Available online at www.ijournalse.org

**Emerging Science Journal**

(ISSN: 2610-9182)

Vol. 8, No. 4, August, 2024

# Enhancing Control Systems with Neural Network-Based Intelligent Controllers

Kevin Puentes [1], Luis Morales [1], David F. Pozo-Espin [2*], Viviana Moya [3]

[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.*

[3] *Facultad de Ciencias Técnicas, Universidad Internacional Del Ecuador UIDE, Quito 170411, Ecuador.*

**Abstract**

The primary challenge faced by a neural controller in the dynamic model of a mobile robot lies in its ability to address the inherent complexity of the system dynamics. Given that mobile robots exhibit nonlinear movements and are subject to diverse environmental conditions, they contend with a challenging dynamic environment. The neural controllers must demonstrate the capability to continuously adapt and effectively learn to manage the variability present in the dynamic of the robot. This paper presents two intelligent controllers utilizing neural networks, showcasing their relevance in the field of robotics. The first controller, referred to as the neural PID (PIDN), integrates the traditional PID controller with a neural component. The second controller leverages the dynamic model of a differential robot to improve trajectory tracking, employing a parallel architecture that combines PID with neural networks (PID+NN). Our proposals adhere to a cascading structure, where the outer loop takes the lead in reducing position errors through a kinematic controller, while concurrently, the inner loop is employed to regulate linear and angular velocities through the proposed controllers. The controllers are validated in the CoppeliaSIM simulator, offering a realistic setting for evaluating the behavior of the chosen Pioneer 3-DX robot. To comprehensively assess controller performance, three strategies are examined: PIDN, PID+NN, and the conventional PID. Through a blend of qualitative and quantitative analyses, employing diverse performance metrics, the advantages of our proposed controllers become apparent.

**Keywords:**

Adaptive Neural Controller;
Mobile Robot;
Neural Networks;
PID;
Trajectory Tracking.

**Article History:**

| | | | |
|---|---|---|---|
| **Received:** | 25 | February | 2024 |
| **Revised:** | 06 | July | 2024 |
| **Accepted:** | 11 | July | 2024 |
| **Published:** | 01 | August | 2024 |

## 1- Introduction

Mobile robotics has become a thriving research domain with proven applicability in different fields [1] such as military [2], health services [3], exploration, agriculture [4], defense, and surveillance, or to fill in the many human tasks that are remote or remain unmanned [5]. The development and implementation of sophisticated controllers for trajectory control have been pivotal in this success; enabling successful navigation in the face of complex scenarios for a robot is by far the most critical factor. This enhancement in feedback systems enabled mobile robots to prove their worth and become very useful to this day in a wide range of operations [6].

Path tracking models need to be evaluated by the given trajectory of a pre-determined time period, where adjustments must be made to the path to prevent any undesirable deflections from the preplanned way [7]. Especially in autonomous operation, it is essential to prevent the plane or spacecraft from colliding with obstacles on the way and the fact that the vessel successfully reaches a planned destination without any accidents [8]. In the initial stages, only kinematics was

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

**DOI:** http://dx.doi.org/10.28991/ESJ-2024-08-04-01

considered in the trajectories control system. Hence, a departure from the predicted dynamics can cause an increase in unscheduled friction, inertia of mobility mechanisms, and deformations in the trajectory plane. This way, the robot's dynamics can change, affecting robot dynamics, trajectory accuracy, and mobile robots [9].

To address these dynamics, researchers' interests have shifted toward employing controllers based on dynamic models [10]. Nevertheless, the intricacy of core models depicting the higher-order system of state variables hampers its fine-tuning as a model design for the control systems. In response to this challenge, the use of artificial intelligence has emerged as a solution, facilitating the development of control algorithms based on an uncertain model [11]. These algorithms achieve precise trajectory tracking in non-holonomic mobile robots, marking a significant advancement in the field [12]. Recently, the incorporation of neural networks into control systems has presented a novel approach that adeptly navigates the modeling challenges posed by higher-order dynamic systems [13]. Traditional methods frequently need help acquiring dynamic models or formulating controllers for such complex systems. This approach is speedily getting better with time, as it is one of the gaining areas of research that showcases neural networks' excellent potential in mobile robotics regarding tracking trajectory [14].

### *1-1- Related Work*

The development of mobile robots has been expanded thanks to artificial intelligence (AI). These algorithms have increasingly become a solution to enhance system responsiveness and performance. Integrating AI into the different fields of robotics, especially mobile, has improved performance and resources, and it has also allowed research for more complex and adaptive behaviors, making AI-driven robots more versatile and capable of several tasks [15]. This section will present different studies, first focusing on research that uses controllers and different artificial intelligence techniques, then using models, and finally estimating uncertainties.

First, it is important to review research that improves the response of trajectory tracking with the use of intelligence algorithms in the controller as studied by Alouache & Wu (2018) [16], which explores visual trajectory tracking control for wheeled mobile robots. It emphasizes employing a Genetic Algorithms (GA) to increase the effectiveness of the PID controller and trajectory estimation. The primary aim is to enable the mobile robot to track a reference trajectory generated by another robot while remaining within the fixed camera's field of view. Simulation results conclusively showed that the proposed approach, utilizing the GA-PID controller, achieves noteworthy enhancements in control performance compared to a conventional PID controller. The research by Matich (2001) [17] proposes the creation of an online neural network controller to mitigate position errors arising from the constraints of subpar binary sensors, rendering the use of a linear controller impractical. The adoption of a neural network enables continuous learning. The suggested approach integrates a feedback structure with a PID-type control mechanism to determine errors. Additionally, the neural network weights are computed using gradient descent, and the error is propagated backwards to establish the weight update rule.

Another study in this line was presented by Asai et al. (2019) [18], where a control structure is employed, utilizing input and output patterns to adapt weights and achieve a controlled output. If the desired output is not reached, adjustments are made to the connection weights to align the obtained output with the desired result closely. This iterative process involves continuous learning. Notably, weight adjustments can only be made with prior knowledge of network patterns. By providing input pattern information and observing the output, correlations can be identified through unsupervised learning, as described in Haykin (2009) [19]. Additionally, in the work of Trujillo et al. (2023) [20], neural network-based controllers are implemented to guide the system to the desired reference trajectory by performing a gain adjustment by a backpropagation algorithm until the error between the current trajectory and the desired trajectory is approximately zero. This proposal has been devoted to improving the robot's positional control and the trajectory tracking performance of the kinematic model. In the study by Puentes & Morales (2023) [21], the cascade control strategy for trajectory tracking of a mobile robot employs a dual scheme, where two distinct control levels are integrated. First, the outer loop minimizes the position error using a model-based kinematic controller. On the other hand, the inner loop oversees controlling the linear and angular velocity of the robot, using a neural controller in combination with a PID to adapt to possible dynamic changes in the system. The neural networks applied to the dynamic system are subjected to a parameter identification process based on the dynamic model. This controller is updated at each sampling interval and collaborates with a PID controller to train the network online.

In the field of using artificial intelligence in models, whether kinematic or dynamic, the research carried out by In Hui & Ji-hong (2014) [22], proposes to integrate a neural network into the dynamic system of a ship, taking into account the principle of asymptotic stability in a closed system and applying Lyapunov's law to estimate nonlinear uncertainties. A trajectory tracking controller is formulated employing the sliding mode method. A Model Predictive Control (MPC) is introduced in Deng et al. (2014) [23], which utilizes a dual primal neural network. The system determines errors in the robot's kinematic model based on the inequality of linear variables. The control problem then shifts into a regulation problem, considering the dynamic effects on the robot, aiming to propagate errors towards the inputs. This is achieved by implementing MPC, wherein estimation and optimization occur at each sampling time to derive the desired variable

vector. The approach proposed in da Silva Lima et al. (2023) [24] uses an adaptive Radial Basis Function (RBF) neural network to model the uncertain robot dynamics within a nonlinear control law based on the Lyapunov principle. To reduce computational complexity, the neural network architecture requires only a single input neuron representing a combined error measure. In addition, the neural network weights are updated online by minimizing the combined error measure, allowing continuous improvement of the controller as the robot moves without requiring direct measurements of the disturbances to be compensated. All control parameters are based on numerical studies, assuming uncertainties in the inertia matrix and no prior knowledge of friction effects.

Additionally, another approach is to use neural networks to compensate for uncertainties. In the paper by Hoang et al. (2013) [25], a technique is presented for using a neural network to employ offset uncertainties stemming from the robot model. The trajectory tracking of a robot involves the derivation of its kinematic model and the application of Lyapunov's law. The external loop integrates the torsion method to manage the robot's dynamics without requiring explicit knowledge of its dynamic model. Likewise, in Dang et al. (2023) [26], the controller applied is backstepping, which provides stability and tracking. At the same time, the Radial Basis Function (RBF) neural networks increase the adaptability to uncertainties and improve the overall control quality of the 3WMR system by regulating the motion angle and performing the robot position control, minimizing uncertainty effects.

### *1-2- Main Contribution*

Neural controllers have emerged as a forefront solution in the realm of mobile robotics, presenting advanced methodologies to navigate through the intricate, nonlinear dynamics characteristic of such systems. These controllers harness the power of neural networks to direct robot behavior within both dynamic and unpredictable contexts. A significant challenge neural controllers confront in the dynamic models of mobile robots includes the modeling and management of nonlinear dynamics, as demonstrated by Lewis et al. (2012) [27]. Furthermore, these controllers exhibit the capacity to adapt in real-time to environmental alterations, thereby enhancing the robot's ability to navigate and perform tasks efficiently without the need for manual recalibration, as illustrated by Tai et al. (2016) [28]. Moreover, the inherent robustness of neural controllers allows mobile robots to effectively manage uncertainties present in sensor data and the incomplete information about their surroundings, a principle supported by Huang et al. (2007) [29].

Based on the previous analysis, this paper provides a complementary to Puentes & Morales (2023) [21] and comprehensive exploration of neural network-based controllers, with a specific focus on their role in guiding a mobile robot through trajectory tracking, especially in scenarios marked by uncertain dynamic models [30]. Our proposals stand out for their cost-effectiveness in computational terms and their simplicity in implementation, making them easily replicable across various programming software. Furthermore, they leverage the conventional PID as a foundational support to enhance the response, acknowledging its widespread use in the industry to date. Within this framework, we introduce two innovative intelligent controller concepts, both constructed on neural networks, which are proposed upon in the subsequent sections of Morales et al. [31]:

1. A Neural PID (PIDN) controller, functioning as an ongoing learning system. This controller continuously refines the weights of the proportional, integral, and derivative gains through the gradient descent method during each control action iteration, aiming to reduce the tracking error.

2. A novel parallel architecture of a PID controller, integrated with Neural Networks (PID + NN), comprises two distinct phases: a learning phase utilizing a neural identification approach to acquire the system model, and an application phase where the neural network functions as the controller, facilitating continuous learning.

