Optimal Tuning of an Active Disturbance Rejection Controller Using a Particle Swarm Optimization Algorithm

Olga L. Jiménez Morales, Diego Tristán Rodríguez, Rubén Garrido, and Efrén Mezura-Montes

1 Introduction

Active Disturbance Rejection Control (ADRC) allows compensating internal and external disturbances accurately or approximately in a control system. The main idea is to estimate a term containing the disturbances affecting a plant as well as parameter uncertainties and unmodelled terms by means of a disturbance observer (DOB). The estimate produced by the latter is injected in the plant as part of the control signal with the goal of counteracting the real disturbance. In some cases the ADRC can be shown to be equivalent to classic controllers including the Proportional Integral Derivative Controller (PID) widely used in industry.

There exist works showing the equivalence between a PID controller and a DOB-based controller [1–3]. They conclude that the integral action of the PID controller compensates for constant disturbances and has been considered as an implicit disturbance observer. On the other hand, the DOB-based controller counteracts constant and time-varying disturbances, so it is considered as a robust controller based on online disturbance estimation and compensation.

O. L. Jiménez Morales (✉) · D. Tristán Rodríguez · R. Garrido
Department of Automatic Control, CINVESTAV-IPN, Av. Instituto Politecnico Nacional 2508, Mexico city 07360, Mexico
e-mail: ojimenez@ctrl.cinvestav.mx

D. Tristán Rodríguez
e-mail: dtristan@ctrl.cinvestav.mx

R. Garrido
e-mail: garrido@ctrl.cinvestav.mx

E. Mezura-Montes
Artificial Intelligence Research Institute, University of Veracruz, Veracruz 91223, Mexico
e-mail: emezura@uv.mx

O. Castillo and P. Melin (eds.), *Hybrid Intelligent Systems Based on Extensions of Fuzzy Logic, Neural Networks and Metaheuristics*, Studies in Computational Intelligence 1096,
https://doi.org/10.1007/978-3-031-28999-6_11

For controller design and tuning, in particular the PID controller, there exists a plethora of tuning techniques including the pole assignment technique, the Ziegler and Nichols method, and optimal tuning techniques such as the Linear Quadratic Regulator among others [4–7]. Interestingly enough, in [8, 9], the disturbance observer is used as a tuning technique for fuzzy controllers. However, for the implementation of an ADRC algorithm, obtaining optimum performance is not a trivial task. The above is due to the fact that ADRC requires the simultaneous tuning of the controller gains plus those of the disturbance observer.

Intelligent optimization techniques such as simulated annealing, pattern search, genetic algorithm, and particle swarm optimization (PSO) are currently being used in several areas of automatic control and industrial applications to improve system stability, for parameter identification, and for controller tuning [10–17]. Among the them, the PSO algorithm has been successful due to the simplicity of its implementation, its computational efficiency and its ability to search large spaces of potential solutions.

There exists interesting past literature regarding the parameter tuning of feedback controllers employing intelligent optimization techniques. The PID controllers has been tuned using several algorithms including the Modified Butterfly Optimization algorithm [18], and the PSO algorithm [19, 20]. On the other hand, several metaheuristics including the Genetic Algorithm, Simulated Annealing, and Tabu Search has been used for PID controller tuning applied to bioprocess control [21]. An interesting survey on applying intelligent optimization techniques to PID controller tuning is found in [22] whereas [23] gives an account on how to apply multiobjective optimization to controller tuning.

In the case of ADRC there are several works where the tuning is performed through a metaheuristic [24–26]. The PSO algorithm is employed in [24] for tuning an ADRC algorithm applied to a ship. The controller is based on an extended state observer and the fitness function correspond to the integral time absolute error (ITAE) performance index. The Whale Optimization algorithm is employed in [25] for tuning of an ADRC algorithm applied to a quadrotor, and the integral of the absolute error is used as a fitness function. The tuning of extended state observers is performed in [26] by means of a PSO algorithm, and the fitness function corresponds to the integral of the absolute error. It is interesting to point out that in all the above references the stability conditions associated to the closed-loop system are not considered into the implementation of the optimization algorithms. Moreover, the control effort is not taken into account in the fitness function, and only simulation results are reported.

The objective of this work is to show the tuning of an ADRC algorithm by means of Particle Swarm Optimization techniques. The controller is implemented in real-time and applied on a low-cost educational prototype endowed with a Makeblock servo motor [27]. One of the key features of the proposed tuning methodology, which is a depart from previous controller tuning procedures using intelligent optimization techniques, is to use the conditions derived from a stability analysis of the closed-loop system composed of the servo motor and the ADRC algorithm, to define the feasible set of solutions. Thus, the PSO algorithms would produce particles associated with optimal gains that guarantee closed-loop stability while minimizing a fitness function.

This work is divided as follows. Section 2 describes the Active Disturbance Rejection Controller. Section 3 presents the Particle Swarm Optimization techniques used for ADRC tuning. Section 4 gives details of the experimental platform, exposes the experimental results and presents the performance analysis of the controller. The work ends with some concluding remarks.

2 Active Disturbance Rejection Controller

2.1 *Preliminaires*

The main idea of the Disturbances Observer (DOB) developed in [28–30] is to use measurements of the input and output of a disturbed plant under control to estimate the disturbance. Then, the disturbance estimate is used to counteract the effects of the real disturbance. According to the diagram shown in Fig. 1 the plant output is given by:

$$Y(s) = P(s)[U(s) + \hat{D}(s)] \tag{1}$$

where s is a complex variable, $G(s)$ is the transfer function which models the plant [4], $Y(s) = \mathcal{L}\{y\}$ and $U(s) = \mathcal{L}\{u\}$ are the Laplace transform of the output y and the input u, $D(s) = \mathcal{L}\{d\}$ and $\hat{D}(s) = \mathcal{L}\{\hat{d}\}$ are respectively the Laplace transform of the disturbance d and its estimate $\hat{d}$, and $\mathcal{L}$ stands for the Laplace operator.

According to Fig. 1, the DOB is composed of the inverse of the nominal model of the plant $P_m(s)$ and a strictly proper stable filter $F(s)$, and the expression for estimating $\hat{D}(s)$ is

$$\hat{D}(s) = [P_m^{-1}(s)Y(s) - U(s)]F(s) \tag{2}$$

Note that the estimation of the disturbance is realized using only input and output measurements.

