relative precision and two permanent phases (maximum and minimum speeds) with overshoots and greater precision although the speed is non-linear. We also note a stability of the speed.

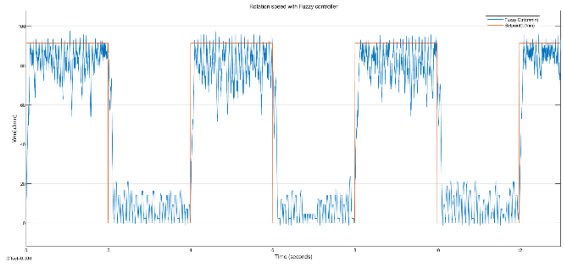

Figure 3.10.1: Square response of the DC motor velocity with the fuzzy closed loop controller of the real model

As before, Figure 3.10.1 shows the velocity of the DC motor with relative accuracy in steady state despite the observed overshoots. The velocity is stable although not linear.

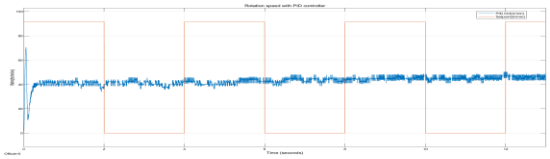

Figure 3.10.2: Square response of the DC motor speed with the real model closed loop PID controller

Figure 3.10.2 shows the speed of the DC motor regulated by the PID controller. We can see a peak in the transient regime, which indicates a fast speed at start-up. However, we note a strong inaccuracy of the speed compared to the speed set point.

c. Performance indicators

TABLE 3.2: PERFORMANCE INDICATOR OF THE DIFFERENT IMPLEMENTED CONTROLLERS RESULTING FROM THE RESPONSE TO A SQUARE SET POINT OF THE ESTIMATED OR SIMULATED MODEL

Controller	Rise time (ms)	Overshoot (%)
PID	**1199**	$\overline{\mathbf{0,779}}$; $\underline{\mathbf{1,978}}$
Fuzzy	**1368**	$\overline{\mathbf{0,652}}$; $\underline{\mathbf{1,983}}$
Anfis	**1345**	$\overline{\mathbf{0,659}}$; $\underline{\mathbf{1,969}}$

From the results in Table 3.2, it can be seen that the PID controller is the fastest (tm-PID = 1199ms) of the three. However, the PID controller is the least accurate in steady state (rising edge or maximum speed) and more accurate than the fuzzy controller in steady state (falling edge or minimum speed). The Anfis controller offers the best accuracy (dAnfis = 0,659% for the rising edge and dAnfis = 1,969% for the falling edge) of the three. Note also that the Anfis controller is faster (tm-Anfis = 1345ms) than the fuzzy controller.

TABLE 3.3: PERFORMANCE INDICATOR OF THE DIFFERENT IMPLEMENTED CONTROLLERS FROM THE RESPONSE TO A SQUARE SET POINT OF THE EXPERIMENTAL OR MEASURED MODEL

Controller	Rise time (ms)	Overshoot (%)
PID	**165,340**	$\overline{\mathbf{30,729}}$; $\underline{\mathbf{13,346}}$
Fuzzy	**85,620**	$\overline{\mathbf{13,107}}$; $\underline{\mathbf{20,237}}$
Anfis	**105,054**	$\overline{\mathbf{12,527}}$; $\underline{\mathbf{5,673}}$

From the results in Table 3.3, it can be seen that the fuzzy controller is the fastest (tm-fuzzy =85,620ms) of the three. However, the PID controller is the least accurate in steady state (rising edge or maximum speed) and more accurate than the fuzzy controller in steady state (falling edge or minimum speed). The Anfis controller offers the best accuracy (dAnfis = 12,527% for the rising edge and dAnfis = 5,673% for the falling edge) of the three. Note also that the Anfis controller is faster (tm-Anfis = 105,054ms) than the PID controller.

