XLIII CILAMCE
21–25, November, 2022
FOZ DO IGUAÇU – PR – BRAZIL

Gaussian Adaptive PID control (GAPID) and the Fuzzy logic PID control (FLPID) Tuned by Particle Swarm Optimization for a speed control in a BLDC motor

Carlos da Conceição Castilho Neto[1], Taysa Millena Marques[1], Hugo Valadares Siqueira[1], Marcella Ribeiro Martins[1], Mauricio dos Santos Kaster[1], Fernanda Cristina Corrêa[1].

[1]*Dept. of Electrical Engineering, Federal University of Technology — Paraná*
Avenue Doutor Washington Subtil Chueire, 330 - Jardim Carvalho, 84017-220, Paraná, Brazil
carlos.castilhoneto@gmail.com, fernandacorrea@utfpr.edu.br

Abstract. The usage of Brushless Direct Current Electric Motors (BLDC) is each time more frequent in industrial appliances such as the automobile segment. In such application the BLDC motor is exposed to many types of charge disturbances which makes the conventional control methods, such as proportional–integral–derivative controller (PID), not reaching its variables with precision in cases of sudden perturbation and variation of the parameters. Thus, the PID controller might have its performance improved with the application of adaptive techniques which collect data from the operating system environment and make adjusts based in the condition where it is found. In this case, two techniques are chosen: the Gaussian Adaptive PID Control (GAPID) and the Fuzzy logic PID Control (FLPID), choosing its parameters is not an easy task, although they can be obtained through the usage of optimization techniques, such as the Particle Swarm Optimization (PSO), ensuring a better performance and robustness of the GAPID and FLPID compared to the linear PID by load and gain sweep tests, achieving fast response and minimal variation. This paper aims to accomplish the comparison between different control techniques for the speed control in a BLDC motor and its practical implementation in a ESP32 microcontroller.

Keywords: BLDC, PID, Particle Swarm Optimization, GAPID, Fuzzy Logic.

1 Introduction

High efficiency, longer service life, lower noise emission, and high dynamic response are some characteristics of the Brushless Direct Current Electric Motors (BLDC). Such attributes make the motor BLDC to be attractive mainly for electric vehicles, robotics, and aviation. The simplicity and easy applicability of the Proportional–Integral–Derivative controller (PID) are its strong point as well as its weak point, in such applications, the BLDC motor is exposed to different types of load disturbance, which can make the controller unresponsive, not reaching the desired performance accurately in a short response time, in other words, the PID controller may not be the most suitable option for specific cases [1].

However, adaptive techniques, which are commonly adopted in dynamic systems in unstable environments, can be applied to the PID controller. This union integrates the advantage of both control structures, robustness and adaptability, when plant parameters vary or sudden disturbance occurs [2].

In a universe of countless adaptive alternatives, the objective of this paper is to study the widely known technique, Fuzzy logic and Gaussian Adaptive Proportional, Integral and Derivative controller (GAPID) strategy, generally applied to typical power supplies of medical equipment with the Buck converter. Even though there are not many examples of the application of the GAPID controller to electric motors, it should be effective, practical and straightforward to apply on it [3].

According to Simões and Shaw [4] and Puchta et al. [3], there is no algebraic solution for the adaptive parameters tuning of both controllers (GAPID and FLPID), which requires the designer's full knowledge of the system whereupon it will be applied, making the choices adopted not being the ideal ones to obtain the maximum controller performance.

Thus, one of the ways to obtain these parameters would be the use of additional tools, such as Particle Swarm Optimization (PSO). This bio-inspired algorithm is based on the interaction between several particles with a range of random values in search of the best resolution for the proposed problem.

This study aims to analyze and compare the behavior of these control techniques for load variations applied to the speed control of a BLDC motor using an ESP32 microcontroller.

2 Literature Survey

2.1 PID Controller

Controllers are subsystems that act on a given system or plant to achieve pre-established results. When correctly dimensioned, controllers can increase the efficiency of the system in which they operate and when used in a closed-loop, where the system feedback with the output signal occurs, the reduction of the output error in relation to the input signal can occur [5].

Because it is simple, robust, and has few adjustment parameters, 90% of industrial loops apply PID controllers [6].