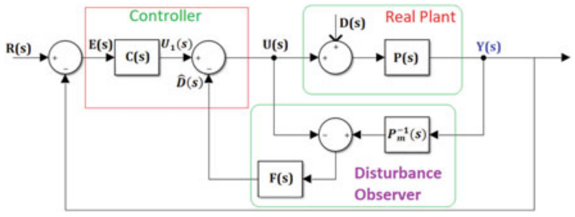


Fig. 1 Diagram of the ADRC algorithm composed of a Disturbance Observer and a controller designed for the nominal model $P_m(s)$ of the plant

2.2 ADRC Applied to a DC Servomotor

Consider the mathematical model of a DC servomotor affected by an unknown disturbance $\bar{d}$ [31]:

$$\ddot{y} = -a\dot{y} + bu + \bar{d} \tag{3}$$

The term $a > 0$ is related to viscous friction and $b >$ is the input gain. It is assumed that a is unknown and the input gain b is known. Moreover, $-a\dot{y}$ and $\bar{d}$ are grouped into a single term $d = \bar{d} - a\dot{y}$, which is assumed to be bounded, i.e. $|d| \leq D$. Then, Eq. (3) simplifies to:

$$\ddot{y} = bu + d \tag{4}$$

The Laplace transform of (4) is given by:

$$s^2Y(s) = bU(s) + D(s) \tag{5}$$

From (5) it follows that the nominal model P_m of the servomechanimsm corresponds to:

$$P_m(s) = \frac{b}{s^2} \tag{6}$$

According to the theory presented previously, and considering [1, 28–30], the filter $F(s)$ is defined as:

$$F(s) = \frac{\beta}{s+\beta} \tag{7}$$

with cutoff frequency $\beta > 0$.

Finally, from (2), (5) and (7), the perturbation estimate is given by:

$$\hat{D}(s) = \frac{\beta s}{s+\beta} sY(s) - \frac{\beta b}{s+\beta} U(s) \tag{8}$$

Note that $sY(s)$ corresponds to the angular velocity of the servomotor. Moreover, for the implementation of the Disturbance Observer, measurements of the angular velocity of the servomotor are required. However, in many practical cases, only angular position measurements are available, so a Luenberguer Observer is designed to provide velocity estimates.

For ease of design we rewrite Eq. (4) in the time domain as:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= bu + d \end{aligned} \tag{9}$$

where $x_1 = y$ and $x_2 = \dot{y}$. and the Disturbance Observer (8) in the time domain is given by:

$$\dot{\hat{d}} = -\beta\hat{d} + \beta(\dot{\hat{x}}_2 - bu) \tag{10}$$

Following the ideas in [30] to avoid the use of $\dot{\hat{x}}_2$, define the following auxiliary variable:

$$\omega = \hat{d} - \beta\hat{x}_2 \tag{11}$$

Thus, performing the time derivative of (11) and substituting (10) produces:

$$\begin{aligned}\dot{\omega} &= -\beta\hat{d} - \beta bu \\ \hat{d} &= \omega + \beta\hat{x}_2\end{aligned} \tag{12}$$

The next control law with state feedback is proposed and allows compensating for the effects of the disturbance d:

$$u = \frac{1}{b}[u_n - \hat{d}] \tag{13}$$

where u_n corresponds to the control signal generated by the controller designed for the nominal plant $P_m(s)$. First, assume that $\hat{d}$ in (13) exactly compensates for the perturbation d in (9), i.e. $\hat{d} - d = 0$. Then, substituting (13) into (9) yields the next undisturbed nominal model of the plant:

$$\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= u_n\end{aligned} \tag{14}$$

Therefore, the term u_n is given by:

$$\begin{aligned}u_n &= \ddot{r} + \alpha_1\dot{e} + \alpha_2 e \\ &= \ddot{r} + \alpha_1(\dot{r} - x_2) + \alpha_2(r - x_1)\end{aligned} \tag{15}$$

where r, $\dot{r}$ and $\ddot{r}$ are the reference and its derivatives, $\dot{e} = \dot{r} - x_2$, and α_1, α_2 are positive constants. Substituting (15) into (14) yields the dynamics of the closed-loop undisturbed system:

$$\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= \ddot{r} + \alpha_1\dot{e} + \alpha_2 e \\ &= \ddot{r} + \alpha_1(\dot{r} - x_2) + \alpha_2(r - x_1)\end{aligned} \tag{16}$$

which has the next Hurwitz stable polynomial:

$$P_c(s) = s^2 + \alpha_1 s + \alpha_2 \tag{17}$$

Then, for the implementation of the Disturbance Observer and the ADRC, the following control law is set as:

$$u = \frac{1}{b}[\ddot{r} + \alpha_1(\dot{r} - \hat{x}_2) + \alpha_2(r - \hat{x}_1) - \hat{d}] \tag{18}$$

The estimates $\hat{x}_1$ and $\hat{x}_2$ are produced by the following Luenberger Observer (LO) built for the nominal system (14):

$$\begin{aligned} \dot{\hat{x}}_1 &= \hat{x}_2 + \gamma_1 \epsilon \\ \dot{\hat{x}}_2 &= u_n + \gamma_2 \epsilon \\ \hat{y} &= \hat{x}_1 \\ \epsilon &= y - \hat{y} \end{aligned} \tag{19}$$

where ϵ is the observation error. The gains $\gamma_1 > 0$ and $\gamma_2 > 0$ are the coefficients of the next polynomial associated to the LO, which is Hurwitz stable:

$$P_{do}(s) = s^2 + \gamma_1 s + \gamma_2 \tag{20}$$

Finally, from the stability analysis [30], the following condition must be fulfilled to guarantee closed-loop stability:

$$0 < \beta < \gamma_1 \tag{21}$$

3 Particle Swarm Optimization

The Particle Swarm Optimization (PSO) algorithm is one of the most widely used swarm intelligence methods since it has been able to resolve a variety of complex optimization problems in several different areas [32].

Proposed in 1995 by Kennedy and Eberhart [33], this algorithm was discovered through a simulation of a simplified social model inspired by the behavior of bird flocks. The PSO is composed of a swarm of particles evolving in a set of feasible solutions where it desired to find a minimum or a maximum of a fitness function. Each one of the particles is a potential solution to the problem. The swarm movements in the search space are called flights, and they are produced from the best individual particle positions, from the best global positions corresponding to all the particles in the swarm, and from past position values. The updating of the best positions, individual and global, is done by the evaluation of a fitness function.

This work uses three variations of the PSO algorithm in order to compare the performance of each one.