Hence, three controllers were subjected to testing on the CoppeliaSim platform to showcase the navigation capabilities of the Pioneer 3-DX robot [32]: the two previously described controllers and the conventional PID controller. This robot, renowned for its two-wheeled differential traction system, has been thoroughly investigated within the realm of control system development. Equipped with a motion controller incorporating encoder feedback, it provides capabilities for monitoring through mapping/location techniques or remote operation [33].

Both proposals are implemented within the framework of the dynamic model of the robot, acknowledging that the environment in which the robot operates can significantly influence its performance. Factors such as variations in terrain or obstacles encountered can impact crucial aspects such as the center of mass of the robot or its ability to navigate smoothly over uneven surfaces. Consequently, these effects must be carefully accounted for and mitigated by the controller system.

For instance, changes in the terrain could lead to shifts in the center of mass of the robot, potentially affecting its stability and maneuverability. Similarly, surface irregularities may impede the movement of the robot, necessitating adjustments in its control inputs to ensure smooth traversal. By incorporating both proposals into the dynamic model, the controller can effectively compensate for these environmental variables, thereby enhancing the overall performance of the robot and adaptability.

In summary, integrating these proposals into the dynamic model enables the controller to proactively address environmental challenges, allowing the robot to navigate more effectively and efficiently across diverse terrains and conditions.

### *1-3- Outline*

The paper is structured as follows: Section 2 delves into the characteristics of the chosen mobile robot, examining both its kinematic and dynamic models along with the corresponding controllers. Section 3 comprehensively discusses the results analysis of the controllers based on neural networks, employing both qualitative and quantitative approaches through performance indicators (ISE and IAE). Finally, Section 4 presents the conclusions drawn from the study and outlines potential applications, and Section 5 outlines directions for future work.

## 2- Material and Methods

To begin with the development of the controllers, all the parameters and variables pertinent to the non-holonomic mobile robot are defined and illustrated in Figure 2. The system is characterized by several parameters, among them, d represents the distance between the wheels, while B denotes the midpoint between the wheels, $(x, y)$ is the reference point of the position with respect to the $XY$ plane, $a$ is the distance between the midpoint of the axis of the wheels and the reference point, $r$ is the radius of the wheels, $\omega$ and $v$ are the angular and linear velocity of robot respectively, and $\psi$ is the orientation angle.

Then, all the parameters and variables pertinent to the non-holonomic mobile robot are defined and illustrated in Figure 1. The system is characterized by several parameters, among them, d represents the distance between the wheels, while B denotes the midpoint between the wheels., $(x, y)$ is the reference point of the position with respect to the $XY$ plane, $a$ is the distance between the midpoint of the axis of the wheels and the reference point, $r$ is the radius of the wheels, $\omega$ and $v$ are the angular and linear velocity of robot respectively, and $\psi$ is the orientation angle.

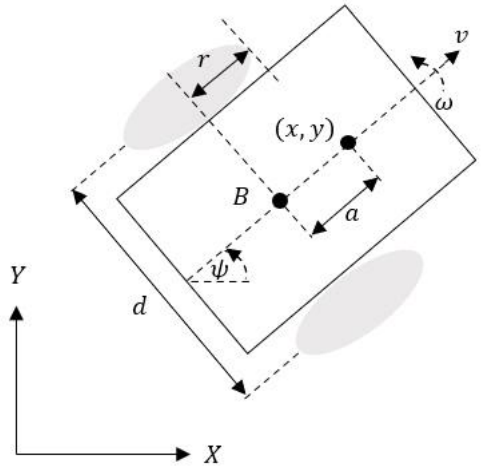



**Figure 1. Mobile robot parameters [21]**

### *2-1- Kinematic Controller*

The kinematic model plays an important role in advancing trajectory tracking within robotics. It provides a robust framework for calculating the position of the robot by considering both its linear and angular velocities (known as the inverse kinematic model). This capability facilitates the implementation of real-time control strategies, ensuring accurate adherence to desired trajectories. However, it's important to note that the kinematic model alone doesn't account for the dynamic forces and torques acting upon the robot's mechanism. To address this aspect, we delve into the dynamic model of the robot in the subsequent subsection.

Also, the kinematic model allows for the analysis of mechanical system motion, disregarding the influence of external forces acting upon them. It models the linear and angular speed of the robot based on the wheel speeds and geometric parameters of the robot [34]. For a given sampling time denoted as $T_s$ , the discretized kinematic model of the differential traction mobile robot can be mathematically represented as: Equation 1.

$$\begin{bmatrix} x(k+1) \\ y(k+1) \\ \psi(k+1) \end{bmatrix} = T_s \begin{bmatrix} cos\ \psi(k) & -a\ sin\ \psi(k) \\ sin\ \psi(k) & a\ cos\ \psi(k) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v(k) \\ \omega(k) \end{bmatrix} + \begin{bmatrix} x(k) \\ y(k) \\ \psi(k) \end{bmatrix} \quad (1)$$

The kinematic controller Equation 2 relies on the kinematic model, taking into account the coordinates $[x, y]^T$ of the points of interest. The control law, as presented in Zheng et al. (2024) [35], governs the controller's behavior.

$$\begin{bmatrix} v_{ref}^{c}(k) \\ \omega_{ref}^{c}(k) \end{bmatrix} = \begin{bmatrix} \frac{cos\,\psi(k)}{T_s} & \frac{sin\,\psi(k)}{T_s} \\ -\frac{1}{a}\frac{sin\,\psi(k)}{T_s} & \frac{1}{a}\frac{cos\,\psi(k)}{T_s} \end{bmatrix} \times \begin{bmatrix} x_{ref}(k+1) + l_x tanh\left(\frac{k_x}{l_x}e_x(k)\right) - x_{ref}(k) \\ y_{ref}(k+1) + l_y tanh\left(\frac{k_y}{l_y}e_y(k)\right) - y_{ref}(k) \end{bmatrix}, \quad (2)$$

where $\left[v_{ref}^{c}(k)\ \ \omega_{ref}^{c}(k)\right]^T$, is the output of the kinematic controller, $e_x(k) = x_{ref}(k) - x(k)$, and $e_y(k) = y_{ref}(k) - y(k)$ are the position errors for the $X$ and $Y$ axes respectively. $k_x, k_y$ are the controller gains, $l_x, l_y \in \mathbb{R}$ are saturation constants and $T_s$ is the capture time used in the case study. The function $tanh\,(\cdot)$ is added to saturate the control actions in case the position error is too large. In the stability analysis, the speed traction is considered $v_{ref}^{c}(k) = v(k)$ and $\omega_{ref}^{c}(k) = \omega(k)$. Substituting Equation 1 in Equation 2, the closed loop equation is:

$$\begin{bmatrix} e_x(k+1) \\ e_y(k+1) \end{bmatrix} + \begin{bmatrix} l_x tanh\left(\frac{k_x}{l_x}e_x(k)\right) \\ l_y tanh\left(\frac{k_y}{l_y}e_y(k)\right) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad (3)$$

Defining the output error vector as $\tilde{h}(k) = \left[e_x(k)\ \ e_y(k)\right]^T$, thus Equation 4:

$$\tilde{h}(k+1) = -\begin{bmatrix} l_x tanh\left(\frac{k_x}{l_x}e_x(k)\right) \\ l_y tanh\left(\frac{k_y}{l_y}e_y(k)\right) \end{bmatrix} \quad (4)$$

In Morales et al. (2021) [31] study the candidate of the Lyapunov function for the kinematic control law has been selected as $V(k) = 1/2\ \tilde{h}^T(k)\tilde{h}(k)$, being positive. The first derivative of the Lyapunov function is:

$$V(k+1) = \frac{1}{2}\tilde{h}^T(k)\tilde{h}(k+1) = -\tilde{x}(k)l_x tanh\left(\frac{k_x}{l_x}\tilde{x}(k)\right) - \tilde{x}(k)l_x tanh\left(\frac{k_y}{l_y}\tilde{y}(k)\right), \quad (5)$$

Equation 5 demonstrates the stability of the kinematic control for trajectory following if the parameters are configured as $k_x > 0, k_y > 0$, $l_x > 0$ and $l_y > 0$, then $\tilde{h}(k) \to 0$ for $k \to \infty$.

### *2-2- Dynamic Controller*

Mobile robots operate in dynamic environments that are inherently complex and unpredictable, a fact that significantly influences their design, control strategies, and operational capabilities. The nonlinear movements of these robots and the varied environmental conditions they encounter introduce layers of complexity that must be skillfully managed to achieve efficient and reliable performance. The primary element adding to the complexity is the variable dynamics; this is because mobile robots frequently exhibit nonlinear dynamics stemming from their modes of movement, such as wheels, legs, or propellers. For example, the relationship between the input commands to a robot (such as motor voltage) and its resulting movement (speed or direction) is rarely linear. This nonlinearity, resulting from factors like friction, slip, and the changing inertia of moving parts, complicates the prediction and control of robot motion. Another significant factor is the necessity for mobile robots to maneuver through environments subject to rapid and unforeseen changes. These variations can encompass moving obstacles (such as people, pets, or other robots), shifts in surface textures (from smooth to uneven terrains), and changes in environmental elements like lighting or weather conditions. Each of these aspects can influence the robot's mobility and the accuracy of its sensor data, further complicating the challenges of navigation and task performance.

Given the numerous parameters inherent in the dynamic model, which include both physical variables and acting forces, the identification process in this study involves deriving a first-order plus dead time (FOPDT) model [36]. This methodology offers a streamlined portrayal of the dynamics system. The process entails utilizing the reaction curves depicted in Figures 2 and 3, where a step-type signal is introduced to the system to assess the behavior of linear and angular velocity, respectively. Leveraging the CoppeliaSIM Platform provides a realistic environment and proves exceptionally valuable for simulating robotic systems at both kinematic and dynamic levels. Specifically, the platform features the Pioneer 3DX robot as proposed in this study.

$$\frac{X(s)}{U(s)} = \frac{Ke^{-t_0 s}}{\tau s + 1} \quad (6)$$

Adhering to the procedure detailed in Morales et al. (2021) [31], the model depicted in Equation 6 is obtained. This model offers a straightforward method for approximating the mathematical representation of the linear and angular velocity of the Pioneer 3-DX robot in the form of First Order Plus Time Delay (FOPTD). The subsequent section presents the results obtained for the linear velocity. $K = 1$, $\tau = 0.33435$ sec and $t_0 = 0{,}12975$ sec., and for angular velocity: $K = 1$, $\tau = 0.1527$ sec and $t_0 = 0{,}0833$ sec.