TABLE 3.4: PERFORMANCE INDICATOR OF THE DIFFERENT IMPLEMENTED CONTROLLERS FROM THE STEP RESPONSE OF THE ESTIMATED OR SIMULATED MODEL

Controller	Rise time (ms)	Overshoot (%)
PID	**2576**	**0,504**
Fuzzy	**2121**	**0,501**
Anfis	**2065**	**0,500**

From the results in Table 3.4 it can be seen that the Anfis controller is the fastest (tm-Anfis = 2065ms) and most accurate (dAnfis =0,500%) of the three. However, the fuzzy controller is more accurate (dm-fuzzy =0,501%) and faster (tm-fuzzy =2121ms) than the PID controller.

TABLE 3.5: PERFORMANCE INDICATOR OF THE DIFFERENT IMPLEMENTED CONTROLLERS FROM THE STEP RESPONSE OF THE EXPERIMENTAL OR MEASURED MODEL

Controller	Rise time (ms)	Overshoot (%)
PID	**51,314**	**34,269**
Fuzzy	**73,789**	**18,843**
Anfis	**43,140**	**12,790**

From the results in Table 3.5 it can be seen that the Anfis controller is the fastest (tm-Anfis = 43,140ms) and most accurate (dAnfis =12,790%) of the three. However, the fuzzy controller is more accurate (dm-fuzzy =18,843%) and less fast than the PID controller (tm-PID = 51,314ms).

It appears from our observations and analysis of the data obtained that, the soft computing controllers offer a better performance in the context of our work therefore, these controllers are more robust than that of the digital or classical PID controller for the control of non-linear systems.

CONCLUSION

The work presented in this paper focused on the speed control of a DC motor (FIT0521 geared motor). For this purpose, we designed three controllers, one of which is a classical controller (PID controller) and two others from soft computing (fuzzy controller and neuro-fuzzy controller of Anfis type). We then implemented the latter on our platform (figures : 3.1, 3.2, 3.3, 3.4, 3.5). It is important to underline that two approaches were addressed in our study: the experimental or measured model (figures: 3.3, 3.4, 3.5) and the simulated or estimated model (figure 3.2). After analysing the tables obtained (Tables : 3.2, 3.3, 3.4, 3.5) and observing the figures obtained (Figures : 3.7, 3.7.1, 3.7.2, 3.8, 3.9), it is clear that the fuzzy and neuro-fuzzy speed controllers (Anfis) offer better performance (stability, accuracy and speed) than the PID speed controller, although in some cases they offer greater speed or response. It goes without saying that soft computing methods are more robust and efficient for non-linear systems.

This performance is sufficient evidence that the digital PID controller is more suitable for linear systems but less robust for non-linear systems. The neuro-fuzzy controller of the Anfis type, because it combines the advantages of neural networks and fuzzy logic, and the fuzzy controller prove their effectiveness and are more robust and efficient for non-linear systems. Future work could focus on the implementation of soft computing methods in order to integrate them into path following and obstacle avoidance for an autonomous mobile robot such as a car in a dynamic or static environment.

REFERENCES

[1] Danilo Alves de Lima et Alessandro Correa Victorino ˆ, « Un contrôleur hybride pour le suivi de route et évitement d'obstacles », nº JDJNMACS'2015, juin 2015.

[2] R. Shamshiri et W. I. W. Ismail, « Design and Simulation of Control Systems for a Field Survey Mobile Robot Platform », *RJASET*, vol. 6, nº 13, p. 2307-2315, août 2013, doi: 10.19026/rjaset.6.3701.

[3] Jianming Zhang, Ning Wang, et Shuqing Wang, « A developed method of tuning PID controllers with fuzzy rules for integrating processes », in *Proceedings of the 2004 American Control Conference*, Boston, MA, USA, 2004, p. 1109-1114 vol.2. doi: 10.23919/ACC.2004.1386720.

[4] Kiam Heong Ang, G. Chong, et Yun Li, « PID control system analysis, design, and technology », *IEEE Trans. Contr. Syst. Technol.*, vol. 13, nº 4, p. 559-576, juill. 2005, doi: 10.1109/TCST.2005.847331.

[5] Zoghmar Mahieddine et Habchi Aboubakar seddik, « Étude comparative entre deux régulateurs PID et FLC appliqués à la Machine à Courant Continu », Université Larbi Ben M'Hidi de Oum Elbouaghi, memoire de fin d'études, juin 2012.

