Research Article/Electrical,Electronic and Telecommunications Engineering http://doi.org/10.15446/ing.investig.114438

Active Disturbance Rejection Control of a DC Brushed Motor Using Simulink and Raspberry Pi

Control por rechazo activo de perturbaciones de un motor DC con escobillas utilizando Simulink y Raspberry Pi

Mario E. González N. [1], Oscar H. Sierra H. [2], and Oscar I. Higuera M. [3]

ABSTRACT

Active disturbance rejection control (ADRC) is a robust methodology that does not require precise knowledge of the plant. Developed in China by Professor Jingqing Han, it is based on PID control, state observers, and nonlinear functions. Brushed DC motors are known for their low cost and the noise they introduce into control circuits. This paper demonstrates that ADRC can effectively control low-power brushed DC motors using a general nonlinear model and Simulink for tuning. The model is simulated using parameters provided by the manufacturer. An ADRC developed and programmed by the authors in MATLAB is then integrated into the simulation. The controller is tuned, and its performance is verified. Subsequently, the ADRC is implemented on a Raspberry Pi 3 using MATLAB's support packages and methods developed by the authors. The controller is tested on a Faulhaber 2342L012CR DC motor (12 V/17 W). The results show that it is possible to control the position of the low-power brushed DC motor through simulation-based tuning. The interaction between Simulink and Raspberry Pi 3 enables an optimal control characterized by a fast response, a minimal steady-state error, and no perceptible overshoot. This implementation demonstrates that ADRC is a practical and efficient control method for brushed DC motors.

Keywords: ADRC, MATLAB, coreless brushed DC motor, automatic control, Raspberry Pi

RESUMEN

El control activo de rechazo de perturbaciones (ADRC) es un método robusto que no requiere un conocimiento preciso de la planta. Desarrollado en China por el profesor Jingqing Han, se basa en control PID, observadores de estado y funciones no lineales. Los motores DC con escobillas son conocidos por su bajo costo y el ruido que introducen en los circuitos de control. Este artículo demuestra que ADRC puede controlar eficazmente motores de DC de baja potencia con escobillas utilizando un modelo general no lineal y Simulink para su sintonización. Este modelo se simula utilizando parámetros proporcionados por el fabricante. Luego, se integra a la simulación un ADRC desarrollado y programado por los autores en MATLAB. Se sintoniza el controlador y se verifica su rendimiento. Posteriormente, el ADRC es implementado en una Raspberry Pi 3 utilizando los paquetes de apoyo de MATLAB y métodos desarrollados por los autores. El controlador es puesto a prueba en un motor de DC Faulhaber 2342L012CR (12 V/17 W). Los resultados muestran que es posible controlar la posición del motor de DC con escobillas de baja potencia mediante una sintonización basada en simulación. La interacción entre Simulink y Raspberry Pi 3 permite un sistema de control óptimo caracterizado por una respuesta rápida, un error mínimo en estado estacionario y ningún sobrepico perceptible. Esta implementación demuestra que el ADRC es un método de control práctico y eficiente para motores de DC con escobillas.

Palabras clave: ADRC, MATLAB, motor DC con escobillas sin núcleo, control automático, Raspberry Pi

Received: May 15th, 2024
Accepted: May 19th, 2025

Introduction

The concept and applications of feedback control have historically attracted significant interest, given the mathematical explanations to the ingenious mechanisms of control used on military applications during the second world war. Feedback control is one of the most used, and it is in constant change, getting better every day. There is always some kind of control device involved in every engineering process [1]. The literature on feedback control spans more than six decades of research. These developments were parallel to the mathematics applied in different academic disciplines as the formulation of

[1] Electronics engineer, Universidad Pedagógica y Tecnológica de Colombia, Colombia. MEng, Universidad Pedagógica y Tecnológica de Colombia, Colombia. Affiliation: Full profesor, Electromechanical Engineering, Universidad Pedagógica y Tecnológica de Colombia, Colombia. Email: marioeduardo.gonzalez@uptc.edu.co

[2] Electronics engineer, Universidad Pedagógica y Tecnológica de Colombia, Colombia. MEng, Universidad Pedagógica y Tecnológica de Colombia, Colombia. Affiliation: Full profesor, Electronics Engineering, Universidad Pedagógica y Tecnológica de Colombia, Colombia. Email: oscarhumberto.sierra@uptc.edu.co

[3] Electronics engineer, Universidad Pedagógica y Tecnológica de Colombia, Colombia. MEng, Universidad Nacional de Colombia, PhD(c) Universidad Pedagógica y Tecnológica de Colombia, Colombia. Affiliation: Full profesor, Electronics Engineering, Universidad Pedagógica y Tecnológica de Colombia. Email: oscar.higuera@uptc.edu.co

optimal control and the way to investigate it became more axiomatic and deductive [2].

Active disturbance rejection control (ADRC) allows controlling complex systems with almost zero knowledge of their dynamics. It represents a new paradigm in control theory since the concepts of *robustness* and *sensitivity to unmodeled dynamic perturbations* must be redefined. ADRC seeks to adapt the dynamic behavior of the system to the control strategy, rejecting external perturbations with the purpose of ensuring that the controller operates effectively in the presence of uncertainties [2]. ADRC has great potential for solving control problems in industrial processes characterized by elevated levels of uncertainty [3].

There is a considerable amount of bibliography related to ADRC, but there are few records of its implementation. This scarcity is due to its use of two nonlinear functions. A more popular variant called *linear active disturbance rejection control* (LADRC) is more commonly implemented and has a variety of tuning strategies available in the literature [4]. Despite this, the original ADRC remains a challenging controller to design and implement [5]. The design of an ADRC controller is particularly difficult due to its nonlinear nature, wherein empirical tuning through simulation before testing is the most widespread and recommended approach [6], [7].

The ADRC theory was developed in China by professor Jingqing Han. Based on proportional-integral derivative (PID) controllers [1], this theory has been simulated and implemented in power electronics [8], drone altitude control [9], robotics [10], tank level control [11], and permanent magnet synchronous motors [12], among others [13]. Since this is a relatively new theory, its use in practical applications has been very limited. Besides, it requires a considerable amount of time and resources for its implementation [14]. There are some implementations involving field programmable gate arrays (FPGAs) [15] and digital signal processors (DSPs) [16], but they are extremely complex and difficult to replicate.

Despite the complexity of ADRC implementations on platforms like DSPs and FPGAs [14]–[16], recent studies have shown that affordable embedded systems such as the Raspberry Pi offer a viable alternative for real-time control experiments [17]–[19]. The Raspberry Pi 3, with its quad-core ARM processor and compatibility with Simulink [20]-[22], enables the implementation of advanced controllers without the need for specialized hardware. Additionally, MATLAB/Simulink environments are widely used in control education and research due to their simulation capabilities and intuitive design tools [23].

The implementation of electronic control systems requires, at a minimum, the use of an analog-to-digital (A/D) converter, power systems, and processing systems, which include the controller, pulse width modulation (PWM) signal generators, and other signal processing components [24]. These systems can be complex and hard to implement on platforms like FPGA, DAQ, DSPs, *etc.* [14].

