Available online at www.ijournalse.org

Emerging Science Journal

(ISSN: 2610-9182)

Vol. 9, No. 2, April, 2025

Enhancing Trajectory Tracking in Humanoid Robots Using Neural Network-Based Dynamic Gain Control

Darwin Trujillo [1], Luis A. Morales [1], Danilo Chávez [1], María Trujillo [1], David F. Pozo-Espín [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 the development and evaluation of a dynamic gain controller utilizing neural networks to enhance trajectory tracking performance in the NAO humanoid robot. The proposed controller employs a differential kinematic model and dynamically adjusts its gains using a backpropagation algorithm, eliminating the need for manual gain tuning and simplifying the robot's setup process. Experimental validation was conducted in a simulated environment using CoppeliaSim, with the NAOqi library facilitating integration. The analysis results demonstrate that the dynamic controller using a neural network provides better trajectory tracking accuracy than the traditional kinematic controller. Adaptability of the dynamic controller, which adjusts gain parameters in real-time, contributes to improved robustness and precision across various trajectory types. These findings demonstrate the potential of dynamic, self-tuning controllers in enhancing the performance, efficiency, and versatility of humanoid robots in complex navigation tasks.

Keywords:
NAO; ANN;
Kinematic Controller;
Coppelia Sim.

Article History:

Received:	28	November	2024
Revised:	19	March	2025
Accepted:	25	March	2025
Published:	01	April	2025

1- Introduction

Humanoid robots resemble the shape and structure of the human body and are designed to move and operate in different environments, mimicking the behavior of a human being. They are considered one of the most advanced forms of robotics, as they use advanced technology to have a more natural and human experience [1]. For a humanoid robot to effectively perform tasks, such as navigating a work environment, it must maintain static stability. This condition is satisfied when the center of gravity lies within the polygon created by the contact points of its feet with a surface [2]. Bipedal locomotion in humanoid robots requires a complex interaction of mechanical features and the control system. This nonlinear, dynamic process is characterized by minor oscillations that arise not from the intended trajectory but rather from data measurement artifacts [3]. The locomotion of a humanoid robot can be influenced by factors such as stability, energy consumption, and task duration. Therefore, it is essential to develop effective and versatile solutions to improve its movement [4].

Artificial intelligence (AI) is increasingly integrated into embedded systems, including humanoid robots, enabling them to perform autonomous tasks [5]. The potential of AI in humanoid robotics holds great promise for improving quality of life, particularly in assistance, healthcare, and various other applications, as highlighted in Obrenovic et al. [6]. By integrating AI, these robots can execute tasks with great precision and flexibility, potentially gaining widespread

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

DOI: http://dx.doi.org/10.28991/ESJ-2025-09-02-02

acceptance across different sectors. As AI technology progresses, humanoid robots' capabilities will grow, presenting fresh solutions and opportunities to improve human well-being and societal operations [7]. Overall, AI in humanoid robots has the potential to improve quality of life and well-being. In Podpečan [8], the NAO robot used in the study primarily moves to engage children in various activities, such as motor development games, theatre performances, and artificial intelligence applications. In Venkataswamy et al. [9], a humanoid doctor uses an artificial intelligence platform in the cloud to get a diagnosis of each individual's disease in real time. However, it does not explicitly mention whether a kinematic or dynamic model is used, and it also does not specify the exact controllers used for the NAO robot, but the use of built-in modules such as custom-developed modules for facial expression recognition are mentioned.

The NAOqi library allows various software modules to communicate with each other, allowing commands made in the Python programming language to be executed and controlled by the humanoid robot [10]. It works as a black box ready for use. In particular, this library provides an API that allows controlling the robot's movement and accessing its sensors and actuators, among other functionalities [11]. The kinematic model, corresponding to a mobile robot, and the NAOqi library are indirectly related in the programming of the NAO robot. In the context of mobile and humanoid robots, controlling trajectory tracking is a widely studied and strongly researched task. Mobile robot controllers often rely on mathematical models that accurately describe the robot's behavior. However, the complexity of these models and the numerous variables involved can lead to significant challenges. Model uncertainty and non-linearity are two common issues that arise, and the severity of these challenges depends on the system's complexity [12].

A variety of intelligent control methods are currently under investigation, encompassing the application of artificial intelligence, fuzzy logic, genetic algorithms, and other approaches. The primary objective is to overcome the challenges of mobile and humanoid robotics, such as navigating complex, unpredictable environments autonomously. Additionally, researchers are working to calibrate controllers based on the robot's specific terrain and desired trajectory, as these factors significantly impact performance [13, 14]. Artificial Neural Networks (ANN) are highly effective in approximating nonlinear systems and processes, even with limited data. This capability makes them ideal for designing controllers that address uncertainty and nonlinearities in robot models, significantly enhancing trajectory tracking for both mobile and humanoid robots [2].

In recent studies, artificial intelligence (AI) has emerged as a critical component in dynamic control systems. Asai et al. [15] proposed utilizing artificial neural networks (ANN) for sensor-equipped automatons. Similarly, Chen et al. [16] suggested an adaptive ANN to approximate the unknown dynamics of mobile robots, incorporating a Lyapunov barrier to regulate the speed of the robot. Mohareri et al. [17] took an online approach to tune controller parameters, leveraging ANNs to capture the characteristics of the direct model. Some studies also explore offline learning phases for neural networks. For example, Yildirim et al. [18] designed a controller using a neural network predictor; however, this approach demands extensive training data to optimize gain values and minimize errors. Another method, described in Mohamed & Hamza [19], builds an ANN controller based on the PID. A learning algorithm identifies the PID parameters to reduce trajectory-tracking errors, but this requires significant effort to optimize gain values for each controller. ANN Additionally, Benbouabdallah & Qi-dan [20] propose a fuzzy logic controller grounded in the Takagi-Sugeno methodology. Their approach calculates velocities to satisfy control objectives and employs a genetic algorithm to optimize fuzzy controller inputs, improving trajectory tracking performance. However, conducting this process offline raises concerns about time efficiency. In Farhat et al. [21], a NAO humanoid robot's trajectory control is achieved using an NDO-based FTSM controller, which enhances trajectory tracking even under disturbances. Finally, Bai et al. [22] described dual-arm humanoid robot control, where fuzzy logic and adaptive techniques enable precise, coordinated actions in response to control inputs. A key limitation in previous studies is the reliance on an experimental training phase to map system inputs to outputs, which can compromise accuracy if insufficient.

This work addresses the issue by proposing an online learning phase for the controller, enabling dynamic adjustments based on the error between current and target positions to improve trajectory tracking and overall performance. In this work, we use ANN to develop and self-tune the controller in real-time. The ANN architecture is modeled as a multilayer perceptron, employing the backpropagation algorithm to minimize errors and improve the precision of humanoid trajectory tracking. An adaptable kinematic controller can significantly enhance trajectory tracking by dynamically adjusting its parameters to account for uncertainties and variations in the robot's dynamics and environment. Unlike fixed-gain controllers, adaptable controllers can modify their behavior in real time, reducing trajectory errors and improving overall performance. This adaptability is particularly beneficial for humanoid robots, such as the NAO, where unmodeled dynamics, sensor noise, or external disturbances can affect tracking accuracy [23]. By integrating learning mechanisms, such as artificial neural networks, these controllers can estimate and compensate for nonlinearities, making them highly effective in complex, real-world scenarios. The ability to self-tune ensures robust performance even under varying conditions, a critical requirement for applications involving dynamic or unpredictable environments.

The effectiveness of the proposed controller is assessed through standard metrics, including ISU, IAE, and ISE. Its performance in trajectory tracking is compared to that of a non-self-tuning kinematic controller, showcasing improved system behavior across multiple scenarios.

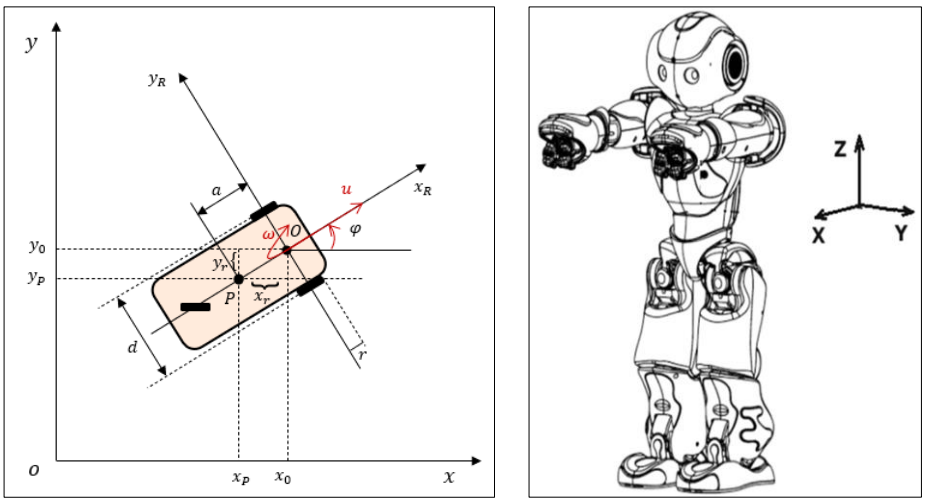
The structure of the paper is as follows: Section 2 presents the mathematical model of the mobile robot. Section 3 provides a brief introduction to the fundamentals of neural networks. Section 4 focuses on the design and implementation of the ANN-based controller. The experimental results are analyzed in Section 5, while Section 6 concludes with a summary and key observations.

