Adaptive neural PD controllers for mobile manipulator trajectory tracking

Jose de Jesus Hernandez Barragan Corresp., 1 , **Jorge Daniel Rios Arrañaga** [1] , **Javier Enrique Gomez Avila** [1] , **Nancy Guadalupe Arana Daniel** [1] , **Carlos Alberto Lopez Franco** [1] , **Alma Yolanda Alanis Garcia** [1]

[1] Department of Computer Science, University of Guadalajara, Guadalajara, Jalisco, México

Corresponding Author: Jose de Jesus Hernandez Barragan
Email address: josed.hernandezb@academicos.udg.mx

Artificial intelligence techniques have been used in the industry to control complex systems; among these proposals, adaptive PID (Proportional, Integrative, Derivative) controllers are intelligent versions of the most used controller in the industry. This work presents an adaptive neuron PD controller and a multilayer neural PD controller for position tracking of a mobile manipulator. Both controllers are trained by an extended Kalman filter (EKF) algorithm. Neural networks trained with the EKF algorithm show faster learning speeds and convergence times than the training based on backpropagation. The integrative term in PID controllers eliminates the steady-state error, but it provokes oscillations and overshoot. Moreover, the cumulative error in the integral action may produce windup effects such as high settling time, poor performance, and instability. The proposed neural PD controllers adjust their gains dynamically, which eliminates the steady-state error. Then, the integrative term is not required, and oscillations and overshot are highly reduced. Removing the integral part also eliminates the need for anti-windup methodologies to deal with the windup effects. Mobile manipulators are popular due to their mobile capability combined with a dexterous manipulation capability, which gives them the potential for many industrial applications. Applicability of the proposed adaptive neural controllers is presented by simulating experimental results on a KUKA Youbot mobile manipulator, presenting different tests and comparisons with the conventional PID controller.

Adaptive neural PD controllers for mobile manipulator trajectory tracking

Jesus Hernandez-Barragan, Jorge D. Ríos, Javier Gomez-Avila, Nancy Arana-Daniel, Carlos Lopez-Franco, and Alma Y. Alanis

University of Guadalajara, Center of Exact Sciences and Engineering, Department of Computer Science, Blvd. Gral. Marcelino García Barragán 1421, Olímpica, 44430 Guadalajara, Jalisco, Mexico.

Corresponding author:
Jesus Hernandez-Barragan*

Email address: josed.hernandezb@academicos.udg.mx

ABSTRACT

Artificial intelligence techniques have been used in the industry to control complex systems; among these proposals, adaptive PID (Proportional, Integrative, Derivative) controllers are intelligent versions of the most used controller in the industry. This work presents an adaptive neuron PD controller and a multilayer neural PD controller for position tracking of a mobile manipulator. Both controllers are trained by an extended Kalman filter (EKF) algorithm. Neural networks trained with the EKF algorithm show faster learning speeds and convergence times than the training based on backpropagation. The integrative term in PID controllers eliminates the steady-state error, but it provokes oscillations and overshoot. Moreover, the cumulative error in the integral action may produce windup effects such as high settling time, poor performance, and instability. The proposed neural PD controllers adjust their gains dynamically, which eliminates the steady-state error. Then, the integrative term is not required, and oscillations and overshot are highly reduced. Removing the integral part also eliminates the need for anti-windup methodologies to deal with the windup effects. Mobile manipulators are popular due to their mobile capability combined with a dexterous manipulation capability, which gives them the potential for many industrial applications. Applicability of the proposed adaptive neural controllers is presented by simulating experimental results on a KUKA Youbot ™ mobile manipulator, presenting different tests and comparisons with the conventional PID controller.

INTRODUCTION

Artificial intelligence (AI) has been very present in our society in the past few years; however, its use in the industry dates back to decades, Bryson (2019). Due to the recent interest in AI, many works have been reported in the literature in many research areas: control, internet of things, natural language processing, machine vision, medicine, robotics, security, social application, among others, Bryson (2019); Maglogiannis et al. (2020). When facing a control problem, the PID (Proportional Integral, Derivative) controllers are commonly used as the first approach. PID controllers still among the most popular controllers in the industry, mainly for their simplicity and good results, even if these results can vary due to uncertainties in operating conditions and environmental parameters Åström and Hägglund (1995); Ogata (2010). The main drawbacks of PID controllers are they are only adequate for a nominal process, they have a bad performance under systems uncertainties in operating conditions, and changing environmental conditions, Tian et al. (1999). It is well-known that with the knowledge of the system plan model, there are techniques to improve the selection of PID parameters; however, most of these techniques are offline methodologies, Johnson and Moradi (2006); Visioli (2006); Ogata (2010). Due to the above drawbacks, artificial intelligence has been used as a tool to solve these inconveniences.

In the literature, there are several works for adapting PID parameters, some based on artificial intelligence methodologies; among them, neural PID controllers stand out Hernandez-Barragan et al. (2020). Neural PID controllers learning capabilities allow them to adapt to unmodeled dynamics, communication time-delays, actuator saturation, among others, Ge et al. (2004); Lopez-Franco et al.