The PID controller is composed of the sum of three gains, proportional gain reduces rise time and steady-state error, integral gain has the effect of eliminating steady-state error but negatively impacts transient response, and derivative gain increases system stability and improves transient response Kumar et al. [7].

According to [8], the control signal provided by the PID controller depends on three parameters, which are given by eq. (1).

$$u(t) = K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{de(t)}{dt} \tag{1}$$

Where:

- u(t) represents the control signal;
- e(t) represents the Steady-state error, which is the difference between the Set point $r(t)$ and the Output $y(t)$;
- K_p represents the Proportional gain;
- K_i represents the Integral gain;
- K_d represents the Derivative gain.

2.2 FLPID Controller

Fuzzy logic seeks to approach human thought, leaving boolean logic and finding answers by dealing with the concept of partial truth. It happens because the basis of fuzzy systems is the Fuzzy set theory, in which its elements have a degree of membership determined by the analysis of the membership functions [9].

One of the main advantages of using the Fuzzy controller is that the designer can control a system based on its behavior without raising its mathematical model [10].

For the processing of numerical variables sent to the controller to happen, it is necessary to start a process that consists of transforming these numerical information into linguistic variables to carry out decision-making based on pre-established rules associated with a numerical value necessary to control the plant. These steps (Figure 1) can be defined as fuzzification, inference, and defuzzification [11].

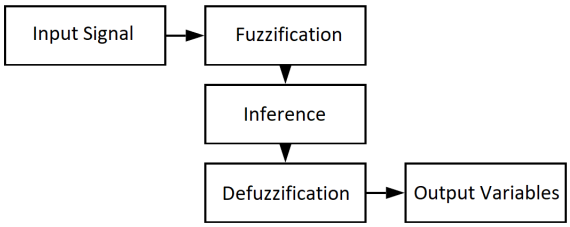

Figure 1. Fuzzy logic structure

The fuzzification step transforms the input data, which are numerical variables, into linguistic variables, where a pre-processing of categories takes place to reduce the number of processes. Then, in inference, decisions

are made based on the If-Then conditional, defined by a base of previously established rules, defining the actions to be taken on a given occasion. The last step of signal processing, defuzzification, ensures the exact interpretation of the linguistic variables obtained in the inference phase into numerical values [11].

As it is a widely studied technique, the fuzzy logic can be applied to solve different situations, from solving economic problems to controlling a motor. This technique proved to be effective for speed control of a BLDC motor where several tests are performed to investigate the performance for sudden disturbances and parameter variations, showing that the controller is robust and suitable for high performance drive applications [12].

2.3 GAPID Controller

Gaussian adaptive control is based on the use of Gaussian functions with well-defined upper and lower limits, adjustable concavities, and smooth derivatives, which causes the gains to be gradually increased or decreased as the stationary error approaches zero. These functions are defined as shown in eq. (2).

$$f(\gamma) = K_1 - (K_1 - K_0)e^{-q.\gamma^2} \tag{2}$$

Where:

- γ represents the error;
- K_1 and K_0 represents the function limits;
- $-q$ represents regulator of the concavity of the Gaussian curve.

The $-q$ regulator adjusts the position of the range of input values where the transition between K_0 and K_1 occurs. K_0 has a more significant influence when the stationary error approaches zero and K_1 when the error is significant. Note also that when K_0 ¡ K_1, the concavity is facing upwards, and when K_0 ¿ K_1, the concavity is facing downwards.

This technique has been studied in recent years in several applications, among the most cited are the application in a DC-DC Buck converter, commonly used in hospital electronic devices. Allied to the PID controller, the GAPID, has shown to obtain short rise time and settling time and low overshoot for non-linear loads [3].

2.4 Particle Swarm Optimization

Particle swarm optimization (PSO) was initially proposed in 1995 by James Kennedy and Russell Eberhart, in which they sought to describe the collective behavior of groups of animals in a mathematical way, where the individual behavior of each member that composes the group is analyzed and the social impact it has on its neighbors [13].