2- Tracking Trajectory of Humanoid Robots

Path-following controllers, traditionally based on robot kinematics, are effective in well-defined environments. However, in scenarios with varying dynamic properties, considering the robot's dynamics becomes essential. While precise trajectory tracking is the primary goal, uncertainties and disturbances can lead to errors in robot motion. Humanoid robots often exhibit periodic oscillations during trajectory following due to the displacement point's location in the chest and the influence of leg movements. This characteristic is reflected in the robot's control signals. The proposed dynamic controller addresses these challenges by adapting to unknown system dynamics [24-26]. This allows the robot to stay on the reference trajectory within a given time limit, while reducing tracking errors.

2-1- Human Robot Model

The humanoid robot is a differential mobile robot and is non-omnidirectional. This type of robot is commonly chosen for studying control methods because of its rapid response and non-linear dynamics [27]. Figure 1 illustrates a standard representation of a mobile robot. The point *C* denotes the center of the axle connecting the right and left wheels, while *G* indicates its center of gravity. Notably, *C* also functions as the control point. The space relating the *C* point and the center of the wheel axle is denoted by *L*, and *r* is the wheel's radius. The space involving the wheels is *d*. The linear velocity is represented by *v*, the angular velocity by ω, and the robot's orientation by θ.



(a) (b)

Figure 1. a) Geometry of the Robot; b) NAO robot orientation [28]

Given the configuration of the robot at a specific time t_k, represented by $[x_k\ y_k\ \varphi_k]$, and the known velocities u_k and ω_k, the Euler integration method is applied to approximate the subsequent configuration of the robot at a time t_k [29]. This estimation is accomplished by determining the robot's position and orientation using its current state and the velocities applied.

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

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

Rossomando et al. [30] proposed a cascade controller approach for humanoid robots, integrating both kinematic and dynamic models. Their work assumed an unknown dynamic model and utilized a neural network-based method to control it.

2-2- Kinematic Controller

The kinematic controller, derived from the robot's kinematic model as presented in Equation 2, focuses on the point coordinates $[x, y]$. Employing a proportional (P) control approach, as described in SoftBank Robotics [31], the control law is:

$$\begin{bmatrix} u_{ref}(t_k) \\ \omega_{ref}(t_k) \end{bmatrix} = \begin{bmatrix} \cos\theta\,(t_k) & \sin\theta\,(t_k) \\ -\frac{1}{a}\sin\theta\,(t_k) & \frac{1}{a}\cos\theta\,(t_k) \end{bmatrix} \times \begin{bmatrix} \frac{x_r(t_k)-x_r\,(t_k-1)}{T_s} + k_x\, e_x(t_k) \\ \frac{y_r(t_k)-y_r\,(t_k-1)}{T_s} + k_y\, e_y(t_k) \end{bmatrix} \quad (2)$$

The output of the kinematic controller is $[u_{ref}\ \omega_{ref}]^T$. For stability the gains of the controller are $k_x > 0$, $k_y > 0$. Errors are $e_x(t_k) = x_r(t_k) - x(t_k), e_y(t_k) = y_r(t_k) - y(t_k)$ with $x_r(t_k)$ and $y_r(t_k)$ the reference coordinates. We consider, $u_{ref}(t_k) = u(t_k), \omega_{ref} = \omega(t_k)$ for our proposal of kinematic controller.

3- Artificial Neural Networks ANNs

Artificial intelligence, particularly neural networks, significantly enhances the trajectory tracking capabilities of humanoid robots such as NAO. By employing machine learning techniques, we can design dynamic controllers that adapt to varying conditions and uncertainties in real-time. This adaptive nature is crucial for achieving precise and robust trajectory tracking, as it allows the controller to learn and improve its performance over time. The ability to automatically tune controller gains eliminates the need for extensive manual calibration, making the robot setup process more efficient and less prone to human error. Consequently, AI-driven controllers offer a promising avenue for advancing the field of robotics and enabling more sophisticated and autonomous behaviors in humanoid robots.

ANN is adept at approximating nonlinear systems. Their adaptive nature helps reduce uncertainty in such systems by fine-tuning internal parameters [10, 29]. ANN have diverse applications, including system control, computer vision, and pattern recognition. The Multi-layer Perceptron (MLP) is especially well-suited for system control [15]. ANN learning focuses on modifying synaptic weights to reduce a defined objective function, typically achieved by employing the backpropagation algorithm [1, 8]. In observed learning, the desired output corresponding to a specific input dataset is pre-determined. This process is guided either by automated systems or direct human intervention [30].

The MLP, a widely used neural network architecture, excels at filtering noise, approximating nonlinear systems, and serving as a universal approximator [1, 8]. Its ability to organize neurons into multiple hierarchical levels is a key feature. However, the nonlinear nature of its components presents challenges in terms of interpretability.

The scheme of a MLP is shown in Figure 2. Each connection between neurons is characterized by synaptic weights. The network consists of three primary layer types: the input layer, which receives signals and transmits them to the next layer; hidden layers, where neurons nonlinearly process received patterns; and the output layer, which generates the network's response to the input values.

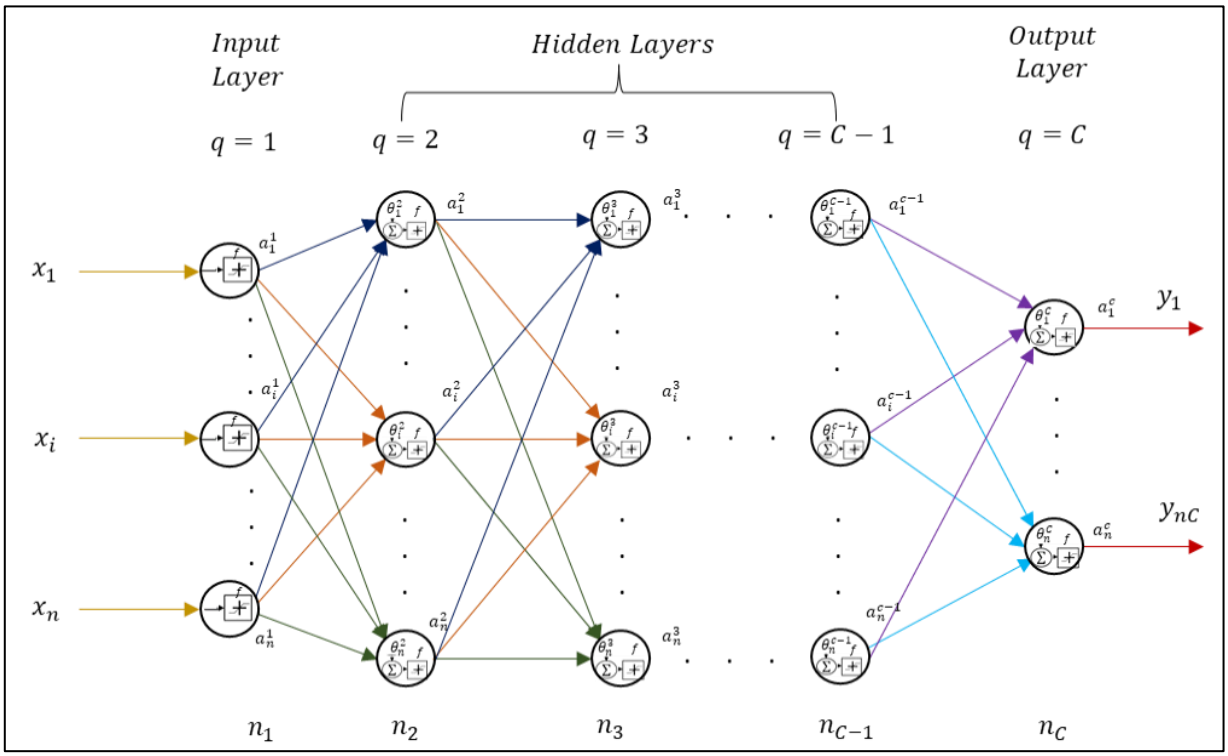


Figure 2. MLP

For a MPL with c layers, $c - 2$ hidden layers and $q = 1, 2, 3 \ldots, c$.

Neurons in the input layer are activated as follows:

$$a_i^1 = x_i \quad \forall\, i = 1,\ 2\ , .. n_1 \tag{3}$$

In the input layer, a_i^1 represents the neuron i, x_i is the input vector of the ANN and n_1 is the number of neurons in the layer 1.

In the hidden layer, a_i^q is calculated as follows:

$$a_i^q = f\left(\sum_{j=1}^{n_{q-1}} w_{ji}^{q-1} a_j^{q-1} + \theta_i^q\right) \quad \forall\, i = 1,\ 2\ , \ldots n_q \ and\ q = 2, \ldots, c-1 \tag{4}$$

In this neural network architecture, the start of the neuron j in layer q is denoted by a_i^q. This activation is calculated based on the activations of neurons j in the preceding layer $q\ -\ 1$, $a(q-1)$, the weights connecting these neurons W_{ji}^{q-1}, and a threshold value θ_i^q.