[6] atef naghmouchi, « Conception des régulateurs classique, flou et neuro-flou pour la régulation de moteur à courant continu », 14:39:35 UTC. Consulté le: mai 18, 2021. [En ligne]. Disponible sur: https://fr.slideshare.net/atef220/article-du-pfe

[7] N. Kharrat, H. Abid, H. H. Abdallah, et A. Wali, « Implantation d'algorithmes de commandes en vitesse d'un moteur à courant continu », p. 6.

[8] Jyh Shing Roger Jang, « ANFIS: adaptive-network-based fuzzy inference system », *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 23, nº 3, p. 665-685, juin 1993.

[9] Erik Bartmann, *Le grand livre d'arduino*, 2ème. eyrolles, 2015.

[10] Myo Maung Maung, Maung Maung Latt, Chaw Myat Nwe, « DC Motor Angular Position Control using PID Controller with Friction Compensation », *International Journal of Scientific and Research Publications*, vol. 8, p. 7, nov. 2018.

[11] Kessari Amel, et Djafer Khodja Ibtissem, « Etude et réalisation d'un robot mobile à trajectoire programmée avec éviteur d'obstacles », Université Akli Mohand Oulhadje-Bouira, Republique algérienne democratique et populaire, Mémoire de fin d'études, 2019 2018.

[12] robot maker, « Utilisation des encodeurs moteurs CC avec Arduino ». déc. 11, 2021. [En ligne]. Disponible sur: https://www.robot-maker.com/shop/blog/32_Utilisation-des-encodeurs.html

[13] Go Tronic, « Moto réducteur + encodeur FIT0521 ». Consulté le: déc. 09, 2021. [En ligne]. Disponible sur: https://www.gotronic.fr/art-motoreducteur-encodeur-fit0521-27897.htm

[14] Pobot, sophia antipolis, « Asservissement d'un moteur à courant continu ». Consulté le: déc. 09, 2021. [En ligne]. Disponible sur: https://pobot.org/Asservissement-d-un-moteur-a.html

[15] Amit Kumar Sahoo, Sweet Suman, « Speed Control of DC motor Using Hybrid Fuzzy-PID Controller », *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, vol. 6, p. 6.

[16] académie Toulouse /SI/STI2D, « Le moteur à courant continu Modélisation causale ». [En ligne]. Disponible sur: https://disciplines.ac-toulouse.fr/sii/sites/sii.disciplines.ac-toulouse.fr/files/se_former/formation_modelisation_multiphysique/2-tp_mcc_modelecausal.pdf

[17] académie de Toulouse, « Le moteur à courant continu Ajuster les paramètres du modèle acausal à partir des mesures ». Consulté le: déc. 11, 2021. [En ligne]. Disponible sur: https://disciplines.ac-toulouse.fr/sii/sites/sii.disciplines.ac-toulouse.fr/files/se_former/formation_modelisation_multiphysique/4c-parametres_modeleacausal.pdf

[18] M. Ajaamoum, M. Kourchi, B. Bouachrine, A. Ihlal, and L. Bouhouch, « Comparison of Takagi-Sugeno fuzzy controller and the command "P & O" for extracting the maximum power from a photovoltaic system », *International Journal of Innovation and Applied Studies*, vol. 10, nº 1, p. 192-206, janv. 2015.

[19] T. Takagi, M. Sugeno, « Fuzzy identification of systems and its application to modeling and control », *IEEE Transactions on Systems,*

Man, and Cybernetics, vol. Vol. SMC-15, nº No 1, p. 116-132, févr. 1985.

[20] N. Martaj et M. Mokhtari, *MATLAB R2009, SIMULINK et STATEFLOW pour Ingénieurs, Chercheurs et Etudiants*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. doi: 10.1007/978-3-642-11764-0.

[21] F. Belhachat, C. Larbes, L. Barazane, et S. Kharzi, « Commande neuro-floue d'un hacheur MPPT », p. 8, 2007.

[22] Bouchiba Fayçal, « Navigation neuro-floue d'un robot mobile dans un environnement inconnu », Thèse, Université des sciences et de la technologie d'Oran Mohamed Boudiaf, 2017.