Brushed DC motors are widely used in robotics and small appliances due to their low cost and high torque at low rpm. However, they are prone to noise from brushes and heat generation, which can alter their performance over time. Therefore, it is important to implement a dynamic controller that is capable of adapting to changing parameters in order to achieve more efficient systems [25].

In the literature on the implementation of controllers with DC brushed motors, significant works include [26], where a permanent-magnet brushed DC motor is controlled via an ADRC to compensate for load torque and variations in the rotor shaft, which do not require measuring the rotor shaft speed since they can be verified only through simulation; [27], where the speed of a brushless DC motor is controlled using a PID and artificial neural network-based controller; and [25], where a software is developed to control brushed DC motors using a PI controller tunned dynamically using fuzzy logic.

Works with more emphasis on theory or other types of motors include [6], which is characterized by its focus on controlling a moving mirror system and tuning ADRC parameters through improved snake optimization (I-SO). Moreover, [28] highlights the use of a genetic algorithm (GA) to optimize control in five-phase motors, enhancing the efficiency of a specific AC brushless motor. [29] proposes a set of rules for tuning ADRC in first-order plus delay time models, focusing on deep theoretical analysis without including practical implementation. [30] introduces a combined ADRC and iterative learning control (ILC) strategy to improve torque control in switched reluctance motors (SRM), offering innovative solutions in this specific field. [31] investigates the use of ADRC for speed control in DC brushless motors, comparing this technique against other control methods such as the PI and PID approaches.

Regarding ADRC and low-power motors, the authors of [32] focus on the angular position control of a servo motor, demonstrating the versatility of their method in various motor configurations. [33] provides a detailed analysis applied to a DC servo motor in real time, including the theory, practical implementation, and verification of linear ADRC in a real hardware environment. A related study is presented in [34], where a Raspberry Pi 4 is used together with Simulink (through external mode over Wi-Fi) to implement an ADRC controller for a DC motor. However, this work does not evaluate the real-world performance of the ADRC when tuned through simulation using a nonlinear DC motor model.

Brushed low-power DC motors are typically controlled using PID controllers. Nevertheless, there are no implementations

of ADRC for brushed low-power DC motors. The authors aim to address this gap by tuning an ADRC controller using a nonlinear DC motor model and Simulink. The tuned parameters will then be used to effectively control the motor.

The objectives of this paper are to tune an ADRC controller through simulation using a nonlinear DC motor model and Simulink, to demonstrate the control of a coreless brushed low-power DC motor, and to compare its simulated and actual performance.

This paper is organized as follows. First, the theoretical background of ADRC and the methodology used to design the control system are presented. Then, the results from both simulation and physical implementation are described. Finally, the main conclusions of this research are discussed.

Methodology

The methodology used in this research was empirical and analytical, which implies a scientific research model based on experimentation and experimental logic, as well as on the observation and application of the concepts around ADRC. To obtain the results described in this work, the control system design was simulated and implemented using a Raspberry Pi 3.

Active disturbance rejection control

To design and implement an ADRC, perturbations must be managed as time-variant signals. This can be done through extended state observers (ESOs). ESOs support the ADRC by estimating the total action of uncertain models and perturbations on the system. Knowing these data allows reacting to and canceling the perturbations and undesirable effects arising from the nonlinearities in the system, thereby enabling robust control [35], [36].

To implement an ADRC, the system under study must be regarded as nonlinear and dynamic. This is described in Eq. (1), where t is the time, ω are the external perturbations, u is the input, y is the output, and b is a real constant of the system, which can be obtained by estimating $b0$ [37].

$$\ddot{y}(t) = f\left(t, y, \dot{y}, w\right) + bu \tag{1}$$

Eq. (1) expresses the unknown dynamics of a time-variant system. The principle governing the ADRC is the estimation and cancelation of (1) by means of derivation. The ADRC consists of three fundamental parts: a tracking differentiator (TD), an ESO, and nonlinear state error feedback (NLSEF) [38]-[39].

The general scheme of the ADRC is presented in Fig. 1.

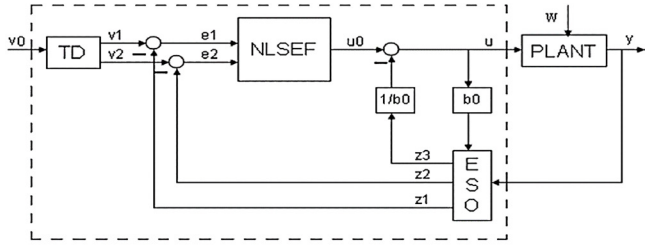

Figure 1. ADRC structure
Source: [40]

The structure of the signals that govern the ADRC is shown in Eqs. (2) to (10) [41].

$$v_1 = v_1 + hv_2 \tag{2}$$

$$v_{2=} v_2 + hfh \tag{3}$$

$$z_1 = z_1 + h\left(z_2 - \beta_{01} e\right) \tag{4}$$

$$z_2 = z_2 + h\left(z_3 - \beta_{02} fe + b_o u\right) \tag{5}$$

$$z_3 = z_1 + h\left(-\beta_{03} fe_1\right) \tag{6}$$

$$e_1 = v_1 - z_1 \tag{7}$$

$$e_2 = v_2 - z_2 \tag{8}$$

$$u_0 = k_p\left(r - z_1\right) + k_d\left(\dot{r} - z_2\right) \tag{9}$$

$$u = \left(u_0 - z_3\right) / b_o \tag{10}$$

Where:

- h1: the coefficient of precision that determines the aggressivity of the control cycle and is often a multiple of the sampling time h
- $\beta 0_1$, $\beta 0_2$, $\beta 0_3$: gain vectors
- e_1, e_2: differential error
- r_0 : tracking speed
- b_0: a constant that ensures system robustness
- h: parameter used to track the range of the differential input signal
- z_1, z_2, z_3: ESO outputs
- v_1, v_2: TD outputs
- v_0: TD input

In addition, β_{01} is the proportional gain parameter analogous to proportional gain of a PID controller, and β_{02} is the differential gain, also analogous to the differential gain of a PID controller. If the value of β_{02} is too high, high-frequency noise may occur. The control law relies on the differential errors e_1 and e_2, so the design will behave like a PD controller. The parameter r defines the tracking speed – a larger value of r means a faster speed. Moreover, the parameter h tracks the range of the differential input signal, and $b0$ is a gain constant used to tune the controller and adjust the control signal magnitude. This constant also helps the ESO with perturbation and system error estimation [40].

The system used to tune the controller was the nonlinear model of the DC motor described in Eqs. (11) and (12) [42].

$$L_A I_A = -R_A\, I_A(t) - \Psi w(t) - k_B \left|w(t)\right| I_A(t) + U_A(t) \quad (11)$$

$$J\dot{w} = \Psi\, I_A(t) - M_{F1} w(t) - M_{F0} sign\left(w(t)\right) - M_L(t) \quad (12)$$

Where:

- U: armature voltage
- IA: armature current
- RA: armature resistance
- ψ: magnetic flux
- KB: voltage drop factor
- $MF1$: viscous friction
- $MF0$: dry friction
- ML: input load

Eqs. (11) and (12) were implemented in MATLAB's Simulink (Fig. 2), considering the technical specifications of the Faulhaber 2342L012CR motor datasheet and assuming a 12 V input with a 100% PWM duty cycle. The input of the motor is a PWM signal, and the output is the position of the motor in radians.