The activation function, *f*, used in this context is a hyperbolic tangent function, defined by Equations 5 and 6.

Its derivative, necessary for the backpropagation algorithm, is given by Equation 6.

$$\mathrm{f(x)} = \tanh(\mathrm{x}) = \frac{1-\mathrm{e}^{-\mathrm{x}}}{1+\mathrm{e}^{-\mathrm{x}}} \tag{5}$$

$$\mathrm{f'(x)} = \mathrm{sech}^2(\mathrm{x}) = 1 - \tanh^2(\mathrm{x}) \tag{6}$$

3-1- Backpropagation Algorithm (BP)

Backpropagation is a supervised learning algorithm that adjusts synaptic weights to minimize the mean squared error between the desired and actual outputs [31]. The error signal is propagated backward through both the hidden and output layers [8]. Consequently, the learning process is formulated as an optimization problem of the following type:

$$Min_W E \tag{7}$$

where E is the error function. W denotes the synaptic weights of the neural network.

Equation 5 expressed the error function as:

$$E(t_k) = \frac{1}{2}\sum_{i=1}^{n_C} (y_{di}(n) - y_i(n))^2 \tag{8}$$

where $y_{di}(n)$ is the required output value and $y_i(n)$ is the output value of the network for pattern n at the time t_k.

To minimize Equation 9, the stochastic gradient descent method is employed to propagate errors backward through the network. This method iteratively adjusts each weight parameter w to reduce the error $E(t_k)$.

The learning law is as follows:

$$W(t_k) = W(t_k - 1) - \alpha \frac{\partial E(t_k)}{\partial W} \tag{9}$$

where α represents the learning rate, typically in the range [0,1], which determines the extent to which each weight is adjusted during updates. The selection of α in this study is heuristic [32]. Higher α values may prevent finding a local minimum and could lead to non-convergence of the algorithm, while lower α values may converge to a local minimum but could result in longer processing times.

Finally, the BP is defined as:

$$W_{kj}^q(t_k) = W_{kj}^q(t_k - 1) - \alpha\, \delta_i^{q-1}(t_k)\, a_k^q(t_k);$$
$$\forall\, i = 1,2, \ldots\ , n_c\ ;\quad j = 1,2, \ldots\ , n_{c-1}\ ; \tag{10}$$
$$\mathrm{q} = 1,2, \ldots, \mathrm{c}-2$$

The term δ ais defined for neuron i in layer $q\ +\ 1$ for pattern n, where $a_k^q(n)$ represents the initiation of neuron k in layer q for pattern n, as follows:

$$\delta_i^{q-1}(t_k) = f'\left(\sum_{k=1}^{n_q} W_{kj}^q a_k^q + \theta_j^q\right) \sum_{i=1}^{n_{q+1}} \delta_i^{q+2}(t_k)\ W_{ji}^q \tag{11}$$

Equation 11 requires the derivative of the activation function, which is provided in Equation 6. Where W_{kj}^{q} is the vector of weights.

The network thresholds are defined by:

$$\theta_j^{q-1}(t_k) = \theta_j^{q-1}(t_k - 1) - \alpha\, \delta_i^{q-1}(t_k)$$
$$\forall\ j = 1,2,3 \dots, n_{c-1};\ q = 2,3 \dots, c-2 \tag{12}$$

where θ_j^{q-1} represents the thresholds for the layer $q - 1$ corresponding to the pattern n.

4- Controller Design

Conventional tracking control systems for humanoid robots often face challenges due to model uncertainties, variations in plant parameters, and external disturbances. Combined with the intrinsic complexity of humanoid dynamics, these factors demand a controller that can ensure reliable performance without requiring explicit compensation [15]. Figure 3 illustrates the proposed neural network architecture, which effectively addresses these challenges by employing an ANN-based controller to drive the system toward the required trajectory.

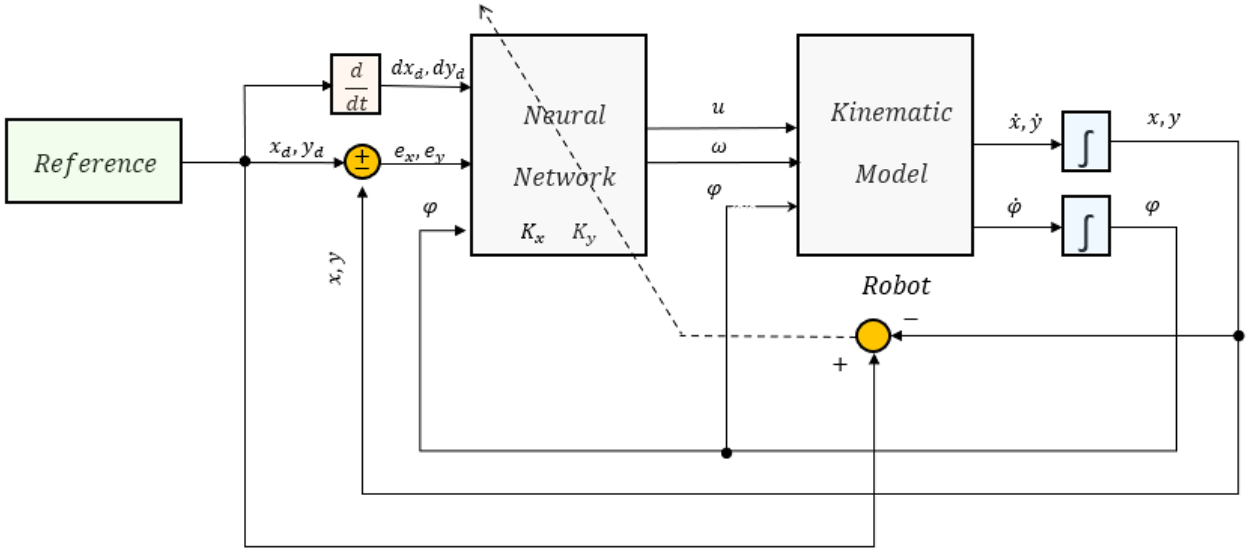


Figure 3. Control Architecture

Figure 4 illustrates the architecture of the proposed controller, which operates as a sequential process built around a neural network inspired by a kinematic controller. During online operation, the backpropagation (BP) algorithm is employed to minimize the positional error between the humanoid robot's current position and the target trajectory. This iterative method dynamically updates the neural network parameters, facilitating continuous optimization and improving the controller's overall performance.

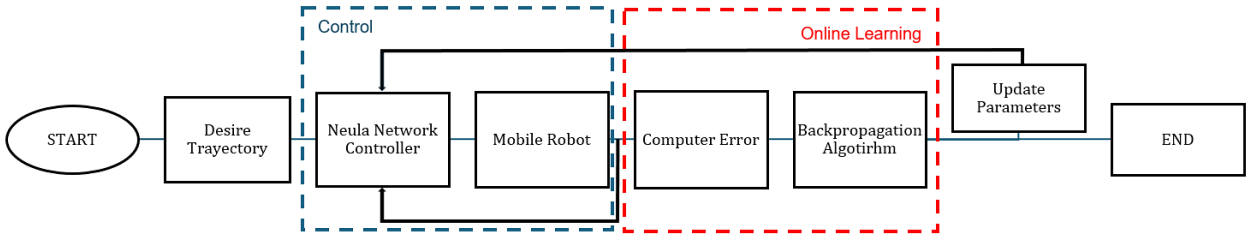


Figure 4. Methodology Flowchart

The neural network architecture was designed based on the kinematic controller's control laws. By deriving the control law from Equation 13, we obtained the following:

$$u\,(t_k) = \cos\theta\,(t_k)\left(\frac{x_r(t_k) - x_r\,(t_k-1)}{T_s} + k_x\, e_x\,(t_k)\right) + \sin\theta\,(t_k)\left(\frac{y_r(t_k) - y_r\,(t_k-1}{T_s} + k_y\, e_y(t_k)\right) \tag{13}$$

$$\omega(t_k) = -\frac{1}{a}\sin\theta\,(t_k)\left(\frac{x_r(t_k) - x_r\,(t_k-1)}{T_s} + k_x\, e_x(t_k)\right) + \frac{1}{a}\cos\theta\,(t_k)\left(\frac{y_r(t_k) - y_r\,(t_k-1)}{T_s} + k_y\, e_y(t_k)\right) \tag{14}$$

The ANN, developed based on Equations as 14 and 15, follows an MLP architecture. Its design is derived from the kinematic controller of the mobile robot (see Figure 5).