(2017); Sarangapani (2018); Gomez-Avila (2019). Moreover, they are capable of being trained online, which is necessary for the task of adapting while operating. Adaptive neural PID controllers trained with the Extended Kalman filter (EKF) algorithm have proved to show faster learning speeds and convergence times than adaptive neural PID based on backpropagation, which makes them ideal for experimental and real-time tests Hernandez-Barragan et al. (2020). Additionally, training algorithms based on Extended Kalman filter (EKF) have been proven reliable for recurrent and feedforward neural networks for control applications, which some of them are real-time applications Haykin (2004); Sanchez et al. (2010); Alanis et al. (2019); Rios et al. (2020) .

Besides the already mentioned withdraws of PID controllers, another common problem is a windup effect due to the accumulative error action of integral part of the controller. This effect produces saturation on actuators and contributes to low-performance, overshoot, high settling time, and instability, losing controllability, Visioli (2006); Kumar and Negi (2012); Hernandez-Barragan et al. (2020). Considering the previously said, including anti-windup strategies when using PID controllers is something to consider. Among proposed anti-windup strategies limiter integrator, back-calculation, and observer approach, Visioli (2006); Kumar and Negi (2012); Kheirkhahan (2017); Angel et al. (2019). The integral term is important as it allows to eliminate the steady-state error that the proportional term cannot suppress with a fixed proportional gain. However, the integral action causes oscillations and overshoot. This work proposes an adaptive neural PD controller that not requires the use of an integral part. This approach adapts its weights dynamically, eliminating the steady-state error, and oscillations and overshoot are highly suppressed. Additionally, the proposed approach does not suffer from windup effects because there is no cumulative error of an integral part.

Mobile manipulator robots combine mobile platforms and robotic arms, extending operational range and functionality, allowing mobile manipulators to accomplish tasks that are difficult or non-doable for a manipulator or a mobile platform by themselves, Sheng Lin and Goldenberg (2001); Li and Ge (2017). Among these applications: construction, health-care, nuclear reactor maintenance, manufacturing, military operations, and planetary exploration. Some of those tasks can put human lives at risk, Sheng Lin and Goldenberg (2001); Li and Ge (2017). However, these advantages come with complexity and difficulty at the time of designing controllers, Li and Ge (2017), which for some tasks, conventional PID performances may not be enough for control objectives, and adaptive intelligent techniques stand out as plausible solutions. This work proposes adaptive neural PD controllers trained online with extended Kalman filter (EKF) based training algorithms for trajectory tracking of mobile robots. The proposal includes a single neuron and a multilayer neuron controllers. Without the integral part of a PID controller, these adaptive controllers achieve a good performance, reduce overshoot and steady-state errors, having better performance than conventional PID controllers. Also, the proposed adaptive neural PD controllers are more robust than classic PID.

The remaining of this paper is organized as follows: Section presents a summary of PID and PD controllers and the description of the adaptive neural PD controller use in this work. Section includes the implementation of the proposal on mobile manipulators. Section shows the performance of the adaptive neural PD controller on simulation and experimental results on a KUKA™[1] mobile manipulator.

ADAPTIVE NEURAL PD CONTROLLERS

PID controllers consist of applying the sum of three types of control actions, proportional, integral, and derivative correctly, Visioli (2006); Temel et al. (2013). Moreover, simpler controllers can be used P, PD, and PI, which may be enough for some applications, especially linear ones and under regulated conditions. Nevertheless, the PID controller appears as the better of them. Even if there are more robust control schemes reported in the literature, the popularity of PID is mainly due to its simple implementation. Inspire in the popularity of the PID, several works have been proposed to improve PID controllers, but most of those works introduce complex methodologies.

[1]KUKA is a registered trademark of KUKA AG

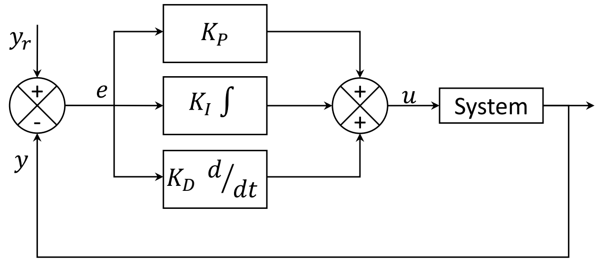


Figure 1. Control PID scheme.

The primary use of the P controller is to reduce the steady-state error of the system. As the proportional gain k_P increases, the steady-state error decreases. However, the steady-state error will not be eliminated because increasing k_P leads to overshoot, smaller amplitude, phase margin, faster dynamics, and more sensitivity to noise. This control is recommended when the system is tolerable to a constant steady-state error. The use of PI controllers is to eliminate the steady-state error resulting from the P controller. However, it harms the speed of response and system stability. This control is used when the speed of the system is not an issue. PI controller cannot decrease the rise time and eliminate the oscillations, and overshoot is always present. PD controller increases system stability by improving control since it can predict the future error of the system response. Derivative controllers respond to changing error signals, but they do not respond to constant error signals. Due to this, derivative control D is combined with proportional control P. PID controller needs the derivative gain component in addition to the PI controller to reduce the overshoot and oscillations occurring in the output response of the system. A control scheme of the PID controller is presented in 1. The manual tuning of the proportional K_P, integrative K_I, and derivative K_D gains represent an inconvenience of conventional PID controllers. This paper introduces the use of neural PID controllers to adjust themselves online during the operation of the system, even with changes in the nature of the problem.