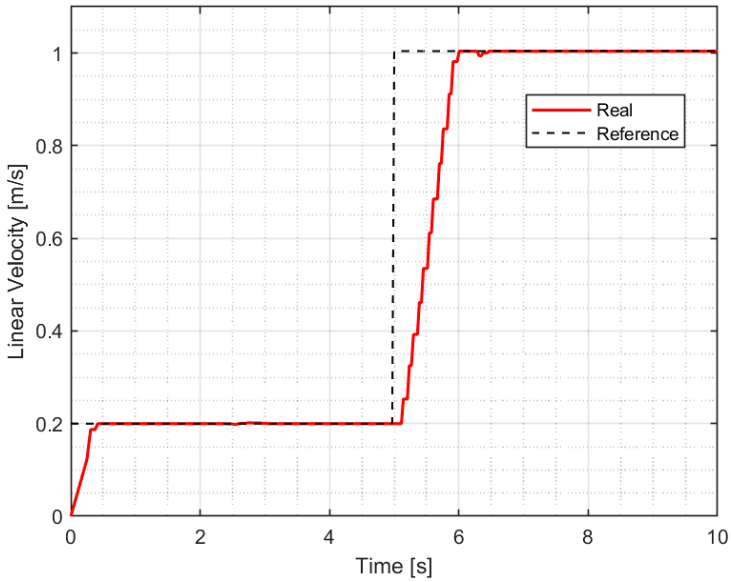



**Figure 2. Linear velocity response to a step input [21]**

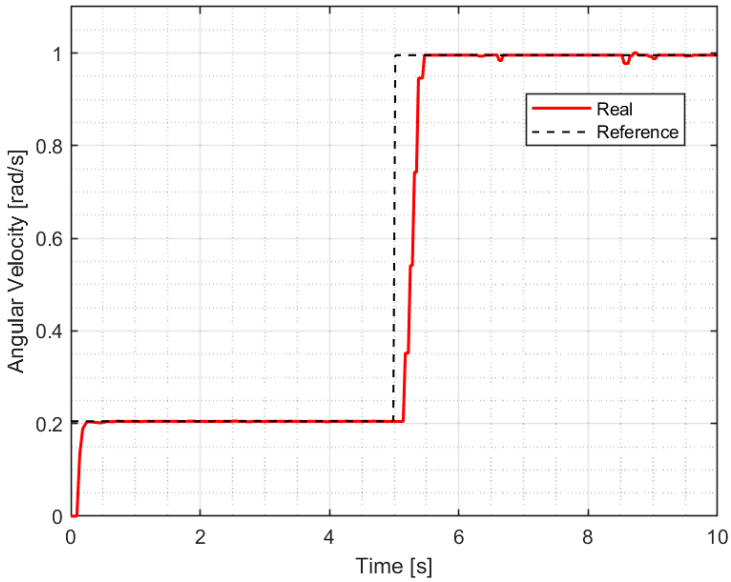



**Figure 3. Angular velocity response to a step input [21]**

Drawing from an approximate dynamic model of the robot, the methodology suggests the development of two controllers to fulfill the dual velocity requisites.

To conduct a comprehensive evaluation of the PID+NN performance, it is advisable to compare its response with that of a neural PID controller. To facilitate a meaningful comparison with our proposed controller, we first introduce the neural PID controller, offering insights into its design principles. By highlighting the common neural characteristics of both controllers, this approach ensures a fair and accurate comparison.

### *2-3- Neural PID Controller (PIDN)*

The proposed controller relies on a Neural PID adaptive controller, incorporating the parameters $K_P$, $K_I$ and $K_D$ which are tuned through the descending gradient method [37]. The main objective is to adjust the controller parameters in each iteration [38, 39] to achieve system stability. Initially, conventional PID controller weights Equation 7 are utilized to establish a stable starting point.

$$o_\lambda(k) = o_\lambda(k-1) + K_{P\lambda}(e_\lambda(k) - e_\lambda(k-1)] + K_{I\lambda}e_\lambda(k)T_s + \frac{K_{D\lambda}}{T_s}[e_\lambda(k) - 2e_\lambda(k-1) + e_\lambda(k-2)] \quad (7)$$

where $e_\lambda(k) = \lambda^c_{ref}(k) - \lambda(k)$, λ represents the velocity linear v and angular ω, and $o_\lambda$ is the control action.

The development of the neural network is derived from the PID control law. Each neuron is governed by an activation Equation 8, as illustrated in the neural network depicted in Figure 4. This function conveys information generated by the

linear combination of weights and inputs, essentially providing a mechanism for transmitting information through the output connections [40].

The $tan\,(.)$ is the activation function that saturates the move of the motors in both directions clockwise and counterclockwise.

$$f\big(o_\lambda(k)\big) = \delta \tanh \frac{o_\lambda(k)}{\delta} = \lambda^d_{ref}(k) \tag{8}$$

where $\delta$ allows values other than 1.

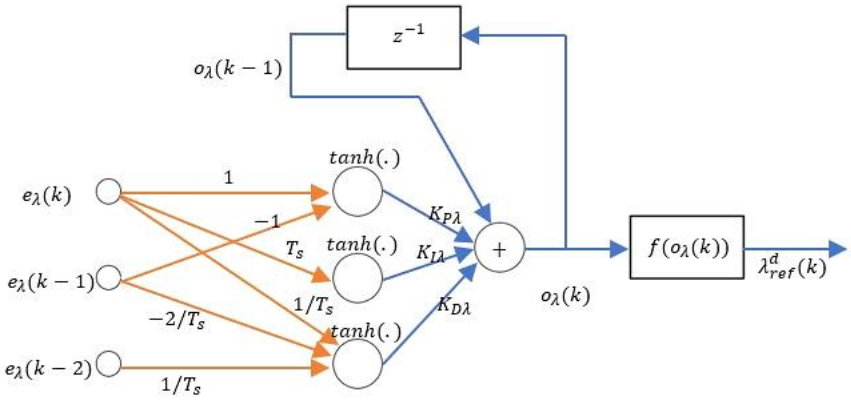



**Figure 4. Neural network in Neural PID controller**

During the weight adjustment process, the error is retroactively propagated. In more straightforward terms, the objective is to minimize the error, as specified by Equation 9, with regard to the variable being adjusted.

$$E_\lambda(k) = \frac{1}{2}(\lambda^c{}_{ref}(k) - \lambda(k))^2 \tag{9}$$

Hence, the application of the chain rule is employed to calculate the derivatives that aid in this computation, as delineated in Equation 10.

$$\frac{\partial E_\lambda(k)}{\partial K_{P\lambda}} = \frac{\partial E_\lambda(k)}{\partial \lambda(k)} \frac{\partial \lambda(k)}{\partial \lambda^d_{ref}(k)} \frac{\partial \lambda^d_{ref}(k)}{\partial o_\lambda(k)} \frac{\partial o_\lambda(k)}{\partial K_{P\lambda}}$$

$$\frac{\partial E_\lambda(k)}{\partial K_{I\lambda}} = \frac{\partial E_\lambda(k)}{\partial \lambda(k)} \frac{\partial \lambda(k)}{\partial \lambda^d_{ref}(k)} \frac{\partial \lambda^d_{ref}(k)}{\partial o_\lambda(k)} \frac{\partial o_\lambda(k)}{\partial K_{I\lambda}} \tag{10}$$

$$\frac{\partial E_\lambda(k)}{\partial K_{D\lambda}} = \frac{\partial E_\lambda(k)}{\partial \lambda(k)} \frac{\partial \lambda(k)}{\partial \lambda^d_{ref}(k)} \frac{\partial \lambda^d_{ref}(k)}{\partial o_\lambda(k)} \frac{\partial o_\lambda(k)}{\partial K_{D\lambda}}$$

Deriving the error with respect to the output of the plant, is obtained Equation 11:

$$\frac{\partial E(k)}{\partial \lambda(k)} = -e_\lambda(k) \tag{11}$$

The output with respect to the control variable $\lambda^d_{ref}(k)$ Equation 12, where $h$ is a change in the output:

$$\frac{\partial \lambda(k)}{\partial \lambda^d_{ref}(k)} = \frac{\lambda(k-1)-\lambda(k-2)}{\lambda^d_{ref}(k-2)-\lambda^d_{ref}(k-3)} \tag{12}$$

From the activation Equation 8, is obtained:

$$\frac{\partial \lambda^d_{ref}(k)}{\partial o_\lambda(k)} = [1 - f^2(o_\lambda(k))], \tag{13}$$

where $f(x) = \tanh(x)$ and $f'(x) = 1 - \tanh^2(x)$:

The controller output with respect to the weight $K_{P\lambda}$, is obtained Equation 14:

$$\frac{\partial o_\lambda(k)}{\partial K_{P\lambda}(k)} = f(e_\lambda(k) - e_\lambda(k-1)), \tag{14}$$

The controller output with respect to the weight $K_{I\lambda}$, is obtained Equation 15:

$$\frac{\partial o_\lambda(k)}{\partial K_{I\lambda}(k)} = f(e_\lambda(k)T_s), \tag{15}$$

The controller output with respect to the weight $K_{I\lambda}$, is obtained Equation 15:

$$\frac{\partial o_\lambda(k)}{\partial K_{D\lambda}(k)} = f\left(\frac{e_\lambda(k)-2e_\lambda(k-1)+e_\lambda(k-2)}{T_s}\right), \tag{16}$$

To obtain the update of the gains at instant $k+1$ Equation 17.

$$K_\theta(k+1) = K_\theta(k) - \eta \frac{\partial e_\lambda(k)}{\partial K_\theta(k)}, \tag{17}$$

where $\eta$ is the learning rate and $\theta$ are: $P\lambda$, $I\lambda$ and $D\lambda$.

Equation 18 shows the output of the neural controller.

$$o_\lambda(k) = o_\lambda(k-1) + K_{P\lambda}(k+1)f[(e_\lambda(k) - e_\lambda(k-1)] + K_{I\lambda}(k+1)f(e_\lambda(k)T_s) + K_{D\lambda}(k+1)f\left(\frac{e_\lambda(k)-2e_\lambda(k-1)+e_\lambda(k-2)}{T_s}\right), \tag{18}$$

Figure 5 shows the complete scheme of the system for the position control of a mobile robot where $\lambda$ has been replaced by the linear velocity $v$ and the angular velocity $\omega$. The linear velocity error is given by $e_v(k) = v^c_{ref}(k) - v(k)$, where $v(k)$ is the linear velocity output and $v^c_{ref}(k)$ is the output of the linear velocity of the kinematic controller. The inputs for Neural Controller 1 are linear velocity error, its two previous states, the differential of two previous states of output linear velocity, and the linear velocity difference of dynamic model $v^d_{ref}(k-2) - v^d_{ref}(k-3)$. The procedure described above is applied in a similar way to control the angular velocity $\omega(k)$, considering the angular velocity error $e_\omega(k) = \omega^c_{ref}(k) - \omega(k)$, where $\omega^c_{ref}(k)$ is the kinematic control action, $\omega(k)$ is the angular velocity output of the system and $\omega^d_{ref}(k)$ is the output the angular velocity of the dynamic model. The motors velocities are given by Equation 19 based on the values of the control actions $[v^d_{ref}(k)\ \omega^d_{ref}(k)]^T$, where left motor is $\Omega_L$ and right motor is $\Omega_R$ [40].

$$\begin{aligned}\Omega_L &= \frac{2v^d_{ref}(k) - d\omega^d_{ref}(k)}{2r} \\ \Omega_R &= \frac{2v^d_{ref}(k) + d\omega^d_{ref}(k)}{2r}\end{aligned} \tag{19}$$