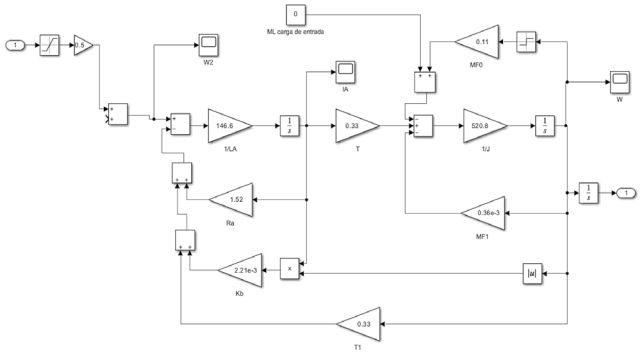

Figure 2. Nonlinear DC motor model implemented in Simulink
Source: Authors

Implementation of the ***fal*** *and* ***fhan*** *nonlinear functions*

Our ADRC was based on nonlinear estimation algorithms and the feedback from nonlinear state variables. In this context, the *fal* and *fhan* nonlinear functions were used. These functions are continuous and linear around the origin d, and they are described in terms of the constants d and α.

The parameters $\beta 01$, $\beta 02$, $\beta 03$ establish the dynamic output of the ESO block and depend proportionally on the magnitude of the perturbation. In addition, the delay relies proportionally on $\beta 03$. The *fal* and *fhan* functions are defined in Eqs. (13) and (14) [40], [43], [44].

$$fal\left(x,\alpha,d\right) = \begin{bmatrix} x / d^{1-\alpha} & ,\left|x\right| \le d \\ \left|x\right|^{\alpha} sign\left(x\right) & ,\left|x\right| > d \end{bmatrix} \quad (13)$$

$$fhan\left(e_1,e2,r,h1\right) = \left\{ \begin{array}{c} d = rh^2, a_0 = hx_2,\ y = x_1 + a_0 \\ a_1 = \sqrt{d\left(d + 8\left|y\right|\right)} \\ a_2 = a_0 + \dfrac{sign\left(y\right)\left(a_1 - d\right)}{2} \\ S_y = \dfrac{sign\left(y+d\right) - sign\left(y-d\right)}{2} \\ a = \left(a_0 + y - a_2\right) S_y + a_2 \\ S_a = \dfrac{sign\left(a+d\right) - sign\left(a-d\right)}{2} \\ fhan = -r\left(\dfrac{a}{d} - sign\left(a\right)\right) S_a - r\, sign\left(a\right) \end{array} \right\} \quad (14)$$

fal and *fhan* were programmed through a MATLAB function. The code used to this effect is presented below.

```
function out = fhan(v1,v2,ro,ho)
  d = ho*ho*ro;
  ao = ho*v2;
  y = v1 + ao;
  a1 = sqrt(d*(d + 8*abs(y)));
  a2 = ao + sign(y)*(a1-d)/2;
  sy = (sign(y+d) - sign(y-d))/2;
  a = (ao + y - a2)*sy + a2;
  sa = (sign(a+d) - sign(a-d))/2;
  out = -ro*(a/d - sign(a))*sa - ro*sign(a);
end

function fe = fal(err,alfa,d)
 if abs(err)<=d
   fe = err/d^(1-alfa);
else
   fe = (abs(err)^alfa)*sign(err);
end
```

Implementation of the ESO block

Different observers have been reported in the literature. The most commonly used for ADRC are the unknown input observer, the perturbation observer, and the ESO. For our ADRC, an ESO was used. An ESO not only estimates and compensates for system perturbations but also each state variable. An adequate ESO design ensures that all the closed-loop variables are bounded, as well as the tracking of the reference signal, even in the presence of uncertainty and perturbations. The ESO uses nonlinear gains for the state observer [45]. The gains fe and fe_1 are expressed in Eqs. (15) and (16). Note that, for fe, $\alpha = 0.25$, and, for fe_1, $\alpha = 0.25$ [40].

$$fe = fal\left(e, 0.5, h\right) \quad (15)$$

$$fe_1 = fal\left(e, 0.25, h\right) \quad (16)$$

The precision of the estimation is determined using ω_0. This is the only parameter in the ESO bandwidth. L is defined in Eq. (17) and corresponds to the bandwidth of the state observer [41].

$$L = \left[l1\ l2\ l3 \right]^T = \left[ew_0, 3w_0^2, w_0^2 \right]^T \quad (17)$$

The code developed for fe is as follows:

```
function y = fcn(err,h)
  alfa=0.5;
  y = fal(err,alfa,h);
end
```

The code developed for fe_1 is presented below.

```
function y = fcn(err,h)
  alfa=0.25;
  y = fal(err,alfa,h);
end
```

The ESO block implemented in Simulink can be seen in Fig. 3. This was done according to Eqs. (2) to (10), (13), (14), and (17).

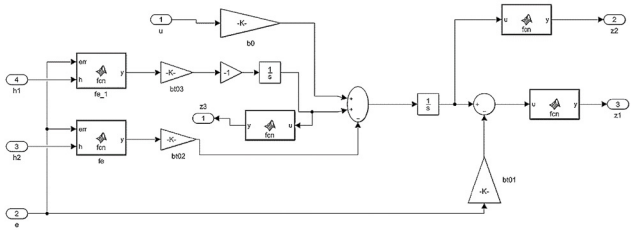

Figure 3. TD block in Simulink
Source: Authors

Implementation of the NLSEF block

The NLSEF block uses the differential signal of the TD and the real-time action of the ESO to control the plant, as shown in Eq. (18) [41].

$$U_0 = -fhan\left(e1, e2, r1, h1\right) \quad (18)$$

The NLSEF block was implemented using a MATLAB function in Simulink. The code used is shown below.

```
function u0 = fcn(r0,h0,c,e1,e2)
  u0=-fhan(e1,c*e2,r0,h0);
end
```

Implementation of the TD block

The code developed in MATLAB for the TD block is as follows:

```
function [fh,v2s] = fcn(h,r0,v0,v1,v2)
  v2s=v2;
  fh=fhan((v1-v0),v2,r0,h);
end
```

The TD block developed in Simulink can be seen in Fig. 4.