3.1 PSO with Inertia Weight

The PSO with inertia weight or ω-PSO algorithm [34] is composed of N particles of dimension L. The particles are represented as $z_n(k) = [z_{n,1}, \dots, z_{n,L}]^T$ where $n \in [1, \dots, N]$. For each particle, there exists a velocity vector $v_n(k) = [v_{n,1}, \dots, v_{n,L}]^T$. The algorithm evolves through the following discrete-time dynamic system:

$$\begin{aligned} v_n(k+1) = {} & \omega v_n(k) \\ & + c_1 \text{rand}()(\text{pBest}(k,n) - z_n(k)) \\ & + c_2 \text{rand}()(\text{gBest}(k) - z_n(k)) \end{aligned} \tag{22}$$

$$z_n(k+1) = z_n(k) + v_n(k+1) \tag{23}$$

where the initial conditions are $z_{n,l}(0) = \text{rand}()$ and $v_{n,l}(0) = 0$ such that $l \in [1, \dots, L]$ and rand() is the uniformly distributed random number function. The parameter $\omega \in [0, 1)$ is called the inertia weight and $c_1 \in [0, 1)$, $c_2 \in [0, 1)$ are called the learning factors. The terms $\text{pBest}(k,n) \in \mathbb{R}^L$ and $\text{gBest}(k) \in \mathbb{R}^L$ are defined as follows:

$$\text{pBest}(k,n) = \arg \min_{0 \le s \le k} J(z_n(s)) \tag{24}$$

$$\text{gBest}(k) = \arg \min_{0 \le s \le k, 1 \le j \le n} J(z_j(s)) \tag{25}$$

such that $J(\cdot)$ is a fitness function. Algorithm 1 shows the pseudo code of the ω-PSO algorithm.

Algorithm 1 ω-PSO algorithm

```
1: Create a random initial swarm $z_n(k)$ $\forall n$
2: Evaluate the fitness function $J(z_n(k))$ $\forall n$
3: Calculate pBest$(k, n)$ $\forall n$
4: Calculate gBest$(k)$
5: while stop condition == false do
6:     for $n = 1$ to $N$ do
7:         Compute the velocity vector $v_n(k+1)$
8:         Perform the flight $z_n(k+1)$
9:         Evaluate the fitness function $J(z_n(k))$
10:        Calculate pBest$(k, n)$
11:    end for
12:    Calculate gBest$(k)$
13: end while
```

3.2 Fractional PSO

The Fractional PSO (FPSO) was developed from the fractional calculus [35] which uses fractional derivatives to control the convergence rate of the PSO. This work uses the Fractional Velocity PSO, where the fractional derivative is applied to the velocity in the classical PSO considering the first four terms of the fractional differential derivative [36]. This algorithm employs N particles of dimension L, which are represented as $z_n(k) = [z_{n,1}, \dots, z_{n,L}]^T$ where $n \in [1, \dots, N]$. For each particle, there exists a velocity vector $v_n(k) = [v_{n,1}, \dots, v_{n,L}]^T$. The discrete dynamic system that represents the FPSO algorithm is as follows:

$$\begin{aligned} v_n(k+1) =& \alpha v_n(k) + \frac{1}{2}\alpha v_n(k-1) + \frac{1}{6}\alpha(1-\alpha)v_n(k-2) \\ &+ \frac{1}{24}\alpha(1-\alpha)(2-\alpha)v_n(k-3) \\ &+ c_1\text{rand}()(\text{pBest}(k,i) - z_n(k)) + c_2\text{rand}()(\text{gBest}(k) - z_n(k)) \end{aligned} \tag{26}$$

$$z_n(k+1) = z_n(k) + v_n(k+1) \tag{27}$$

where the initial conditions are $z_{n,l}(0) = \text{rand}()$ and $v_{n,l}(0) = 0$ such that $l \in [1, \dots, L]$ and rand() is the uniformly distributed random numbers function, pBest() and gBest() are defined by (24) and (25). The terms c_1 and c_2 are computed through the next formulae:

$$c_j = (c_{ji} - c_{jf})\frac{k_{max} - k}{k_{max}} + c_{jf}, \quad j = 1, 2 \tag{28}$$

such that c_{ji} and c_{jf} are the initial and final value of the learning factors respectively. The variable α is linearly and adaptively regulated as follows:

$$\alpha = 0.9 - \frac{k}{(1 + e^{-E_f(k)})k_{max}} \tag{29}$$

where

$$E_f(k) = \frac{di_{gb}(k) - di_{\min}(k)}{di_{\max}(k) - di_{\min}(k)} \tag{30}$$

$$di(z_n(k)) = \frac{1}{N-1}\sum_{j=1, j\neq n}^{N} \|z_n(k) - z_j(k)\| \tag{31}$$

and

$$di_{gb}(k) = di(\text{gBest}(k)) \tag{32}$$

$$di_{min}(k) = \min_{1\le s\le n} di(z_s(k)) \tag{33}$$

$$di_{max}(k) = \max_{1\le s\le n} di(z_s(k)) \tag{34}$$

3.3 PSO-AWDV

The PSO with an adaptive weighted delay velocity (PSO-AWDV) is proposed by Xiu [37] in order to deal with problems in the optimization as premature convergence and local stagnation. This algorithm is composed of N particles of dimension L. The particles are represented as $z_n(k) = [z_{n,1}, \ldots, z_{n,L}]^T$ where $n \in [1, \ldots, N]$. For each particle, exists a velocity vector $v_n(k) = [v_{n,1}, \ldots, v_{n,L}]^T$. Its discrete-time dynamics are as follows:

$$\begin{aligned} v_n(k+1) =& \omega v_n(k) + (1-\omega)v_n(k-1) \\ &+ c_1\text{rand}()(\text{pBest}(k,i) - z_n(k)) \\ &+ c_2\text{rand}()(\text{gBest}(k) - z_n(k)) \end{aligned} \tag{35}$$

$$z_n(k+1) = z_n(k) + v_n(k+1) \tag{36}$$

where the initial conditions are $z_{n,l}(0) = \text{rand}()$ and $v_{n,l}(0) = 0$ such that $l \in [1, \ldots, L]$ and rand() is the uniformly distributed random numbers function, pBest() and gBest() are defined by (24) and (25), and c_1 and c_2 decreases linearly as in the FPSO algorithm and ω changes according to the following equation:

$$\omega = 1 - \frac{a}{1 + e^{bE(k)}} \tag{37}$$

$$E(k) = \frac{\max\limits_{1\le s\le N} J(z_s(k)) - \min\limits_{1\le s\le N} J(z_s(k))}{\max\limits_{1\le s\le N} J(z_s(k))} \tag{38}$$

3.4 Fitness Function

Each particle in the PSO algorithms is defined using the terms β in (12) and α_1, α_2 in (18) to minimize a fitness function. To this end, define each particle in the PSO algorithms as $z_n = [\alpha_1, \alpha_2, \beta]^T$. The fitness function is:

$$J(z_n) = \int_0^T \left(w_1|e_t| + w_2|e_v| + w_3|u| + w_4\left|\frac{du}{dt}\right| \right) dt \tag{39}$$

where $e_t = r - x_1$ and $e_v = \dot{r} - \hat{x}_2$ are the tracking error and the velocity error from the dynamic system (9) in closed-loop with (18) respectively, u is the control signal and the last term is its time derivative. Note that each element of the fitness function has a weight w_i, where $i = 1, \ldots, 4$, that gives priority to one of the terms in the integral. Unlike previous works, the control signals and its time derivative are considered in the fitness functions. The control signal is used to avoid excessive energy consumption, and the derivative of the control signal helps reducing the impact of measurement noise on the closed-loop system. If this term is not taken into account, then the DOB term β may take large values that produces high levels of chatter in the control signal thus wreaking havoc the performance of the closed-loop system.