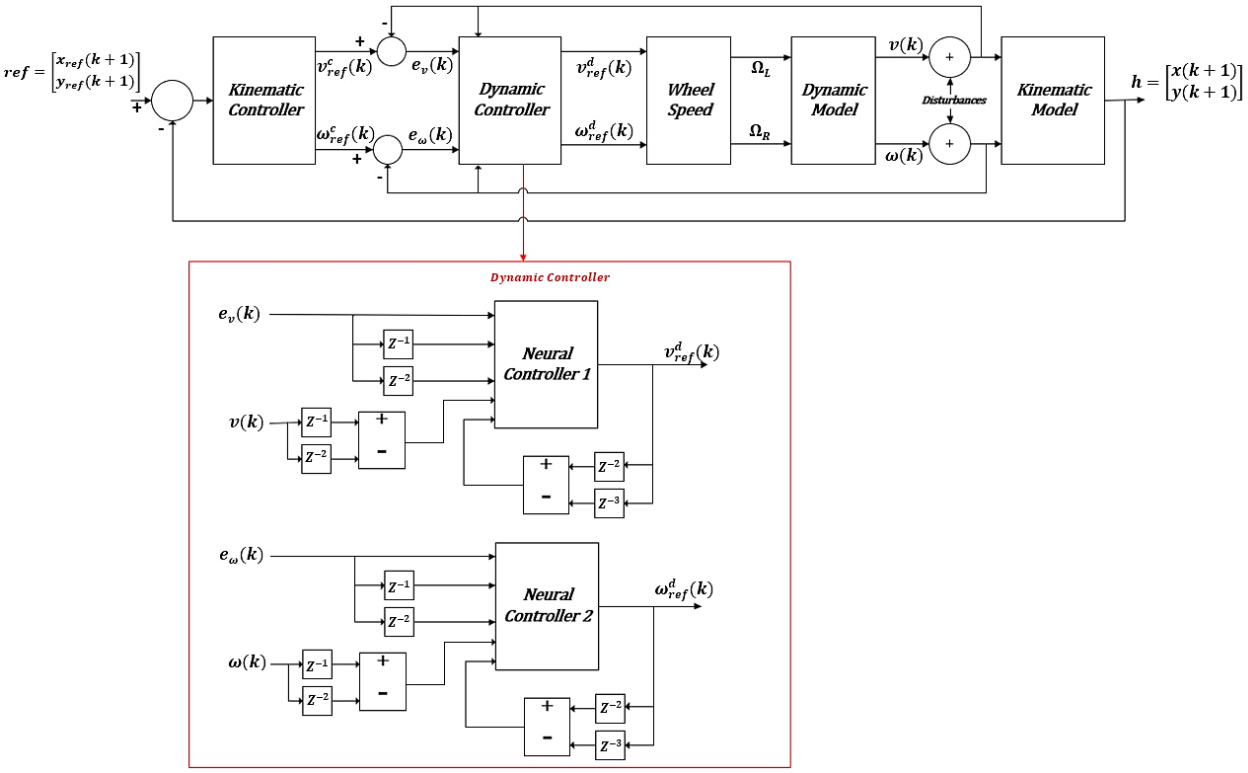



**Figure 5. Neural PID control scheme for trajectory tracking of a mobile robot using neural networks**

The proposed control structure is based on a cascade-type architecture, where an internal loop is employed to adaptively control linear and angular velocities. This is achieved through the utilization of a dynamic control block and behavior predictions derived from the robot's dynamic model. Meanwhile, the external control loop follows a classical architecture aimed at minimizing position errors through kinematic control. The primary concept underlying this

architecture is its emphasis on early correction of linear and angular velocities to robustly handle disturbances that could significantly impact the robot's position thereafter.

Within the dynamic controller block, the proposed PIDN controller is situated, comprising two neural networks. These neural networks are tasked with adjusting the Kp, Ki, and Kd parameters for both linear and angular velocity PID controllers.

### *2-4- PID Controller combined with Neural Network (PID+NN)*

The second proposition integrates a PID controller with a parallel neural network [21]. This strategy seeks to minimize errors by fine-tuning the control output via the neural network, leveraging the system's dynamic model, which follows a first order system model without considering the delay due to its low value Equation 20.

$$\frac{\lambda}{o_\lambda} = \frac{K_\lambda / z}{1 - \beta_\lambda / z} \tag{20}$$

where $\lambda$ represents the velocities: linear $v(k)$ and angular $\omega(k)$. $K_\lambda$ represents the gain of the system and $\beta_\lambda$ represents stability time.

Discretizing the first order model, is obtained Equation 21.

$$\lambda(k) = \beta_\lambda \lambda(k-1) + K_\lambda o_\lambda(k-1) \tag{21}$$

The control law Equation 22 is obtained by rearranging the terms.

$$o_\lambda(k) = \frac{1}{K_\lambda}\lambda(k+1) - \frac{\beta_\lambda}{K_\lambda}\lambda(k), \tag{22}$$

making a change of variable where: $\frac{1}{K_\lambda} = b$ and $\frac{\beta_\lambda}{K_\lambda} = c$, is obtained Equation 23.

$$o_\lambda(k) = b_\lambda(k+1) - c_\lambda(k) \tag{23}$$

The variables $b$ and $c$ are the weights of the neural network to be modified. Figure 6 illustrates the schematic representation of the neural network in this approach, where $\tanh(.)$ denotes the activation function. This choice is equally applicable for the reasons expounded in the preceding case (8), yielding the output from the neural network Equation 24:

$$f(o_\lambda(k)) = \delta \tanh \frac{o_\lambda(k)}{\delta} = \lambda^N_{ref}(k) \tag{24}$$

where the error to be minimized is given by (25).

$$E_\lambda(k) = \frac{1}{2}(\lambda^d_{ref}(k) - \lambda^N_{ref}(k))^2 \tag{25}$$

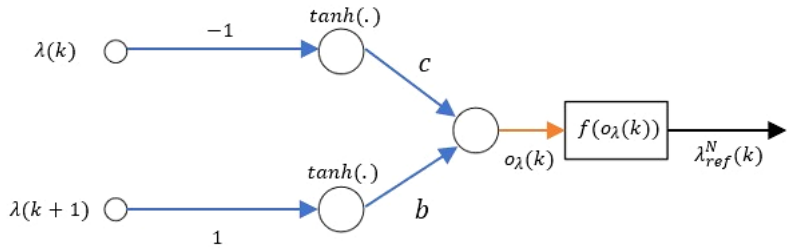



**Figure 6. PID+NN neural network [21]**

And applying the chain rule, the partial derivatives are obtained Equation 26.

$$\frac{\partial E_\lambda(k)}{\partial b} = \frac{\partial E_\lambda(k)}{\partial \lambda^N_{ref}(k)} \frac{\partial \lambda^N_{ref}(k)}{\partial o_\lambda(k)} \frac{\partial o_\lambda(k)}{\partial b}$$

$$\frac{\partial E_\lambda(k)}{\partial c} = \frac{\partial E_\lambda(k)}{\partial \lambda^N_{ref}(k)} \frac{\partial \lambda^N_{ref}(k)}{\partial o_\lambda(k)} \frac{\partial o_\lambda(k)}{\partial c} \tag{26}$$

The derivative of the error with respect to the output of the system, is obtained Equation 27:

$$\frac{\partial E_\lambda(k)}{\partial \lambda} = -e_\lambda(k) \tag{27}$$

From the activation function Equation 28, is obtained:

$$\frac{\partial \lambda_{ref}^{N}(k)}{\partial o_{\lambda}(k)} = \left[1 - f^{2}\left(o_{\lambda}(k)\right)\right] \tag{28}$$

where $f(x) = \tanh(x)$ and $f'(x) = 1 - \tanh^2(x)$:

The output of the controller with respect to the weight is $b$ Equation 29:

$$\frac{\partial o_{\lambda}(k)}{\partial b} = \lambda(k+1) \tag{29}$$

The output of the controller with respect to the weight is c Equation 30:

$$\frac{\partial o_{\lambda}(k)}{\partial c} = \lambda(k) \tag{30}$$

Obtaining neural network parameters at instant $k+1$, where $\eta$ is the learning rate and $\theta$ are $b, c$:

$$\theta(k+1) = \theta(k) - \eta \frac{\partial e_{\lambda}(k)}{\partial \theta(k)} \tag{31}$$

The Equation 32 corresponding to the neural network is obtained finally.

$$o_{\lambda}(k) = bf(\lambda(k+1)) - cf(\lambda(k)) \tag{32}$$

Figure 7 shows the proposed scheme, featuring the learning block of linear velocity (Neural Identifier 1). This block takes as inputs the measured variable $v$ and its previous state, utilizing them to minimize the error $e_v(k) = v_{ref}^{d}(k-1) - v_{ref}^{N}(k-2)$, where $v_{ref}^{N}$ is the output of the identifier, $v_{ref}^{d}$ is the sum of the Neural control action $v_{NN}^{C}$ and output of the PID controller $v_{PID}^{C}$.

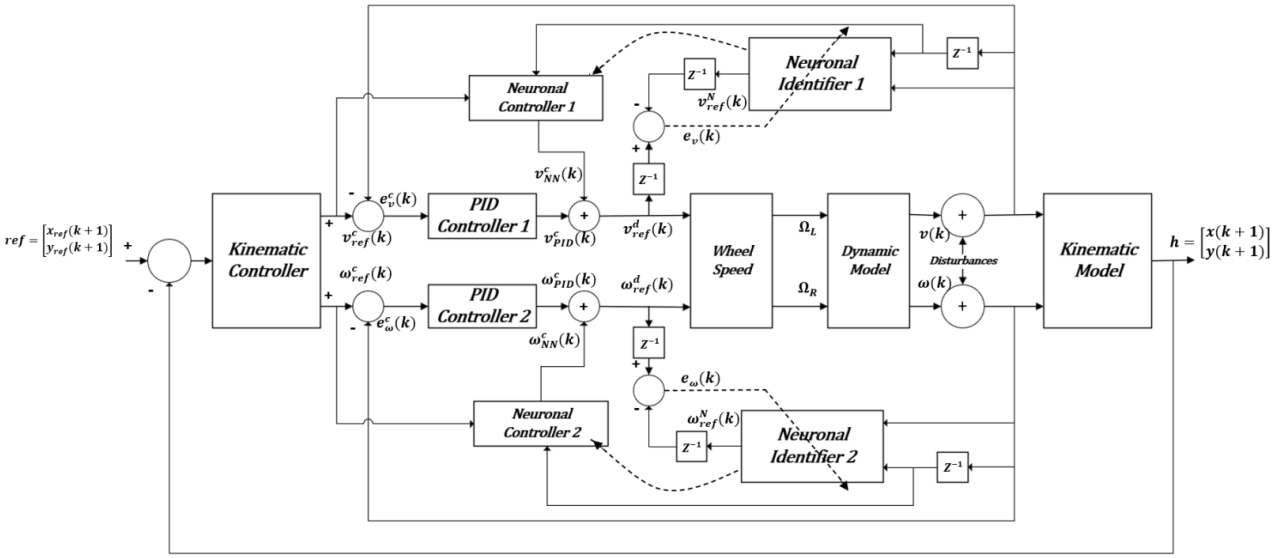



**Figure 7. PID+NN control scheme for trajectory tracking of a mobile robot using neural networks [21]**

The Neural Identifier and the Neural Controller share the same neural network architecture as depicted in Figure 7. The primary distinction lies in the application block (Neural Controller 1), where the output $v_{NN}^{C} = o_{\lambda}(k) = b_{updated} f(\lambda(k+1)) - c_{updated} f(\lambda(k))$ is obtained, This output is based on inputs including the linear velocity of the kinematic controller $v_{ref}^{C}$ , the linear velocity $v$ and the identified parameters.