As it is based on biological models, the algorithm uses basic rules to generate competitive and/or cooperative behavior among individuals to find the best solution for a given problem [14]. As it is simple and robust, this method has been successfully applied in several areas of engineering to find solutions to various problems [14].

Over time, several researchers sought ways to increase the performance of the PSO, among the improvements, it is worth highlighting the stability analysis and the understanding of the dynamics of the particle swarm, which is based on the influence that the group has on the particle.

This cultural adaptation can be summarized in three principles: The individual and collective perception of the particle, the comparison between individuals, and the imitation of the best particles [15].

To optimize a problem, it is necessary to initialize a randomly distributed population composed of individuals or particles with unique positions x_i and speeds v_i. These particles are represented by a vector whose dimension is the domain of the fitness function that is evaluated by each particle in each iteration, resulting in a change in the position and velocity of each individual. The number of iterations (i) and the number of individuals influence the amount of processing used, and the accuracy of the result obtained. Thus, the Particle Swarm Optimization is represented by eq. (3).

$$V_i(t+1) = \omega v_i + c_1 r_1(pBest - x_i) + c_2 r_2(gBest - x_i) \tag{3}$$

Where:

- ω represents the inertia weight;
- v_i represents the previous velocity of the particle;

- c_1 represents the cognitive parameter;
- r_1 and r_2 represents a random number;
- $pBest$ represents the local best position;
- c_2 represents the social scaling parameter;
- $gBest$ represents the global best position;
- x_i represents the previous position of the particle.

3 Experimental Development

The control techniques previously presented were applied to a system consisting of a BLDC Racerstar BR2212 1800KV motor, a 40 amps electronic speed controller (ESC), the HTR-W2-360-3PP encoder to close the control loop that emits 360 pulses per revolution, and an ESP32 microcontroller.

As the access to all the constructive parameters that constitute the BLDC motor is not an easy task, the system was compared to a black box.Thus, in order to obtain a mathematical model that resembles the behavior of the system, it is necessary to carry out experimental open-loop tests [8].

Initially, open-loop tests were performed, applying 11.1 volts to the motor input to obtain the steady-state speed and, consequently, the maximum number of pulses emitted by the encoder in the sampling time Ts.

Based on the results obtained, it was evident that it was a first-order system, facilitating the choice of the time constant that in first-order systems can be obtained when reaches 63% of its maximum capacity [8], in this case, the maximum speed.Therefore, the transfer function in the Z plane is represented by eq. (4).

$$P(z) = \frac{344.5}{z - 0.3679} \tag{4}$$

3.1 PID Controller

The PI controller design was based on Ziegler–Nichols method without the derivative gain (K_d) because it is a first-order system. Thus, with the integral gain (K_i), it is possible to correct the stationary error and with the proportional gain (K_p) to obtain an improvement in the response speed.In this way, the performance parameters adopted for the development of the controller were the percentage overshoot (PO) of at most 1.3% and a stabilization time (Te) of 0.5 seconds, resulting in eq. (5).

$$K(z) = \frac{0.0009113z + 0.0002364}{z - 1} \tag{5}$$

3.2 FLPI Controller

The design of this controller was carried out in an empirical way, where it has the function of improving the previously presented PI controller. In this case, two inputs were adopted the first input was the error sign (E) which is the difference between the chosen reference and the output signal, and the second input is the error derivative (de) which indicates the proximity of the output signal with the reference helping fine-tune the controller, resulting in two outputs called K_p and K_i which in turn will be multiplied by the PI controller gains.

As shown in Figure 2, it was chosen for the error inputs and the error derivative to use five membership functions of the triangular type because it is the easiest topology to manipulate and also because it is the most common in many applications of Fuzzy controllers. The membership functions are named NP, NE, ZO, PO e PP, meaning negative plus, negative, zero, positive, and positive plus, respectively. Four membership functions of the triangular type were used for the output variables: Z, S, M, and B, meaning zero, small, medium, and big.

3.3 GAPI Controller Optimized by PSO

The determination of the 6 parameters (Table 1), K_{p_1}, K_{i_1}, K_{p_0}, K_{i_0}, q_p and q_i, is not an easy task to perform empirically. Due to the complexity of the problem, it was decided to use the PSO to obtain them.