3.5 Set of Feasible Solutions

Since the PSO algorithms does not explicitly take into account the limits in the particles to ensure that the dynamic system is stable in closed-loop, it is important to define a set of feasible solutions. This set is called $\Omega \in \mathbb{R}^D$. In this paper, the Projection Boundary method [38] is implemented in the PSO algorithms to limit the particle positions to the set Ω. In this technique, if an element of the solution $z_n(k+1)$ falls out of Ω, the Projection Boundary method projects it into the boundary of Ω:

$$z_{n,d}(k+1) = \begin{cases} z_{n,d}(k+1) & \text{if } \min(\Omega_d) < z_{n,d}(k+1) < \max(\Omega_d) \\ \min(\Omega_d) & \text{if } \min(\Omega_d) > z_{n,d}(k+1) \\ \max(\Omega_d) & \text{if } \max(\Omega_d) < z_{n,d}(k+1) \end{cases} \tag{40}$$

where $d = 1, \ldots, D$.

4 Experiments

4.1 Experimental Platform

The experimental platform shown in Fig. 2 used for the evaluation of the algorithm is composed of a personal computer equipped with Matlab/Simulink real-time programming software and the QUARC real-time environment from Quanser consulting. A Quanser Q2 card provides data acquisition. The control signal produced by the acquisition board feeds a low-cost educational prototype composed of a linear power amplifier and a Makeblock DC servomotor. A galvanic isolator is used to protect the computer and the data acquisition board from the power amplifier.

4.1.1 Technical Characteristics of the Makeblock Servomotor

The servomotor presented in Fig. 2 is sold by the Makeblock company, and in the sequel it will be called the Makeblock servomotor. It has a 360 ppr optical encoder that allows the measurement of the angular position of the motor, and a steel output shaft that allows direct coupling of inertial loads, gears, pinions and synchronous belts, among others. Table 1 depicts its technical specifications, availability in the national market and cost in USD.

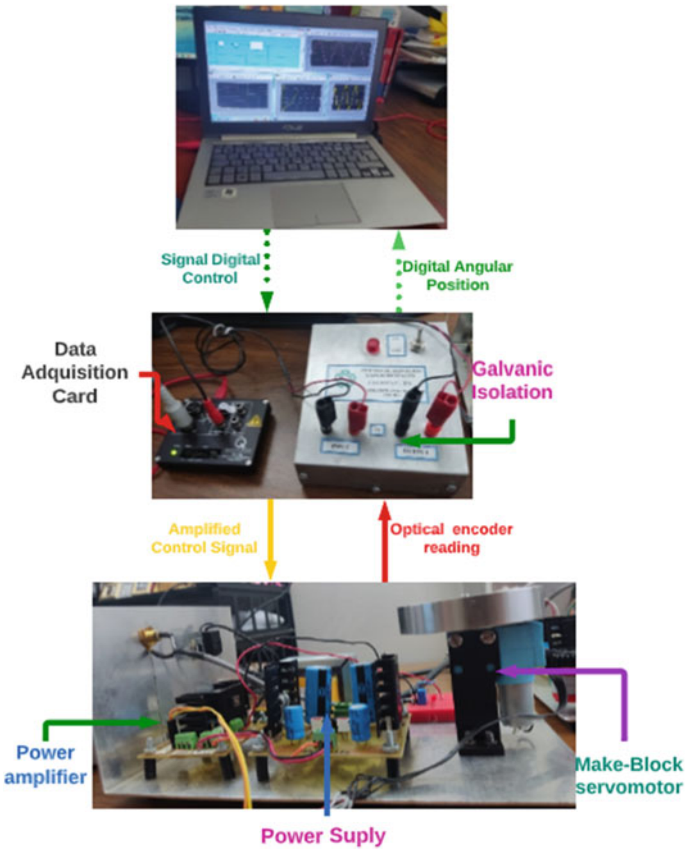


Fig. 2 Experimental platform

Table 1 Technical specifications of the *MakeBlock* servomotor [39]

Technical characteristic	
Rated voltage	7.4 V
No-load current	240 mA
Rotation speed	178 ± 10 RPM
Start torque	5 kg·cm
Feedback	Optical encoder
Rotation	Unlimited
Encoder accuracy	360 ppr
Weight	0.0615 kg kg
Availability	Highly available
Cost	30 USD

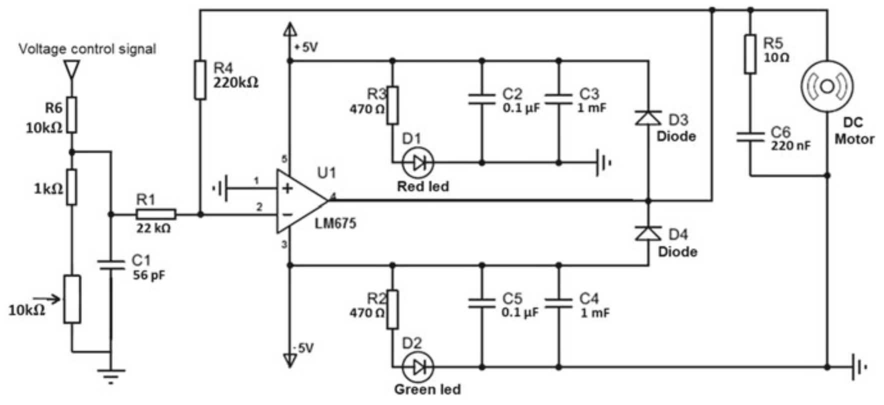


Fig. 3 Power amplifier circuit

4.1.2 Technical Characteristics of the Linear Power Amplifier

A National Semiconductor linear power amplifier model LM675, set with a gain of 10, is used for driving the Makeblock servomotor. The diagram of the power amplifier and its components is shown in Fig. 3.