The procedure described above is applied in a similar way to control the angular velocity $\omega$, considering the minimization of the error $e_{\omega}(k) = \omega_{ref}^{d}(k-1) - \omega_{ref}^{N}(k-2)$, where $\omega_{ref}^{N}$ is the output of the neural network identifier and $\omega_{ref}^{d}$ is the control action calculated by dynamic control, $\omega_{NN}^{C}$ is the angular control action of the network and $\omega_{PID}^{C}$ is the angular control action of the PID controller. The control variables are the motor velocities (19), so it is necessary to calculate the speed of the left wheel $\Omega_{\mathrm{L}}$ and right wheel $\Omega_{\mathrm{R}}$, based on the values of the control actions $[v_{ref}^{d}(k) \quad \omega_{ref}^{d}(k)]^{T}$.

The proposed controllers integrate neural networks with traditional PID structures through dynamic adjustment of their parameters using real-time data from the system. Neural networks learn from the inputs and outputs of the system,

allowing for adaptive responses, especially beneficial for non-linear systems. This fusion combines the flexibility of neural network with the stability of PID, offering improved accuracy, adaptability to changing conditions and reduced manual adjustment. In robotics, these controllers excel at handling complex dynamics, ensuring smoother operation in various environments. They facilitate continuous learning, improving performance over time, making them indispensable for robust and efficient robotic control.

Due to the Neural Identifier and the Neural Controller in Figure 7 sharing the same neural network architecture, the parameters $b$ and $c$ of the neural identifier are continually adjusted based on the dynamic model response. Meanwhile, in the neural controller, these previously updated parameters are utilized to generate a corrective output $v_{NN}^{c}$ that is added to the output of the classical PID controller $v_{PID}^{c}$. The primary concept behind this architecture is to have an adaptive correction running parallel to the classical PID controller, based on changes in the dynamic model response of the system. This setup aims to maintain the response characteristics of the PID controller in its initial configuration, even in the face of disturbances in the linear and angular velocities of the system.

Finally, as a summary, Figure 8 illustrates the methodology employed for the proposed controllers. Following trajectory generation, the kinematic controller ensures the maintenance of the desired position, while the neural network-based controller mitigates the impact of disturbances arising from dynamic changes in the robot.

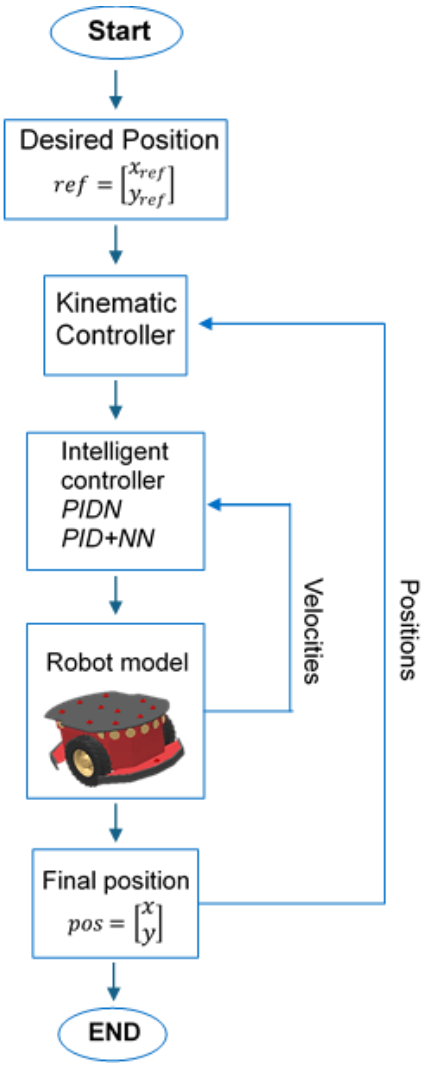



**Figure 8. Flowchart of the proposed methodology**

## 3- Simulations and Results

The Pioneer 3DX robot, employed to validate the controller, operates within the CoppeliaSIM Platform. This platform enables the simulation of robotic systems, incorporating considerations for their kinematics, dynamics, and environment. The software's versatility is amplified by the availability of plugins that facilitate connections with other computational tools, such as Matlab, housing the programmed algorithms. The connection between CoppeliaSIM and the Matlab environment for controlling the Pioneer 3DX robot using the proposed Intelligent controllers (PIDN and PID+NN) is illustrated in Figure 9.

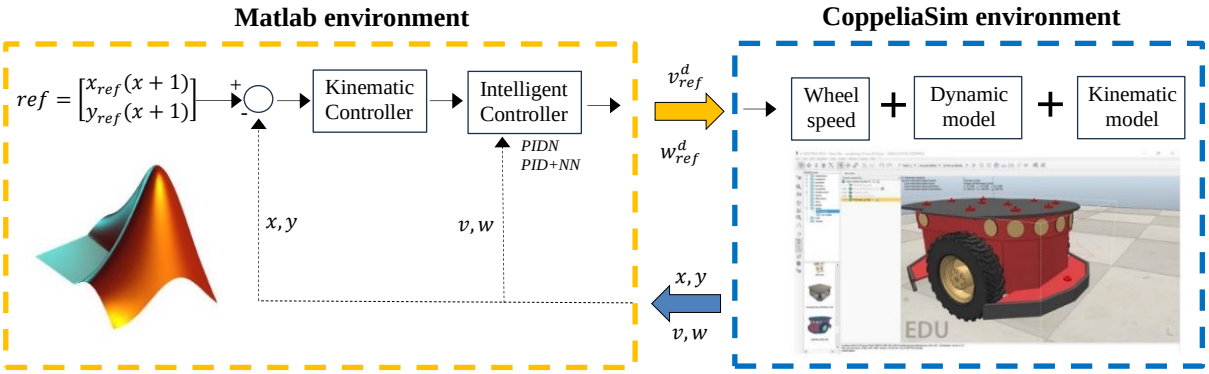



**Figure 9. Connection between CoppeliaSIM and MatLab**

The number of the samples Equation 33 of the trajectory is computed considering the duration time of the simulation and the sample time.

$$m = \frac{t}{T_S}, \tag{33}$$

Below are the square Equation 34 and circular Equation 35 trajectory, where $L$ is the side of the square, and $J$ is the radius of the circle. Creating the vector of positions $x_{ref}(k)$ and $y_{ref}(k)$, and $k$ is the iteration number.

$$\begin{cases} x_{ref}(k) = \frac{L}{2} \forall km \in [0, km]; \left(\frac{L}{2} - 4kmL\right) \forall\, km \in [km, 2km] - \frac{L}{2} \forall\, km \in [2km, 3km]; \left(-\frac{L}{2} + 4kmL\right) \forall\, km \in [3km, 4km], \\ y_{ref}(k) = \left(-\frac{L}{2} + 4kmL\right) \forall\, km \in [0, km]; \frac{L}{2} \forall\, k \in [km, 2km]; \left(\frac{L}{2} - 4kmL\right) \forall\, km \in [2km, 3km]; -\frac{L}{2} \forall\, km \in [3km, 4km], \end{cases} \tag{34}$$

$$\begin{cases} x_{ref}(k) = J\ cos(2\pi km) \\ y_{ref}(k) = J\ sin(2\pi km) \end{cases} \tag{35}$$

In Section 2, a detailed description of the design process of the proposed PID+NNN and PIDN controllers is given, laying the foundation for our subsequent comparative analysis. This section involves subjecting these controllers to rigorous testing through their application to the Pioneer 3-DX mobile robot. The implementation and programming of the controller are executed in Matlab. To validate its performance, experiments were performed in CoppeliaSIM, using two different types of trajectories. The kinematic controller was tuned heuristically, with the physical parameters of the robot set as $a = 0.12$, representing the distance between the robot reference point and the center point of the wheel axis, $k_x, k_y = 0.07$, are gains that allow minimizing the position error, the constants $l_x, l_y = 0.1$, which allow the saturation of the linear and angular velocity of the robot, and $T_s = 0.1$ sec being the sampling time. To tune the dynamic controllers, we begin by obtaining the constants of the conventional PID controller using the Dahlin method [41-43], which proposes.

$K_P$ is in charge of increasing the response speed and decreasing the system error, and is calculated with Equation 36:

$$K_P = \frac{1}{2K}\left(\frac{t_0}{\tau}\right)^{-1}, \tag{36}$$

$K_D$ is responsible for increasing the response of the system, and is calculated with Equation 37:

$$K_D = K_P\left(\frac{t_0}{2}\right)^{-1}, \tag{37}$$

$K_I$ is in charge of decreasing the error of the system in steady state and increasing the speed of the system moderately and is calculated with Equation 38:

$$K_D = K_P\left(\frac{t_0}{2}\right)^{-1}, \tag{38}$$

The traditional PID controller parameters serve as the foundational framework for the PIDN and PID+NN controllers. In the PIDN configuration, these parameters serve as initial values to commence the learning process. To ensure equitable comparisons, both neural controllers are configured with identical learning rates, designated as α. Specifically, the learning rates are designated as 0.009 for linear speed and 0.00005 for angular speed. To evaluate the system's response to disturbances, the robot traverses an inclined plane inclined at a 10° angle at various time intervals, as illustrated in Figure 10. This incline induces adjustments in the robot's dynamics attributable to shifts in its center of mass):

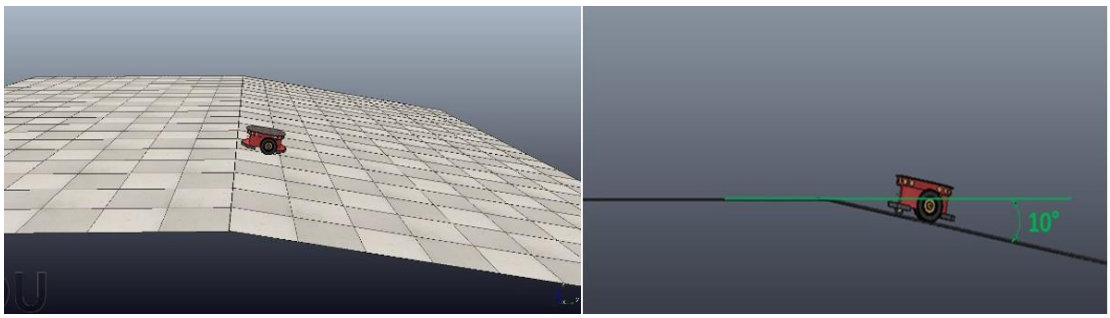



**Figure 10. Programmed inclination in CoppeliaSim**

### *3-1- Square Trajectory*

The square trajectory comprises a series of right-angle turns and linear movements, acting as a standardized measure to assess the controllers' ability to handle sudden shifts in orientation while upholding the robot's stability. In real-world scenarios, such as in manufacturing, logistics, and material transportation [39] robots must adeptly navigate unexpected changes in orientation. This is particularly pertinent as certain trajectory planning algorithms may generate such maneuvers. The results, as depicted in Figure 11 and Figure 12-a, offer a qualitative assessment of performance. Notably, along the straight path, the instantaneous mean square distance error attributed to the kinematic controller remains consistently below 2 cm. Even during encounters with sharp corners involving abrupt changes in orientation, the error does not exceed 15 cm. It is noteworthy that the highest error peak occurs during the initial sharp change in orientation along the trajectory. This is primarily due to the robot descending an inclined plane, necessitating a reduction in linear velocity. Despite this challenge, the dynamic controllers effectively maintain the robot's position.