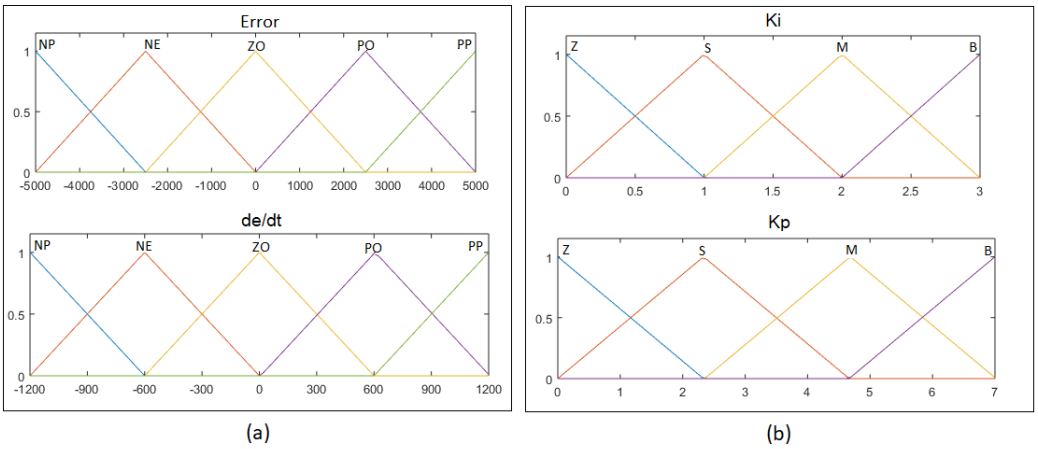

Figure 2. (a) Entries of membership function E and de, (b) Output membership function K_p and K_i

Instead of using the specialized solution, which consists of the direct application of the Gaussian function into the plant, it was decided to link the set of parameters to the gains of the linear PI controller and, in this way, take advantage of the same project requirements but seeking better performance. For the optimization algorithm (PSO), 10 simulations were compared, and the best result was chosen among them, and in each simulation 50 particles with 150 iterations, c_1 and c_2 were empirically chosen 1.5 and 2.5, respectively and fixed inertia coefficient (ω) of 0.5 were adopted.

Table 1. GAPI Parameters

Parameters	Value
K_{p_1}	3.17901
K_{i_1}	7.73366
K_{p_0}	-1.03353
K_{i_0}	0.39996
q_p	6.59485
q_i	5.16543

3.4 FLPI Controller Optimized by PSO

In each simulation 40 particles with 40 iterations, c_1 and c_2 were empirically chosen 1.5 and 2.5, respectively and fixed inertia coefficient (ω) of 0.5 were adopted. With the simulations results, the membership functions continued to be of the triangular type, but the universe of discourse for each variable underwent changed (Figure 3).

4 Implementation Results

Figure 4 shows closed-loop simulations were performed for a set point of 2900 pulses/Ts for the four controllers, also a load disturbance was inserted to investigate the robustness of each control technique used, where the load is placed in 5 and 15 seconds and withdrawn at 10 and 20 seconds.

According to the graph presented and the measured data in Table 2, it is noted that the FLPI controller optimized by the PSO has a faster response to load disturbances, managing to maintain the lowest percentage of overshoot between the four analyzed controllers.

In addition, it is also worth mentioning the robustness of the PI controller, which despite having a slow response to the load variation, still manages to adapt and reach the established reference, having a shorter stabilization time than the other three controllers.

Although the GAPI controller does not have a reasonable overshoot control as the adaptive controllers based on fuzzy logic, it fulfilled its function being effective and leading to a significant improvement for the PI controller