4.2 Optimization Procedure

To perform the optimization by the PSO algorithms to find the parameters α_1, α_2 and β, the following steps are performed:

- Dynamic simulation conditions. In order to perform the dynamic simulation of the DC servomotor employed in the optimization procedure, consider the model (9) under the following conditions. A quantizer with a quantization interval of 1440 is added to the simulation to take into account the output of an optical incremental encoder used to measure the servomotor angular position. The values $a = 19.2519$ and $b = 12.2809$, obtained through a Least Squares algorithm [40], are considered in the simulation. The gains of the Luenberger Observer (19) are set to $\gamma_1 = 160$ and $\gamma_2 = 6400$. The disturbance d is simulated according to the next model:

$$\bar{d} = 0.05 \sin 2t + 0.1 \sin 0.2t + 0.1 \sin 0.5t + 0.1 \tag{41}$$

  This model represents a generic disturbance and it does not necessarily correspond to disturbances found in real servomotors. Nevertheless, it helps tuning de ADRC algorithm.
- Optimization problem definition. Define the feasible set of solutions employing the restrictions on $\alpha_1 > 0$, $\alpha_2 > 0$ associated to the characteristic polynomial (17),

Table 2 Values of the weights in the fitness function

	Test 1	Test 2	Test 3	Test 4
Value of w_1	100	100	100	100
Value of w_2	10	50	100	10
Value of w_3	0.1	0.1	0.1	0.1
Value of w_4	0.1	0.1	0.1	0

and β, which is a positive constant according to (7), and whose restriction imposed by the stability analysis corresponds to the inequality (21):

$$\Omega = [\alpha_1, \alpha_2, \beta | \alpha_1 \in (0, \infty), \alpha_2 \in (0, \infty), \beta \in (0, \gamma_1)] \tag{42}$$

Therefore, from (39) and (42) the optimization problem is defined as follows:

$$\begin{aligned} &\min J \\ &\text{subject to } \Omega \end{aligned} \tag{43}$$

A dynamic simulation of the servomotor model in closed-loop with the ADRC algorithm is performed with every particle generated by the PSO algorithms. The initial conditions of the DC servomotor model and of the Luenberger Observer are $x(0) = [1, 1]^T$, $\hat{x}(0) = [1, 1]^T$ and the reference and its time derivative in control law (18) are set to $r = 0$ and $\dot{r} = 0$ respectively. Each dynamic simulation lasts 6s. During the simulation the signals $e_t = r - x_1$, $e_v = \dot{r} - \hat{x}_2$, and u are stored in a file. Subsequently, the time derivative $\frac{du}{dt}$ is computed. Then, the above signals allow computing the fitness function (39) and the pBest and gBest terms used in the PSO algorithms.

For the implementation of the PSO algorithms, four tests are performed. Every test uses a different set of weights in the fitness function (39). The values are depicted in Table 2.

The parameters of the PSO algorithms are set using the IRACE package [41]. In the case of the ω-PSO algorithm the parameters are set as $\omega = 0.7$,$c_1 = 0.7$, $c_2 = 0.9$, $N = 23$ and a maximum number of iterations $MI = 180$. In the case of the FPSO algorithm the parameters are $c_{1i} = 0.9$, $c_{1f} = 0.7$, $c_{2i} = 0.9$, $c_{2f} = 0.8$, $N = 18$ and a maximum number of iterations $MI = 180$. The parameters of the PSO-AWDV algorithm are set as $c_{1i} = 0.9$, $c_{1f} = 0.5$, $c_{2i} = 0.9$, $c_{2f} = 0.8$, $N = 16$ and $MI = 180$. Each run of the algorithm stops after (N×MI) evaluations. An statistical test is performed with 30 runs of each algorithm to ensure that it gives congruent results. Using parallel processing with 6 cores of a PC computer, which runs at 4.5 GHz, the 30 runs of every algorithm are executed in 26 min.

For the sake of space, only the time evolution of the particles for the three PSO algorithms corresponding to the Test 1 is reported.

Table 3 Results of Test 1

	ω-PSO	FPSO	PSO-AWDV
Dimension 1 [α_1]	32.62	32.50	32.2454
Dimension 2 [α_2]	307.42	305.69	301.6532
Dimension 3 [β]	71.89	72.0607	72.1876
Minimum evaluation of J	**29.9782**	29.9823	29.9797
Median evaluation of J	29.9888	30.0119	29.9837
Average evaluation of J	29.9985	30.0090	29.9893
Standard deviation evaluation of J	$2.28e^{-2}$	$2.20e^{-2}$	$1.47e^{-2}$

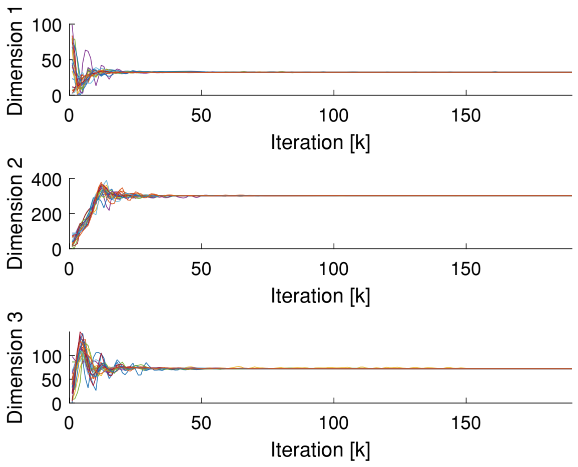
4.2.1 Test 1

Figures 4, 5 and 6 show the time evolution of the particles in each dimension with respect to the iterations in the three different PSO algorithms. In this case the particles in the PSO-AWDV exhibits more oscillations. Table 3 depicts the results for every PSO algorithm where the lowest value of the fitness function is obtained through the ω-PSO algorithm.

4.2.2 Test 2

In the second test, the weight w_2 is increased. This change clearly affects the first dimension of the particles as it is shown in Table 4. Note that the gain α_1 increases compared with the value obtained in the Test 1. Moreover, the PSO-AWDV algorithm gives the lowest value of the fitness function.