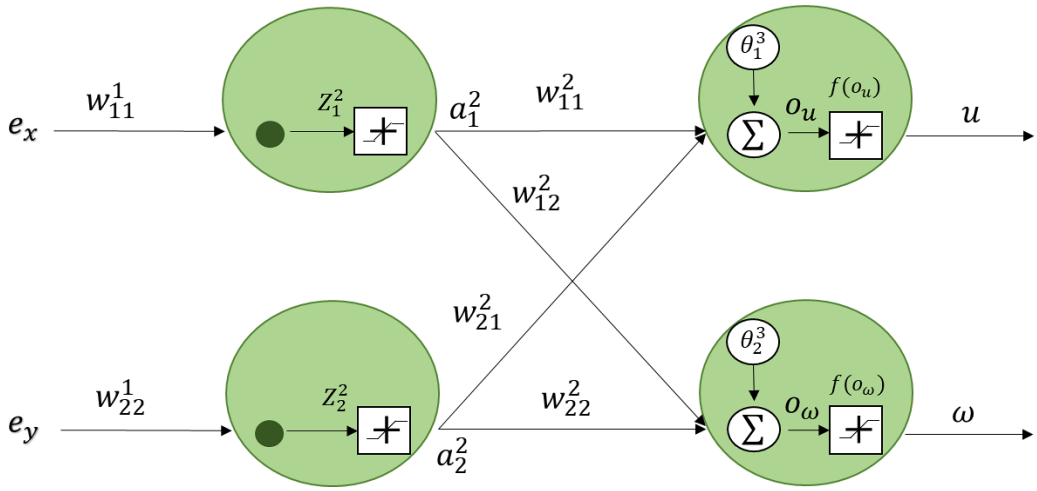


Figure 5. Architecture of the ANN designed

The inputs are directly connected to the first layer, with no gains, thus $w_{11}^1 = k_x$, $w_{22}^1 = k_y$. This simplified structure based in a P controller for a mobile robot, eliminates the need for thresholds in the first hidden layer. k_x and k_y, are the weights of the ANN, which are adapted to minimize position error and ensure rapid convergence to the desired reference. The hyperbolic tangent activation function (Equation 4) is employed to saturate the control signals, u and ω, preventing actuator overload.

In the input layer, the neurons' activation is:

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

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

With q = 2, the hidden layer shows the relationships:

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

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

In the hidden layer, the neurons' activation is calculated using Equation 4:

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

$$a_2^2 = f(Z_2^2) = tanh\ \left(k_y\ e_y(t_k)\right) \tag{20}$$

The weights in the hidden layer are computed as:

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

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

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

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

The thresholds are obtained by:

$$\varphi_1^3(t_k) = \left(\frac{x_r(t_k) - x_r\,(t_k-1)}{T_s}\right)\cos\theta\,(t_k) + \left(\frac{y_r(t_k) - y_r\,(k-1)}{T_s}\right)\sin\theta\,(t_k) \tag{25}$$

$$\varphi_2^3(t_k) = -\frac{1}{a}sin\theta(t_k)\,\left(\frac{x_r(t_k) - x_r\,(t_k-1)}{T_s}\right) + \frac{1}{a}cos\theta(t_k)\,\left(\frac{y_r(t_k) - y_r\,(t_k-1)}{T_s}\right) \tag{26}$$

Using the Equation 5, in the layer $q = 3$, the neurons' activation is:

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

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

From:

$$o_u\,(t_k) = cos\theta(t_k)\ f(Z_1^2) + sin\theta\,(t_k)\,f\,(Z_2^2) + \varphi_1^3\,(t_k) \tag{29}$$

$$o_\omega\,(t_k) = -\frac{1}{a}sin\theta(t_k)\ f(Z_1^2) + \frac{1}{a}cos\theta\,(t_k)\,f\,(Z_2^2) + \varphi_2^3\,(t_k) \tag{30}$$

The controller is designed to guarantee that the robot's actual position closely matches the desired trajectory, minimizing tracking errors in both axes (x, y). The gains in the controller (k_x,k_y) are dynamically modified using the backpropagation algorithm. This iterative process iteratively refines the gains until the difference relating the robot's current path and the needed trajectory is virtually eliminated.

The total error in trajectory tracking is:

$$E\,(t_k) = (1/2)\left(e^2{}_x + e^2{}_y\right) \tag{31}$$

To minimize Equation 31, k_x and k_y are going to be adjusted. The learning is posed as:

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

From Equation 10, the learning laws are:

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

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

It is necessary derivate to solve Equations 33 and 34:

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

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

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

Clearing e_x from Equation 13 and substituting $e_x(t_k) = x_r(t_k) - x(t_k)$:

$$x(t_k) = x_r(t_k) - \frac{u(t_k)}{k_x \cos\theta(t_k)} + \frac{\sin\theta(t_k)\left(y_r(t_k+1) - y_r\,(t_k) + k_y\,e_y(t_k)T_s\right)}{k_x \cos\theta(t_k)T_s} + \frac{x_r(t_k+1) - x_r\,(t_k)}{k_x T_s} \tag{39}$$

Clearing e_y from Equation 13 and substituting $e_y(t_k) = y_r\,(t_k) - y(t_k)$:

$$y(t_k) = y_d(t_k) - \frac{u(t_k)}{k_y \sin\theta(t_k)} + \frac{cos\,\theta(t_k)(x_r(t_k+1) - x_r\,(t_k) + k_x\,e_x(t_k)T_s)}{k_y \sin\theta(t_k)\,T_s} + \frac{y_r(t_k+1) - y_r\,(t_k)}{k_y T_s} \tag{40}$$

Clearing e_x from Equation 14 and substituting $e_x(t_k) = x_r\,(t_k) - x(t_k)$:

$$x(t_k) = x_r(t_k) - \frac{cos\,\theta(t_k)\left(y_r(t_k+1) - y_r(t_k) + k_y\,e_y(t_k)T_s\right)}{k_x\,sin\,\theta(t_k)T_s} + \frac{\omega(t_k)\,a}{k_x \sin\theta(t_k)} + \frac{x_r(t_k+1) - x_r\,(t_k)}{k_x T_s} \tag{41}$$

Clearing e_y from Equation 14 and substituting $e_y(k) = y_r\,(t_k) - y(t_k)$:

$$y(t_k) = y_r(t_k) - \frac{sin\,\theta(t_k)(x_r(t_k+1) - x_r\,(t_k) + k_x\,e_x(t_k)T_s)}{k_y\,cos\,\theta(t_k)\,T_s} - \frac{\omega(t_k)\,a}{k_y\,cos\,\theta(t_k)} + \frac{y_r(t_k+1) - y_r\,(t_k)}{k_y T_s} \tag{42}$$

Obtaining the partial derivative:

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

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

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

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

From Equation 31:

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

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

Replacing Equation 5 in Equations 27 and 28:

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

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

From Equations 29 and 30, it is obtained:

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

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

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

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

By Equation 5, the derivative of the activation function:

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

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

The derivatives, from Equations 18 and 19 are:

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

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

In general, the proposed approach uses a library to send control inputs to the robot and receive sensor data. By representing the NAO robot kinematics as a black box, the kinematic model of a mobile robot is used to compute positions and velocities based on the provided inputs. This abstraction simplifies and enhances the efficiency of both the controller design process and the simulation of trajectory tracking.

This methodology is inherently generalizable to other robotic platforms, including humanoid robots and robots with different kinematic models, due to its modular structure and reliance on a neural network-based dynamic gain control. The neural network's ability to learn and adapt to different kinematic configurations enables the approach to accommodate a wide range of robots, from those with simpler motion dynamics to those with more complex mechanisms.

However, generalizing this method to robots with significantly different motion dynamics or without dedicated control libraries for joint-level actuation presents challenges. In such cases, the absence of a predefined control interface may affect the implementation and performance of the kinematic or dynamic model. To address this, additional preprocessing layers or tailored neural network training may be required to ensure accurate trajectory tracking.

Despite these considerations, the adaptability of the proposed controller to diverse robot types highlights its potential for broader application, provided that necessary adjustments are made to account for variations in kinematic and dynamic properties.

5- Experimental Tests

CoppeliaSim Edu, a comprehensive robotic simulation platform, offers an integrated development environment for a wide range of robotics applications. Its strength lies in its ability to precisely simulate individual robot components, considering their kinematics, dynamics, and interactions with the environment. This granular control is achieved through various control mechanisms [31, 32]. MATLAB-based controllers are seamlessly integrated into CoppeliaSim via plug-ins, enabling flexible and efficient simulation. Figure 6 illustrates the navigation interface of the corresponding software, including the NAO robot.

