Available online at www.ijournalse.org

Emerging Science Journal

(ISSN: 2610-9182)

Vol. 7, No. 6, December, 2023

Trajectory Tracking Control of a Mobile Robot using Neural Networks

Darwin Trujillo [1], Luis A. Morales [1], Danilo Chávez [1], David F. Pozo [2*]

[1] *Departamento de Automatización Y Control Industrial, Escuela Politécnica Nacional, Quito, Ecuador.*

[2] *Facultad de Ingeniería y Ciencias Aplicadas, Ingeniería en Electrónica y Automatización, Universidad de Las Américas, Quito, Ecuador.*

Abstract

This paper presents a novel soft computing-based machine learning technique designed to enhance the trajectory tracking capabilities of mobile robots through the application of neural networks. The goal of this approach is to enhance the accuracy and overall performance of trajectory tracking without the need for manual gain recalibration, which is a tedious and time-consuming task for the designer when setting up the robot. This improvement is achieved by creating a kinematic controller based on neural networks, which are constructed using the kinematic model of the robot. In the initial phase, the controller requires gains defined by the designer. Subsequently, during the application phase, the backpropagation algorithm is used to dynamically adjust the gains of the neural network, aiming to minimize the closed-loop error. One of the key innovations introduced by this controller is the potential for automatic online gain tuning, thereby eliminating the need for a pre-learning phase, typically required by traditional neural controllers. To validate the effectiveness of this approach, the results are systematically analyzed and compared against those obtained using a conventional kinematic controller. Performance metrics reveal the improved precision in trajectory tracking achieved by the controller, with reduced effort, highlighting the performance enhancements in different trajectories.

Keywords:

Backpropagation;
Kinematics; Robot;
Neural Network;
Trajectory Tracking.

Article History:

Received:	12	September	2023
Revised:	19	November	2023
Accepted:	28	November	2023
Published:	01	December	2023

1- Introduction

Artificial Intelligence (AI) is an advanced and highly promising field in modern computing and science. It originated in the 1950s [1] and has made significant progress, demonstrating its usefulness in various applications. The fundamental aim of AI is to create intelligent systems, encompassing computers, machines, and various artifacts, capable of emulating human-like intelligence, which includes cognitive abilities, learning, and decision-making. These intelligent systems possess the ability to adapt to novel situations, solve problems, and undertake a wide range of tasks [2]. Nowadays, AI is swiftly transforming the world, and its influence is evident across numerous research domains, including robotics [3], medicine [4], and education [5], as well as in applications like classification and grouping problems [1]. Therefore, it is essential to understand the possibilities and limitations of AI to assess when and how to effectively leverage its potential.

The field of mobile robotics is in a state of continuous advancement, with a primary focus on the development and construction of robots capable of autonomous movement and operation [6]. These robots have diverse applications, including infrastructure inspection [7], agriculture [8], and performing tasks in hazardous environments unsuitable for humans [9]. In the realm of mobile robotics, trajectory tracking control stands as a prevalent and actively researched task. Controllers for mobile robots are typically constructed using mathematical models that diligently capture the

* **CONTACT**: david.pozo@udla.edu.ec

DOI: http://dx.doi.org/10.28991/ESJ-2023-07-06-01

behavior of the robot. Nevertheless, owing to the intricate nature of these systems and the multitude of variables at play, a spectrum of challenges emerges, including issues related to model uncertainty and non-linearity. The degree of complexity inherent to the system dictates the magnitude of these challenges [10]. There are three main categories of modeling [11]:

- ***Analytical Modelling***: Models are calculated using differential equations of the system.
- ***System Identification***: The modeling process is performed based on the measurement of the input-output data of the system.
- ***Soft Computing***: Contemporary computational methods within the field of computer science rely on technologies such as artificial intelligence and Deep Learning, among others. Techniques within this category encompass neural networks, fuzzy systems, and evolutionary algorithms.

Soft computing covers a wide range of problem-solving approaches, especially in engineering areas, inspired by algorithms that emulate human learning to improve their behavior in dynamic environments. Within this context, the use of neural networks for control strategies as a soft computing technique is grounded in their ability to mimic human cognitive processes and their aptitude for tackling vaguely defined problems efficiently when compared to traditional methodologies [12].

Currently, research is being conducted on a variety of intelligent control techniques, including the use of artificial neural networks, genetic algorithms, fuzzy logic, and more, aimed at tackling challenges within the realm of mobile robotics. These challenges encompass tasks such as autonomous navigation in complex and unpredictable environments, as well as the calibration of controllers based on the surface the robot traverses and the desired path to follow, factors that frequently influence the robot's performance [13, 14]. Artificial neural networks exhibit a remarkable attribute, enabling them to approximate nonlinear systems or processes effectively, even when information is limited. Consequently, the development of a controller with the ability to address uncertainty parameters and nonlinearities within the model offers the potential to significantly improve the trajectory tracking performance of mobile robots [6].

In recent works, artificial neural networks have been employed as a fundamental component in intelligent control systems. Liu & Cong [15] used a neural network to solve the nonlinearity problem present in mobile robots. However, the use of IoT sensors is necessary for the robot to be able to perform effective control under the use of a tracking scheme and a radial basis function adaptive algorithm. This self-adapting algorithm based on IoT and neural networks allows for compensating for the uncertainty of the parameters. However, the accuracy of the neural network model decreases when the parameters are changed, and it is also necessary to use sensors to provide the necessary information from the environment to transmit to the robot. Asai et al. [16] proposed a neural network for robots with sensors. Chen et al. [17] proposed an adaptive neural network to approximate the unknown dynamics of the robot as well as a Lyapunov barrier to limit the robot's speed. Another study, Mohareri et al. [18], adopts an online approach for tuning the controller parameters. In this case, the neural network is designed to learn the characteristics of the direct model and determine the Jacobian of the system.

There are studies in which the neural network training phase is performed offline. In Yildirim et al. [19], the design of the controller is done by means of a neural network predictor. This is not so feasible because it is necessary to use training data until the appropriate values of the gains are obtained so that the error is minimized. In Mohamed & Hamza [20] and Gou & Liu [21] studies, the neural controller is made from the principles of a neural network of the structure of a PID controller, and with a learning algorithm, the parameters of the PID controller are found to minimize the trajectory tracking error. This would represent more work because it is necessary to find the appropriate values of the gains of each controller so that the error is minimized. Hassan & Saleem [10] propose a combination of a neural network-based controller and model reference adaptive control (MRAC). The inclusion of the model reference adaptive control helps ensure stable tracking even in the presence of parameter uncertainties. The neural network in this model consists of 108 neurons distributed across three layers, utilizing the sigmoidal function as the activation function. The inputs to the network are the reference state and the current state, while the outputs correspond to the controller gains, k_x and k_y. The training process involves an offline phase, during which a set of reference trajectories are applied as inputs to the closed-loop system. The backpropagation algorithm is used to determine the gains, and the optimal values are selected once the network is trained. Consequently, if the network is not well-trained, the provided outputs may not be as accurate.

In the Rossomando et al. [22] study, a feedback linearization based on a nominal model and a Radial Basis Function Neural Network (RBF-NN) adaptive dynamic compensation is proposed. A kinematic controller and an inverse dynamics controller are implemented separately. The uncertainty of the dynamics model is compensated by an adaptive neural controller; however, it is not considered a kinematic controller adaptable to changes in trajectory, and the calibration of this controller is not specified. In the Nath et al. [23] study, a radial basis function neural network (RBF-NN) is used to realize a sliding mode adaptive controller (SMC). To improve the error convergence and reduce chattering, the weights of the RBF-NN and the parameters of the SMC are estimated with an adaptive tuning law.

Morales, Aguilar, et al. [24] propose an artificial intelligence-based method called LAMDA (Learning Algorithm for Multivariate Data Analysis) with an adaptive approach. In this study (adaptive LAMDA), the initial parameters of the controller are set, and the control strategy is calculated and updated by an online learning process that evaluates the closed-loop model. The primary challenge with this proposal lies in its dependence on an initial learning phase, which, in specific systems, may not be practically feasible unless it can be executed through simulation. LAMDA has satisfactory results in classification and clustering tasks [25–27], has been tested in a novel way, and has good results in the area of control systems.

In the study conducted by Naveed et al. [28], trajectory tracking for a mobile robot was achieved through the implementation of two distinct control techniques. One of these techniques, referred to as the Direct Reference Adaptive Controller (D-MRAC), embodies a model-dependent approach. The other, known as the Adaptive Neuro-Fuzzy Inference System (ANFIS), represents a data-driven approach. The ANFIS technique is trained using a straightforward State Dependent Riccati Equation (SDRE)-based controller and demonstrates the ability to adapt in the presence of parametric uncertainties. While D-MRAC stands out for its simplicity, it is evident that, in the context of real-world applications characterized by uncertainties, ANFIS emerges as the more suitable choice for effective control. Benbouabdallah & Qi-dan [29] proposed a fuzzy logic controller based on Takagi-Sugeno. This method calculates the linear and angular velocity of the mobile robot to meet the control objectives. Then, a genetic algorithm is applied to improve the Fuzzy Logic Controller (FLC) through the optimization of the scaling factors of the fuzzy controller inputs to improve the accuracy and robustness of the trajectory tracking; however, this methodology is recursive and must be done offline.

The key aspects overlooked in the aforementioned studies refer to the requirement of an initial training phase to establish the correlation between the inputs and outputs of the system. This poses a challenge because inadequate training could lead to less accurate results. Therefore, our contribution involves the incorporation of an online learning phase for the controller, which improves precision and performance in tracking trajectories by adapting its behavior based on the error measured between the desired and actual position. In this paper, we use neural networks for the design and self-tuning of the controller in online mode. The neural network architecture closely resembles that of a multilayer perceptron model, and it employs the backpropagation algorithm to reduce the trajectory tracking error of the mobile robot. The advantages of the proposed proposal are the following:

- To enhance the performance of the robot in following a desired trajectory by adjusting the controller gains.
- The controller does not require an offline learning phase.
- The controller initializes its operation with the gains obtained from a conventional kinematic control. Subsequently, it learns from the behavior of the robot and autonomously adjusts the gains to improve its performance.
- Optimizing the controller gains by employing a neural network rooted in the kinematic model. Calibrating these parameters is typically a complex and time-consuming task for designers.

Finally, the performance of the proposed controller is assessed using performance indices commonly employed in control systems, such as ISE, IAE, and ISU, and is compared with a non-self-adjusting kinematic controller to observe enhancements in system behavior when tracking various trajectories.

The paper is organized as follows: Section II presents a theoretical and mathematical overview of neural networks. Section III shows the mathematical representation of the complete model of the mobile robot. The neural network-based controller design is described in Section IV. Experimental results are presented in Section V, a brief conclusion in Section VI, and future work in Section VII.