Fig. 4 Evolution of the particles in the ω-PSO algorithm: Test 1



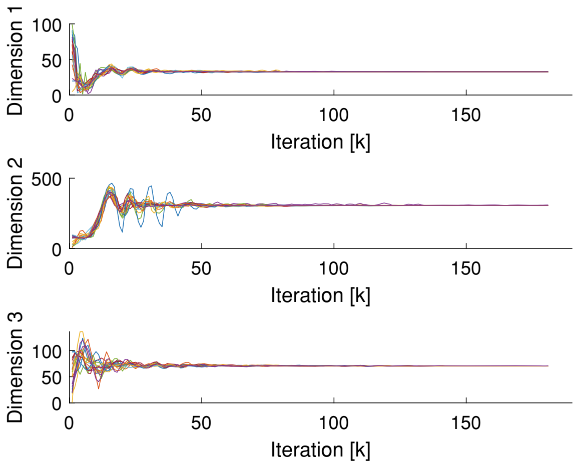


Fig. 5 Evolution of the particles in the FPSO algorithm: Test 1

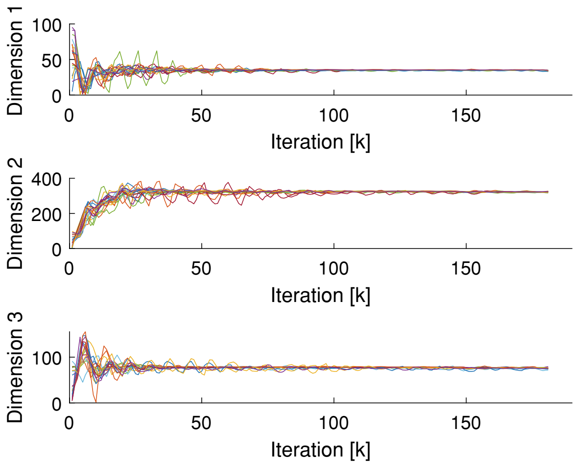


Fig. 6 Evolution of the particles in the PSO-AWDV algorithm: Test 1

4.2.3 Test 3

In the Test 3, the term w_2 is further increased and the corresponding gains are shown in Table 5. In this case, the PSO-AWDV algorithm provided the lowest value of the fitness function.

4.2.4 Test 4

The goal of this test is to show the behavior of the PSO algorithm when the term w_4 of the fitness function (39) is equal to zero. The above means that the fitness function does not take into account the chatter in the control signal. The test is done with

Table 4 Results in Test 2

	ω-PSO	FPSO	PSO-AWDV
Dimension 1 [α_1]	37.26	37.69	36.95
Dimension 2 [α_2]	326.37	327.38	322.14
Dimension 3 [β]	78.52	79.63	79.18
Minimum evaluation of J	69.9758	70.0083	**69.9745**
Median evaluation of J	29.9888	70.0154	69.98045
Average evaluation of J	29.9985	70.0177	69.9860
Standard deviation evaluation of J	$5.42e^{-2}$	$9.52e^{-3}$	$1.44e^{-2}$

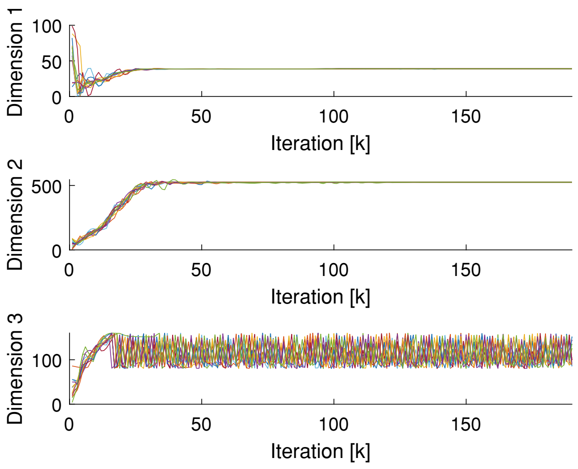


Fig. 7 Evolution of the particles with $w_4 = 0$ proved with ω-PSO algorithm

the ω-PSO algorithm, nevertheless, similar outcomes are obtained with the other PSO algorithms. Figure 7 shows that the third particle never reaches a constant value and holds sustained oscillations. This behaviour shows how important is to take into account the time derivative of the control signal in the fitness function.

Table 5 Results in Test 3

	ω-PSO	FPSO	PSO-AWDV
Dimension 1 [α_1]	38.93	39.98	39.29
Dimension 2 [α_2]	342.87	358.41	346.86
Dimension 3 [β]	92.14	88.12	90.74
Minimum evaluation of J	119.2806	119.3604	**119.2599**
Median evaluation of J	119.3941	119.3978	119.3815
Average evaluation of J	119.4374	119.3963	119.3671
Standard deviation evaluation of J	$1.66e^{-1}$	$1.20e^{-3}$	$4.51e^{-2}$

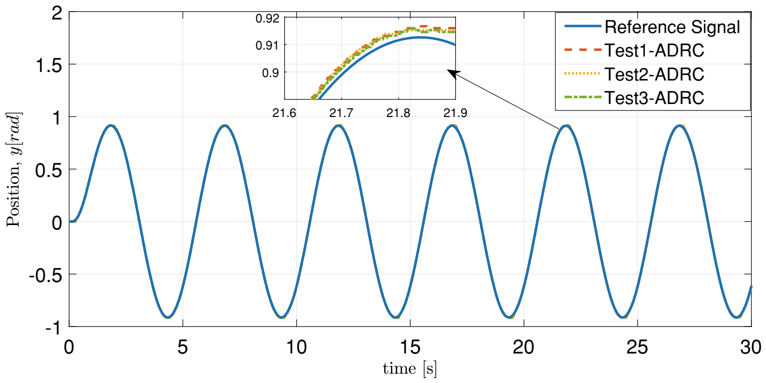


Fig. 8 Reference signal r verses servomotor output signal y using the ADRC corresponding to each of the tests

4.3 *Experimental Results*

The goal of this section is to show the performance of the ADRC algorithm using the gains obtained through the PSO algorithms. The ADRC algorithm is coded in the MATLAB/SIMULINK programming platform under the QUARC real-time environment from Quanser Consulting with a sampling time of 1 ms and the Euler01 integration method. The Makeblock servomotor used in the experiments drives an inertia disk. A sinusoidal function is used as a reference signal $r = 0.8 \sin 0.2t$, filtered by a first order low-pass filter with a cut-off frequency of 10 rad/s.

Three experiments were performed, in which the controller gains α_1 and α_2 of the control law (18) and the cutoff frequency β of the DOB are tuned using the parameters shown in Table 6. These gains correspond to the PSO algorithms producing the lowest value of the fitness function.

Figure 8 shows the trajectory tracking position results of the DC servomotor. The three set of gains produce similar results. Figure 9 shows the corresponding control signals of the ADRC algorithm. Note that the control signal generated by the gains corresponding to Test 1 produces larger peaks compared with the other signals. The tracking error signals are depicted in Fig. 10. It is worth noting that the tuning produced by the PSO-AWDV algorithm produces the lowest tracking error.