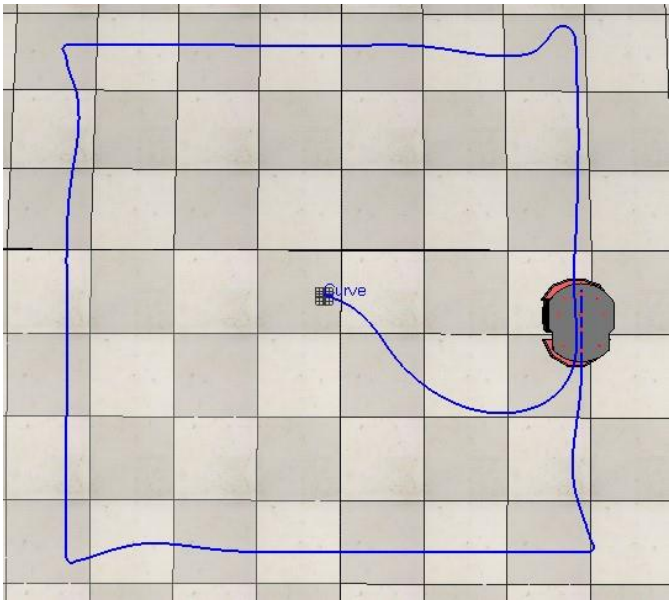



**Figure 11. Square path tracking in CoppeliaSIM environment**

The results depicted in Figure 12-b provide a qualitative evaluation of performance. It is evident that, along the straight path, the instantaneous mean square distance error attributed to the kinematic controller remains below 2 cm. During encounters with sharp corners involving abrupt orientation changes, the error does not surpass 15 cm. Notably, the highest error peak occurs during the initial sharp change in orientation along the trajectory. This is mainly due to the robot descending an inclined plane, necessitating a reduction in linear velocity. Despite this challenge, the dynamic controllers effectively preserve the robot's position. In the context of maintaining a reference linear velocity of approximately 0.2 m/s, as depicted in Figure 12-c, it is evident that the intelligent controllers exhibit a more aggressive control action. This results in a rapid attainment of the desired reference, leading to the presence of overshoots. Regarding angular velocity, as illustrated in Figure 12-d, it is demonstrated how the intelligent controllers efficiently reduce the error to zero during the straight lines of the trajectory. They showcase faster response times in reaching the reference, enabling swifter turns while maintaining velocity stability.

Comparatively, the PIDN controller displays fewer oscillations than the PID+NN controller. Although the proposed PID+NN and PIDN showed a maximum over peak higher to the conventional PID, both controllers are characterized by their assertive response, reaching the reference values quicker without compromising the integrity of the actuators. Despite experiencing oscillations and overshoots, these controllers maintain consistent linear and angular velocities in the face of perturbations, such as the presence of inclined planes, resulting in a significant improvement in trajectory tracking accuracy compared to conventional PID.

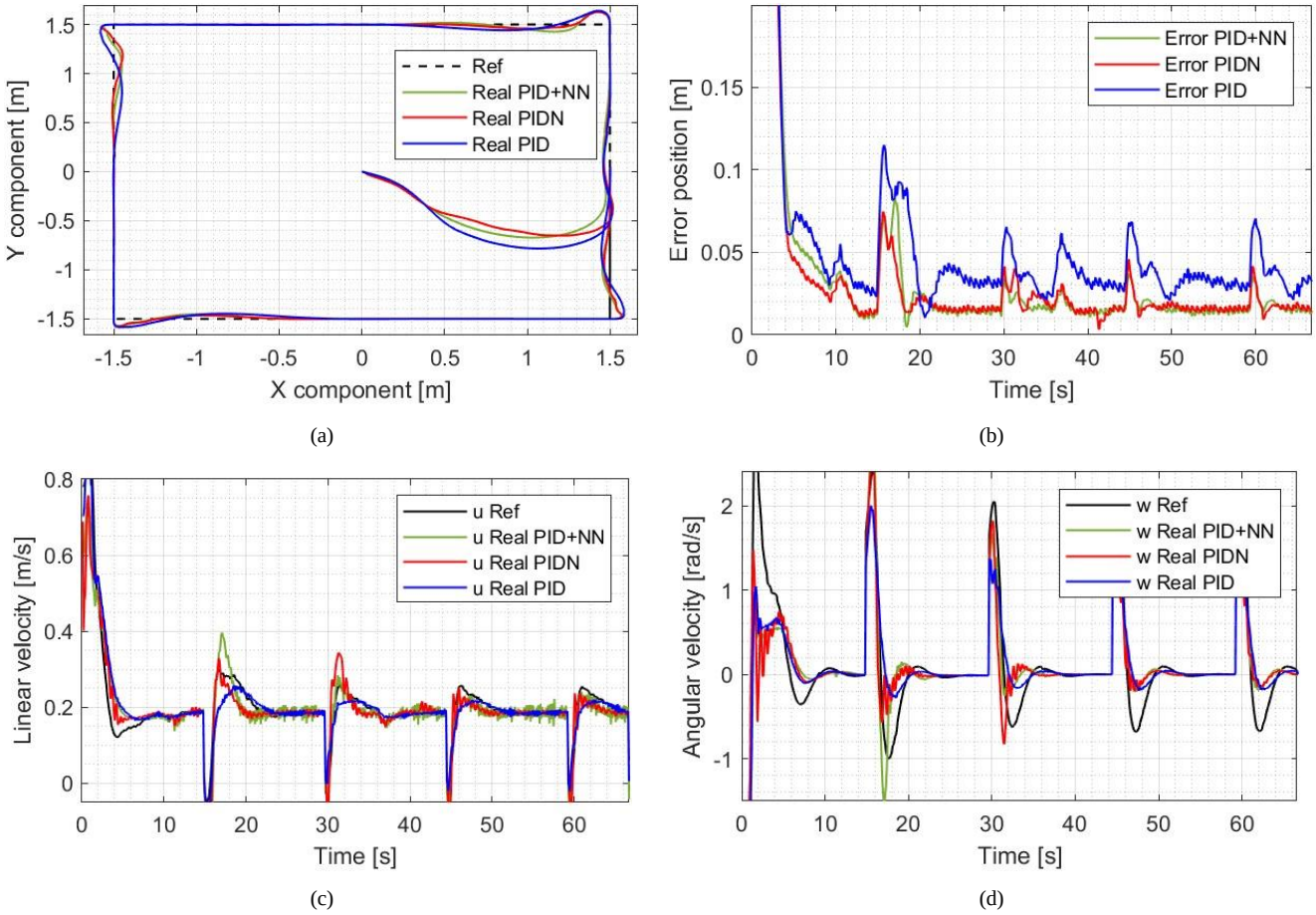



**Figure 12. (a) Square Trajectory Tracking, (b) Position error for Square trajectory, (c) Linear velocity for Square trajectory, (d) Angular velocity for Square trajectory**

### *3-2- Circular Trajectory*

The circular trajectory, distinguished by its smooth curve, ensures a constant change of orientation, resulting in more precise turns compared to the square trajectory. The consistent curvature facilitates the attainment of a constant linear and angular speed, demanding precise control from the dynamic controllers. Figure 13 shows the circular trajectory followed by the mobile robot in the environment using the PID+NN controller and Figure 14-a shows the analysis between the PID, PIDN, PID+NN controllers and the expected trajectory.