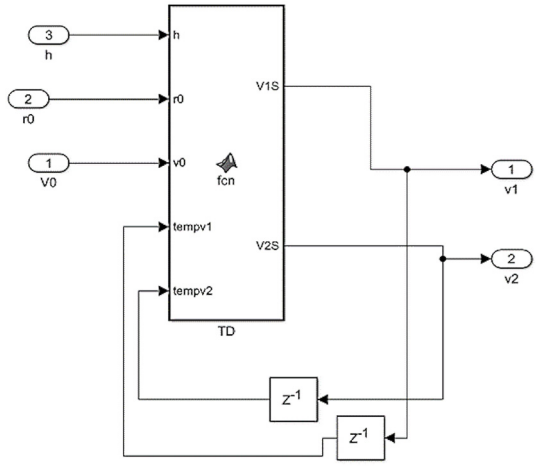

Figure 4. TD block in Simulink
Source: Authors

The scheme developed in Simulink for the ADCR is connected to the nonlinear model of the DC motor (Fig. 5). The implemented ADRC includes the ESO, TD, and NLSEF blocks introduced earlier. The parameters for the ADRC are obtained as shown in [46] *i.e.*, h = 0.01, $\beta 01$ = 100, $\beta 02$ = 300, $\beta 03$ = 10 000, $r_0 = 30$, and $b0 = 1$.

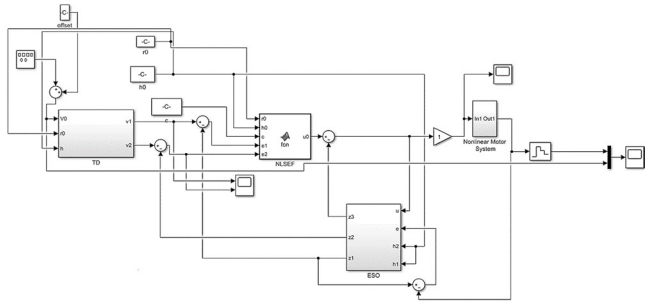

Figure 5. The ADRC and the nonlinear DC motor implemented in Simulink
Source: Authors

Implementation of the ADRC on the Raspberry Pi 3

To program the Raspberry Pi 3 from Simulink, the first step was to install MATLAB's support hardware package for the card. Then, in Simulink, the connection through the ethernet port was configured. This included setting the model of the card and its IP. Afterwards, the libraries developed for the Raspberry Pi 3 were installed. These libraries are used to connect an A/D or a D/A converter to the Pi 3, and they can read or write data from Simulink. The connection is carried out through an S-function. In this case, the converter used was a PCF8591F using the I2C protocol.

The implementation of the controller from Simulink on the Raspberry Pi 3 can be done in two ways, the first of which is maintaining the Pi 3 connected to the computer. This method features the use of scopes and sliders, wherein signals can be visualized in real time and sliders can be used to change parameters while the Raspberry Pi 3 is controlling

the system. To program the card in this way, the *Build model* option is selected in the peripheral section before running the model.

The second method is programming the Pi 3 and using it as an embedded system disconnected from the computer. To this effect, the *external* option must be selected in the simulation settings of the implemented scheme. In this work, the selected sample time was 0.05 s, as there is a noticeable delay when using sample times shorter than 0.01 s, which causes errors during real-time implementation. This delay seems to be due to the processing power of the card.

Fig. 6 presents the scheme of our implementation. Here, the Pi 3 is connected to a computer through an ethernet cable, and the card is connected to a PCF8591F A/D-D/A converter and a L298 H-bridge used to drive the DC motor through a PWM signal using a GPIO pin, wherein the turning sense is controlled by two additional GPIO pins. The L298 is connected to a Faulhaber DC motor coupled to a 10KΩ linear potentiometer, which is in turn connected to a voltage divider. This divider is connected to an instrumentation amplifier, which is in turn connected to the A/D-D/A converter. The converter is used to provide feedback to the Raspberry Pi 3 with information about the motor's position.

Our ADRC, developed in Simulink and implemented on the Raspberry Pi 3, can be seen in Fig. 7. This figure also shows the amplitude and offset correction from the input signal, which are employed to obtain the position of the motor in degrees. The ADRC uses this set of values and reference to calculate the error and uncertainty and compensate for them, using TD, ESO, and NSELF blocks to send a control signal to an A/D converter. Then, these digital signals are sent through GPIO pins to control the DC motor (PWM and turning sense). Simulink scopes can be used to examine the signals coming in and out of the Pi 3 in the computer while the simulation is running. Furthermore, the values of the reference or the ADRC can be modified in real time.

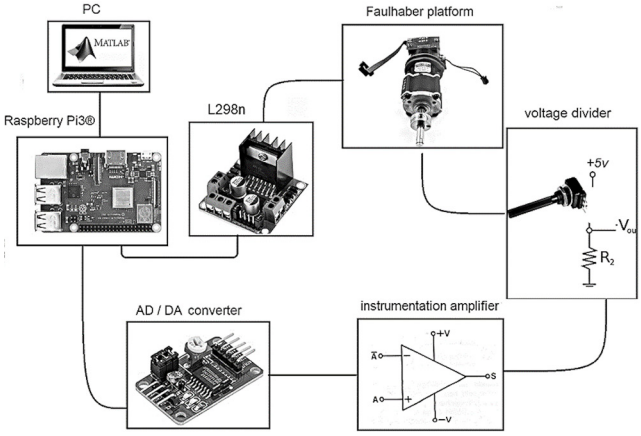

Figure 6. General scheme of the ADRC implemented on a DC motor
Source: Authors

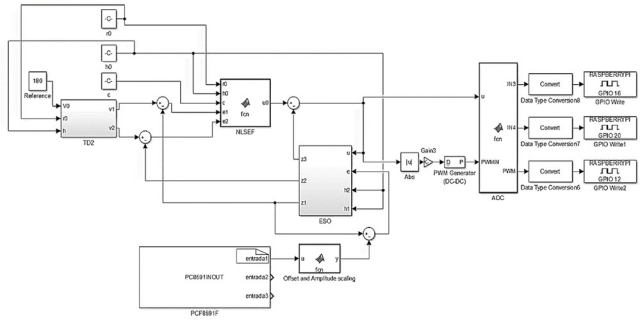

Figure 7. Interaction between Simulink and the Raspberry Pi 3
Source: Authors

Fig. 8 shows the implemented prototype, with the following components: 1) linear potentiometer, 2) Faulhaber DC motor, 3) H-bridge, 4) Raspberry Pi 3, 5) PCF8591 A/D-D/A converter, 6) Protoboard with instrumentation circuit, and 7) DC power supply.

Results and discussion

As previously mentioned, the ADRC was simulated, tuned, and implemented using Simulink and a Raspberry Pi 3, resulting in a robust control system with parameters that can be quickly and easily reconfigured.

Simulation results

Fig. 9 displays the output (motor position in radians) and reference signals obtained in the Simulink simulation using the nonlinear DC motor model (Fig. 2) when controlled by the ADRC. The ADRC effectively controlled the system for a given reference. The settling time was approximately 1.5 s. There was no noticeable overshoot, and the steady-state error was near zero. The control signal (Fig. 9) starts at 100% of the PWM duty cycle and decreases to zero as the output reaches the reference. The nonlinear motor model was controlled successfully in the simulation.