Table 6 Gains used in the implementation of the ADRC algorithm

Gains	Test 1: ω-PSO	Test 2: PSO-AWDV	Test 3: PSO-AWDV
α_1	32.62	36.95	39.29
α_2	307.42	322.14	346.86
β	71.89	79.18	90.74

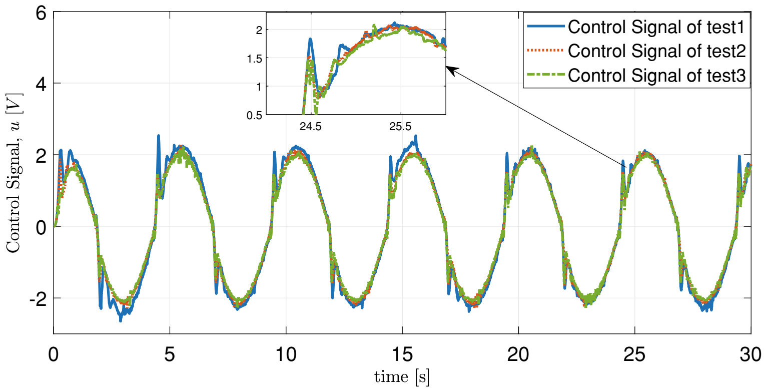


Fig. 9 ADCR control signal u tuning with the ω-PSO and PSO-AWDV algorithms

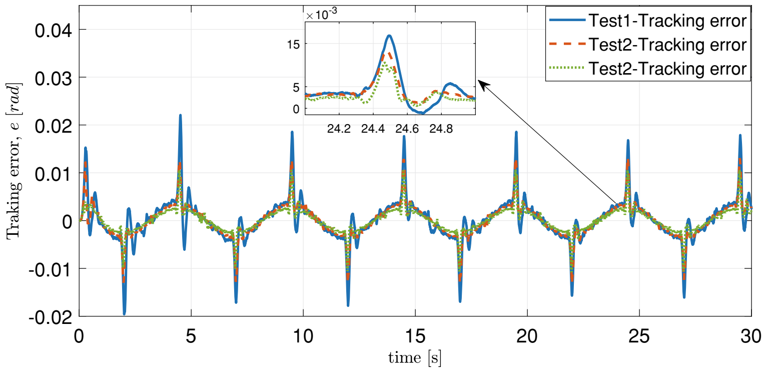


Fig. 10 The tracking error e and corresponding to each one of the tests

Three indices are used to measure the performance of the ADRC, with respect to the error signal, the control signal and its time derivative. These indices are represented mathematically as follows:

$$ISE = \int_{T_1}^{T_2} k[e(t)]^2 dt \tag{44}$$

$$IAC = \int_{T_1}^{T_2} |u(t)| dt \tag{45}$$

$$IACV = \int_{T_1}^{T_2} |\frac{du(t)}{dt}| dt \tag{46}$$

Table 7 ADRC Performance

Indices	Test 1 ω-PSO	Test 2 PSO-AWDV	Test 3 PSO-AWDV
IEC	0.7715	0.5721	**0.3185**
IAC	7.108	6.986	**6.774**
IACV	**38.07**	39.94	47.71

where k represents a scaling factor and (T_1, T_2) defines the time interval during which the performance indices are calculated. The indices are evaluated using $k = 100$ and the interval $T_2 - T_1 = 5$s with $T_1 = 20$s and $T_2 = 25$s.

Table 7 shows the results obtained in the experiments. The smallest Integral Squared Error (ISE) and Integral of the Absolute Value Control (IAC) indices are produced by the gains tuned using the Test 3.

The above is due to the fact that the cutoff frequency $\beta = 90.74$ is the largest of the three values. High values of this term promotes lower tracking errors. On the other hand, the index (IACV), which evaluates the control signal chatter increases for increasing values of β. Note also that a good performance trade-off between signal chatter and tracking error is obtained with the gains corresponding to the Test 2 since the chatter level measured by the IACV index is close to the one obtained with the gains corresponding in the Test 1 and the tracking error quality, measured using the ISE index takes values ISE $= 0.5721$, which is close to the mean of the ISE values obtained using the gains produced by the Tests 1 and 3, i.e. 0.545.

5 Conclusion

The results reported in this work show that the tuning produced by the Particle Swarm Optimization (PSO) algorithm are able to generate optimal gains in an Active Disturbance Rejection Control algorithm. These gains produce good performance in an low-cost educational prototype composed by an low-cost servomotor and a simple linear amplifier. Moreover, the proposed fitness function as well as the use of a realistic servomotor model, which includes the effects of an optical encoder and a disturbances, enhance the optimization procedure. A salient feature of the proposed optimization procedure is that it takes into account the stability conditions of the ADRC algorithm to define the set of feasible solutions. It is also worth remarking that the three PSO algorithms produce similar results, nevertheless, the PSO-AWDV algorithm generated the lowest value of the fitness function in two of three tests.

Acknowledgements The authors would like to thank Gerardo Castro and Jesús Meza for setting up the educational prototype and for installing the MatLab and QUARC software in the computer used in the experiments. The first and second authors thank CONACyT Mexico for its support through the PhD scholarships CVU 860371 and CVU 996816 respectively.

References

1. Luna, L., & Garrido, R. (2018). On the equivalence between P+ DOB and set point weighted PI controllers for velocity control of servodrives under load disturbances. In *2018 XX Congreso Mexicano de Robótica (COMRob)* (pp. 1–6). IEEE.
2. Yamada, K., Komada, S., Ishida, M., & Hori, T. (1997). Analysis and classical control design of servo system using high order disturbance observer. In *Proceedings of the IECON'97 23rd International Conference on Industrial Electronics, Control, and Instrumentation (Cat. No. 97CH36066)* (vol. 1, pp. 4–9). IEEE.
3. Garrido, R., & Luna, J. L. (2018). On the equivalence between PD+ DOB and PID controllers applied to servo drives. *IFAC-PapersOnLine, 51*(4), 95–100.
4. Ogata, K. (2002). *Modern Control Engineering*. Upper Saddle River: Prentice Hall.
5. Meshram, P., & Kanojiya, R. G. (2012). Tuning of pid controller using ziegler-nichols method for speed control of dc motor. In *IEEE-International Conference on Advances in Engineering, Science and Management (ICAESM-2012)* (pp. 117–122). IEEE.
6. Patel, V. V. (2020). Ziegler-nichols tuning method. *Resonance, 25*(10), 1385–1397.
7. Poznyak, A. S. (2009). *Advanced Mathematical Tools for Automatic Control Engineers: Stochastic Techniques*. Oxford: Elsevier.
8. Kim, B. K., Chung, W. K., & Ohba, K. (2009). Design and performance tuning of sliding-mode controller for high-speed and high-accuracy positioning systems in disturbance observer framework. *IEEE Transactions on Industrial Electronics, 56*(10), 3798–3809.
9. Lee, Y. S., Kim, D. S., & Kim, S.-K. (2018). Disturbance observer-based proportional-type position tracking controller for dc motor. *International Journal of Control, Automation and Systems, 16*(5), 2169–2176.
10. Singh, M., Patel, R. N., & Jhapte, R. (2016). Performance comparison of optimized controller tuning techniques for voltage stability. In *2016 IEEE First International Conference on Control, Measurement and Instrumentation (CMI)* (pp. 11–15). https://doi.org/10.1109/CMI.2016.7413701
11. Amine, K. (2019). Multiobjective simulated annealing: Principles and algorithm variants. *Advances in Operations Research 2019*.
12. Mishra, S., Prusty, R. C., & Panda, S. (2020). Design and analysis of 2dof-pid controller for frequency regulation of multi-microgrid using hybrid dragonfly and pattern search algorithm. *Journal of Control, Automation and Electrical Systems, 31*(3), 813–827.
13. Beigi, A. M., & Maroosi, A. (2018). Parameter identification for solar cells and module using a hybrid firefly and pattern search algorithms. *Solar Energy, 171*, 435–446.
14. Ibrahim, M. A., Mahmood, A. K., & Sultan, N. S. (2019). Optimal pid controller of a brushless dc motor using genetic algorithm. *International Journal of Power Electronics and Drive Systems ISSN, 2088*(8694), 8694.
15. Ramesh, H., & Xavier, S. (2022). Optimal tuning of servo motor based linear motion system using optimization algorithm. *Journal of Electrical Engineering & Technology,* 1–16.
16. Nyong-Bassey, B., & Epemu, A. (2022). Systems identification of servomechanism parameters using jellyfish, particle swarm and constraint optimization. *Nigerian Journal of Technology, 41*(3), 569–577.
17. Cortez, R., Garrido, R., & Mezura-Montes, E. (2022). Spectral richness pso algorithm for parameter identification of dynamical systems under non-ideal excitation conditions. *Applied Soft Computing, 128*, 109–490. https://doi.org/10.1016/j.asoc.2022.109490
18. Arulvadivu, J., Manoharan, S., Lal Raja Singh, R., & Giriprasad, S. (2016). Optimal design of proportional integral derivative acceleration controller for higher-order nonlinear time delay system using m-mboa technique. *International Journal of Numerical Modelling: Electronic Networks, Devices and Fields, 3016*.
19. Bouallègue, S., Haggège, J., Ayadi, M., & Benrejeb, M. (2012). Pid-type fuzzy logic controller tuning based on particle swarm optimization. *Engineering Applications of Artificial Intelligence, 25*(3), 484–493.