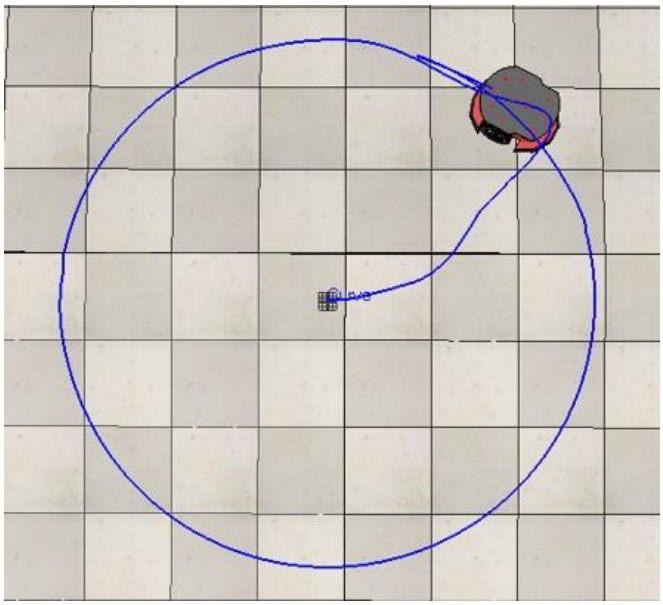


**Figure 13. Circular path tracking in VREP environment**

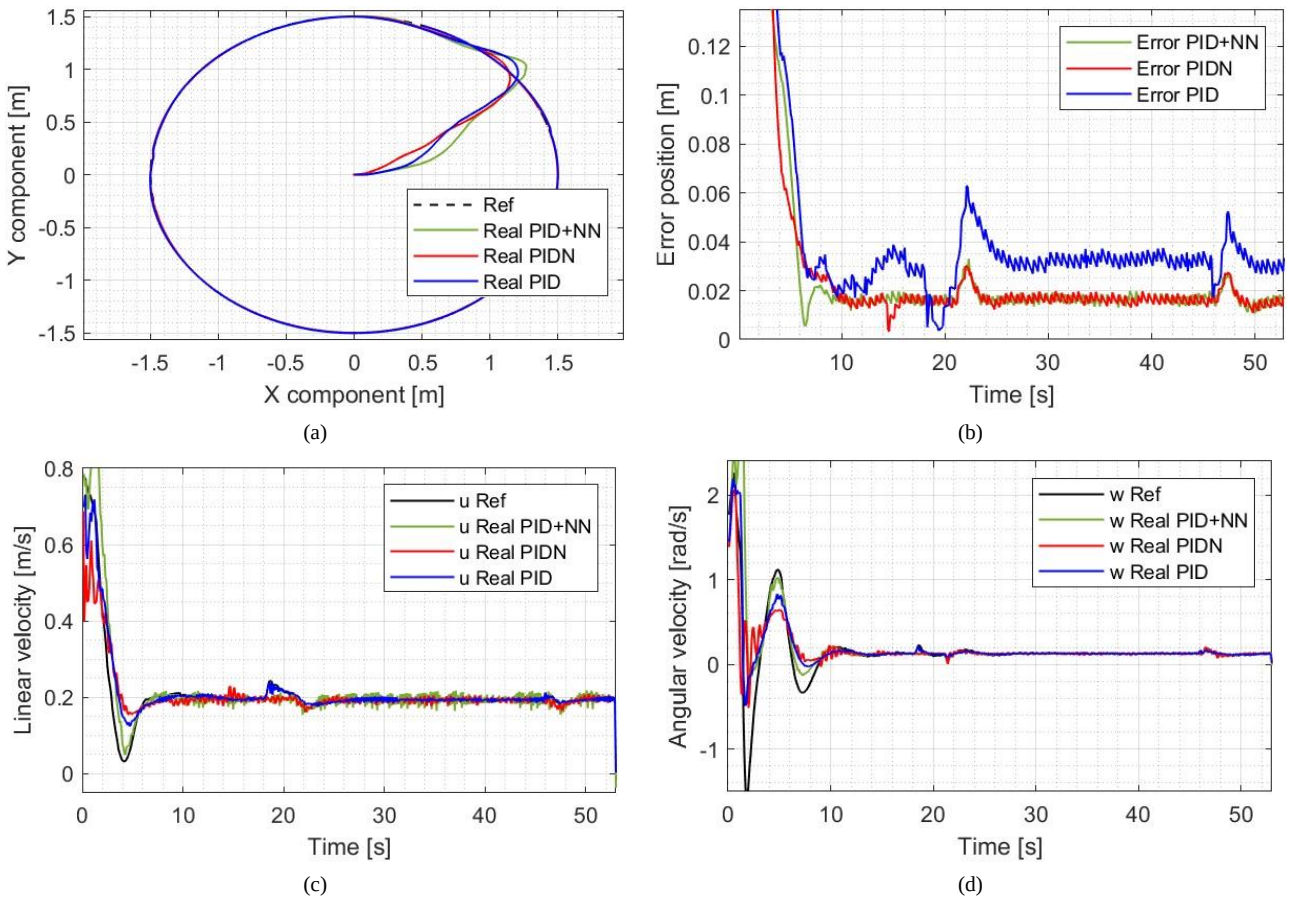



**Figure 14. (a) Circular Trajectory Tracking, (b) Position error for Circular trajectory, (c) Linear velocity for Circular trajectory, (d) Angular velocity for Circular trajectory**

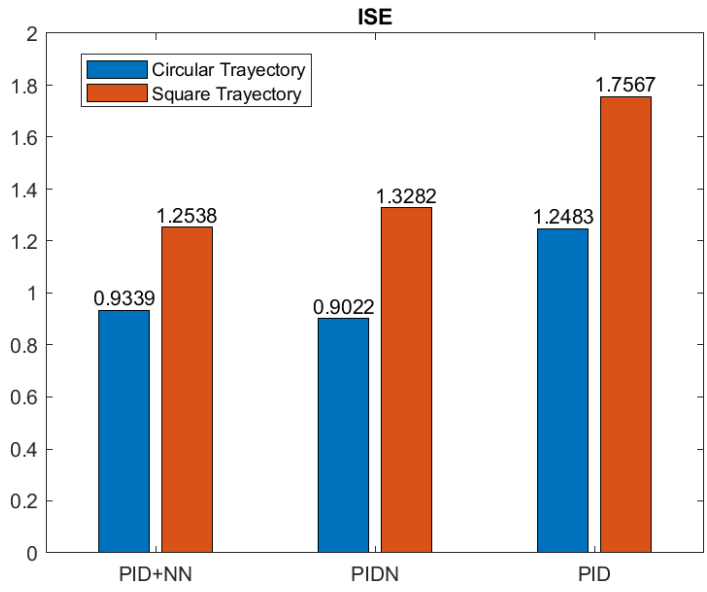
The kinematic control results provide a qualitative insight, showcasing an instantaneous mean square distance error consistently below 2 cm (refer to Figure 14-b). As the robot transitions onto the inclined plane, an increase in velocity is observed, coupled with adjustments facilitated by the controllers. Notably, controllers based on neural networks exhibit a swifter response in reaching the linear velocity reference of 0.2 m/s, as illustrated in Figure 14-c. Additionally, a distinct velocity correction is noted upon entering the inclined plane, showcasing a lesser overshoot compared to the PID controller. Regarding angular velocity at 0.2 rad/s (refer to Figure 14-d), a steady orientation change is encountered. However, the PIDN controller exhibits a more pronounced overshoot in tracking the reference compared to the other two controllers. Despite this, it effectively enables precise tracking of the circular trajectory, even in the face of dynamic changes induced by the inclined plane.

During the initial phase, the PIDN controller shows less over peak, followed by the traditional PID and, finally, the PID+NN controller. All three controllers maintain this until the trajectory is reached. Once the robot is on the path, all three controllers present acceptable behavior, and the trajectory error decreases compared to the square trajectory for all three variants.

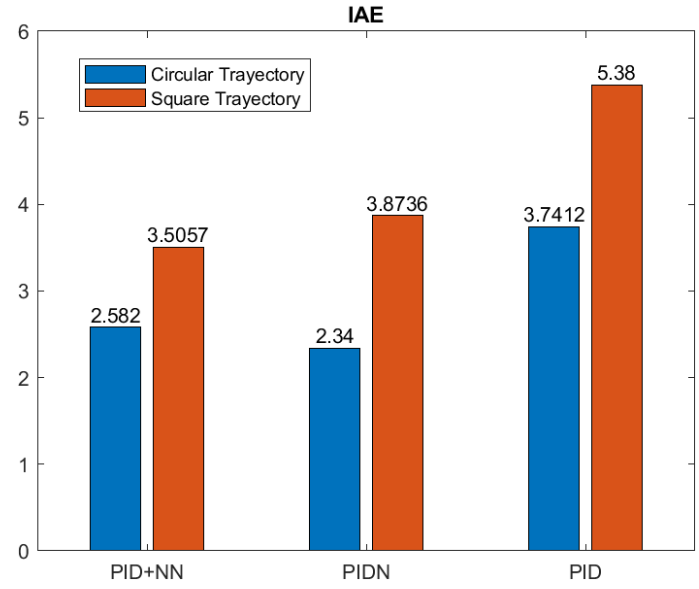
As depicted in Figures 12-c, 12-d, 14-c, and 14-d, the control actions of the PID+NN proposal exhibit the highest level of aggressiveness among the three, facilitating quicker attainment of the reference. Nevertheless, the overshoot compared to the others is minimal and even comparable, suggesting that despite the higher energy expenditure, this is offset by the ability to swiftly reach the reference. Additionally, it is noteworthy that the control action of PIDN demonstrates the smoothest energy consumption profile, as depicted graphically, presenting a significantly acceptable and superior response compared to the PID, owing to its adaptive characteristics.

### *3-3- Quantitative Analysis*

The effectiveness of the system in minimizing position error becomes apparent when analyzing the ISE index, as illustrated in Figure 15-a. Controllers based on neural networks showcase superior precision in trajectory tracking and maintaining positional reference, even in the presence of disturbances like the transition from a horizontal to an inclined plane. To evaluate error minimization, the IAE index is utilized, as depicted in Figure 15-b. The results indicate that the intelligent controllers (PIDN and PID+NN) exhibit smaller absolute errors compared to the conventional controller (PID). In broader terms, it is evident that the PID+NN controller enhances the indices by approximately 8.4% in square trajectories, which involve abrupt changes in orientation. On the other hand, its performance in circular trajectories is comparable to that of the PIDN controller, which is considered the preferable choice for accurately following the desired path.




(a)




(b)

**Figure 15. Quantitative comparison for the different controllers based on (a) ISE (b) IAE**

## 4- Conclusion

In this paper, we developed and deployed two neural network-based controllers for trajectory tracking in a differential traction mobile robot. Assessments in CoppeliaSIM demonstrated their impressive performance across diverse trajectories. The setup includes a kinematic controller in the outer loop, based on the robot's kinematic model, and a dynamic controller in the inner loop, integrating neural networks (PID+NN and PIDN). This configuration ensured consistent maintenance of velocities despite disturbances like inclined planes, achieving the reference trajectory swiftly compared to conventional PID.

The findings of this paper not only refine trajectory tracking precision but also carry substantial practical implications. Neural network integration improves response speed, diminishes tracking error, and enhances adaptability, especially beneficial in navigating robots through intricate environments. As the controller adapts, performance thrives even amidst uncertainties, rendering it invaluable for real-world robotic tasks. Notably, the PIDN and PID+NN controllers excel for their distinct attributes. They effectively manage oscillations and overshoots within a tolerable range, safeguarding actuator operability and robot structural integrity. This blend of swift response and adept disturbance handling underscores their efficacy and superiority in motion control applications. Their assertive response, coupled with meticulous control action regulation within the robot's physical and mechanical limitations, mitigates overload scenarios and minimizes component damage risks.

The successful fusion of a traditional PID with neural networks paves the way for future exploration of hybrid algorithms featuring diverse and intricate control techniques applicable in agricultural contexts, military operations, and service robotics, among others. Improving human-robot interaction is a promising avenue for future research as it provides insights into human behavior. Additionally, neural network-based controllers can facilitate collaborative efforts, enabling the design and implementation of cooperation and coordination between multiple robots. Beyond controller development, neural networks can assist in plant identification and predictive control refinement, addressing uncertainties in the environment to improve controller performance.

## 5- Declarations

### *5-1- Author Contributions*

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

### *5-2- Data Availability Statement*

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

### *5-3- Funding*

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

### *5-4- Institutional Review Board Statement*

Not applicable.

### *5-5- Informed Consent Statement*

Not applicable.

### *5-6- 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.

## 6- References

[1] Loganathan, A., & Ahmad, N. S. (2023). A systematic review on recent advances in autonomous mobile robot navigation. Engineering Science and Technology, an International Journal, 40, 101343. doi:10.1016/j.jestch.2023.101343.

[2] Kot, T., & Novák, P. (2018). Application of virtual reality in teleoperation of the military mobile robotic system TAROS. International Journal of Advanced Robotic Systems, 15(1), 172988141775154. doi:10.1177/1729881417751545.

[3] Nevot Cercós, J. (2000). Design of an advanced controller based on neural networks for the management of the air-gasoline mixture in a reciprocating engine. Ph.D. Thesis, Universitat Politècnica de Catalunya, Barcelona, Spain. (In Spanish).