2- Artificial Neural Networks

Artificial Neural Networks (ANN), a soft computing technique [11], has the ability to approximate nonlinear systems. Its adaptive schemes can reduce the uncertainty of nonlinear systems [30] by tuning their internal parameters. ANNs are generally used for pattern recognition, system control, and computer vision. The most useful neural network for system control purposes is the Multi-layer Perceptron (MLP) [10]. The learning process in ANN consists of modifying the values of the synaptic weights. This is achieved by minimizing a given objective function using the backpropagation algorithm [1, 6]. In supervised learning, the correct output data for a set of input data is known in advance. This data is provided semi-automatically or by a supervisor [31].

Multi-layer Perceptron

It is one of the most widely used schemes at present because it is capable of approximating nonlinear systems, filtering noise, and being a universal approximator, among others. It has limitations due to the difficulty of performing an interpretation of the network because of its nonlinear components. It is characterized because the neurons are grouped at various levels [1, 6].

Figure 1 shows the model of a Multilayer Perceptron Model (MLP). In each interconnection, there are values called synaptic weights. Three types of layers are distinguished. The input layer receives signals from other neurons or from the outside and propagates them to the neurons in the next layer. In the hidden layers, neurons perform nonlinear processing of the received patterns. In the last layer, a network response is given for the input values entered. In general, neurons in one layer are connected to all neurons in the next layer, so there is connectivity throughout the network [1].

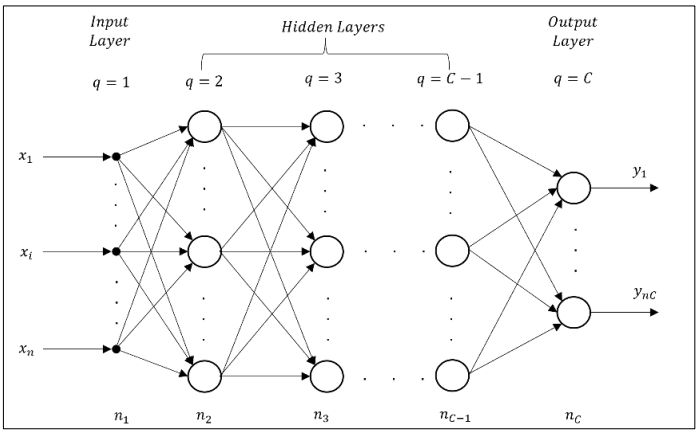


Figure 1. Multilayer Perceptron Model

For a MLP with C layers, $C-2$ hidden layers and $q = 1, 2, \ldots, C$.

The activation of neurons in the input layer is:

$$a_i^1 = x_i \ \forall\, i = 1,\ 2\ ,..\, n_1 \tag{1}$$

Where a_i^1 is the activation of neuron i in the input layer, x_i is the input vector of the artificial neural network, n_1 are the neurons in the input layer.

The activation of the neurons of the hidden layer q is:

$$a_i^q = f\left(\sum_{j=1}^{n_{q-1}} w_{ji}^{q-1} a_j^{q-1} + \theta_i^q\right)$$
$$\forall\, i = 1,\ 2\ ,..\, n_q\ \text{^}\ q = 2,3, \ldots, C-1 \tag{2}$$

where a_i^q is the activation of neuron i in layer q, a_j^{q-1} is the activation of neurons j in layer $q-1$, n_q are the neurons in layer q, W_{ji}^{q-1} is the vector of weights from neuron j to neuron i between layer $q-1$ and layer q, θ_i^q is the vector of thresholds in layer q.

The activation function f corresponds to a hyperbolic tangent (4) and its derivative (5).

$$f(x) = \tanh(x) = \frac{1-e^{-x}}{1+e^{-x}} \tag{3}$$

$$f'(x) = \operatorname{sech}^2(x) = 1 - \tanh^2(x) \tag{4}$$

Multi-layer Perceptron

It is a supervised learning algorithm [32] responsible for adapting the synaptic weights to minimize the mean square error between the desired output and the actual output. Errors are propagated in the hidden layers and in the output layer [6]. Therefore, learning is posed as a minimization problem of the form:

$$Min_W E \tag{5}$$

where W are the weights of the network, E is the error function.

The error function is expressed as:

$$E(n) = \frac{1}{2}\sum_{i=1}^{nC}(s_i(n) - y_i(n))^2 \tag{6}$$

where $s_i(n)$ is the desired output value for pattern n, $y_i(n)$ is the output value of the network.

To find the minimum of Equation 7 we use the stochastic gradient descent rule also known as backpropagation algorithm since the error committed by the network propagates backwards. This rule consists in modifying each parameter w of the network to minimize the errors committed by the network for each input pattern n. [1]. The learning law is as follows:

$$W(n) = W(n-1) - \alpha \frac{\partial E(n)}{\partial W} \tag{7}$$

Where α is the learning rate [0,1] and corresponds to the percentage change in which each weight is updated. In this paper the value of alpha has been selected heuristically [33]. However, it can be considered that at high α values a local minimum may not be found, and the algorithm may not even converge, while at low α it may converge to the local minimum but at the cost of longer processing time.

The generalization of the backpropagation algorithm is described as:

$$\begin{aligned} &W_{kj}^{q}(n) = W_{kj}^{q}(n-1) - \alpha\, \delta_i^{q-1}(n)\, a_k^q(n); \\ &\forall\, i = 1,2,\dots,n_c\,; \quad j = 1,2,\dots,n_{c-1}\,; \\ &q = 1,2,\dots,C-2 \end{aligned} \tag{8}$$

Where $a_k^q(n)$ is the activation of neuron k of layer q for pattern n and defining the term δ associated with neuron i of layer $q + 1$ for pattern n, as:

$$\delta_i^{q-1}(n) = f'\left(\sum_{k=1}^{n_q} W_{kj}^q a_k^q + \theta_j^q\right) \sum_{i=1}^{n_{q+1}} \delta_i^{q+2}(n)\; W_{ji}^q \tag{9}$$

Where W_{kj}^q is the vector of weights from neuron k to neuron j between layer q and layer $q + 1$. Equation 10 needs the derivative of the activation function. This derivative is given by Equation 5.

The generalization for the network thresholds is described as:

$$\begin{aligned} &\theta_j^{q-1}(n) = \theta_j^{q-1}(n-1) - \alpha\, \delta_i^{q-1}(n) \\ &\forall\; j = 1,2,\dots,n_{c-1};\; q = 2,\dots,C-2 \end{aligned} \tag{10}$$

where θ_j^{q-1} is the vector of thresholds in the $q-1$ layer for the n pattern.

3- Tracking Trajectory of Mobile Robots

Path-following control is used in scenarios where the operational surroundings are known [34]. Typically, these controllers are designed based on the kinematics of the robot. However, in cases where dynamic characteristics such as inertia, mass, or center of mass undergo changes, the robot's dynamics are taken into consideration [35]. After generating the desired trajectory, the primary objective for the robot is to accurately track it. However, due to the complexity involved in obtaining dynamic models for these systems and the presence of uncertainties, disturbances, sensor measurement errors, and other factors, errors in robot motion may arise [36]. To address these challenges, the proposed controller is intended to be implemented in such systems, as it can learn from the unknown dynamics of the system's behavior [37]. This controller enables the robot to achieve and follow the parameterized reference within a specified time period while minimizing tracking errors in a simulated robot environment.

Robot Model

The differential mobile robot is a non-omnidirectional system used in typical applications with low mechanical complexity and energy consumption [6]. It consists of two independently controlled conventional wheels and a passive wheel to maintain balance and stability. It is frequently used in control systems due to its fast and nonlinear dynamics [38]. Figure 2 shows the representation of a differential mobile robot, where O is the center of the axis between the left and right wheels, P is the center of gravity of the mobile robot and corresponds to the control point, a is the distance between the central point O joining the drive wheels to point P, r is the radius of each wheel and d is the distance between the wheels, u and ω correspond to the linear velocity and angular velocity respectively, φ is the orientation of the robot.

Assuming a robot configuration $[x_k\ y_k\ \varphi_k]$ and that the velocity inputs u_k and ω_k are known at discrete time t_k, then using the Euler integration method [39], the estimate of the robot configuration at time t_k is calculated as:

$$\begin{bmatrix} x_k \\ y_k \\ \varphi_k \end{bmatrix} = \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \varphi_{k-1} \end{bmatrix} + \begin{bmatrix} \cos\varphi_k & -a\sin\varphi_k \\ \sin\varphi_k & a\cos\varphi_k \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \Delta S \\ \Delta\varphi \end{bmatrix} \tag{11}$$

where $\Delta S = u_k T_s$, $\Delta\varphi = \omega_k T_s$ and $T_s = t_k - t_{k-1}$, T_s is the sampling time.

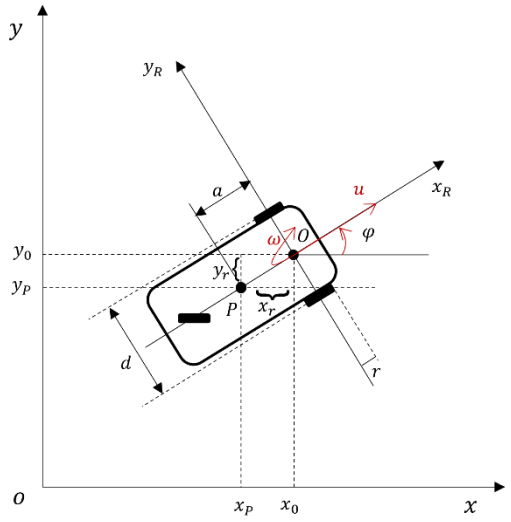


Figure 2. Geometry Differential Mobile Robot

In Rossomando et al. [22], the application of two controllers based on the kinematics and dynamics of the robot is presented. In this work, the dynamic model is considered unknown, so the identification of the behavior of its dynamics is performed with the controller based on neural networks.

Kinematic Controller

The kinematic controller presented in Equation 12 is based on the kinematic model of the robot represented by (12), taking into account the coordinates of the point $[x\ y]^T$. The control law, considering a P-type controller [24], is:

$$\begin{bmatrix} u_{ref}(k) \\ \omega_{ref}(k) \end{bmatrix} = \begin{bmatrix} \cos\varphi\,(k) & \sin\varphi\,(k) \\ -\frac{1}{a}\sin\varphi\,(k) & \frac{1}{a}\cos\varphi\,(k) \end{bmatrix} \times \begin{bmatrix} \frac{x_d(k)-x_d\,(k-1)}{T_s} + k_x\, e_x(k) \\ \frac{y_d(k)-y_d\,(k-1)}{T_s} + k_y\, e_y(k) \end{bmatrix} \tag{12}$$

where $a < 0$. $[u_{ref}\ \omega_{ref}]^T$ is the output of the kinematic controller, $k_x > 0\,, k_y > 0$ are the controller gains, $e_x(k) = x_d(k) - x(k), e_y(k) = y_d(k) - y(k)$ are the position errors in the X and Y axes respectively and $x_d(k)$ and $y_d(k)$ are the desired coordinates. For the implementation of the kinematic controller based on neural networks we consider, $u_{ref}(k) = u(k), \omega_{ref} = w(k)$.