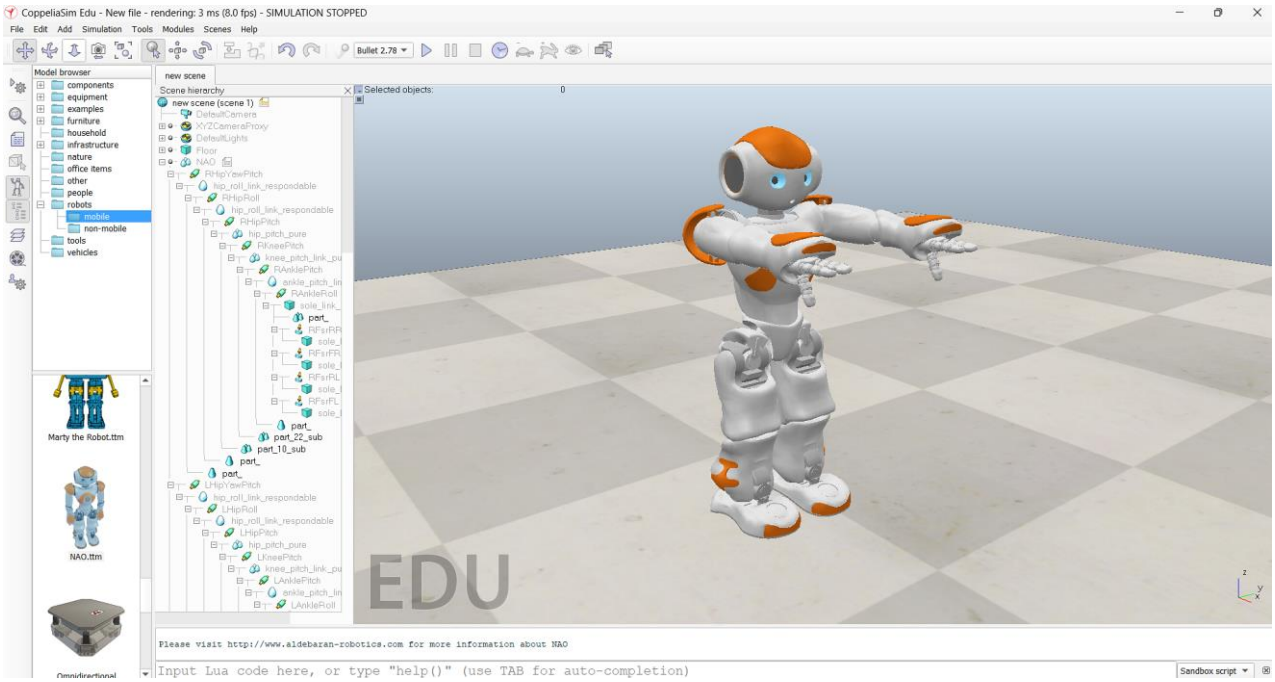


Figure 6. NAO in CoppeliaSim

The proposed neural network-based controller is implemented and tested on the NAO humanoid robot, a programmable and interactive platform developed by Aldebaran Robotics. NAO, widely used in research, education, and human-robot interaction applications, features 25 degrees of freedom: 5 in each leg, 5 in each arm, 2 in each hand, and 2 in the neck. Its advanced hardware enables real-time data acquisition, which is essential for ensuring precise trajectory tracking and maintaining system stability [24, 31].

Our method employs an online learning mechanism that dynamically updates the neural network during operation, removing the need for offline training. The computational load of the neural network is minimal and carefully managed to align with the robot processing capabilities. This is achieved by optimizing the architecture, limiting the number of layers and neurons, and using efficient algorithms such as backpropagation. These measures ensure that the controller can dynamically adjust gains without introducing delays, maintaining real-time performance.

The maximum speed of the NAO robot, 0.6 km/h, helps ensure that the controller operates within feasible computational limits, preventing saturation of the processing cache or excessive power consumption. Energy efficiency is a critical consideration for small humanoid robots like NAO, and the controller is specifically designed to minimize unnecessary computations. By focusing on error correction at 20 ms intervals, the proposed approach keeps energy consumption within acceptable bounds, even during extended operations. This balance between real-time control and energy efficiency makes the method practical and effective for trajectory tracking in small humanoid robots.

Empirical tests in the CoppeliaSim environment confirm that the real-time learning process does not negatively affect the robot responsiveness. Instead, it improves trajectory tracking by generating smoother control signals and enabling faster convergence to the desired path, even in the presence of disturbances. For robots with more degrees of freedom or more complex dynamics, additional optimization of the learning algorithm or hardware acceleration may be required to maintain similar performance.

Overall, the proposed method achieves a balance between adaptability, computational efficiency, and energy consumption. It is designed to operate efficiently within the NAO robot constraints while improving trajectory tracking performance in real time.

The controller will be evaluated in the following trajectories: circular, square, and lemniscate (Equations 56, 57, and 58, respectively). To assess the controller's performance, both graphical and numerical comparisons will be conducted $(x, y) = (0, 0)m$ is the starting point for all trajectories.

$$\begin{cases} x_{ref}(k) = \cos(0.0167\pi k T_0) \\ y_{ref}(k) = \sin(0.0167\pi k T_0) \end{cases} \quad (59)$$

$$\begin{cases} x_{ref}(k) = (0.5 - 0.005kT_0) \forall kT_0 \in [0{,}200];\ -0.5\ \forall kT_0 \in [200{,}400]; \\ (0.005 * (kT_0 - 400) - 0.5) \forall kT_0 \in [400{,}600];\ 0.5\ \forall kT_0 \in [600{,}800]; \\ y_{ref}(k) = 0.5\ \ \forall kT_0 \in [0{,}200];\ (0.5 - 0.005 * (kT_0 - 200))\ \forall kT_0 \in [200{,}400]; \\ -0.5\ \forall kT_0 \in [400{,}600];\ (0.005 * (kT_0 - 600) - 0.5) \forall kT_0 \in [600{,}800]; \end{cases} \quad (60)$$

$$\begin{cases} x_{ref}(k) = \sin(0.0143\pi k T_0) \\ y_{ref}(k) = \sin(0.0286\pi k T_0) \end{cases} \quad (61)$$

$k_x = 1, k_y = 1, \alpha = 0.00004$ are the initial parameters for ANN controller presented in Equations 13 and14. In addition, $k_x = 1, k_y = 1$ are the initial parameters for the conventional kinematic controller Equation 12.

The gain values were determined through a heuristic method, balancing various trade-offs. Lower gains result in slower convergence to the desired trajectory, whereas higher gains can speed up convergence but may reduce tracking accuracy and compromise stability. Moreover, elevated gain values increase linear and angular velocities, which could place additional stress on the actuators.

To determine the efficacy of the neural network-based kinematic controller, we will compare it to a conventional proportional kinematic controller. This choice is motivated by the widespread use of proportional controllers in trajectory-tracking literature. Many existing methods concentrate on addressing the dynamic within cascade structures, highlighting the exterior loop that controls kinematics. Our proposed neural network-based controller aims to improve upon this aspect.

The comparison of controller performance will be based on performance indexes, numerical metrics used in control systems to evaluate effectiveness and guide parameter selection. Optimal performance is typically achieved by minimizing these indices.

The ISE index (Equation 62) is a key metric for system tuning. Minimizing the ISE improves the performance and reduce the energy consumption required to reach the corresponding reference [33].

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

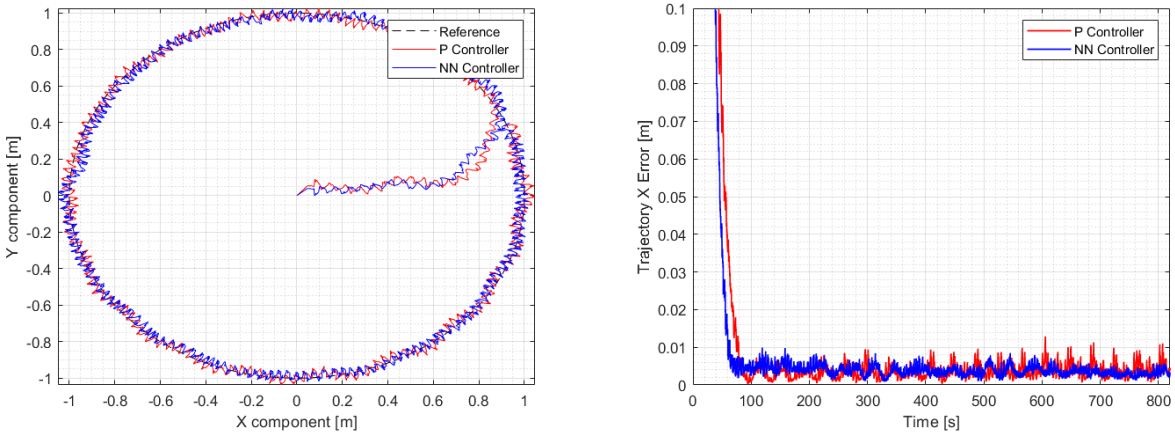
IAE index (Equation 63), is employed in system calibration. Minimizing the IAE index ensures an adequate damping and transient response [25].

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

ISU index (Equation 64) quantifies the controller effort. Minimizing the ISU index reduces the energy consumption required by the controller [26].