Figure 8. Physical implementation of the prototype
Source: Authors

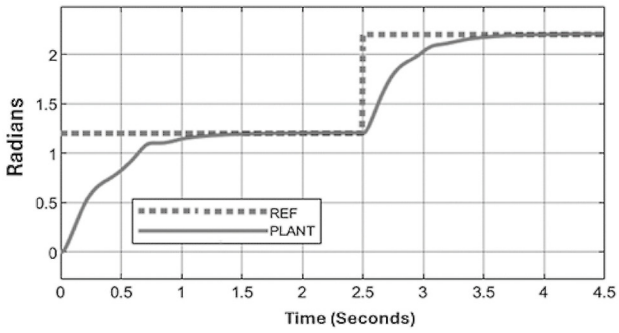

Figure 9. Motor position in the simulation with the ADRC
Source: Authors

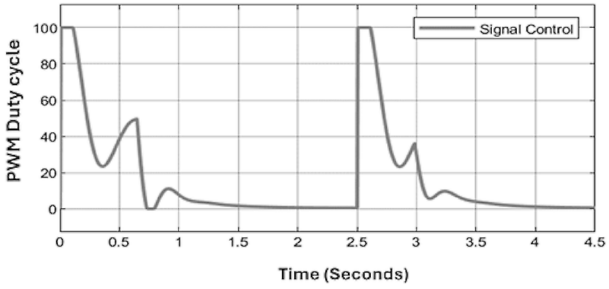

Figure 10. PWM duty cycle of the control signal obtained in the simulation
Source: Authors

To evaluate the controller's simulation performance, a PI controller was implemented, tuned, and simulated in Simulink using the nonlinear DC motor model. The parameters for the controller were P = 10 and I = 0.1 The integral of the absolute error (IAE), the integral of the time-weighted absolute error (ITAE), the integral of the squared error (ISE), and the integral of time-weighted squared error (ITSE) were calculated for both the PI controller and the ADRC. The simulation setup for the PI controller is shown in Fig. 11, and its response is illustrated in Fig. 12

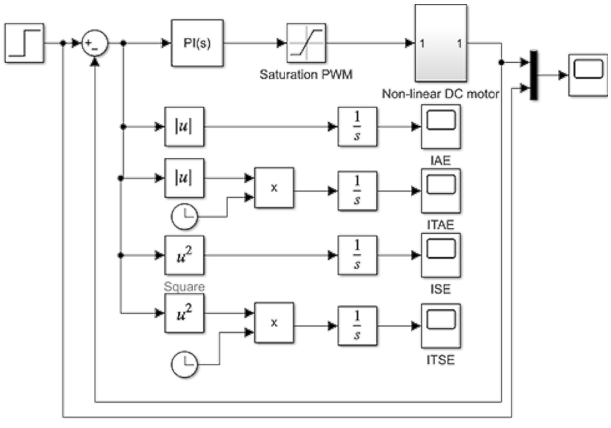

Figure 11. PI controller with its IAE, ITAE, ISE, and ITSE parameters in Simulink
Source: Authors

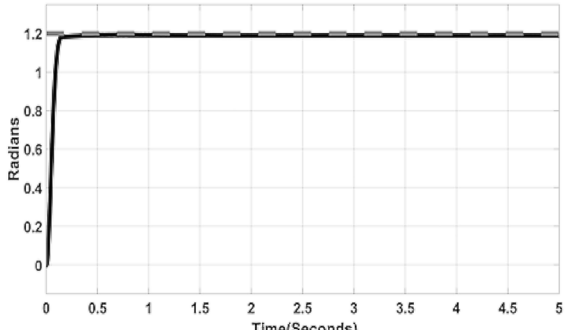

Figure 12. Output signal of the motor position during the simulation of the PI controller
Source: Authors

Fig. 13 compares the IAE values for the ADRC and the PID controller. The latter demonstrates a faster response, but it exhibits a persistent error accumulation due to its steady-state error, indicating higher sensitivity to system nonlinearities. In contrast, the ADRC shows a slower response but stabilizes around 1.5 s with no steady-state error, which highlights its ability to adapt to the model's nonlinear dynamics.

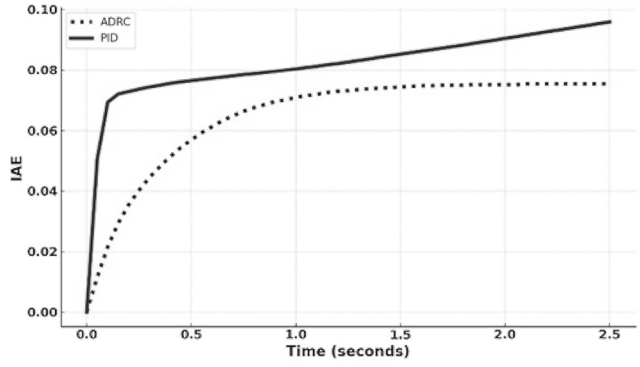

Figure 13. IAE of the control signal obtained in the simulation
Source: Authors

Fig. 14 plots the ISE of both controllers. The PID controller shows a rapid increase in this metric during the first 0.2 s, stabilizing at 0.3 s, with slight increases thereafter. In contrast, the ADRC exhibits a more gradual and continuous increase, stabilizing at 1 s with a lower final ISE than its PID counterpart. This indicates that, while the PID controller responds faster, the ADRC is more efficient at minimizing the accumulated error and provides greater long-term stability, making it more suitable for nonlinear models.

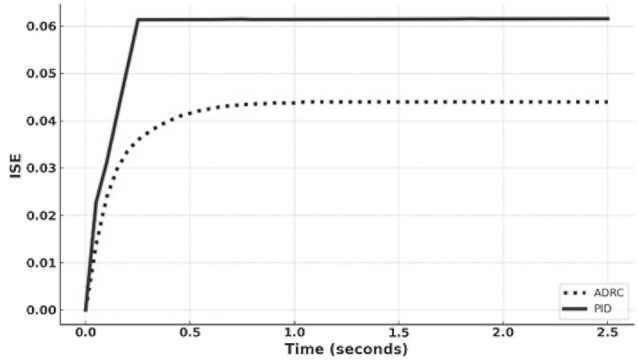

Figure 14. ISE of the control signal obtained in the simulation
Source: Authors

Fig. 15 shows the ITAE for both controllers. The PID controller starts slowly but continues accumulating errors throughout the analyzed interval, suggesting inefficiency in the long term and making it less suitable for systems requiring stability. In contrast, the ADRC shows a faster initial increase but stabilizes at 1 s, significantly reducing the accumulated errors. This behavior makes the ADRC more efficient and better suited for systems that demand precision and robustness.