4- Neural Network-based Controller Design

Traditional tracking control systems for mobile robots do not address model uncertainties, fluctuations in plant parameters, or external disturbances. A proficient controller should deliver satisfactory performance even in the absence of these compensations [10]. The block diagram presented in Figure 3 outlines the suggested neural network architecture and the utilization of neural network-based controllers to guide the system towards the desired reference trajectory.

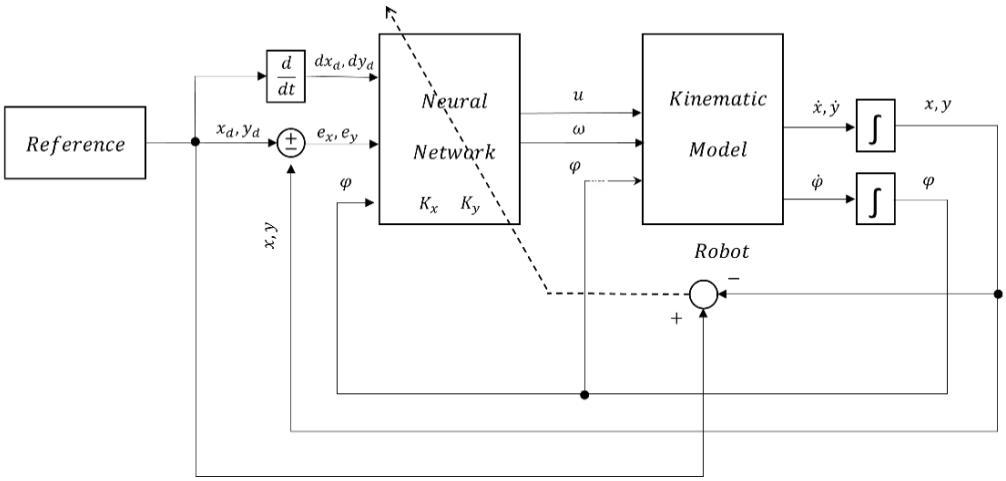


Figure 3. Block diagram of the proposed control architecture

Figure 4 depicts a flowchart illustrating the proposed controller. It delineates the control stages, featuring a neural network structured akin to a kinematic controller. Additionally, the online learning phase employs the backpropagation algorithm, with the objective of minimizing the position error between the robot and the trajectory points and facilitating the parameter updates of the neural network.

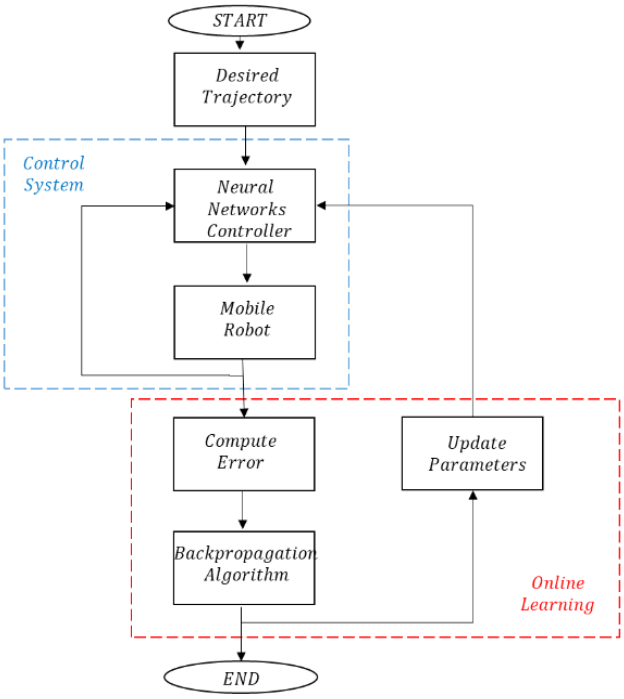


Figure 4. Flowchart of the proposed methodology

From the control laws of the kinematic controller, the design of the Artificial Neural Network (ANN) is carried out. Developing the control law of Equation 13, it is obtained:

$$u(k) = \cos\varphi\,(k)\left(\frac{x_d(k)-x_d\,(k-1)}{T_s} + k_x\,e_x(k)\right) + \sin\varphi\,(k)\left(\frac{y_d(k)-y_d\,(k-1)}{T_s} + k_y\,e_y(k)\right) \quad (13)$$

$$\omega(k) = -\frac{1}{a}\sin\varphi\,(k)\left(\frac{x_d(k)-x_d\,(k-1)}{T_s} + k_x\,e_x(k)\right) + \frac{1}{a}\cos\varphi\,(k)\left(\frac{y_d(k)-y_d\,(k-1)}{T_s} + k_y\,e_y(k)\right) \quad (14)$$

The ANN is created using Equations 14 and 15 and is modelled after a MLP design. This architecture is based on the kinematic controller model of a mobile robot, as shown in Figure 5.

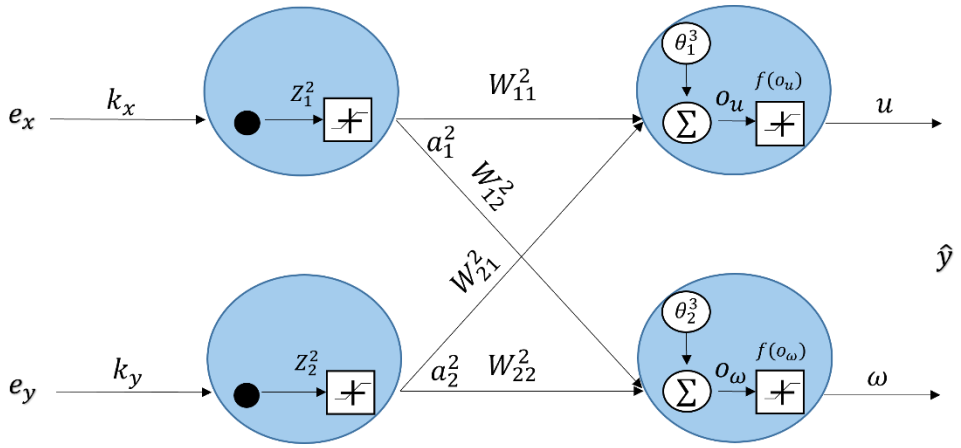


Figure 5. Neural Network designed from the model of the kinematic controller of the mobile robot

The input layer is connected 1 to 1 with the first hidden layer since the weights W_{12}^1 and W_{21}^1 are zero. In addition, no threshold is considered in the first hidden layer. These considerations are taken from the neural network design based on the model of the kinematic controller of the mobile robot shown in Figure 5. The gains k_x and k_y correspond to the synaptic weights of the neural network that will be adjusted so that the system achieves fast convergence to the desired reference by decreasing the position error. In addition, the activation function used is the hyperbolic tangent described

by Equation 4. This function serves as a saturation mechanism for the control signals u and ω, ensuring that their values do not exceed certain limits imposed by the actuators. It can take both positive and negative values, preventing the delivery of control signals that surpass the permissible energy levels.

The activation of the neurons in the input layer q = 1 is found with Equation 1 and is given by:

$$a_1^1 = e_x(k) \tag{15}$$

$$a_2^1 = e_y(k) \tag{16}$$

In the hidden layer $q = 2$ we have the following relations:

$$Z_1^2 = k_x\, e_x(k) \tag{17}$$

$$Z_2^2 = k_y\, e_y(k) \tag{18}$$

The activation of neurons in the hidden layer is found with Equation 2 and is given by:

$$a_1^2 = f(Z_1^2) = \tanh(k_x\, e_x(k)) \tag{19}$$

$$a_2^2 = f(Z_2^2) = \tanh\big(k_y\, e_y(k)\big) \tag{20}$$

The synaptic weights in the hidden layer are given by:

$$W_{11}^2 = \cos\varphi \tag{21}$$

$$W_{12}^2 = -\frac{1}{a}\sin\varphi \tag{22}$$

$$W_{21}^2 = \sin\varphi \tag{23}$$

$$W_{22}^2 = \frac{1}{a}\cos\varphi \tag{24}$$

The thresholds in the output layer are given by:

$$\theta_1^3(k) = \left(\frac{x_d(k) - x_d\,(k-1)}{T_s}\right)\cos\varphi\,(k) + \left(\frac{y_d(k) - y_d\,(k-1)}{T_s}\right)\sin\varphi\,(k) \tag{25}$$

$$\theta_2^3(k) = -\frac{1}{a}sin\varphi(k)\,\left(\frac{x_d(k) - x_d\,(k-1)}{T_s}\right) + \frac{1}{a}cos\varphi(k)\,\left(\frac{y_d(k) - y_d\,(k-1)}{T_s}\right) \tag{26}$$

The activation of the neurons of the output layer $q = 3$ is found with Equation 3 and is given by:

$$u(k) = a_1^3 = \tanh(o_u(k)) \tag{27}$$

$$\omega(k) = a_2^3 = \tanh(o_\omega(k)) \tag{28}$$

where:

$$o_u(k) = cos\varphi(k)\, f(Z_1^2) + sin\varphi(k)\, f(Z_2^2) + \theta_1^3(k) \tag{29}$$

$$o_\omega(k) = -\frac{1}{a}sin\varphi(k)\, f(Z_1^2) + \frac{1}{a}cos\varphi(k)\, f(Z_2^2) + \theta_2^3(k) \tag{30}$$

The objective of the designed controller is to guarantee that the output of the plant output closely tracks the model reference while minimizing the error on both the x and y axes. The gains k_x and k_y will be adjusted by the backpropagation algorithm until the error between the current trajectory and the desired trajectory is approximately zero.