Adaptive neural PD controller

The proposed adaptive single neuron PD (SNPD) controller is illustrated in Figure 2. The value e represents an error (1) between the reference y_r and the system output y. The inputs x_1 and x_2 are defined as the proportional (2) and the derivative (3) errors. The weights ω_1 and ω_2, are adapted online using the EKF algorithm. The weight ω_1 represents the proportional gain, and ω_2 represents the derivative gain. The value v is computed as the weighted sum of the inputs of the neuron (4). Finally, the output of the neuron $\hat{y}$ is computed with (5), where the activation function is selected as $\tanh(\cdot)$ and α scales its amplitude. The activation function reacts in the range $[-1,1]$. However, the parameter α can be selected to adequate the control action, since the output of the neuron is directly considered as the control law $u(k) = \hat{y}(k)$.

$$
\begin{aligned}
e(k) &= y_r(k) - y(k), && (1)\\
x_1(k) &= e(k), && (2)\\
x_2(k) &= e(k) - e(k-1), && (3)\\
v(k) &= \omega_1(k)x_1(k) + \omega_2(k)x_2(k), && (4)\\
\hat{y}(k) &= \alpha \tanh(v(k)). && (5)
\end{aligned}
$$

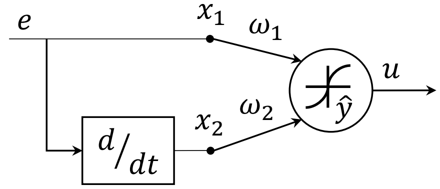


Figure 2. Adaptive single neuron PID controller.

This work proposed an adaptive neuron PD trained with the EKF algorithm. The EKF provides faster learning rates and convergence time than backpropagation, which is crucial for online training.

Adaptive multilayer PD controller

The most critical disadvantage of conventional PD controllers is that it is not suitable for nonlinear, time-variant systems. The Multilayer network PD (MNPD) scheme is depicted in Figure 3, and it consists of a fully connected neural network with one hidden layer with multiples nodes and one node at the output layer. The network input is the error and the derivative between a reference value and the system output. The neural network is trained online using an extended Kalman filter-based algorithm; the objective is to reduce the tracking error by adapting online the output of the network, which is the control signal to the system, it is $u(k) = \hat{y}(k)$.

Consider a neural network as shown in Figure 3 with 2 input signals and q nodes in the hidden layer.

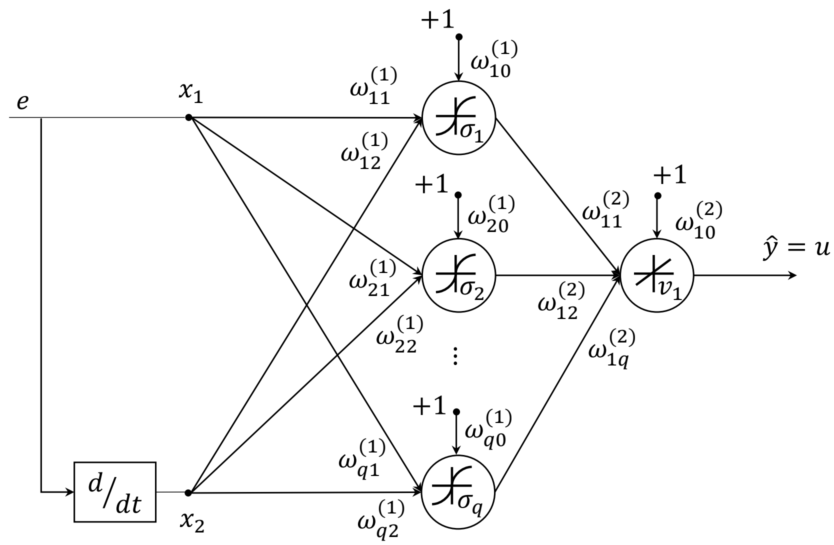


Figure 3. MLP architecture. In this case, the network has one hidden layer whose weights are denoted by $\omega_{ij}^{(1)}$ and the output layer has one node and its weights are represented with $\omega_{1j}^{(2)}$.