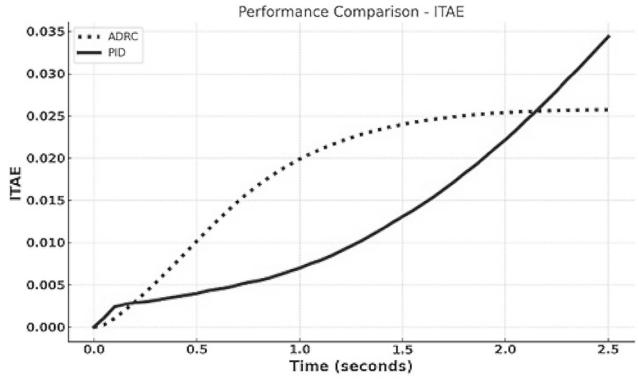

Figure 15. ITAE of the control signal obtained in the simulation
Source: Authors

Fig. 16 presents the ITSE for both controllers. The PID controller exhibits a gradual and uniform increase, reflecting a continuous error accumulation over time. In contrast, the ADRC shows a steeper initial growth but stabilizes quickly, demonstrating greater efficiency in minimizing the accumulated error.

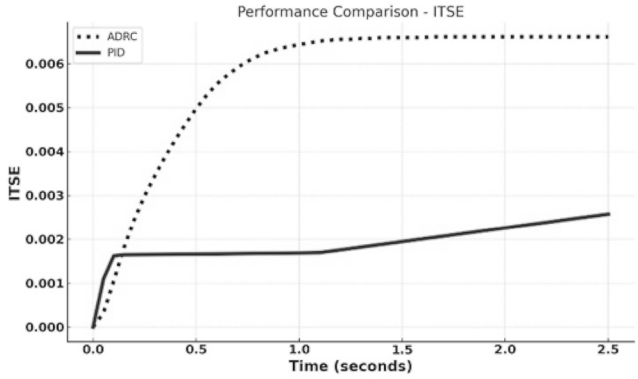

Figure 16. ITSE of the control signal obtained in the simulation
Source: Authors

An analysis of the aforementioned performance indices reveals that the ADRC excels at minimizing the accumulated error and maintaining stability in nonlinear systems, while the PID controller struggles with error accumulation over time, highlighting its limitations in such applications.

Table I presents a comparison in terms of settling time, overshoot, and steady-state error. The PID controller is faster than the ADRC but exhibits persistent steady-state errors. While this can be addressed by adjusting parameters, it leads to an overshoot greater than 10%. On the other hand, the PID controller is even slower and always exhibits overshoot. These findings are consistent with the results of [31], which compares ADRC, PI, and PID controllers on a DC motor.

Table I. Comparison of ADRC and PI controllers

Controller	Settling time (S)	Overshoot (%)	Steady state error (%)
ADRC	1.5	0	0
PI	0.3	0	1

Source: Authors

Implementation results

The results obtained from the Raspberry Pi 3-Simulink implementation are presented in Figs. 17 and 18.

Fig. 17 depicts the position of the motor and the reference signal in radians. The output signal is very noisy due to the brushes. The Faulhaber motor reaches the desired position without oscillation or overshoot, with a settling time of around 2 s. Moreover, the steady-state error is around zero, proving that ADRC can control the position of the motor without precise knowledge of the plant's dynamics – all this, while overcoming the noise. The parameters used in the ADRC were tuned using an approximated nonlinear model, but this is far from a close interpretation of the actual model, especially in terms of friction and noise. Moreover, there are delays related to the Raspberry Pi 3-Simulink connection, as well as others generated within the Raspberry Pi 3 due to

processing power limitations. This becomes evident when the sample time is set to values under 0.01 s.

Fig. 18 presents the control signal applied to the DC motor. It is noisy and quickly becomes zero after reaching the reference.

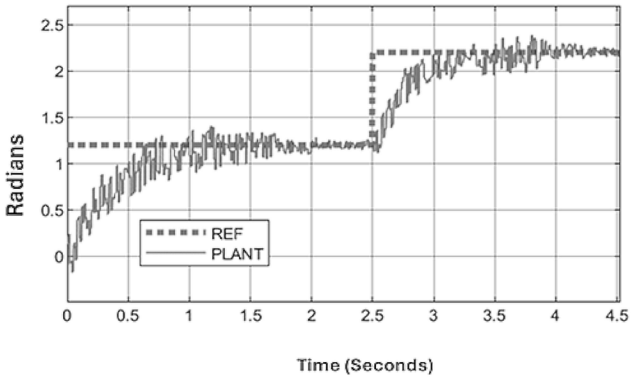

Figure 17. Output signals of the implemented system
Source: Authors

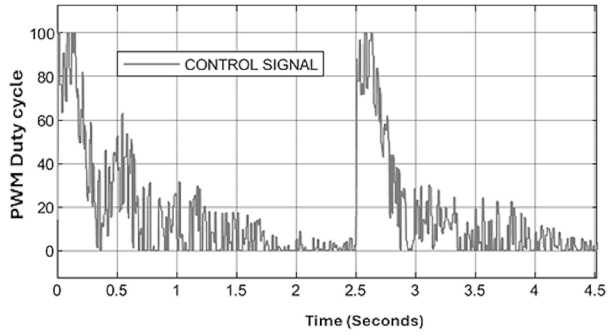

Figure 18. Control signal of the implemented system
Source: Authors

Conclusions

This paper presented the implementation of an active disturbance rejection control for a coreless brushed DC motor. Brushed motors are known for generating noise that can interfere with the control circuits. To effectively manage this motor, the general model of a nonlinear DC motor was used for tuning in simulation. The parameters obtained from the simulation were then applied in practice, demonstrating that ADRC requires minimal information about the plant to achieve effective control. The system actively estimates and compensates for dynamic and unknown disturbances, such as the noise generated by the motor's brushes. Tuning, simulation, and implementation were all carried out using Simulink and a Raspberry Pi 3. The latter was used as an embedded system, facilitating live data exchange with a computer via an Ethernet connection.

In this context, the careful selection of the sample time is crucial, as it must align with the board's processing power to avoid delays that could negatively impact controller performance. The performance indices used (IAE, ITAE, ISE, and ITSE) highlight the advantages of ADRC over a PID controller when dealing with nonlinear systems. While the latter responds faster, it accumulates more error over time, making it less efficient and stable for long-term control. In contrast, ADRC stabilizes more quickly and minimizes the accumulated error, demonstrating greater robustness and precision. This analysis underscores the suitability of ADRC for applications that demand stability and adaptability to nonlinear dynamics. The development presented in this paper enables the implementation of a portable, easy-to-tune embedded control system, where simulation proved to be an effective tuning tool. Future work will focus on exploring additional nonlinear systems using ADRC, linear active disturbance rejection control (LADRC), and other control strategies. Additionally, deep learning-based tuning strategies will be tested, and other alternatives such as fuzzy controllers will be developed for implementation in Simulink's external mode using Raspberry Pi boards.

CRediT author statement

Mario Gonzalez, Oscar Sierra, and Oscar Higuera conceived the idea and conducted the background research. *Oscar Higuera* devised the methodology, and all authors collaborated with simulation, implementation, and data collection. The writing of the manuscript was entrusted to *Mario González* and *Oscar Sierra*, and its review and editing was performed by *Oscar Higuera*.