The total trajectory tracking error is:

$$E(n) = \frac{1}{2}\left(e^2{}_x + e^2{}_y\right) \tag{31}$$

The synaptic weights k_x and k_y will be adapted to minimize Equation 31. Thus, the learning is posed as a minimization problem of the form:

$$Min_{k_x,k_y}E \tag{32}$$

From Equation 8, learning laws are defined as:

$$k_x(n) = k_x(n-1) - \alpha \frac{\partial E(n)}{\partial k_x} \tag{33}$$

$$k_y(n) = k_y(n-1) - \alpha \frac{\partial E(n)}{\partial k_y} \tag{34}$$

To solve Equations 33 and 34 it is necessary to calculate the following derivates:

$$\frac{\partial E}{\partial k_x} = \frac{\partial E}{\partial x} + \frac{\partial E}{\partial y} \tag{35}$$

$$\frac{\partial E}{\partial k_y} = \frac{\partial E}{\partial x} + \frac{\partial E}{\partial y} \tag{36}$$

Using the chain rule for Equations 35 and 36 we have:

$$\frac{\partial E}{\partial k_x} = \frac{\partial E}{\partial x}\left[\left(\frac{\partial x}{\partial u}\frac{\partial u}{\partial o_u}\frac{\partial o_u}{\partial a_1^2} + \frac{\partial x}{\partial \omega}\frac{\partial \omega}{\partial o_\omega}\frac{\partial o_\omega}{\partial a_1^2}\right)\frac{\partial a_1^2}{\partial Z_1^2}\frac{\partial Z_1^2}{\partial k_x}\right] + \frac{\partial E}{\partial y}\left[\left(\frac{\partial y}{\partial u}\frac{\partial u}{\partial o_u}\frac{\partial o_u}{\partial a_1^2} + \frac{\partial y}{\partial \omega}\frac{\partial \omega}{\partial o_\omega}\frac{\partial o_\omega}{\partial a_1^2}\right)\frac{\partial a_1^2}{\partial Z_1^2}\frac{\partial Z_1^2}{\partial k_x}\right] \tag{37}$$

$$\frac{\partial E}{\partial k_y} = \frac{\partial E}{\partial x}\left[\left(\frac{\partial x}{\partial u}\frac{\partial u}{\partial o_u}\frac{\partial o_u}{\partial a_2^2} + \frac{\partial x}{\partial \omega}\frac{\partial \omega}{\partial o_\omega}\frac{\partial o_\omega}{\partial a_2^2}\right)\frac{\partial a_2^2}{\partial Z_2^2}\frac{\partial Z_2^2}{\partial k_y}\right] + \frac{\partial E}{\partial y}\left[\left(\frac{\partial y}{\partial u}\frac{\partial u}{\partial o_u}\frac{\partial o_u}{\partial a_2^2} + \frac{\partial y}{\partial \omega}\frac{\partial \omega}{\partial o_\omega}\frac{\partial o_\omega}{\partial a_2^2}\right)\frac{\partial a_2^2}{\partial Z_2^2}\frac{\partial Z_2^2}{\partial k_y}\right] \tag{38}$$

Next, we find the partial derivatives needed to solve Equations 37 and 38.

By subtracting e_x from Equation 13 and replacing $e_x(k) = x_d(k) - x(k)$, we have:

$$x(k) = x_d(k) - \frac{u(k)}{k_x \cos\varphi(k)} + \frac{\sin\varphi(k)\left(y_d(k+1) - y_d(k) + k_y\, e_y(k) T_s\right)}{k_x \cos\varphi(k) T_s} + \frac{x_d(k+1) - x_d(k)}{k_x T_s} \tag{39}$$

By subtracting e_y from Equation 13 and replacing $e_y(k) = y_d(k) - y(k)$, we have:

$$y(k) = y_d(k) - \frac{u(k)}{k_y \sin\varphi(k)} + \frac{\cos\varphi(k)\left(x_d(k+1) - x_d(k) + k_x\, e_x(k) T_s\right)}{k_y \sin\varphi(k)\, T_s} + \frac{y_d(k+1) - y_d(k)}{k_y T_s} \tag{40}$$

By subtracting e_x from Equation 14 and replacing $e_x(k) = x_d(k) - x(k)$, it is obtained:

$$x(k) = x_d(k) - \frac{\cos\varphi(k)\left(y_d(k+1) - y_d(k) + k_y\, e_y(k) T_s\right)}{k_x \sin\varphi(k) T_s} + \frac{\omega(k)\, a}{k_x \sin\varphi(k)} + \frac{x_d(k+1) - x_d(k)}{k_x T_s} \tag{41}$$

By subtracting e_y from Equation 14 and replacing $e_y(k) = y_d(k) - y(k)$, it is obtained:

$$y(k) = y_d(k) - \frac{\sin\varphi(k)\left(x_d(k+1) - x_d(k) + k_x\, e_x(k) T_s\right)}{k_y \cos\varphi(k)\, T_s} - \frac{\omega(k)\, a}{k_y \cos\varphi(k)} + \frac{y_d(k+1) - y_d(k)}{k_y T_s} \tag{42}$$

Obtaining the partial derivative:

$$\frac{\partial x}{\partial u} = -\frac{1}{k_x \cos\varphi} \tag{43}$$

$$\frac{\partial y}{\partial u} = -\frac{1}{k_y \sin\varphi} \tag{44}$$

$$\frac{\partial x}{\partial \omega} = \frac{a}{k_x \sin\varphi} \tag{45}$$

$$\frac{\partial y}{\partial \omega} = -\frac{a}{k_y \cos\varphi} \tag{46}$$

From Equation 31 the derivative of the error with respect to the position in x and y is:

$$\frac{\partial E}{\partial x} = -e_x \tag{47}$$

$$\frac{\partial E}{\partial y} = -e_y \tag{48}$$

From Equations 27 and 28 and replacing Equation 5 we obtain:

$$\frac{\partial u}{\partial o_u} = f'(o_u) = 1 - \tanh^2(o_u) \tag{49}$$

$$\frac{\partial \omega}{\partial o_\omega} = f'(o_\omega) = 1 - \tanh^2(o_\omega) \tag{50}$$

From Equations 29 and 30, it is obtained:

$$\frac{\partial o_u}{\partial a_1^2} = \cos\varphi \tag{51}$$

$$\frac{\partial o_\omega}{\partial a_1^2} = -\frac{1}{a}\sin\varphi \tag{52}$$

$$\frac{\partial o_u}{\partial a_2^2} = \sin\varphi \tag{53}$$

$$\frac{\partial o_\omega}{\partial a_2^2} = \frac{1}{a}\cos\varphi \tag{54}$$

Considering the derivative of the activation function described by Equation 5, it is obtained:

$$\frac{\partial a_1^2}{\partial Z_1^2} = 1 - \tanh^2(e_x k_x) \tag{55}$$

$$\frac{\partial a_2^2}{\partial Z_2^2} = 1 - \tanh^2(e_y k_y) \tag{56}$$

Finally, from Equations 18 and 19 the derivatives respect k_x and k_y are:

$$\frac{\partial Z_1^2}{\partial k_x} = e_x \tag{57}$$

$$\frac{\partial Z_2^2}{\partial k_y} = e_y \tag{58}$$

5- Experimental Tests

The neural network and the proposed controller are deployed on a 3DX Pioneer robot within a 3D virtual robotic interface. CoppeliaSim Edu serves as a robotic simulator equipped with an integrated development environment, commonly employed for system simulation, hardware control, remote monitoring, product presentation, control algorithm development, and various robotics applications. Its effectiveness and versatility in simulating robotic systems arise from the capability to individually control each component of the system, taking into account its kinematics, dynamics, and the surrounding environment using diverse control mechanisms [40]. The controllers have been programmed in MATLAB and seamlessly incorporated into the software through plug-ins. Figure 6 shows the navigation interface within CoppeliaSim, featuring the 3DX Pioneer robot.

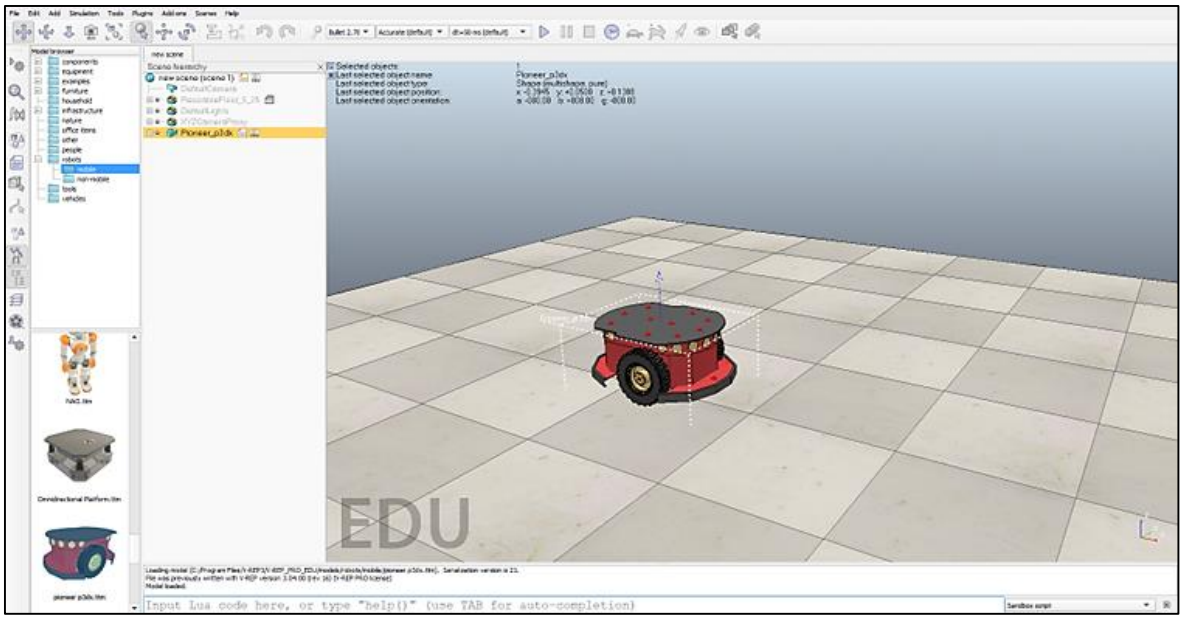


Figure 6. Development Environment of CoppeliaSim

The robot trajectory tracking control will be applied in three different trajectories. The trajectories tested are: Circular (56), Square (57), Lemniscate (58). Graphical and numerical comparisons are performed to check the performance of the designed controller in this control task.

The starting point for all trajectories will be equal to $(x, y) = (0, 0)m$.

$$\begin{cases} x_{ref}(k) = 2\ \cos(0.033\pi k T_0) \\ y_{ref}(k) = 2\ \sin(0.033\pi k T_0) \end{cases} \tag{59}$$

$$\begin{cases} x_{ref}(k) = 1.5\ \forall k T_0\ \epsilon\ [0{,}15];\ (4.5 - 0.2k T_0)\ \forall k T_0\ \epsilon\ [15{,}30]; \\ \quad -1.5\ \forall k T_0\ \epsilon\ [30{,}45];\ (-10.5 + 0.2k T_0) \forall k T_0\ \epsilon\ [45{,}60] \\ y_{ref}(k) = (-1.5 + 0.2k T_0)\ \forall k T_0\ \epsilon [0{,}15];\ 1.5\ \forall k T_0\ \epsilon\ [15{,}30]; \\ \quad (7.5 - 0.2k T_0)\ \forall k T_0\ \epsilon [30{,}45]; -1.5\ \forall k T_0\ \epsilon\ [45{,}60]; \end{cases} \tag{60}$$

$$\begin{cases} x_{ref}(k) = 1.2\sin(0.063\pi k T_0) \\ y_{ref}(k) = 2\sin(0.0315\pi k T_0) \end{cases} \quad (61)$$

The initial parameters for the kinematic controller based on neural networks presented in Equations 14 and 15 are: $k_x = 1, k_y = 1, \alpha = 0.03$. For the conventional kinematic controller (12) the initial parameters are $k_x = 1, k_y = 1$.