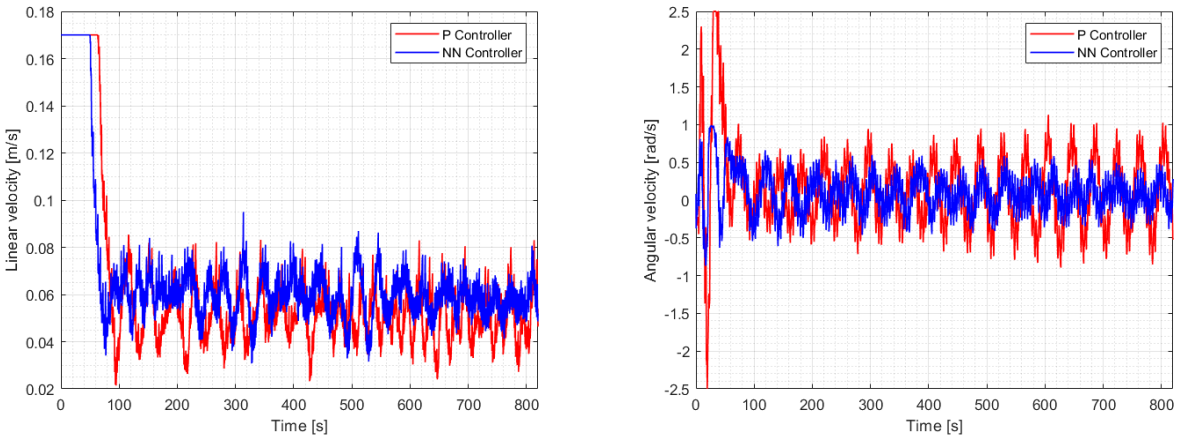
$$ISU = \int_0^t u^2(t)dt \quad (64)$$

Figure 7-a highlights the superior trajectory tracking performance of the ANN controller compared to the proportional controller, while Figure 7-b depicts the corresponding error. The ANN controller achieves smoother and less oscillatory behavior, leading to faster convergence, especially during the initial stages of the simulation. In contrast, as illustrated in Figure 8, the proportional controller produces significant and excessive fluctuations in the control signals for both velocities. These oscillations not only reduce tracking accuracy but also pose a risk to the longevity of the humanoid actuators. Moreover, the saturation of the linear velocity indicates that the humanoid is operating at its maximum speed limits. Although the oscillations diminish over time as the humanoid aligns with the desired trajectory, the ANN controller's stable and efficient control signals clearly outperform the proportional controller, offering improved tracking precision and enhanced actuator durability.



(a) (b)

Figure 7. a) Circular Trajectory; b) Tracking Error



(a) (b)

Figure 8. Velocity a) Linear; b) Angular

Figure 9 illustrates the dynamic evolution of the controller gains as the humanoid navigates the trajectory, showcasing the real-time adjustments achieved through online learning. The ANN controller effectively reduces oscillations, resulting in smoother and more accurate trajectory tracking. This improvement stems from the controller's ability to automatically fine-tune the plant gains, enabling faster convergence to the desired path. As a result, the linear speed control signal remains within acceptable limits, avoiding saturation and ensuring optimal performance.

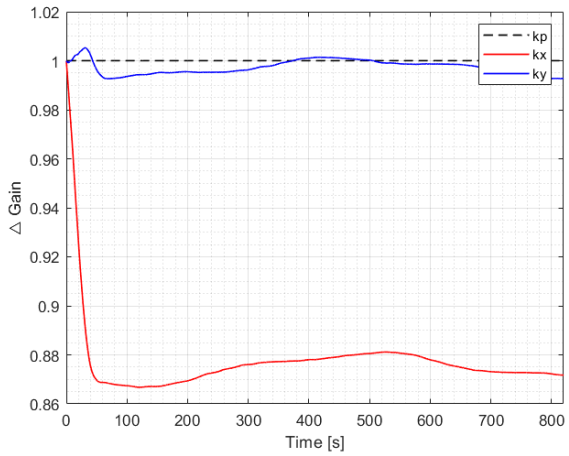
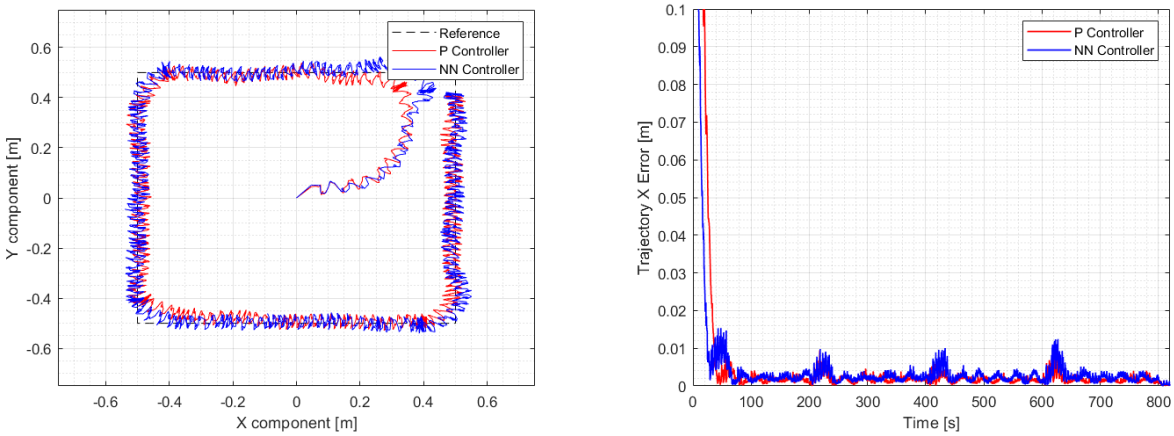


Figure 9. Adaptive Kinematic Control with Neural Networks for Circular Trajectories

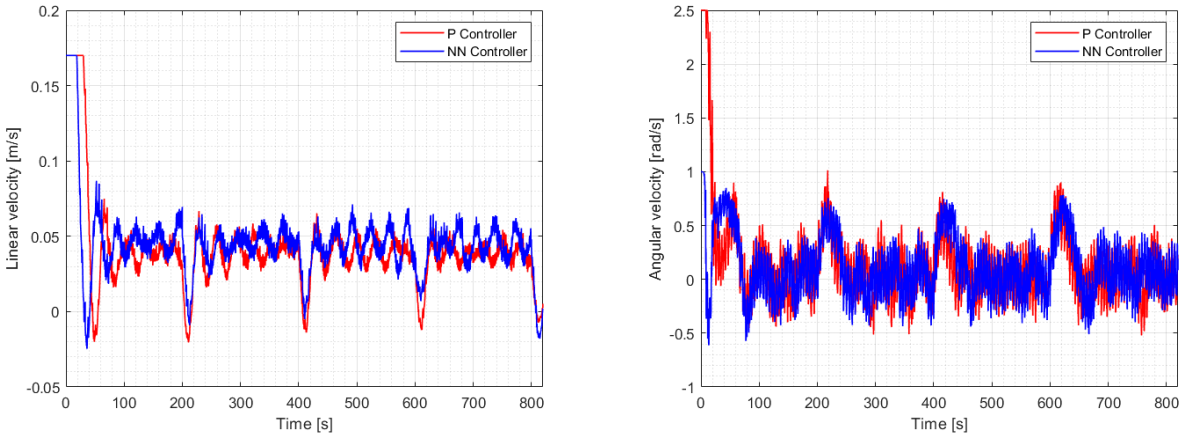
Figure 10-a demonstrates the enhanced performance of the humanoid robot on a 1-meter square trajectory achieved using the ANN controller. Figure 10-b highlights the significant reduction in oscillations and the faster convergence achieved by the ANN controller compared to the proportional (P) controller. These advantages are particularly evident at the test's starting point and when navigating around corners.



(a) (b)

Figure 10. a) Square Trajectory; b) Tracking Error

As shown in Figure 11, the Proportional controller exhibits significant and excessive fluctuations in the control signals, which hinder the humanoid's ability to maintain effective control.



(a) (b)

Figure 11. Velocity a) Linear; b) Angular

Figure 12 illustrates the dynamic adaptive gain adjustments of the controller as the robot follows its trajectory. The neural network-based kinematic controller successfully reduces oscillations, resulting in a smoother and more precise path. This improvement is due to the controller's capability to automatically optimize plant gains in real time, allowing the humanoid's trajectory to rapidly align with the desired path.

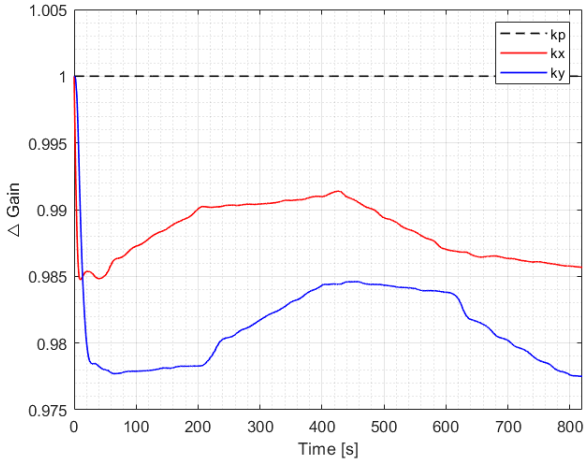
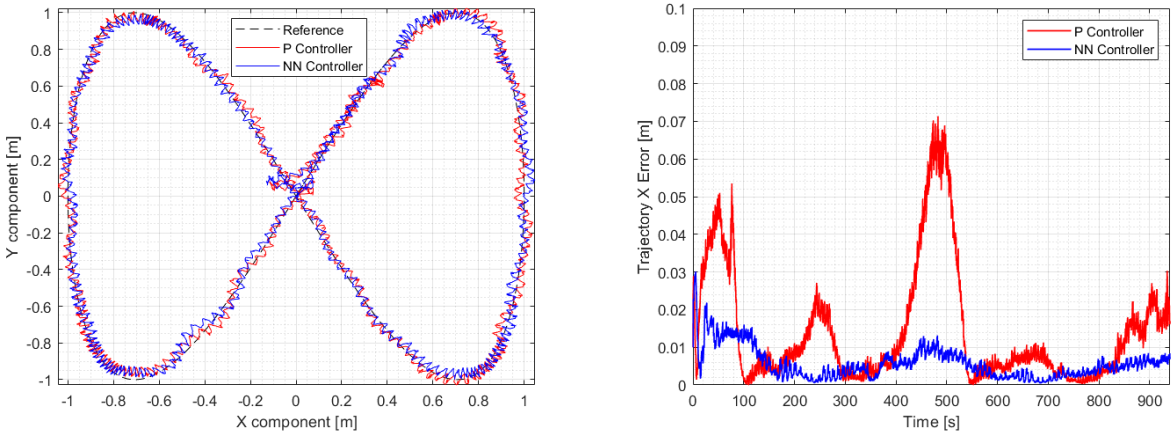