References

[1] J. Han, "From PID to active disturbance rejection control," *IEEE Trans. Ind. Elect.*, vol. 56, no. 3, pp. 900–906, 2009. https://doi.org/10.1109/TIE.2008.2011621

[2] Zhiqiang Gao, Yi Huang, and Jingqing Han, "An alternative paradigm for control system design," in *Proc. 40th IEEE Conf. Dec. Control* (Cat. No.01CH37228), vol. 5, no. February, pp. 4578–4585, 2001. https://doi.org/10.1109/.2001.980926

[3] T. A. Khaled, O. Akhrif, and I. A. Bonev, "Dynamic path correction of an industrial robot using a distance sensor and an ADRC controller," *IEEE/ASME Trans. Mechatronics*, vol. 26, no. 3, pp. 1646–1656, 2020. https://doi.org/10.1109/TMECH.2020.3026994

[4] C. Liu, G. Luo, X. Duan, Z. Chen, Z. Zhang, and C. Qiu, "Adaptive LADRC-based disturbance rejection method for electromechanical servo system," *IEEE Trans. Ind. Appl.*, vol. 56, no. 1, pp. 876–889, 2019. https://doi.org/10.1109/TIA.2019.2955664

[5] H. Jin, J. Song, W. Lan, and Z. Gao, "On the characteristics of ADRC: A PID interpretation," *Sci. China Info. Sci.*, vol. 63, no. 10, art. 209201, 2020.

[6] L. Zhi, M. Huang, L. Qian, Z. Wang, Q. Wen, and W. Han, "Research on active disturbance rejection control with parameter autotuning for a moving mirror control system based on improved snake optimization," *Electronics*,

vol. 13, no. 9, May 2024. https://doi.org/10.3390/electronics13091650

[7] C. Du, Z. Yin, Y. Zhang, J. Liu, X. Sun, and Y. Zhong, "Research on active disturbance rejection control with parameter autotune mechanism for induction motors based on adaptive particle swarm optimization algorithm with dynamic inertia weight," *IEEE Trans. Power Electron.*, vol. 34, no. 3, pp. 2841–2855, Mar. 2019. https://doi.org/10.1109/TPEL.2018.2841869

[8] L. Xu, S. Zhuo, J. Liu, S. Jin, Y. Huangfu, and F. Gao, "Advancement of active disturbance rejection control and its applications in power electronics," in *IEEE Trans. Ind. Appl.*, vol. 60, no. 1, pp. 1680-1694, Jan.-Feb. 2024. https://doi.org/10.1109/TIA.2023.3312653

[9] H. Lu, X. Zhu, C. Ren, S. Ma, and W. Wang, "Active disturbance rejection sliding mode altitude and attitude control of a quadrotor with uncertainties," in *2016 12th World Cong. Intell. Control Autom.* (WCICA), Guilin, China, 2016, pp. 1366-1371. https://doi.org/10.1109/WCICA.2016.7578812

[10] K. Xu, L. Lang, Q. Wei, and H. Ma, "Design of single leg foot force controller for hydraulic actuated quadruped robot based on ADRC," in *Chinese Control Conf., CCC*, vol. 2015-Sept, no. 61473304, pp. 1228–1233, 2015. https://doi.org/10.1109/ChiCC.2015.7259809

[11] X. Meng, H. Yu, J. Zhang, T. Xu, and H. Wu, "Liquid level control of four-tank system based on active disturbance rejection technology," *Measurement*, vol. 175, art. 109146, 2021. https://doi.org/10.1016/j.measurement.2021.109146

[12] H. Sira-Ramírez, J. Linares-Flores, A. Luviano-Juárez, and J. Cortés-Romero, "Ultramodelos globales y el control por rechazo activo de perturbaciones en sistemas no lineales diferencialmente planos," *Rev. Iberoamer. Autom. Infor. Ind.*, vol. 12, no. 2, pp. 133–144, 2015. https://doi.org/10.1016/j.riai.2015.02.001

[13] Y. Lei, J. Xu, and Q. Hao, "Application of ADRC in stability control of tank gun system," in *Proc. 2018 IEEE 7th Data Driven Control Learn. Syst. Conf., DDCLS 2018*, 2018, pp. 670–675. https://doi.org/10.1109/DDCLS.2018.8515958

[14] G. Pasolini, F. Zabini, A. Bazzi, and S. Olivieri, "A software defined radio platform with Raspberry Pi and Simulink," in *2016 24th Euro. Signal Proc. Conf.* (*EUSIPCO*), *2016*, pp. 398–402. https://doi.org/10.1109/EUSIPCO.2016.7760278

[15] A. Ferdjali, M. Stanković, S. Manojlović, R. Madonski, D. Bujaković, and A. Djenadbia, "Systematic design of nonlinear ADRC for laser seeker system with FPGA-based rapid prototyping validation," *Aircraft Eng. Aerospace Tech.*, vol. 94, no. 7, pp. 1087–1099, 2022. https://doi.org/10.1108/AEAT-06-2021-0188

[16] H. Zhang, Q. Zhang, Y. Zhang, A. Zhao, T. Ni, and K. Yang, "A DSP-based magnetic compensation system for optically pumped magnetometer," *IEEE Sens. J.*, 2023.

[17] H. K. Kondaveeti, D. Bandi, S. E. Mathe, S. Vappangi, and M. Subramanian, "A review of image processing applications based on Raspberry Pi," *2022 8th Int. Conf. Adv. Comp. Comm. Syst.* (*ICACCS*), Coimbatore, India, 2022, pp. 22–28. https://doi.org/10.1109/ICACCS54159.2022.9784958

[18] J. Á. Ariza and C. N. Galvis, "RaspyControl Lab: A fully open-source and real-time remote laboratory for education in automatic control systems using Raspberry Pi and Python," *HardwareX*, vol. 13, art. e00396, 2023. https://doi.org/10.1016/j.ohx.2023.e00396

[19] S. E. Mathe, A. C. Pamarthy, H. K. Kondaveeti, and S. Vappangi, "A review on Raspberry Pi and its robotic applications," 2022 2nd Int. Conf. Art. Intell. Signal Proc. (AISP), Vijayawada, India, 2022, pp. 1–6. https://doi.org/10.1109/AISP53593.2022.9760590

[20] A. James, A. Seth, and S. C. Mukhopadhyay, "Programming Raspberry Pi for IoT system," in *IoT System Design*, S. C. Mukhopadhyay, Ed., Cham, Switzerland: Springer, 2022, vol. 41, pp. 51-79. https://doi.org/10.1007/978-3-030-85863-6_4

[21] D. Papakyriakou and I. Barbounakis, "Benchmarking and review of Raspberry Pi (RPi) 2B vs RPi 3B vs RPi 3B+ vs RPi 4B (8GB)," *Int. J. Comp. Appl.*, vol. 185, 2023. https://doi.org/10.5120/ijca2023922693

[22] E. Ilten and M. Demirtas, "Liquid level control interface design on Simulink external mode with Raspberry Pi," in *Proc. Int. Conf. Mod. Adv. Res.*, Konya, Turkey, Aug. 2023, pp. 85–88.