These values were determined using a heuristic method. However, when adjusting them, opting for lower gain values would result in an extended timeframe for the mobile robot to reach the trajectory. Conversely, choosing higher gain values could expedite trajectory completion, but this might entail potential drawbacks such as reduced trajectory tracking precision or an increased risk of instability. Elevated gain values would also amplify both linear and angular velocities, thereby increasing the mechanical strain on the actuators.

For experimentation, we propose to compare the neural network-based kinematic controller (NN Controller) with a conventional proportional kinematic controller (P Controller). It is worth noting that the comparison is only made with the latter controller because the literature reports it as the most widely used method for trajectory tracking. Many proposals in the literature focus on compensating the dynamic model in cascade structures, where the outer loop is responsible for controlling the kinematics of the controller, which is the aspect we aim to improve with our proposal.

The comparison is performed by calculating performance indices. These are numerical indicators employed in control systems to evaluate the efficacy of controller and its selection. Typically, the chosen parameters should be such that they minimize the values associated with the performance indexes [41].

The Integral of the Squared Error, as described in Equation 62, serves as a metric for system calibration. Systems calibrated using this metric demonstrate a rapid decrease in error during the transient state. Minimizing this metric not only leads to improved system performance but also reduces energy consumption in achieving the reference state [42].

$$ISE = \int_0^t e^2(t)dt \quad (62)$$

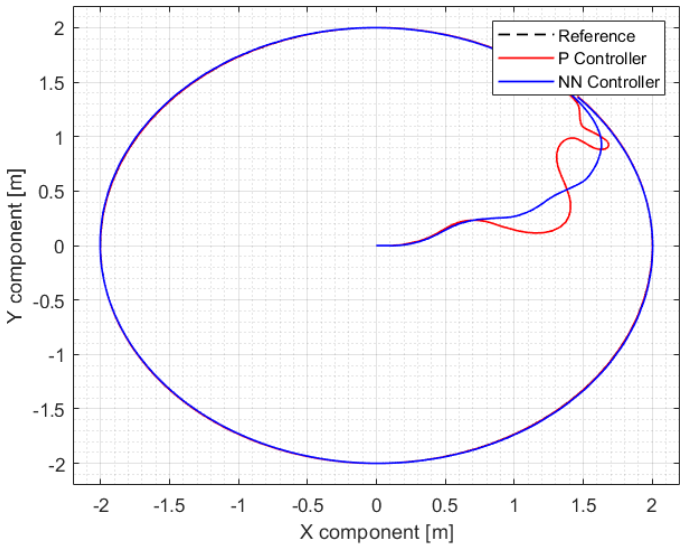
Integral of the Absolute Value of the Error, as depicted in Equation 63, is utilized for calibrating systems. When systems are calibrated using this index, they exhibit satisfactory damping and acceptable transient response [43].

$$IAE = \int_0^t |e(t)|dt \quad (63)$$

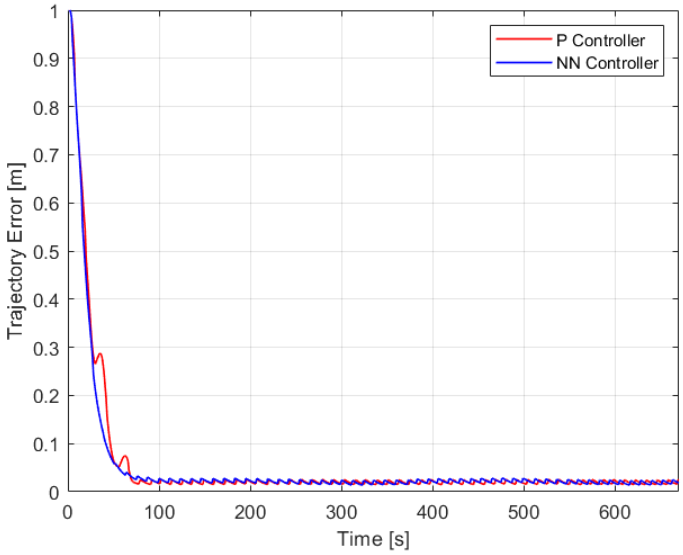
Integral of the Control Output Squared is defined by Equation 64. This index represents the control effort employed by the controller. Reducing this index will result in decreased energy consumption by the controller [44].

$$ISU = \int_0^t u^2(t)dt \quad (64)$$

Figure 7-a shows the tracking of the mobile robot for a circular trajectory with radius $2m$, while Figure 7-b presents the tracking error in that trajectory. The trajectory followed by the NN controller is less oscillatory and reached more quickly than the P controller specially at the beginning of the simulation. With the conventional P Controller (see Figure 8), The control signals for linear and angular velocity exhibit pronounced and high-amplitude oscillations, which may impact the durability of the robot's actuators. Furthermore, the saturation point for linear velocity on the mobile robot is reached. These oscillations diminish as the robot converges to the desired trajectory.

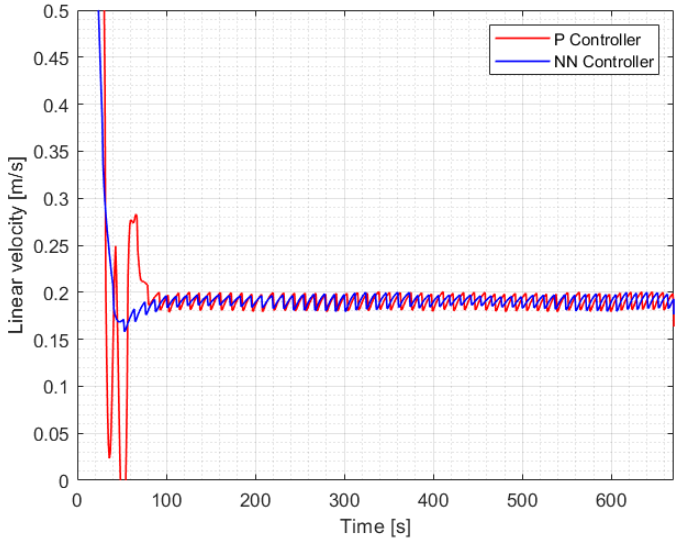


(a)

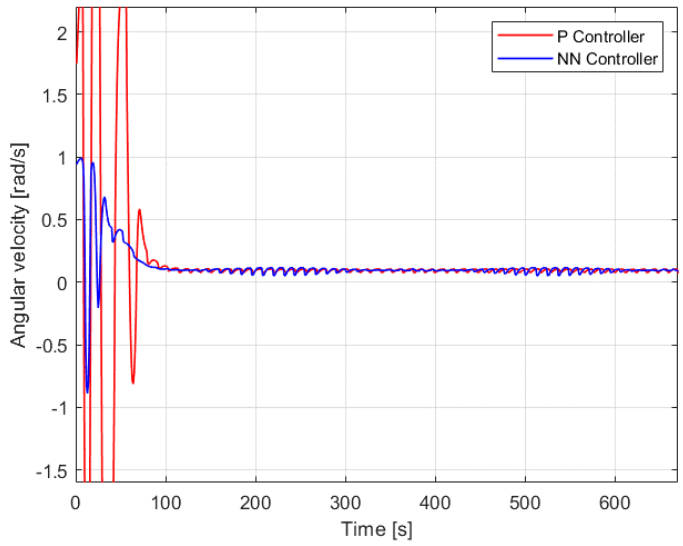


(b)

Figure 7. a) Circular Trajectory Followed by the Robot, b) Tracking Error



(a)



(b)

Figure 8. a) Linear Velocity in Circular Trajectory, b) Angular Velocity in Circular Trajectory

Figure 9 illustrates the variation of controller gains along the path followed by the robot, showing the adjustments made during the online adaptation process. The kinematic controller based on neural networks effectively mitigates these oscillations, rendering them smoother. This improvement arises from the self-adjustment of plant gains, facilitating the rapid convergence of the current trajectory to the desired one. Consequently, the controller ensures that the linear speed control signal does not reach its maximum saturation value.

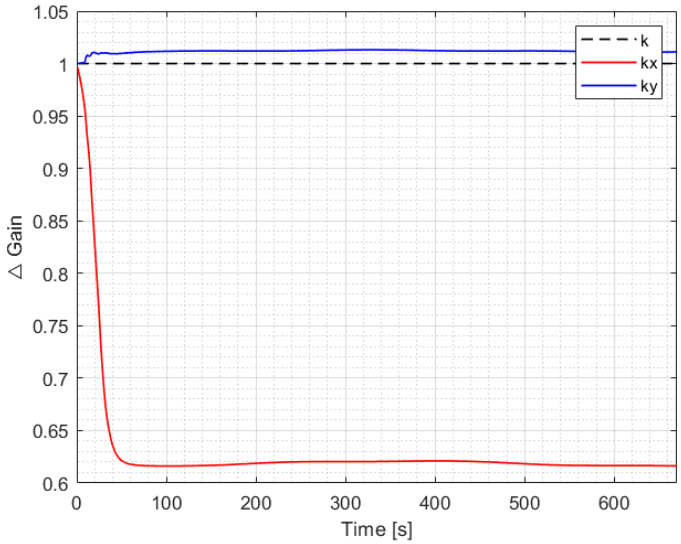


Figure 9. Variation of the Gains of the Kinematic Controller based on Neural Networks in the Circular Trajectory

Figure 10-a presents the performance of the mobile robot for a square trajectory with side of $3m$. Figure 10-b illustrates the tracking error for this path. Notably, the NN controller exhibits less oscillation and achieves quicker convergence compared to the P controller, particularly at the beginning of the simulation and in the corners. In contrast, the conventional P Controller (as seen in Figure 11) leads to substantial and high-amplitude oscillations in the control signals for linear and angular velocity. Additionally, the angular velocity of the mobile robot reaches its saturation point.