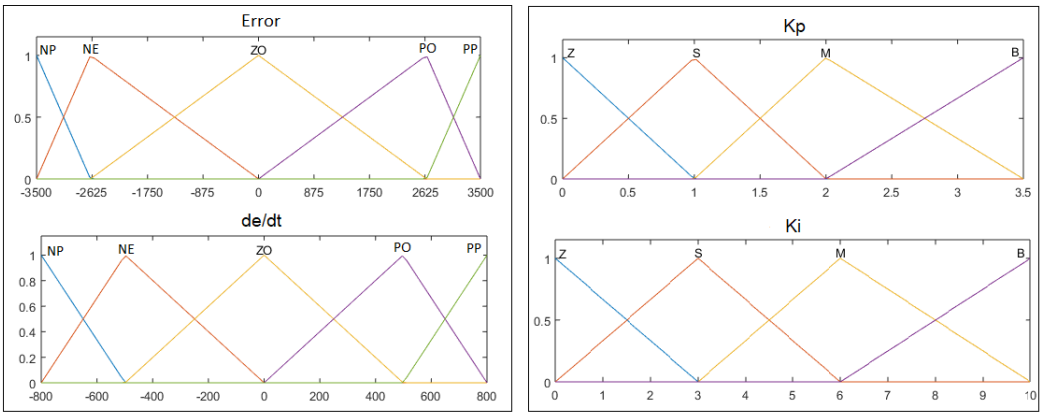

Figure 3. Entries of membership function E and de, (b) Output membership function K_p and K_i obtained by PSO

with an accommodation time 2 seconds faster than the FLPI controller optimized by PSO and a 20% reduction in overshoot. For better understanding, the best results were highlighted in bold on Table 2.

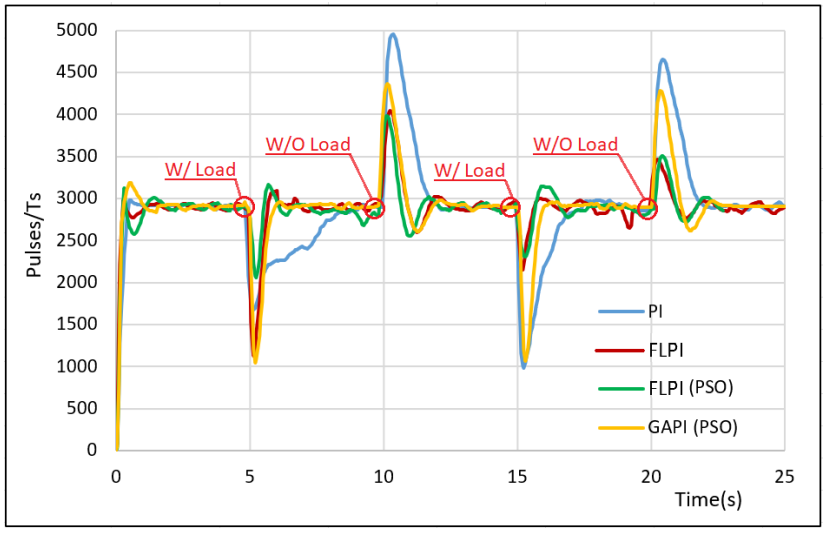

Figure 4. Curve of the four controllers implemented with load for a reference of 2900 pulses/Ts

Table 2. Implementation Results with load variation

Section	PI	FLPI	FLPI (PSO)	GAPI (PSO)
Rise Time (s)	0.2970	0.1391	0.1512	**0.1289**
Settling Time (s)	**21.8275**	24.71	24.4115	22.3991
Overshoot (%)	70.9310	39.3793	**37.3448**	50.5513
Peak Time (s)	10.35	10.25	**10.15**	**10.15**

5 Conclusions

For the proposed study, the control technique widely used in the industry, the proportional-integral controller, was not enough to meet the needs of the project precisely because it does not behave linearly at times when there are load variations.

Thus, one of the ways to mitigate this problem would be the application of adaptive techniques such as Fuzzy or Gaussian. However, for such techniques to be used correctly in the controllers, it is necessary that the designer has full knowledge of the system to be controlled and, as it is a completely empirical control, the values used in the simulation and implementation may not be ideal.

In this way, using an optimization algorithm was the best way to get reliable results, the FLPI controller, when properly designed, proves to be a powerful tool. The GAPI controller fulfilled its role by supplying some evident deficiencies in the PI controller for the test proposal of this paper.

Both controllers (GAPI and FLPI) fulfilled their role by increasing the performance of the PI controller with a shorter rise time, shorter peak time and mainly its adaptability to load variations.