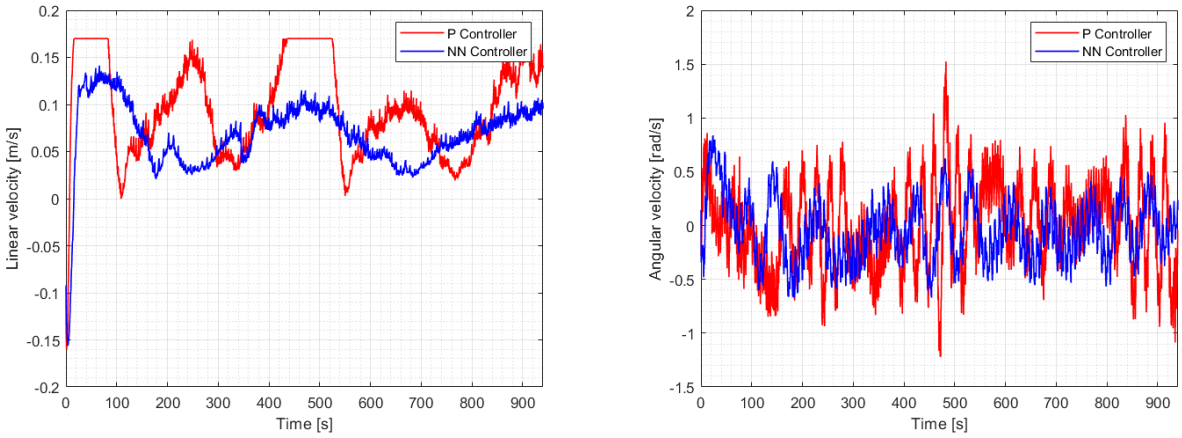
Figure 12. Adaptive Gain Control Using Neural Networks for Square Trajectories

Figure 13-a illustrates the humanoid's performance on a lemniscate trajectory, while Figure 13-b depicts the corresponding tracking error. Notably, the ANN controller achieves a significant reduction in oscillations within the angular velocity, as shown in Figure 14. Despite this improvement, it is important to emphasize that both controllers effectively track the trajectory.



(a) (b)

Figure 13. a) Lemniscate Trajectory; b) Tracking Error



(a) (b)

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

Figure 15 visually illustrates the dynamic adjustment of controller gains as the robot navigates its intended trajectory. These real-time adjustments highlight the adaptive capabilities of the control system, enabling it to respond effectively to changes in the operating conditions.

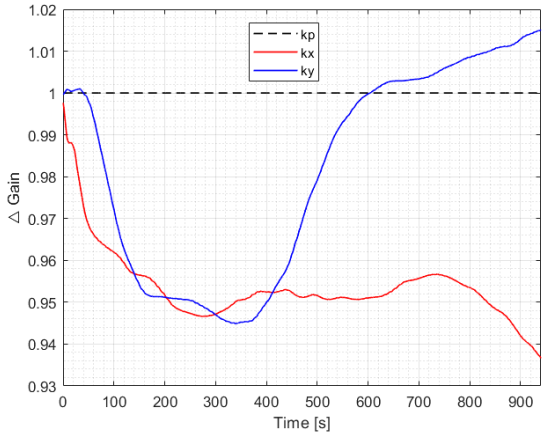


Figure 15. Optimizing Neural Network Gains for Lemniscate Control

This adaptability is crucial for handling the inherent nonlinearities and disturbances within the system, such as variations in the robot's workspace or unexpected perturbations. By dynamically fine-tuning the gains, the controller maintains stability and ensures that the trajectory tracking remains precise, even under challenging conditions. This behavior underscores the robustness of the control strategy and its ability to optimize performance while minimizing oscillations and maintaining the control signals within acceptable bounds.

In general terms, the traditional Proportional (P) controller and the ANN controller are both capable of tracking the chosen trajectory. However, when the robot deviates from the reference point, the advantages of the proposed ANN controller become evident. The ANN controller effectively minimizes oscillations, resulting in a smoother and more stable approach. These oscillations, particularly in linear and angular velocity, can negatively impact actuator performance if they become excessive.

The neural network-based controller significantly reduces oscillations in angular velocity, producing smoother signals that remain within acceptable limits. This rapid convergence to the reference value represents a key advantage of this approach over traditional methods.

Oscillations in linear velocity primarily arise from the system's inherent nonlinearities. While controllers can stabilize the system, mechanical and dynamic nonlinearities may still induce minor oscillations, especially near workspace boundaries or under varying operating conditions. Furthermore, interactions between axes in robots with multiple degrees of freedom can contribute to unintended oscillations, even with independent axis stabilization.

Figure 16 provides a quantitative comparison of the ANN controller's performance against the conventional P controller. The ANN controller achieves a substantial improvement of over 10% in linear speed indexes, with even greater gains in angular velocity indexes, exceeding 100% on certain trajectories. These results validate the effectiveness of the proposed methodology. Although the quantitative improvements in the ISE and IAE indexes are relatively modest, the ANN controller delivers a notable qualitative enhancement in trajectory tracking accuracy.

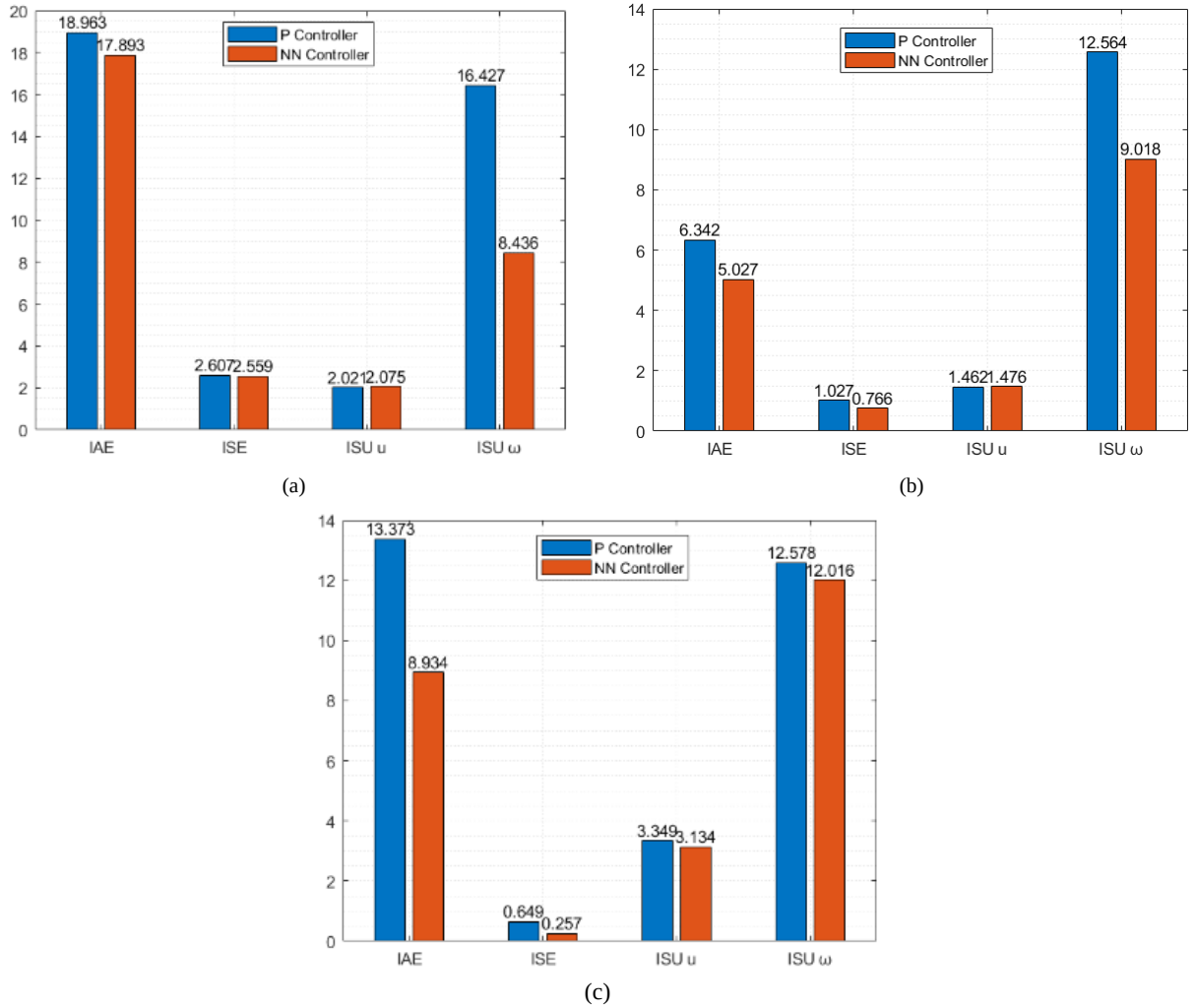


Figure 16. Evaluation Indexes a) Circle, b) Square, c) Lemniscate

6- Conclusion

The present paper shows a neural network-based controller specifically designed for trajectory tracking in the NAO humanoid robot. The controller is rooted in a kinematic controller model and is structured as a multilayer perceptron. It employs an online self-tuning mechanism using backpropagation to dynamically adjust controller gains and minimize tracking errors. Unlike traditional approaches, this method eliminates the need for offline training.