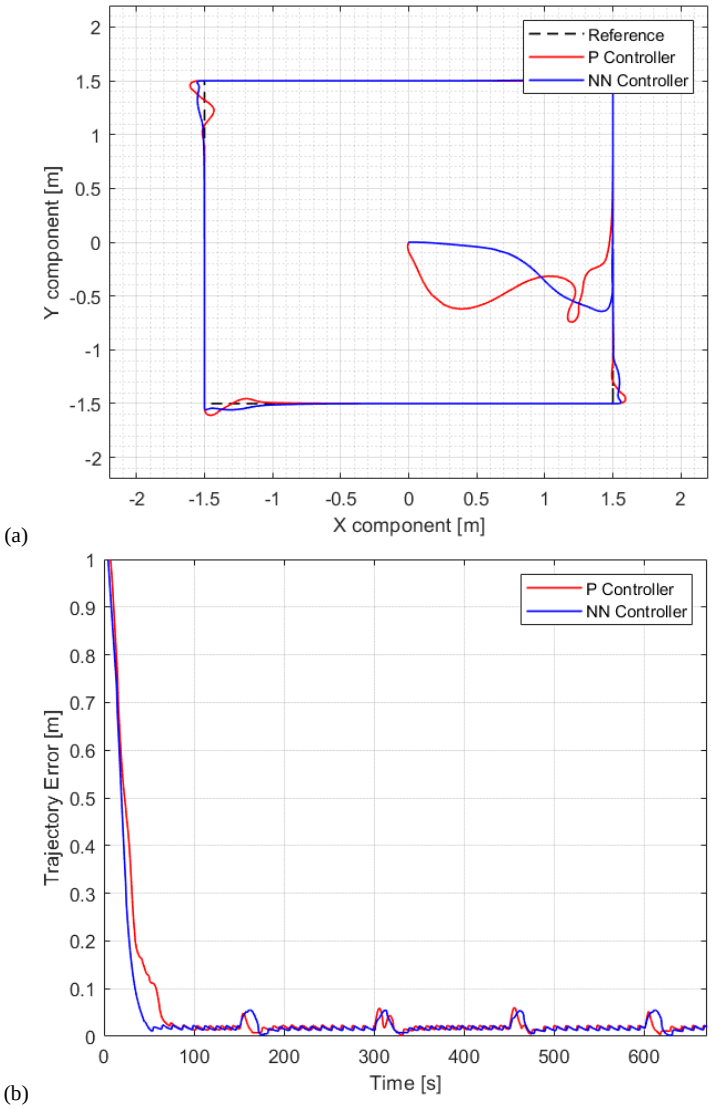


Figure 10. a) Square Trajectory Followed by the Robot, b) Tracking Error

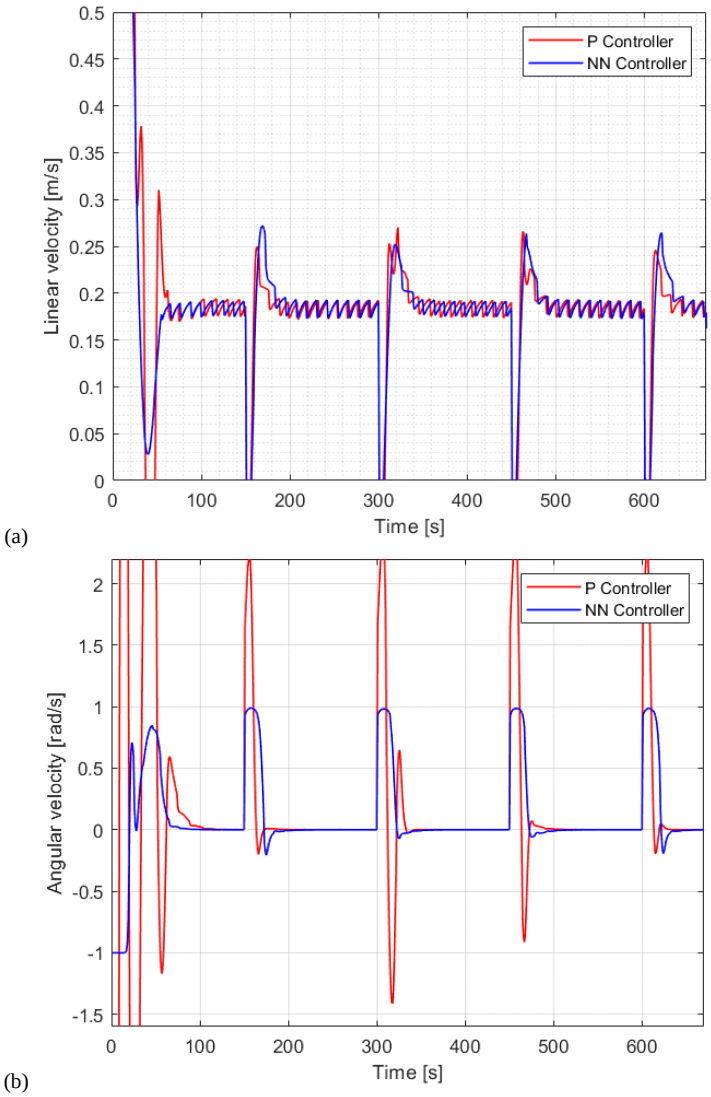


Figure 11. a) Linear Velocity in Circular Trajectory, b) Angular Velocity in Circular Trajectory

Figure 12 provides a visual representation of the dynamic changes in controller gains as the robot traverses its path, revealing the adaptive adjustments executed during the online adaptation process. The kinematic controller, which is founded on neural networks, excels at attenuating these oscillations, resulting in a smoother trajectory. This enhancement is attributed to the autonomous fine-tuning of plant gains, expediting the swift alignment of the current trajectory with the desired one.

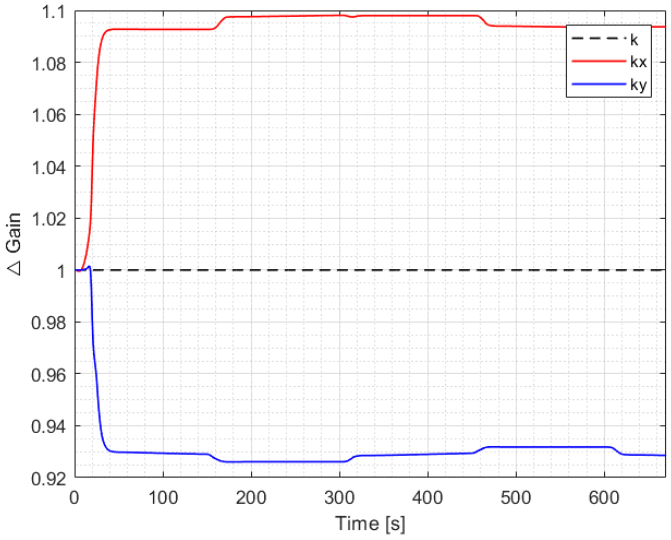
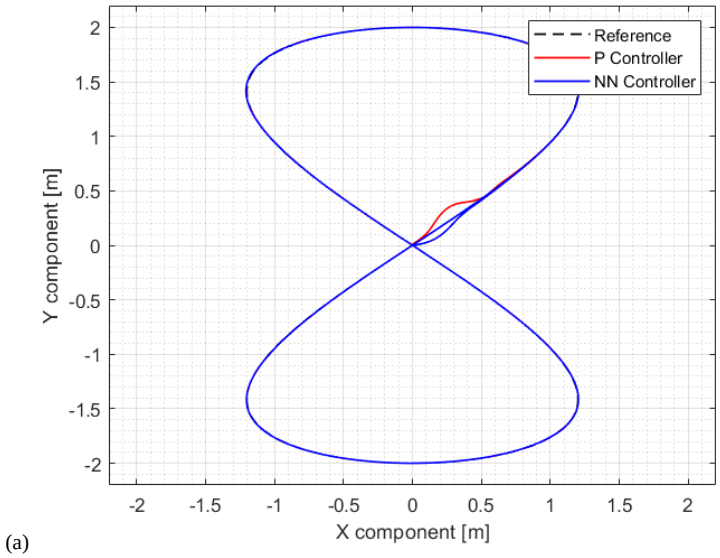
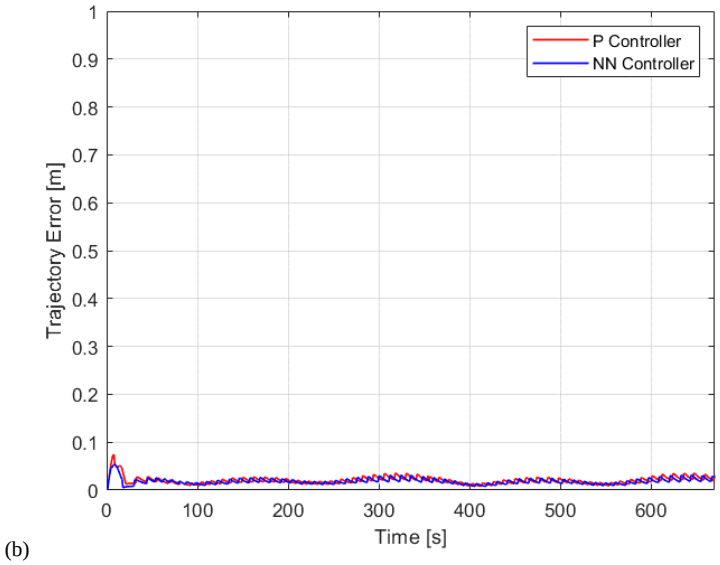


Figure 12. Variation of the Gains of the Kinematic Controller based on Neural Networks in the Square Trajectory

Figure 13-a illustrates the mobile robot's performance along a lemniscate trajectory, and Figure 13-b depicts the tracking error related to this path. It is worth noting that a significant reduction in angular velocity oscillations (see Figure 14) is observed when analyzing the NN controller. Nonetheless, it is crucial to emphasize that both controllers visibly adhere to the trajectory effectively.

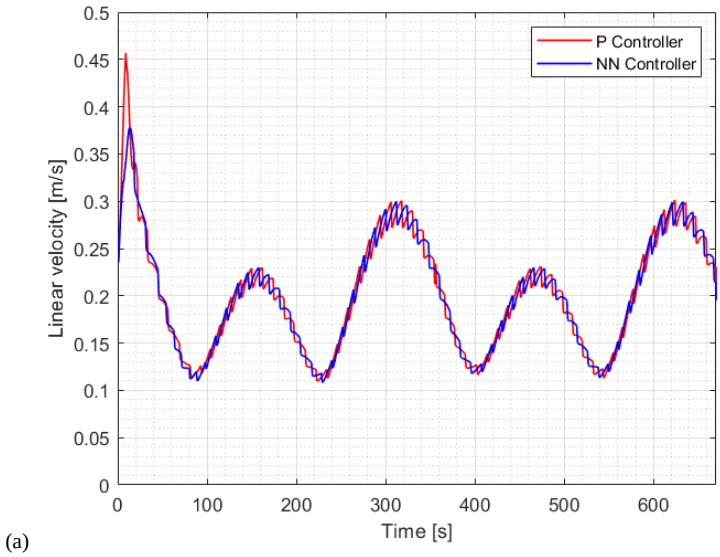


(a)

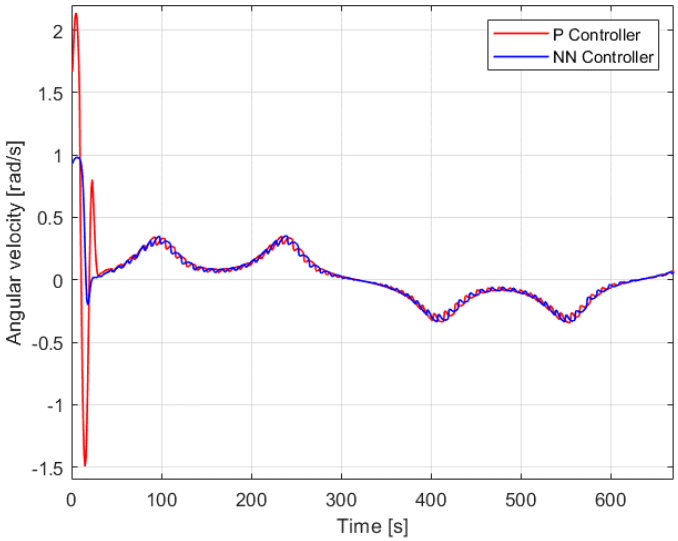


(b)

Figure 13. a) Lemniscate Trajectory Followed by the Robot, b) Tracking Error



(a)



(b)

Figure 14. a) Linear Velocity in Lemniscate Trajectory, b) Angular Velocity in Lemniscate Trajectory

Figure 15 offers a graphical depiction of the dynamic alterations in controller gains as the robot navigates its trajectory, showcasing the adaptive modifications implemented during the online adaptation process.