The output of the network is given by

$$\sigma_i(k) = \tanh\left(n_i(k)\right), \quad i = 1 \dots q, \tag{6}$$

$$n_i(k) = \sum_{j=0}^{2} \omega_{ij}^{(1)}(k) x_j(k), \quad x_0(k) = +1, \tag{7}$$

$$v_1(k) = \sum_{k=0}^{q} \omega_{1j}^{(2)}(k) u_k(k), \quad u_0(k) = +1, \tag{8}$$

$$\hat{y}(k) = v_1(k). \tag{9}$$

Extended Kalman filter based training algorithm for neural networks

The most critical disadvantage of conventional PD controllers is that it is not suitable for nonlinear, time-variant systems. The Multilayer network PD (MNPD) scheme is depicted in Figure 3, and it consists of a fully connected neural network with one hidden layer with multiples nodes and one node at the output layer. The network input is the error and the derivative between a reference value and the system output. The neural network is trained online using an extended Kalman filter-based algorithm; the objective is to reduce the tracking error by adapting online the output of the network, which is the control signal to the system; it is $u(k) = \hat{y}(k)$.

$$\mathbf{K}(k) = \mathbf{P}(k)\mathbf{H}(k)\left[\mathbf{R}(k)+\mathbf{H}^{\top}(k)\mathbf{P}(k)\mathbf{H}(k)\right]^{-1}, \tag{10}$$
$$\omega(k+1) = \omega(k)+\eta\mathbf{K}(k)\mathbf{e}(k), \tag{11}$$
$$\mathbf{P}(k+1) = \mathbf{P}(k)-\mathbf{K}(k)\mathbf{H}^{\top}(k)\mathbf{P}(k)+\mathbf{Q}(k), \tag{12}$$

$$\mathbf{h}_{ij}(k) = \left[\frac{\partial \mathrm{y}_i(k)}{\partial \omega_j(k)}\right]. \tag{13}$$

where $\omega \in \mathbb{R}^n$ is the weight vector, $\mathbf{K} \in \mathbb{R}^{n\times m}$ is the Kalman gain vector with n as the number of weights, and m the number of outputs of the neural network; $\mathbf{P} \in \mathbb{R}^{n\times n}$, $\mathbf{Q} \in \mathbb{R}^{n\times n}$, and $\mathbf{R} \in \mathbb{R}^{m\times m}$ are covariance matrices of weight estimation error, estimation noise, and error noise, respectively; $\eta \in \mathbb{R}$ is the Kalman filter learning rate, and $\mathbf{H} \in \mathbb{R}^{n\times m}$ is a matrix whose entries h_{ij} are the derivative of the neural network output with respect to each weight Eq. (13), $\mathrm{y}_i \in \mathbb{R}$ is the i-th output of the neural network and $j = 1\cdots n$, the error $\mathbf{e} \in \mathbb{R}^m$ is defined as the difference between the desired output and the neural network output, Sanchez and Alanis (2006).

Single neuron EKF training algorithm. The EKF algorithm adjusts the wights ω_1 and ω_2 for the single neuron using an online training. The single neuron scheme is composed by $n = 2$ weights and $m = 1$ neuron output. Then, the dimension of EKF matrices are $\mathbf{K} \in \mathbb{R}^{2\times 1}$, $\mathbf{P} \in \mathbb{R}^{2\times 2}$, $\mathbf{Q} \in \mathbb{R}^{2\times 2}$, $\mathbf{R} \in \mathbb{R}^{1\times 1}$ and $\mathbf{H} \in \mathbb{R}^{2\times 1}$. The weight vector is defined as $\omega \in \mathbb{R}^2$ that includes ω_1 and ω_2, and the error $e \in \mathbb{R}$ is given by (1). The matrix $\mathbf{H}$ can be computed as

$$\mathbf{H}(k) = \begin{bmatrix}\frac{\partial \hat{y}(k)}{\partial \omega_1(k)} & \frac{\partial \hat{y}(k)}{\partial \omega_2(k)}\end{bmatrix}^T = \begin{bmatrix}\frac{\partial \hat{y}(k)}{\partial v(k)}\frac{\partial v(k)}{\partial \omega_1(k)} & \frac{\partial \hat{y}(k)}{\partial v(k)}\frac{\partial v(k)}{\partial \omega_2(k)}\end{bmatrix}^T = \begin{bmatrix}\alpha\,\mathrm{sech}^2(v(k))\,x_1(k)\\ \alpha\,\mathrm{sech}^2(v(k))\,x_2(k)\end{bmatrix}. \tag{14}$$

Multilayer network EKF training algorithm. The EKF algorithm adjusts the wights $\omega_{ij}^{(1)}(k)$ and $\omega_{j1}^{(2)}(k)$ for the multilayer network using an online training. The multilayer network scheme is composed by n weights and $m = 1$ neuron output. Then, the dimension of EKF matrices are $\mathbf{K} \in \mathbb{R}^{n\times 1}$, $\mathbf{R} \in \mathbb{R}^{1\times 1}$ and $\mathbf{H} \in \mathbb{R}^{n\times 1}$. The error $e \in \mathbb{R}$ is given by (1). The matrix $\mathbf{H}$ can be expressed as

$$\mathbf{H}(k) = \begin{bmatrix}\frac{\partial \hat{y}(k)}{\partial w_{10}^{(1)}(k)} & \frac{\partial \hat{y}(k)}{\partial w_{11}^{(1)}(k)} & \cdots & \frac{\partial \hat{y}(k)}{\partial w_{1q}^{(2)}(k)}\end{bmatrix}, \tag{15}$$
$$= \begin{matrix}[\gamma(n_1(k))x_0(k) & \cdots & \gamma(n_1(k))x_p(k) & \gamma(n_2(k))x_0(k) & \cdots \\ \gamma(n_q(k))x_p(k) & u_0(k) & u_1(k) & \cdots & u_q(k)]\end{matrix}, \tag{16}$$

with

$$\gamma(n_i(k)) = w_{1i}^{(2)}(k)\left(\mathrm{sech}^2(n_i(k))\right), \qquad i = 1,\ldots,q, \tag{17}$$