It is evident that the FLPI controller obtained the best results, but its implementation is more complex, requiring the use of a microcontroller with greater processing capacity, such as the one applied in this study, the ESP32. However, the GAPI controller is simpler and obtained similar results.

In conclusion, the union of these control and optimization techniques resulted in two controllers that were able to supply the low response time of the PI controller for load variations and reduce the complexity and eliminate the empiricism of the applied adaptive techniques. Also, for this specific case, it is up to the designer to decide whether to use a more precise and complex controller that requires a powerful processor (FLPI), or a simple controller with reasonable results (GAPI).

Acknowledgements. This work was conducted during graduate and master graduate scholarships supported by the Brazilian Federal Agency for Support and Evaluation of Graduate Education (CAPES), National Council for Scientific and Technological Development (CNPq), and the Federal Technological University of Paraná (UTFPR-PG).

Authorship statement. The authors hereby confirm that they are the sole liable persons responsible for the authorship of this work, and that all material that has been herein included as part of the present paper is either the property (and authorship) of the authors, or has the permission of the owners to be included here.

References

[1] M. A. Ghany, M. A. Shamseldin, and A. A. Ghany. A novel fuzzy self tuning technique of single neuron pid controller for brushless dc motor. In *2017 nineteenth international middle east power systems conference (MEPCON)*, pp. 1453–1458. IEEE, 2017.

[2] R. Goswami and D. Joshi. Performance review of fuzzy logic based controllers employed in brushless dc motor. *Procedia computer science*, vol. 132, pp. 623–631, 2018.

[3] E. D. P. Puchta, P. Bassetto, L. H. Biuk, M. A. Itaborahy Filho, A. Converti, M. d. S. Kaster, and H. V. Siqueira. Swarm-inspired algorithms to optimize a nonlinear gaussian adaptive pid controller. *Energies*, vol. 14, n. 12, pp. 3385, 2021.

[4] M. G. Simões and I. S. Shaw. *Controle e modelagem fuzzy*. Editora Blucher, 2007.

[5] K. Ogata. *Engenharia de controle moderno*. PRENTICE HALL BRASIL, 2011.

[6] V. Chopra, S. K. Singla, and L. Dewan. Comparative analysis of tuning a pid controller using intelligent methods. *Acta Polytechnica Hungarica*, vol. 11, n. 8, pp. 235–249, 2014.

[7] B. Kumar, S. K. Swain, and N. Neogi. Controller design for closed loop speed control of bldc motor. *International Journal on Electrical Engineering and Informatics*, vol. 9, n. 1, pp. 146, 2017.

[8] N. S. Nise and da F. R. Silva. *Engenharia de sistemas de controle*, volume 3. LTC, 2002.

[9] F. M. De Azevedo, L. M. Brasil, and de R. C. L. Oliveira. *Redes neurais com aplicações em controle e em sistemas especialistas*. Visual Books, 2000.

[10] G. Feng, G. Lu, D. Sun, and S. Zhou. A model reference adaptive control algorithm for fuzzy dynamic systems. In *Proceedings of the 4th World Congress on Intelligent Control and Automation (Cat. No. 02EX527)*, volume 4, pp. 3242–3246. IEEE, 2002.

[11] K.-M. Choi, S.-W. Cho, D.-O. Kim, and I.-W. Lee. Active control for seismic response reduction using modal-fuzzy approach. *International Journal of Solids and Structures*, vol. 42, n. 16-17, pp. 4779–4794, 2005.

[12] A. A. El-Samahy and M. A. Shamseldin. Brushless dc motor tracking control using self-tuning fuzzy pid control and model reference adaptive control. *Ain Shams Engineering Journal*, vol. 9, n. 3, pp. 341–352, 2018.

[13] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95-International Conference on Neural Networks*, volume 4, pp. 1942–1948. IEEE, 1995.

[14] E. Garcia-Gonzalo and J. L. Fernandez-Martinez. A brief historical review of particle swarm optimization (pso). *Journal of Bioinformatics and Intelligent Control*, vol. 1, n. 1, pp. 3–16, 2012.

[15] R. C. Eberhart, Y. Shi, and J. Kennedy. *Swarm intelligence*. Elsevier, 2001.