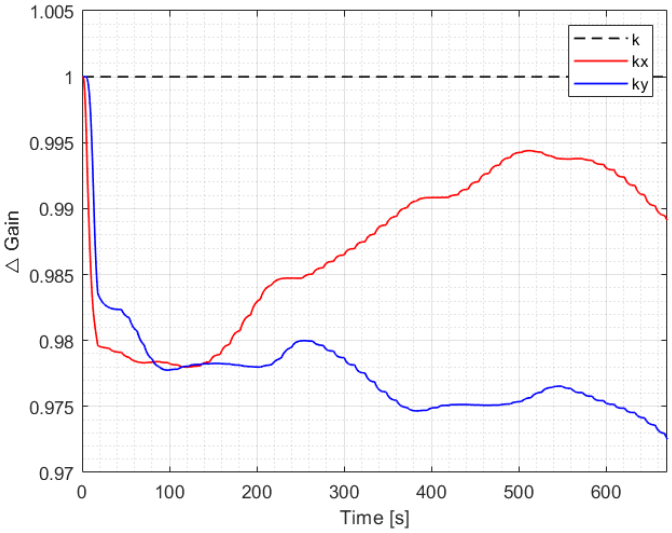


Figure 15. Variation of the Gains of the Kinematic Controller based on Neural Networks in the Lemniscate Trajectory

In general terms, both the conventional P controller and the NN controller follow the desired trajectory. The main distinction becomes apparent when the robot is far from the reference point, where our proposed method significantly reduces oscillations in order to approach it. These oscillations are particularly reflected in the linear and angular velocities, which, when of considerable magnitude, can potentially impact the performance of the actuators.

In the context of angular velocity, the results indicate a noticeable reduction in oscillations, leading to smoother signals that remain within the expected ranges. Consequently, this behavior facilitates a rapid convergence of the system towards the reference value. The significant decrease in oscillations in this control signal stands out as a notable advantage of employing this neural network-based control approach.

Regarding linear velocity, the observed vibrations are primarily attributed to the inherent nonlinearity of the system. Even though controllers can stabilize the system around an equilibrium point, the mechanical and dynamic nonlinearities within the robot's system can introduce minor oscillations, especially when operating near the boundaries of its workspace or under varying dynamic conditions. Another potential contributing factor could be axis coupling issues. In robots with multiple degrees of freedom, motion on one axis can influence the others. Although the controller independently stabilizes each axis, interactions between them can lead to unintended oscillations.

The results presented in Figure 16 highlight a significant quantitative performance enhancement achieved by the NN controller compared to the conventional P controller. Specifically, in terms of the IAE, ISE, and ISU performance indices for linear speed, the NN controller exhibits an improvement of over 10% compared to the conventional P-type controller. Notably, the most remarkable performance enhancement becomes evident in the angular velocity indices, with some

trajectories showing improvements exceeding 100%. This substantial improvement underscores the validity of our approach. While the ISE and IAE indices demonstrate a more modest quantitative enhancement, qualitatively, a considerable improvement in the following trajectory is observed.

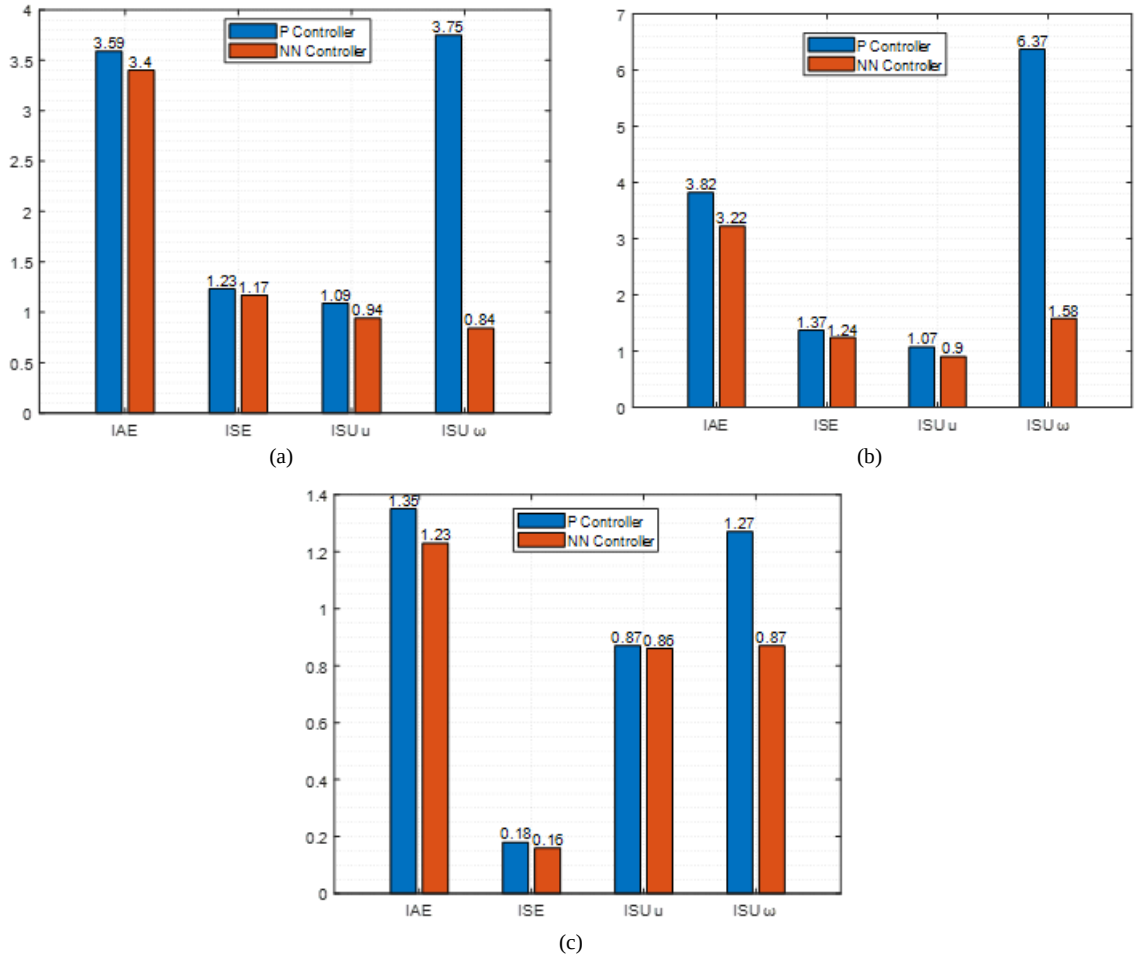


Figure 16. Performance Indexes for a) Circle Trajectory, b) Square Trajectory, c) Lemniscate Trajectory

6- Conclusions

This paper introduces the development of a neural network-based controller. Starting with the kinematic controller model of a mobile robot, we design a neural network. The neural network architecture resembles that of a multilayer perceptron model and is utilized for the design and self-tuning of the controller. The controller gains, representing the adjustable parameters, are fine-tuned using the backpropagation algorithm to minimize the trajectory tracking error of the mobile robot in online mode. An important difference from traditional approaches is that this designed controller eliminates the necessity for an offline learning phase.

In qualitative terms, the proposed controller demonstrates better control performance when compared to the commonly used conventional P controller for trajectory tracking, as extensively observed in the reviewed literature. Hence, it becomes evident that, particularly at the beginning of the simulations, the NN controller surpasses the conventional P controller, reducing the oscillations considerably until reaching the reference. This advantage stems from the reduced trajectory error and diminished oscillations in both linear and angular velocity control signals, which in turn prolongs the operational life of the actuators.

It is noteworthy that control schemes employed in these systems typically follow a cascade structure, where the inner loop governs the system dynamics while the outer loop manages the kinematics. In this context, our proposal has been dedicated to enhancing the robot's positional control and trajectory tracking performance. Subsequently, we intend to further concentrate on optimizing the dynamic aspects of the system.

6-1-Future Work

As future work, we plan to conduct real-world testing of the NN controller in order to thoroughly validate its performance. This will involve considering the real response of the system to dynamic changes and assessing the ability of the controller to adapt to such variations, as demonstrated in previous studies. Moreover, we underscore the significance of testing this controller on more intricate robotic systems, such as quadcopters or UAVs, to broaden its scope of application.

7- Declarations

7-1- Author Contributions

Conceptualization, L.M. and D.C.; methodology, L.M. and D.T.; software, D.T.; formal analysis, L.M. and D.P.; investigation, D.T.; writing—original draft preparation, L.M.; writing—review and editing, L.M. and D.P.; supervision, L.M. All authors have read and agreed to the published version of the manuscript.

7-2- Data Availability Statement

The data presented in this study are available on request from the corresponding author.

7-3- Funding

The authors received no financial support for the research, authorship, and/or publication of this article.

7-4- Institutional Review Board Statement

Not applicable.

7-5- Informed Consent Statement

Not applicable.

7-6- Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this manuscript. In addition, the ethical issues, including plagiarism, informed consent, misconduct, data fabrication and/or falsification, double publication and/or submission, and redundancies have been completely observed by the authors.

8- References

[1] Isasi Vinuela, P., & Galván León, I. M. (2004). Artificial neural networks: A practical approach. Pearson Educación, Madreid, Spain. (In Spanish).

[2] Chen, L., Chen, P., & Lin, Z. (2020). Artificial Intelligence in Education: A Review. IEEE Access, 8, 75264–75278. doi:10.1109/ACCESS.2020.2988510.

[3] Sarker, S., Jamal, L., Ahmed, S. F., & Irtisam, N. (2021). Robotics and artificial intelligence in healthcare during COVID-19 pandemic: A systematic review. Robotics and Autonomous Systems, 146. doi:10.1016/j.robot.2021.103902.

[4] Calderón Velasco, R. (2023). Inteligencia artificial en medicina. Diagnóstico, 62(1), e431. doi:10.33734/diagnostico.v62i1.431.

[5] Matas, C. R. (2018). Artificial intelligence, robotics and public administration models. Revista del CLAD Reforma y Democracia, 72, 5-42. (In Spanish).