[23] K. Utari, N. Mulyaningsih, I. Astuti, Y. Bhakti, and Z. Zulherman, "Physics calculator application with MATLAB as a learning media to thermodynamics concept," *Momentum: Phys. Edu. J.*, vol. 5, no. 2, pp. 101–110, 2021. https://doi.org/10.21067/mpej.v5i2.5133

[24] R. P. Borase, D. K. Maghade, S. Y. Sondkar, and others, "A review of PID control, tuning methods and applications," Int. J. Dynamics Control, vol. 9, pp. 818–827, 2021. https://doi.org/10.1007/s40435-020-00665-4

[25] J. A. Niembro-Ceceña, R. A. Gómez-Loenzo, and J. Rodríguez-Reséndiz, "SoftCtrlDC-M: Embedded control software for brushed direct current motors," *SoftwareX*, vol. 25, art. 101643, Feb. 2024. https://doi.org/10.1016/j.softx.2024.101643

[26] S. Liu, M. Xue, Y. Qiu, X. Zhou, and Q. Zhao, "Design of the missile attitude controller based on the active disturbance rejection control," *J. Aerospace Tech. Manage.*, vol. 14, art. 1255, 2022. https://doi.org/10.1590/jatm.v14.1255

[27] M. Megrini, A. Gaga, and Y. Mehdaoui, "Processor in the loop implementation of artificial neural network controller for BLDC motor speed control," *Results Eng.*, vol. 23, art. 102422, Sep. 2024. https://doi.org/10.1016/j.rineng.2024.102422

[28] R. Zeng, J. Zhao, Y. Xiong, and X. Luo, "Active disturbance rejection control of five-phase motor based on parameter setting of genetic algorithm," *Processes*, vol. 11, no. 6, art. 1712, Jun. 2023. https://doi.org/10.3390/pr11061712

[29] M. V. Srikanth and N. Yadaiah, "A magnitude optimum approach for tuning Reduced-order ADRC with FOPDT models," in *2021 7th Indian Control Conf.*, 2021, pp. 46–51. https://doi.org/10.1109/ICC54714.2021.9703133

[30] W. Ai, M. Wu, X. Li, and X. Li, "Active disturbance rejection based iterative learning control for direct torque control of switched reluctance motor drive," in Proc. 2021 IEEE 10th Data Driven Control Learn. Syst. Conf., May 2021, pp. 1379–1384. https://doi.org/10.1109/DDCLS52934.2021.9455497

[31] M. M. Rahman, M. A. Al Mamon, and M. M. Rahaman, "Active disturbance rejection control based speed control of DC motor," in *2022 Int. Conf. Adv. Elect. Electron. Eng.*, Gazipur, Bangladesh, 2022. https://doi.org/10.1109/ICAEEE54957.2022.9836391

[32] S. Shafi, P. S. Hamid, S. A. Nahvi, M. H. Koul, and M. A. Bazaz, "Active disturbance rejection control of angular position of a DC servo motor," *2023 Int. Conf. Power Inst. Energy Control* (*PIECON*), Aligarh, India, 2023, pp. 1–5. https://doi.org/10.1109/PIECON56912.2023.10085795

[33] S. Bose, Y. V. Hote and D. Sengupta, "Analysis and control of real-time DC servo system using linear ADRC," *2018 15th IEEE India Council Int. Conf.* (INDICON), Coimbatore, India, 2018, pp. 1–6. https://doi.org/10.1109/INDICON45594.2018.8987040

[34] E. Ilten, "Active disturbance rejection control of a DC motor with Raspberry Pi on Simulink external mode," *AS-Proceedings*, vol. 1, art. 124, 2023. https://doi.org/10.59287/as-proceedings.124

[35] H. Sira-Ramírez, R. Castro-Linares, and G. Puriel-Gil, "An active disturbance rejection approach to leader-follower controlled formation," *Asian J. Control*, vol. 16, no. 2, pp. 382–395, 2014. https://doi.org/10.1002/asjc.714

[36] C. Zhang and S. He, "Generalized output feedback active disturbance rejection control for uncertain lower-triangular nonlinear systems," *2016 35th Chinese Control Conf.* (*CCC*), Chengdu, China, 2016, pp. 533–538. https://doi.org/10.1109/ChiCC.2016.7553140

[37] D. Wu and K. Chen, "Frequency-domain analysis of nonlinear active disturbance rejection control via the describing function method," *IEEE Trans. Ind. Electron.*, vol. 60, no. 9, pp. 3906–3914, 2013. https://doi.org/10.1109/TIE.2012.2203777

[38] Y. Wang, J. Zhang, and W. Dong, "A new algorithm in blackbody temperature control system based on ADRC," *2017 IEEE Int. Conf. Mechatronics Autom.*, 2017, pp. 226–230. https://doi.org/10.1109/ICMA.2017.8015818

[39] Z. Shen, "Design and simulation of naval gun servo controller based on ADRC," *Proc. 2019 11th Int. Conf. Meas. Tech. Mechatronics Autom.*, pp. 345–349, 2019. https://doi.org/10.1109/ICMTMA.2019.00083

[40] P. Jiang, J. Y. Hao, X. P. Zong, and P. G. Wang, "Modeling and simulation of active-disturbance-rejection controller with Simulink," *2010 Int. Conf. Machine Learn. Cyber.*, 2010, vol. 2, no. July, pp. 927–931. https://doi.org/10.1109/ICMLC.2010.5580604

[41] D. Liu, C. Min, J. Cui, and D. Feng, "Design of attitude controller for hyper velocity projectile based on active disturbance rejection control," in *Int. Conf. Guid. Nav. Control*, 2022, pp. 2709–2719. https://doi.org/10.1007/978-981-19-6613-2_264

[42] R. Isermann, *Fault-diagnosis systems: An introduction from fault detection to fault tolerance*, 1st ed. Berlin, Germany: Springer-Verlag, 2006. https://doi.org/10.1007/3-540-30368-5

[43] X. Ruan, X. Wang, X. Zhu, Z. Chen, and R. Sun, "Active disturbance rejection control of Single wheel robot," in *Proc. World Cong. Intell. Control Autom.* (*WCICA*), 2015, pp. 4105–4110. https://doi.org/10.1109/WCICA.2014.7053403

[44] K. Xu, L. Lang, Q. Wei, and H. Ma, "Design of single leg foot force controller for hydraulic actuated quadruped robot based on ADRC," *Chinese Control Conf., CCC*, 2015, vol. 2015-Sept, no. 61473304, pp. 1228–1233. https://doi.org/10.1109/ChiCC.2015.7259809

[45] S. Liu, M. Xue, Y. Qiu, X. Zhou, and Q. Zhao, "Design of the missile attitude controller based on the active disturbance rejection control," J. Aerospace Tech. Manage., vol. 14, art. e1322, 2022. https://doi.org/10.1590/jatm.v14.1255

[46] X. Ruan, X. Wang, X. Zhu, Z. Chen, and R. Sun, "Active disturbance rejection control of Single wheel robot," in *Proc. World Cong. Intell. Control Autom.* (*WCICA*) (WCICA), 2015, pp. 4105–4110. https://doi.org/10.1109/WCICA.2014.7053403