20. Chang, W.-D. (2022). An improved particle swarm optimization with multiple strategies for pid control system design. *International Journal of Modeling and Optimization, 12*(2)
21. Roeva, O., & Slavov, T. (2012). Pid controller tuning based on metaheuristic algorithms for bioprocess control. *Biotechnology & Biotechnological Equipment, 26*(5), 3267–3277.
22. Joseph, S. B., Dada, E. G., Abidemi, A., Oyewola, D. O., & Khammas, B.M. (2022). Metaheuristic algorithms for pid controller parameters tuning: Review, approaches and open problems. Heliyon, 309–399.
23. Rodríguez-Molina, A., Mezura-Montes, E., Villarreal-Cervantes, M. G., & Aldape-Pérez, M. (2020). Multi-objective meta-heuristic optimization in intelligent control: A survey on the controller tuning problem. *Applied Soft Computing, 93*, 106–342.
24. Hu, J., & Chen, W.: Design of active disturbance rejection controller for dynamic positioning based on improved particle swarm optimization. *Mathematical Problems in Engineering, 2022*.
25. Liu, X., Gao, Q., Ji, Y., Song, Y., & Liu, J. (2022). Active disturbance rejection control of quadrotor uav based on whale optimization algorithm. In *2022 IEEE International Conference on Mechatronics and Automation (ICMA)* (pp. 351–356). IEEE.
26. Zhang, D., Yao, X., & Wu, Q. (2016). Parameter tuning of modified active disturbance rejection control based on the particle swarm optimization algorithm for high-order system. In *2016 IEEE International Conference on Aircraft Utility Systems (AUS)* (pp. 290–294). IEEE.
27. Olga Jimenez, J. M., & Ruben, G. (2020). Estudio comparativo de servomotores de cd orientados a la construccion de prototipos educativos. In *Congreso Internacional de Robotica Y Computacion (CIRC-2020)* (pp. 32–40). IEEE.
28. Ohishi, K., Ohnishi, K., & Miyachi, K. (1988). Adaptive dc servo drive control taking force disturbance suppression into account. *IEEE Transactions on Industry Applications, 24*(1), 171–176.
29. Ohnishi, K., Shibata, M., & Murakami, T. (1996). Motion control for advanced mechatronics. *IEEE/ASME Transactions on Mechatronics, 1*(1), 56–67.
30. Garrido, R., & Luna, L. (2021). Robust ultra-precision motion control of linear ultrasonic motors: A combined adrc-luenberger observer approach. *Control Engineering Practice, 111*, 104–812. https://doi.org/10.1016/j.conengprac.2021.104812
31. Spong, M.W., Hutchinson, S., Vidyasagar, M., et al. (2006). *Robot Modeling and Control* (vol. 3). New York: Wiley.
32. Wang, D., Tan, D., & Liu, L. (2018). Particle swarm optimization algorithm: an overview. *Soft computing, 22*(2), 387–408.
33. Kennedy, J., Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks* (vol. 4, pp. 1942–1948). https://doi.org/10.1109/ICNN.1995.488968
34. Sidorov, G. (2018). Artificial Intelligence. Alfa-Omega.
35. Pires, E., Tenreiro Machado, J., Moura Oliveira, P., Cunha, J., & Mendes, L. (2010). Particle swarm optimization with fractional-order velocity. *Nonlinear Dynamics, 61*, 295–301. https://doi.org/10.1007/s11071-009-9649-y
36. Song, B., Wang, Z., & Zou, L. (2021). An improved pso algorithm for smooth path planning of mobile robots using continuous high-degree bezier curve. *Applied Soft Computing, 100*, 106–960.
37. Xu, L., Song, B., & Cao, M. (2021). An improved particle swarm optimization algorithm with adaptive weighted delay velocity. *Systems Science & Control Engineering, 9*(1), 188–197. https://doi.org/10.1080/21642583.2021.1891153
38. Juarez-Castillo, E., Acosta-Mesa, H., & Mezura-Montes, E. (2019). Adaptive boundary constraint-handling scheme for constrained optimization. *Soft Computing,* 1–34. https://doi.org/10.1007/s00500-018-3459-4
39. Makeblock. (2019). 180 Optical Encoder Motor. https://store.makeblock.com/180-optical-encoder-motor Consultado: Septiembre 2022. https://store.makeblock.com/180-optical-encoder-motor

40. Ioannou, P. A., & Sun, J. (2012). *Robust Adaptive Control*. Mineola, New York: Courier Corporation.
41. Manuel López Ibáñez, e.a (2020). Leslie Pérez Cáceres: The Irace Package: User Guide. Université Libre de Bruxelles, Brussels, Belgium. https://books.google.com.mx/books?id=pfJHAQAAIAAJ