IMPLEMENTATION TO MOBILE MANIPULATOR TRAJECTORY TRACKING

This section presents a kinematics model for omnidirectional mobile manipulators. Then, the main concepts of differential kinematics are introduced for position control. Finally, the conventional PID and the proposed adaptive PD controllers are provided for the trajectory tracking of omnidirectional mobile manipulators.

Mobile manipulator kinematics

Mobile manipulators are composed of one or more manipulators attached to a mobile platform. Conventional mobile robots such as unicycles, differential drives, and car-like mobile robots are used to increase the workspace of manipulators. However, these platforms have limited movement capabilities due to their nonholonomic kinematics constraints, Li et al. (2016). In contrast, omnidirectional mobile platforms improve the movement capabilities, allowing them to move towards any position and reach any desired orientation, Zhang et al. (2016); Wu et al. (2017); Kundu et al. (2017). This section introduces a kinematic model of a mobile manipulator composed of a robotic manipulator of n Degrees of Freedom (DOF) attached to an omnidirectional mobile platform.

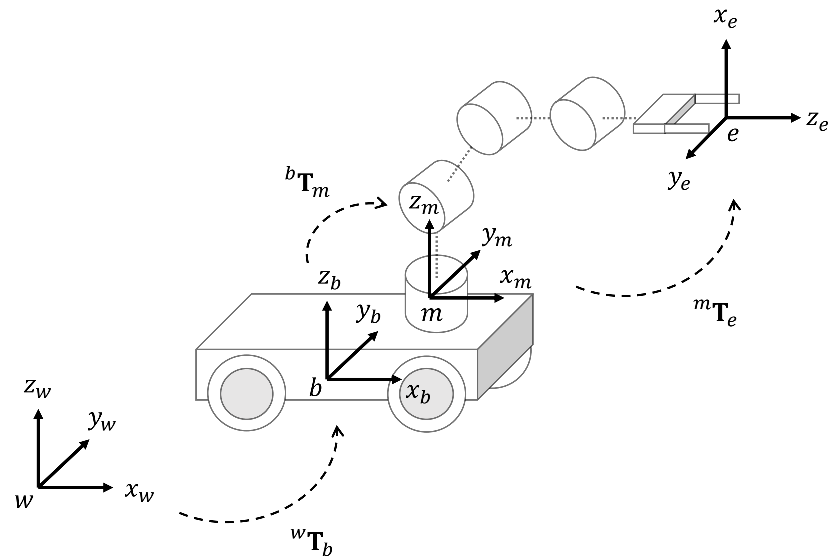


Figure 4. Kinematic chain of mobile manipulators. The transformation ${}^w\mathbf{T}_b$ is the homogeneous matrix from the world frame w to the mobile platform base frame b, ${}^b\mathbf{T}_m$ is the homogeneous matrix from b to the manipulator base frame m, ${}^m\mathbf{T}_e$ is the homogeneous matrix from m to the end-effector frame e.

The Kinematics chain of mobile manipulators is described in Figure 4. The homogeneous matrix ${}^w\mathbf{T}_b$ defines the position and orientation of the mobile platform. The transformation ${}^b\mathbf{T}_m$ is a constant homogeneous matrix between the mobile platform frame and the manipulator base. The matrix ${}^m\mathbf{T}_e$ can be computed based on the Denavit-Hartenberg (DH) model of the manipulator, Spong and Vidyasagar (2008); Lopez-Franco et al. (2018).

Considering an omnidirectional mobile platform, the pose of the robot with respect to the world frame w is given by 3 DOF, which are the positions x_b and y_b, and the orientation θ_b. Then, the matrix ${}^w\mathbf{T}_b$ can be defined as

$$
{}^w\mathbf{T}_b = \begin{bmatrix} \cos(\theta_b) & -\sin(\theta_b) & 0 & x_b \\ \sin(\theta_b) & \cos(\theta_b) & 0 & y_b \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{18}
$$

The matrix ${}^b\mathbf{T}_m$ is constant, and it adjusts the distance from the mobile platform base frame b to the manipulator base frame m. The values t_x, t_y and t_z are used to adjust the distance in the direction of the x-axis, y-axis and z-axis, respectively. If it does not need to adjust the frame orientation, then the matrix ${}^b\mathbf{T}_m$ can be described by

$$
{}^b\mathbf{T}_m = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{19}
$$

Let consider a joint variable $\mathbf{q}$ to represent the platform configuration $\mathbf{q}_\mathsf{b} = \begin{bmatrix} x_b & y_b & \theta_b \end{bmatrix}^T$ and the manipulator configuration $\mathbf{q}_\mathsf{m} = \begin{bmatrix} q_1 & q_2 & q_3 & \cdots & q_n \end{bmatrix}^T$, where q_i is a joint value for the articulation i. The joint variable for the mobile manipulator is given by $\mathbf{q} = \begin{bmatrix} \mathbf{q}_\mathsf{b}^T & \mathbf{q}_\mathsf{m}^T \end{bmatrix}^T$.