[6] Tzafestas, S. G. (2013). Introduction to Mobile Robot Control. Elsevier, Amsterdam, Netherlands. doi:10.1016/C2013-0-01365-5.

[7] Byambasuren, B. E., Kim, D., Oyun-Erdene, M., Bold, C., & Yura, J. (2016). Inspection robot based mobile sensing and power line tracking for smart grid. Sensors (Switzerland), 16(2), 250. doi:10.3390/s16020250.

[8] Bengochea-Guevara, J. M., Conesa-Muñoz, J., Andújar, D., & Ribeiro, A. (2016). Merge fuzzy visual servoing and GPS-based planning to obtain a proper navigation behavior for a small crop-inspection robot. Sensors (Switzerland), 16(3), 276. doi:10.3390/s16030276.

[9] Klamt, T., Kamedula, M., Karaoguz, H., Kashiri, N., Laurenzi, A., Lenz, C., Leonardis, D., Mingo Hoffman, E., Muratore, L., Pavlichenko, D., Porcini, F., Rodriguez, D., Ren, Z., Schilling, F., Schwarz, M., Solazzi, M., Felsberg, M., Frisoli, A., Gustmann, M., … Holmquist, K. (2019). Flexible Disaster Response of Tomorrow: Final Presentation and Evaluation of the Centauro System. IEEE Robotics & Automation Magazine, 26(4), 59–72. doi:10.1109/MRA.2019.2941248.

[10] Hassan, N., & Saleem, A. (2022). Neural Network-Based Adaptive Controller for Trajectory Tracking of Wheeled Mobile Robots. IEEE Access, 10, 13582–13597. doi:10.1109/ACCESS.2022.3146970.

[11] Parsianmehr, S., Moosavian, S. A. A., & Fakharian, A. (2016). An experimental system identification modeling and robust control for NAO humanoid robot. 2016 4th International Conference on Robotics and Mechatronics (ICROM), Tehran, Iran. doi:10.1109/icrom.2016.7886793.

[12] Jang, J. S. R., Sun, C. T., & Mizutani, E. (2005). Neuro-Fuzzy and Soft Computing-A Computational Approach to Learning and Machine Intelligence. IEEE Transactions on Automatic Control, 42(10), 1482–1484. doi:10.1109/tac.1997.633847.

[13] Zalama, E., Paul, M., & Perán, J. R. (1998). Neural Network for the Behavioral Navigation of a Mobile Robot. IFAC Proceedings Volumes, 31(3), 93–98. doi:10.1016/s1474-6670(17)44067-5.

[14] Marichal, G. N., Toledo, J., Acosta, L., González, E. J., & Coll, G. (2007). A neuro-fuzzy method applied to the motors of a stereovision system. Engineering Applications of Artificial Intelligence, 20(7), 951–958. doi:10.1016/j.engappai.2006.12.010.

[15] Liu, Q., & Cong, Q. (2022). Kinematic and dynamic control model of wheeled mobile robot under internet of things and neural network. Journal of Supercomputing, 78(6), 8678–8707. doi:10.1007/s11227-021-04160-1.

[16] Asai, M., Chen, G., & Takami, I. (2019). Neural network trajectory tracking of tracked mobile robot. 16th International Multi-Conference on Systems, Signals and Devices, SSD 2019, 225–230. doi:10.1109/SSD.2019.8893152.

[17] Chen, Z., Liu, Y., He, W., Qiao, H., & Ji, H. (2021). Adaptive-Neural-Network-Based Trajectory Tracking Control for a Nonholonomic Wheeled Mobile Robot with Velocity Constraints. IEEE Transactions on Industrial Electronics, 68(6), 5057–5067. doi:10.1109/TIE.2020.2989711.

[18] Mohareri, O., Dhaouadi, R., & Rad, A. B. (2012). Indirect adaptive tracking control of a nonholonomic mobile robot via neural networks. Neurocomputing, 88, 54–66. doi:10.1016/j.neucom.2011.06.035.

[19] Yildirim, S., Savas, S., & Andruskiene, J. (2021). Controller Gain Tuning of a Nonholonomic Mobile Robot via Neural Network Predictor. 2021 25th International Conference Electronics. doi:10.1109/ieeeconf52705.2021.9467455.

[20] Mohamed, M., & Hamza, M. (2019). Design PID Neural Network Controller for Trajectory Tracking of Differential Drive Mobile Robot Based on PSO. Engineering and Technology Journal, 37(12A), 574–583. doi:10.30684/etj.37.12a.12.

[21] Gou, W., & Liu, Y. (2022). Trajectory tracking control of wheeled mobile robot based on improved LSTM-DDPG algorithm. Journal of Physics: Conference Series, 2303(1). doi:10.1088/1742-6596/2303/1/012069.

[22] Rossomando, F. G., Soria, C., & Carelli, R. (2011). Autonomous mobile robots navigation using RBF neural compensator. Control Engineering Practice, 19(3), 215–222. doi:10.1016/j.conengprac.2010.11.011.

[23] Nath, K., Bera, M. K., & Jagannathan, S. (2022). Concurrent Learning-Based Neuro-Adaptive Robust Tracking Control of Wheeled Mobile Robot: An Event-Triggered Design. IEEE Transactions on Artificial Intelligence, 4(6), 1514–1525. doi:10.1109/TAI.2022.3207133.

[24] Morales, L., Aguilar, J., Rosales, A., Chávez, D., & Leica, P. (2020). Modeling and control of nonlinear systems using an Adaptive LAMDA approach. Applied Soft Computing, 95, 106571. doi:10.1016/j.asoc.2020.106571.

[25] Botía Valderrama, J. F., & Botía Valderrama, D. J. L. (2018). On LAMDA clustering method based on typicality degree and intuitionistic fuzzy sets. Expert Systems with Applications, 107, 196–221. doi:10.1016/j.eswa.2018.04.022.

[26] Morales, L., Lozada, H., Aguilar, J., & Camargo, E. (2019). Applicability of LAMDA as classification model in the oil production. Artificial Intelligence Review, 53(3), 2207–2236. doi:10.1007/s10462-019-09731-6.

[27] Polit, M. (2006). An optimization method for the data space partition obtained by classification techniques for the monitoring of dynamic processes. Artificial Intelligence Research and Development, IOS Press, Amsterdam, Netherlands.

[28] Naveed, K., Khan, Z. H., & Hussain, A. (2014). Adaptive trajectory tracking of wheeled mobile robot with uncertain parameters. Studies in Computational Intelligence, 540, 237–262. doi:10.1007/978-981-4585-36-1_8.

[29] Benbouabdallah, K., & Qi-dan, Z. (2013). Genetic Fuzzy Logic Control Technique for a Mobile Robot Tracking a Moving Target. International Journal of Computer Science Issues, 10(1), 607–613.

[30] Mendez, E., Baltazar-Reyes, G., MacIas, I., Vargas-Martinez, A., De Jesus Lozoya-Santos, J., Ramirez-Mendoza, R., Morales-Menendez, R., & Molina, A. (2020). ANN based MRAC-PID controller implementation for a furuta pendulum system stabilization. Advances in Science, Technology and Engineering Systems, 5(3), 324–333. doi:10.25046/aj050342.

[31] Duong, H. Q., Nguyen, Q. H., Nguyen, D. T., & Van Nguyen, L. (2022). PSO based Hybrid PID-FLC Sugeno Control for Excitation System of Large Synchronous Motor. Emerging Science Journal, 6(2), 201-216. doi:10.28991/ESJ-2022-06-02-01.

[32] Siregar, S. P., & Wanto, A. (2017). Analysis of artificial neural network accuracy using backpropagation algorithm in predicting process (Forecasting). International Journal of Information System & Technology, 1(1), 34. doi:10.30645/ijistech.v1i1.4.

[33] Jepkoech, J., Mugo, D. M., Kenduiywo, B. K., & Too, E. C. (2021). The Effect of Adaptive Learning Rate on the Accuracy of Neural Networks. International Journal of Advanced Computer Science and Applications, 12(8), 736–751. doi:10.14569/IJACSA.2021.0120885.

[34] Shirong Liu, Huidi Zhang, Yang, S. X., & Jinshou Yu. (2004). Dynamic control of a mobile robot using an adaptive neurodynamics and sliding mode strategy. Fifth World Congress on Intelligent Control and Automation (IEEE Cat. No.04EX788). doi:10.1109/wcica.2004.1343669.

[35] Gracia, L., & Tornero, J. (2008). Kinematic control of wheeled mobile robots. Latin American Applied Research, 38(1), 7–16.

[36] Rossomando, F. G., Soria, C., & Carelli, R. (2010). Control of Mobile Robots with Dynamic Uncertainties using Radial Base Networks. Revista Iberoamericana de Automática e Informática Industrial RIAI, 7(4), 28–35. doi:10.1016/s1697-7912(10)70057-1.

[37] Morales, L., Herrera, M., Camacho, O., Leica, P., & Aguilar, J. (2021). LAMDA Control Approaches Applied to Trajectory Tracking for Mobile Robots. IEEE Access, 9, 37179–37195. doi:10.1109/access.2021.3062202.

[38] Eusebio, B.-C., & Ana Yaveni, A.-B. (2014). Visual control for the formation of unicycle-type mobile robots under the leader-follower scheme. Ingeniería, Investigación y Tecnología, 15(4), 593–602. doi:10.1016/s1405-7743(14)70657-2.

[39] Valencia, J. A., Montoya, A., & Rios, L. H. (2009). Kinematic model of a differential type mobile robot and navigation from odometric estimation. Scientia et Technica, 1(41). (In Spanish).

[40] Rohmer, E., Singh, S. P. N., & Freese, M. (2013). V-REP: A versatile and scalable robot simulation framework. 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan. doi:10.1109/iros.2013.6696520.

[41] Rodríguez-Mariano, A., Reynoso-Meza, G., Páramo-Calderón, D. E., Chávez-Conde, E., García-Alvarado, M. A., & Carrillo-Ahumada, J. (2015). Analysis of the Performance of Tuned Linear Controllers in Different Steady States of the Cholette Bioreactor using Multi-Criteria Decision Techniques. Revista mexicana de Ingeniería Química, 14(1), 167-204. (In Spanish).

[42] Proudfoot, C. G. (1987). Principles and practice of automatic process control. Carlos A. Smith and Armando B. Corripio (Book Review). Automatica, 23(3), 414. doi:10.1016/0005-1098(87)90018-5.

[43] Duarte-Mermoud, M. A., & Prieto, R. A. (2004). Performance index for quality response of dynamical systems. ISA Transactions, 43(1), 133–151. doi:10.1016/s0019-0578(07)60026-3.

[44] Salgado, M. E., Oyarzún, D. A., & Silva, E. I. (2007). H2 optimal ripple-free deadbeat controller design. Automatica, 43(11), 1961–1967. doi:10.1016/j.automatica.2007.03.014.