[4] Åstrand, B., & Baerveldt, A. J. (2002). An agricultural mobile robot with vision-based perception for mechanical weed control. Autonomous Robots, 13(1), 21–35. doi:10.1023/A:1015674004201.

[5] Cumbajin, A. (2020). Development of a navigation system based on the Pioneer P3-DX platform for the transport of materials. Universidad de las Fuerzas Armadas, Sangolquí, Ecuador. (In Spanish).

[6] Zhang, L. J., Jia, H. M., & Qi, X. (2011). NNFFC-adaptive output feedback trajectory tracking control for a surface ship at high speed. Ocean Engineering, 38(13), 1430–1438. doi:10.1016/j.oceaneng.2011.07.006.

[7] Zhao, T., Qin, P., & Zhong, Y. (2023). Trajectory Tracking Control Method for Omnidirectional Mobile Robot Based on Self-Organizing Fuzzy Neural Network and Preview Strategy. Entropy, 25(2), 248. doi:10.3390/e25020248.

[8] Somlyai, L., & Vamossy, Z. (2012). Map building with RGB-D camera for Mobil robot. 2012 IEEE 16th International Conference on Intelligent Engineering Systems (INES), 489-493. doi:10.1109/ines.2012.6249883.

[9] Csaba, G., Somlyai, L., & Vamossy, Z. (2018). Mobil robot navigation using 2D LIDAR. 2018 IEEE 16th World Symposium on Applied Machine Intelligence and Informatics (SAMI). doi:10.1109/sami.2018.8324002.

[10] Sarabia Morales, B. F. (2017). Design and simulation of three classic and robust control techniques applied to trajectory tracking in the presence of fixed delays for the Pioneer 3DX robotic platform. Bachelor's Thesis, Escuela Politécnica Nacional, Quito, Ecuador.

[11] Seghour, S., & Tadjine, M. (2017). Consensus-based approach and reactive fuzzy navigation for multiple no-holonomic mobile robots. 2017 6th International Conference on Systems and Control, ICSC 2017, 492–497. doi:10.1109/ICoSC.2017.7958658.

[12] Shen, X., & Shi, W. (2019). Adaptive Trajectory Tracking Control of Wheeled Mobile Robot. 2019 Chinese Control and Decision Conference (CCDC), 5161-5165. doi:10.1109/ccdc.2019.8833019.

[13] Vo, A. T., Kang, H.-J., & Nguyen, V.-C. (2017). An output feedback tracking control based on neural sliding mode and high order sliding mode observer. 2017 10th International Conference on Human System Interactions (HSI), 161-165. doi:10.1109/hsi.2017.8005020.

[14] Ben Halima Abid, D., Allagui, N. Y., & Derbel, N. (2017). Navigation and trajectory tracking of mobile robot based on kinematic PI controller. 2017 18th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA), 252-256. doi:10.1109/sta.2017.8314966.

[15] Sanchez, E. M., Ramirez, J. P., & Angeles, A. R. (2021). Autonomous navigation of a mobile robot using a network of Hindmarsh-Rose (HR) neurons. 2021 18th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE). doi:10.1109/cce53527.2021.9633035.

[16] Alouache, A., & Wu, Q. (2018). Genetic Algorithms for Trajectory Tracking of Mobile Robot Based on PID Controller. 2018 IEEE 14th International Conference on Intelligent Computer Communication and Processing (ICCP), 237-241. doi:10.1109/iccp.2018.8516587.

[17] Matich, D. J. (2001). Neural Networks: Basic Concepts and Applications. National University of Technology—Faculty Regional Rosario. Group of Applied Research in the Chemical Engineering (GIAIQ), Mexico City, Mexico.

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

[19] Haykin, S. (2009). Neural networks and learning machines. Pearson Education, Inc., Upper Saddle River, United States.

[20] Trujillo, D., Morales, L. A., Chávez, D., & Pozo, D. F. (2023). Trajectory Tracking Control of a Mobile Robot using Neural Networks. Emerging Science Journal, 7(6), 1843–1862. doi:10.28991/ESJ-2023-07-06-01.

[21] Puentes, K., & Morales, L. (2023). Trajectory Tracking of a Mobile Robot Using a PID Controller Combined with Neural Networks. 2023 IEEE Seventh Ecuador Technical Chapters Meeting (ECTM). doi:10.1109/etcm58927.2023.10309094.

[22] Hui, Z., & Ji-hong, S. (2014). Neural network robust control of ship trajectory tracking. 2014 IEEE International Conference on Mechatronics and Automation. doi:10.1109/icma.2014.6885899.

[23] Deng, J., Li, Z., & Su, C.-Y. (2014). Trajectory tracking of mobile robots based on model predictive control using primal dual neural network. Proceedings of the 33rd Chinese Control Conference. doi:10.1109/chicc.2014.6896401.

[24] da Silva Lima, G., Moreira, V. R. F., & Bessa, W. M. (2023). Accurate trajectory tracking control with adaptive neural networks for omnidirectional mobile robots subject to unmodeled dynamics. Journal of the Brazilian Society of Mechanical Sciences and Engineering, 45(1), 48. doi:10.1007/s40430-022-03969-y.

[25] Hoang, T. T., Hiep, D. T., Duong, B. G., & Vinh, T. Q. (2013). Trajectory tracking control of the nonholonomic mobile robot using torque method and neural network. 2013 IEEE 8th Conference on Industrial Electronics and Applications (ICIEA). doi:10.1109/iciea.2013.6566660.

[26] Dang, S. T., Dinh, X. M., Kim, T. D., Xuan, H. Le, & Ha, M. H. (2023). Adaptive Backstepping Hierarchical Sliding Mode Control for 3-Wheeled Mobile Robots Based on RBF Neural Networks. Electronics (Switzerland), 12(11), 2345. doi:10.3390/electronics12112345.

[27] Lewis, F. L., Vrabie, D. L., & Syrmos, V. L. (2012). Optimal Control (3rd Ed.), John Wiley & Sons, Hoboken, United States. doi.:10.1002/9781118122631.

[28] Tai, L., Li, S., & Liu, M. (2016). A deep-network solution towards model-less obstacle avoidance. 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). doi:10.1109/iros.2016.7759428.

[29] Huang, C.-M., & Wang, F.-L. (2007). An RBF Network with OLS and EPSO Algorithms for Real-Time Power Dispatch. IEEE Transactions on Power Systems, 22(1), 96–104. doi:10.1109/tpwrs.2006.889133.

[30] Cheon, H., Kim, T., Kim, B. K., Moon, J., & Kim, H. (2023). Online Waypoint Path Refinement for Mobile Robots Using Spatial Definition and Classification Based on Collision Probability. IEEE Transactions on Industrial Electronics, 70(7), 7004–7013. doi:10.1109/TIE.2022.3203684.

[31] 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.

[32] Khoshlessan, M., Asaei, B., & Farhangi, B. (2015). Analysis of fly-back PV micro-inverter and optimizing control system using Finite Gradient Descent Method. 2015 Intl Aegean Conference on Electrical Machines & Power Electronics (ACEMP), 2015 Intl Conference on Optimization of Electrical & Electronic Equipment (OPTIM) & 2015 Intl Symposium on Advanced Electromechanical Motion Systems (ELECTROMOTION), 287-292. doi:10.1109/optim.2015.7427011.

[33] Anushree, R., & Prasad, B. K. S. (2016). Design and development of novel control strategy for trajectory tracking of mobile robot: Featured with tracking error minimization. IEEE Annual India Conference, 1-6. doi:10.1109/indicon.2016.7839162.

[34] Schuler, A. J., Nachbar, P., Nossek, J. A., & Chua, L. O. (1992). Learning state space trajectories in cellular neural networks. CNNA '92 Proceedings Second International Workshop on Cellular Neural Networks and Their Applications, 68-73. doi:10.1109/cnna.1992.274353.

[35] Zheng, Y., Zheng, J., Shao, K., Zhao, H., Xie, H., & Wang, H. (2024). Adaptive Trajectory Tracking Control for Nonholonomic Wheeled Mobile Robots: A Barrier Function Sliding Mode Approach. IEEE/CAA Journal of Automatica Sinica, 11(4), 1007–1021. doi:10.1109/JAS.2023.124002.

[36] Rossomando, F. G., Soria, C., & Carelli, R. (2014). Sliding mode neuro adaptive control in trajectory tracking for mobile robots. Journal of Intelligent and Robotic Systems: Theory and Applications, 74(3–4), 931–944. doi:10.1007/s10846-013-9843-5.

[37] Minguez, J., Montano, L., & Santos-Victor, J. (2002). Reactive navigation for non-holonomic robots using the ego-kinematic space. Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292), 3, 3074–3080. doi:10.1109/robot.2002.1013699.

[38] Masutani, Y., Mikawa, M., Maru, N., & Miyazaki, F. (1994). Visual servoing for non-holonomic mobile robots. Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'94), 2, 1133–1140. doi:10.1109/iros.1994.407471.

[39] Moudoud, B., Aissaoui, H., & Diany, M. (2020). Robust trajectory tracking control based on sliding mode of Differential Driving Four-Wheeled Mobile Robot. 2020 IEEE 6th International Conference on Optimization and Applications (ICOA), 1-5. doi:10.1109/icoa49421.2020.9094510.

[40] Singh, S., & Mitra, R. (2014). Comparative analysis of robustness of optimally offline tuned PID controller and Fuzzy supervised PID controller. 2014 Recent Advances in Engineering and Computational Sciences (RAECS). doi:10.1109/raecs.2014.6799546.

[41] Yu, B. H., Kim, D. H., Yu, B. G., Lee, S. Y., & Han, C. S. (2008). Development of prototype of an Unmanned Transport Robot for transport of construction materials. 2008 International Conference on Control, Automation and Systems, 448-452. doi:10.1109/iccas.2008.4694682.

[42] Alippi, C. (1991). Weight update in back-propagation neural networks: the role of activation functions. Proceedings 1991 IEEE International Joint Conference on Neural Networks, 560-565. doi:10.1109/ijcnn.1991.170459.

[43] Khoukhi, A., Hamdan, M., & Al-Sunni, F. (2012). ANFIS Based-Kinematic Modeling of Mobile Parallel Robot. 2012 UKSim 14th International Conference on Computer Modelling and Simulation, 242-247. doi:10.1109/uksim.2012.42.

[44] Yu, N., Luo, J., Shu, S., & Sun, B. (2010). Application of Delta-bar-Delta Rules Trained Back-Propagation Neural Networks in Nuclear Fusion Pattern Recognition. 2010 International Symposium on Intelligence Information Processing and Trusted Computing, 258-261. doi:10.1109/iptc.2010.167.

[45] Hong, X., & Tao, X. (2009). Implementation of the Dahlin Digital Controller by IIR Network Method. 2009 Second International Conference on Intelligent Computation Technology and Automation, 621-624. doi:10.1109/icicta.2009.155.