Given the joint variable $\mathbf{q}$, the computation of ${}^{w}\mathbf{T}_{e}(\mathbf{q})$ which is the forward kinematics of the mobile manipulator can be obtained as

$$ {}^{w}\mathbf{T}_{e}(\mathbf{q}) = {}^{w}\mathbf{T}_{b}(\mathbf{q}_{\mathsf{b}})\, {}^{b}\mathbf{T}_{m}\, {}^{m}\mathbf{T}_{e}(\mathbf{q}_{\mathsf{m}}), \tag{20} $$

where ${}^{w}\mathbf{T}_{e}(\mathbf{q})$ represents the end-effector pose respect to the world frame w. The matrix ${}^{w}\mathbf{T}_{e}$ is expressed as

$$ {}^{w}\mathbf{T}_{e}(\mathbf{q}) = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}, \tag{21} $$

where the orientation of the end-effector is represented by the matrix $\mathbf{R}$, and its Cartesian position is given by the vector $\mathbf{t}$. More information about homogeneous matrices, manipulators kinematics, and forward kinematics can be found in, Spong and Vidyasagar (2008); J.Craig (2005); Sciavicco and Siciliano (2008).

Differential kinematics

The inverse kinematics consists in the computation of the joint variables $\mathbf{q}$ given the end-effector pose ${}^{0}\mathbf{T}_{n}$. This computation can be solved by minimizing an error function using an iterative process based on the differential kinematics, Sciavicco and Siciliano (2008). Differential kinematics aims to find the relationship between the joint velocities $\dot{\mathbf{q}}$ and the end-effector velocity $\dot{\mathbf{t}}$. The following differential kinematics equation gives this relationship

$$ \dot{\mathbf{t}} = \mathbf{J}(\mathbf{q})\,\dot{\mathbf{q}}, \tag{22} $$

where $\mathbf{J}$ is the matrix relating the contribution of the joint velocities $\dot{\mathbf{q}}$ to the end-effector velocity $\dot{\mathbf{t}}$. The matrix $\mathbf{J}$ is called the geometric Jacobian. This Jacobian matrix can be computed as

$$ \mathbf{J}(\mathbf{q}) = \begin{bmatrix} \frac{\partial t_x}{\partial q_1} & \frac{\partial t_x}{\partial q_2} & \cdots & \frac{\partial t_x}{\partial q_n} \\ \frac{\partial t_y}{\partial q_1} & \frac{\partial t_y}{\partial q_2} & \cdots & \frac{\partial t_y}{\partial q_n} \\ \frac{\partial t_z}{\partial q_1} & \frac{\partial t_z}{\partial q_2} & \cdots & \frac{\partial t_z}{\partial q_n} \end{bmatrix}, \tag{23} $$

where $\mathbf{t} = \begin{bmatrix} t_x & t_y & t_z \end{bmatrix}^T$ is the end-effector position related to the joint variable $\mathbf{q} = \begin{bmatrix} q_1 & q_2 & \cdots & q_n \end{bmatrix}^T$.

An inverse kinematics approach consists in minimizing the error between an actual end-effector position $\mathbf{t}$ and the desired position $\mathbf{t}^*$. This error is defined as $\mathbf{e} = \mathbf{t}^* - \mathbf{t}$. The error $\mathbf{e}$ can be mapped to the joint velocities $\dot{\mathbf{q}}$ based on the differential kinematics equation. Equation (22) is rewritten to compute $\dot{\mathbf{q}}$ given $\mathbf{e}$ as

$$ \dot{\mathbf{q}} = \mathbf{J}(\mathbf{q})^{\dagger}\,\dot{\mathbf{t}} = \mathbf{J}(\mathbf{q})^{\dagger}\,\mathbf{e}, \tag{24} $$

where $\mathbf{J}^{\dagger}$ is the pseudo-inverse of $\mathbf{J}$.