Qualitative comparisons with the widely used Proportional (P) controller highlight the improved performance characteristics of the proposed neural network controller. The neural network controller exhibits significantly smoother operation and faster convergence to the reference trajectory, particularly during the initial stages of the simulation. These improvements result from the controller's ability to minimize trajectory errors and generate more stable control signals for both linear and angular velocity. This stability not only enhances tracking accuracy but also contributes to increased actuator durability, ensuring better long-term performance.

The proposed method is designed for trajectory tracking of a single robot, but it is scalable to multi-robot systems with certain adaptations. Extending the approach would require addressing the combined dynamics and kinematics of the entire system, which increases complexity. If the NAOqi library supports simultaneous robot control, the method could leverage this capability as a black-box system to manage the position and velocity of each robot, streamlining implementation through centralized data acquisition and control. However, additional mechanisms for inter-robot communication, collision avoidance, and coordination would be necessary to ensure effective interaction among robots in shared workspaces. Modifying the neural network architecture to handle multiple inputs and outputs representing all robot states, combined with distributed control strategies, could further enhance scalability and maintain computational efficiency in multi-robot scenarios.

6-1- Future Work

The simulation results demonstrate the potential of the proposed neural network-based controller; however, real-world scenarios introduce challenges. A key concern is the computational load, as real-time calculations may strain the robot's onboard processing capabilities, potentially affecting performance. In future work, the proposed controller will be implemented on the NAO robot to validate its adaptability and reliability under real-world conditions. Challenges such as sensor noise, communication delays, hardware constraints, and environmental factors like uneven terrain or external disturbances will be addressed through strategies including advanced filtering techniques, hardware-in-the-loop testing, and real-time optimization using external processors or hardware accelerators. Additionally, experiments on physical robots will assess the controller's robustness to dynamic interactions, and reinforcement learning or hybrid AI methods will be explored to enhance performance in complex environments. These efforts aim to ensure the method's practicality, efficiency, and robustness for diverse robotic applications.

7- Declarations

7-1- Author Contributions

Conceptualization, L.M. and D.T.; methodology, L.M.; software, D.T.; validation, L.M., D.T., and D.P.; formal analysis, L.M., D.C., and D.P.; investigation, D.T.; resources, D.T.; data curation, L.M.; writing—original draft preparation, L.M. and D.T.; writing—review and editing, D.P.; visualization, M.T.; 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] Kahraman, C., Deveci, M., Boltürk, E., & Türk, S. (2020). Fuzzy controlled humanoid robots: A literature review. Robotics and Autonomous Systems, 134. doi:10.1016/j.robot.2020.103643.

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

[3] Parsianmehr, S., Moosavian, S. A. A., & Fakharian, A. (2017). An experimental system identification modeling and robust control for NAO humanoid robot. 4th RSI International Conference on Robotics and Mechatronics, ICRoM 2016, 506–511. doi:10.1109/ICRoM.2016.7886793.

[4] Almeida, L., Santos, V., & Ferreira, J. (2024). Enhancement of humanoid robot locomotion on slippery floors using an adaptive controller. Robotica, 42(4), 1055–1073. doi:10.1017/S0263574724000080.

[5] Iffath Unnisa Begum. (2024). Role of Artificial Intelligence in Higher Education- An Empirical Investigation. International Research Journal on Advanced Engineering and Management, 2(03), 49–53. doi:10.47392/irjaem.2024.0009.

[6] Obrenovic, B., Gu, X., Wang, G., Godinic, D., & Jakhongirov, I. (2024). Generative AI and human–robot interaction: implications and future agenda for business, society and ethics. AI & Society, 40(2), 677–690. doi:10.1007/s00146-024-01889-0.

[7] Mahamood, S. F., Fikry, A., Hamzah, M. I., Khalid, M. M., & Bhari, A. (2023). Fiqh Robotic For Artificial Intelligent In Humanoids Used For Therapy, Services and Other Social Activities: An Integration of Artificial Intelligence (AI) and Maqasid Shariah. Journal of Fatwa Management and Research, 28(2), 1–13. doi:10.33102/jfatwa.vol28no2.527.

[8] Podpečan, V. (2023). Can You Dance? A Study of Child–Robot Interaction and Emotional Response Using the NAO Robot. Multimodal Technologies and Interaction, 7(9), 85. doi:10.3390/mti7090085.

[9] Venkataswamy, R., Janamala, V., & Cherukuri, R. C. (2023). Realization of Humanoid Doctor and Real-Time Diagnostics of Disease Using Internet of Things, Edge Impulse Platform, and ChatGPT. Annals of Biomedical Engineering, 52(4), 738–740. doi:10.1007/s10439-023-03316-9.

[10] Pot, E., Monceaux, J., Gelin, R., & Maisonnier, B. (2009). Choregraphe: a graphical tool for humanoid robot programming. RO-MAN 2009 - The 18th IEEE International Symposium on Robot and Human Interactive Communication, 46-51. doi:10.1109/roman.2009.5326209.

[11] Ramkumar, A., Akhil Krishna, U., Madhan, M. S., & Prajit, K. K. (2019). Control of Nao robot arm using Myo Armband. International Journal of Innovative Technology and Exploring Engineering, 8(9S2), 409–413. doi:10.35940/ijitee.I1087.0789S219.

[12] 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.

[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] 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), 225–230. doi:10.1109/ssd.2019.8893152.

[16] 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.

[17] 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.

[18] Yildirim, S., Savas, S., & Andruskiene, J. (2021). Controller Gain Tuning of a Nonholonomic Mobile Robot via Neural Network Predictor. 2021 25th International Conference Electronics, 1–6. doi:10.1109/ieeeconf52705.2021.9467455.

[19] 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.

[20] 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.

[21] Farhat, M., Kali, Y., Saad, M., Rahman, M. H., & Lopez-Herrejon, R. E. (2024). Walking position commanded NAO robot using nonlinear disturbance observer-based fixed-time terminal sliding mode. ISA Transactions, 146, 592–602. doi:10.1016/j.isatra.2023.12.026.

[22] Bai, K., Jiang, G., Jiang, G., & Liu, Z. (2019). Based on fuzzy-approximation adaptive backstepping control method for dual-arm of humanoid robot with trajectory tracking. International Journal of Advanced Robotic Systems, 16(3), 1-14. doi:10.1177/1729881419831904.

[23] 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.

[24] Alcaraz-Jiménez, J. J., Herrero-Pérez, D., & Martínez-Barberá, H. (2013). Robust feedback control of ZMP-based gait for the humanoid robot Nao. International Journal of Robotics Research, 32(9–10), 1074–1088. doi:10.1177/0278364913487566.

[25] 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.

[26] 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.

[27] Eusebio, B.-C., & Ana Yaveni, A.-B. (2014). Visual control for training unicycle-type mobile robots under the leader-follower scheme. Engineering, Research, and Technology, 15(4), 593–602. doi:10.1016/s1405-7743(14)70657-2.

[28] Kofinas, N., Orfanoudakis, E., & Lagoudakis, M. G. (2015). Complete Analytical Forward and Inverse Kinematics for the NAO Humanoid Robot. Journal of Intelligent and Robotic Systems: Theory and Applications, 77(2), 251–264. doi:10.1007/s10846-013-0015-4.

[29] Li, S., & Shen, L. (2024). Seismic Optimization Design and Application of Civil Engineering Structures Integrated with Building Robot System Technology. HighTech and Innovation Journal, 5(4), 1118-1134. doi:10.28991/HIJ-2024-05-04-017.

[30] 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.

[31] SoftBank Robotics. (2025). NAO6 the versatile humanoid robot. Available online: https://developer.softbankrobotics.com/nao6 (accessed March 2025).

[32] Shamsuddin, S., Ismail, L. I., Yussof, H., Ismarrubie Zahari, N., Bahari, S., Hashim, H., & Jaffar, A. (2011). Humanoid robot NAO: Review of control and motion exploration. IEEE International Conference on Control System, Computing and Engineering, 511-516. doi:10.1109/iccsce.2011.6190579.

[33] Hwang, J. H., Tsay, S. Y., & Hwang, C. (2000). Tuning PID Controllers for Minimizing ISE and Satisfying Specified Gain and Phase Margins. IFAC Proceedings Volumes, 33(4), 601-606. doi:10.1016/S1474-6670(17)38309-X.