A robot system with a Jacobian matrix $\mathbf{J} \in \mathbb{R}^{3\times n}$ where $n > 3$, the robot is considered redundant. Because there are more n DOF than necessary to perform a task with 3 DOF. Commonly, the combination of DOF of the mobile platform and the manipulator, represent a redundant robot. In the case of a redundant robot, the solution (24) can be generalized into

$$ \dot{\mathbf{q}} = \mathbf{J}(\mathbf{q})^{\dagger}\,\mathbf{e} + \left(\mathbf{I} - \mathbf{J}(\mathbf{q})^{\dagger}\,\mathbf{J}(\mathbf{q})\right)\dot{\mathbf{q}}_0, \tag{25} $$

where the first term minimizes the error e, the matrix $\left(\mathbf{I}-\mathbf{J}^{\dagger}\mathbf{J}\right)$ allows the protection of vector $\dot{\mathbf{q}}_0$ in the null space of $\mathbf{J}$, and $\mathbf{I}$ is the identity matrix. In the case that $\mathbf{e}=\mathbf{0}$, the result of the second term $\left(\mathbf{I}-\mathbf{J}^{\dagger}\mathbf{J}\right)\dot{\mathbf{q}}_0$ can reconfigure the joint variable $\mathbf{q}$ without changing the end-effector position $\mathbf{t}$.

In this work, it is proposed to design the vector $\dot{\mathbf{q}}_0$ to avoid singularities based on the manipulability measure $m(\mathbf{q})$, which is defined as

$$m(\mathbf{q}) = \sqrt{\det\left(\mathbf{J}(\mathbf{q})\mathbf{J}(\mathbf{q})^T\right)}. \tag{26}$$

Then, vector $\dot{\mathbf{q}}_0$ can be computed as

$$\dot{\mathbf{q}}_0 = k_0\left(\frac{\partial m(\mathbf{q})}{\partial \mathbf{q}}\right), \tag{27}$$

where $k_0 > 0$. By maximizing the manipulability measure, redundancy is exploited to move away from singularities. More detailed information about differential kinematics can be found in, Spong and Vidyasagar (2008); J.Craig (2005); Sciavicco and Siciliano (2008).

PID control design

To solve a position tracking for the mobile manipulator, the controller has to compute the joint velocities $\dot{\mathbf{q}}(k)$ at step time k, to control the motion of the mobile manipulator from the actual end-effector position $\mathbf{t}(k)$ to the desired position $\mathbf{t}(k)^*$. This section introduces the use of a discrete PID to control the mobile manipulator motion based on the error $\mathbf{e}(k) = \mathbf{t}(k)^* - \mathbf{t}(k)$, which is described as $\mathbf{e}(k) = \begin{bmatrix} e_x(k) & e_y(k) & e_z(k)\end{bmatrix}^T$.

A discrete PID control Moradi et al. (2001) can be used for each error $e_x(k)$, $e_y(k)$, and $e_z(k)$ as follows

$$u_x(k) = K_P^x e_x(k) + K_I^x \sum_{j=1}^{k} e_x(j) + K_D^x\left[e_x(k) - e_x(k-1)\right], \tag{28}$$

$$u_y(k) = K_P^y e_y(k) + K_I^y \sum_{j=1}^{k} e_y(j) + K_D^y\left[e_y(k) - e_y(k-1)\right], \tag{29}$$

$$u_z(k) = K_P^z e_z(k) + K_I^z \sum_{j=1}^{k} e_z(j) + K_D^z\left[e_z(k) - e_z(k-1)\right], \tag{30}$$

where K_P^x, K_I^x and K_D^x are the proportional, integrative and derivative gains for error e_x, respectively. Similarly, the parameters K_P^y, K_I^y and K_D^y are the gains for error e_y, and K_P^z, K_I^z and K_D^z are the gains for error e_z. The control output $\mathbf{u}(k) = \begin{bmatrix} u_x(k) & u_y(k) & u_z(k)\end{bmatrix}^T$ can be mapped to the joint velocities $\dot{\mathbf{q}}(k)$ based on (25) to control the system. This is

$$\dot{\mathbf{q}}(k) = \mathbf{J}(\mathbf{q}(k))^{\dagger}\mathbf{u}(k) + \left(\mathbf{I} - \mathbf{J}(\mathbf{q}(k))^{\dagger}\mathbf{J}(\mathbf{q}(k))\right)\dot{\mathbf{q}}_0, \tag{31}$$

Neural PD controllers implementation

In general, PID controllers are widely used due to their simplicity and performance. However, the inconvenience of PID controllers is the manual tuning of the proportional, integrative, and derivative gains. This paper presents an adaptive PID approach to overcome this inconvenience. The proposed approach can adjust this gains itself online during the tracking task. The implementation of the mobile manipulator consists of implementing both schemes presented in sections and , with their respective extended Kalman filter-based training algorithm. Figure 5 shows the general scheme for both implementation.

An adaptive neural PD control module is designed to minimize the error e_x, e_y and θ_z. Each control output u_x, u_y and u_z, are provided for each control module. These control signals $\mathbf{u}(k) = \begin{bmatrix} u_x(k) & u_y(k) & u_z(k)\end{bmatrix}^T$ are mapped to the joint velocities $\dot{\mathbf{q}}(k)$ using (25) to control the system.

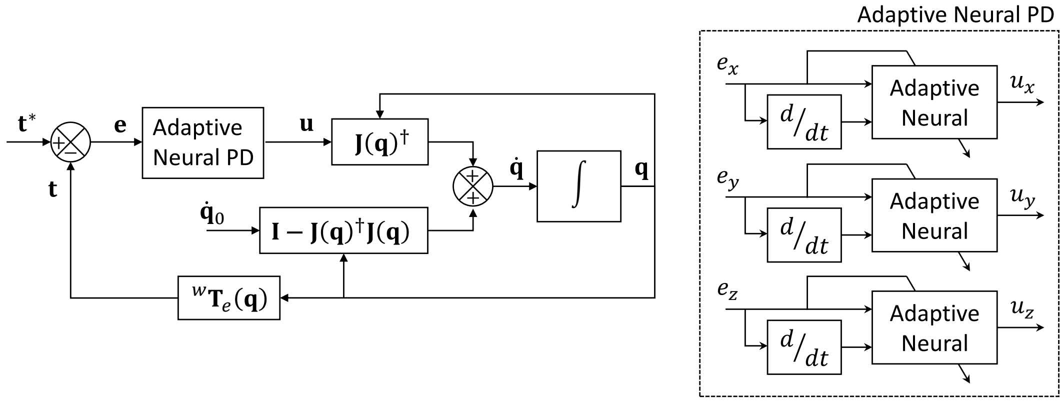


Figure 5. Adaptive Neuronal PD control scheme for the position control of mobile manipulators. The block called Adaptive Neural, can represent the single neuron scheme or the multilayer network scheme.

RESULTS

In order to show the effectiveness of the algorithms, the performance of the proposed adaptive single neuron PD (SNPD) and multilayer network PD (MNPD) controllers are compared against the conventional PD and PID controllers. Trajectories with different degrees of difficulty are considered for simulations, and real experiments on the KUKA Youbot ™ mobile manipulator, see Figure 6.

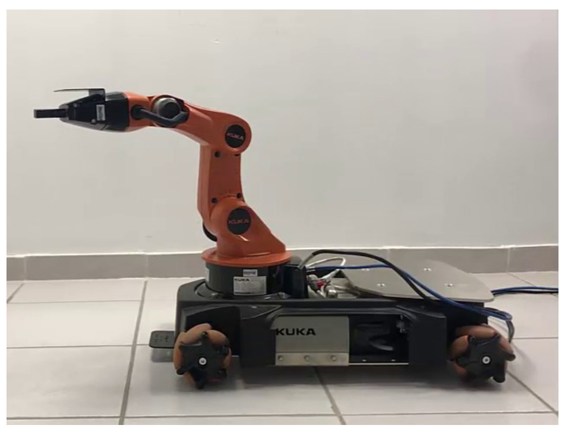


Figure 6. Omnidirectional mobile manipulator KUKA Youbot ™.

The KUKA Youbot ™ is composed of a manipulator of 5 DOF, and an omnidirectional mobile platform of 3 DOF. Respect to the mobile manipulator kinematics, the transformation ${}^{w}\mathbf{T}_b$ can be computed with the mobile platform pose, which is given by x_b, y_b and θ_b, see (18). The constant transformation ${}^{b}\mathbf{T}_m$ is considered to be

$${}^{b}\mathbf{T}_m = \begin{bmatrix} 1 & 0 & 0 & 0.140 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.151 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

these values were obtained based on the KUKA Youbot ™ technical specifications. Finally, the DH table in Table 1, is used to compute the transformation ${}^{m}\mathbf{T}_e$. The joint variable $\mathbf{q}$ for the mobile manipulator is

$$\mathbf{q} = \begin{bmatrix} x_b & y_b & \theta_b & \theta_1 & \theta_2 & \theta_3 & \theta_4 & \theta_5 \end{bmatrix}^T$$

Table 1. DH table for KUKA Youbot ™ manipulator. Values a, α, and d are parameters of the DH convention.

Joint	a (mm)	α (rad)	d (mm)	θ(rad)
1	33	$\pi/2$	147	θ_1
2	155	0	0	θ_2
3	135	0	0	θ_3
4	0	$\pi/2$	0	θ_4
5	0	0	217.5	θ_5

where the joint values $\theta_1 - \theta_5$ represent the joint configuration of the manipulator.

For simulations and real experiments, the weights in the SNPD and MNPD controllers are set randomly in every trajectory test. For PD and PID controllers, proportional gains are set as $K_P^x = K_P^y = K_P^z = 1.5$, integrative gains $K_I^x = K_I^y = K_I^z = 0.001$, and derivative gains $K_D^x = K_D^y = K_D^z = 0.5$. The gains of the PD and PID controllers were heuristically selected. The parameter setting for the EFK are: matrices $\mathbf{P}$ and $\mathbf{Q}$ are initialized as diagonal matrices with $\mathbf{P}_{ii} = 1$ and $\mathbf{Q}_{ii} = 0.1$ with $i = 1, 2, \cdots, n$, the parameter $\mathbf{R} = 0.001$, the Kalman filter learning rate $\eta = 0.2$ and $\alpha = 1$. The selection of these parameters was chosen experimentally.

The considered trajectories, at step time k are generated as follows:

$$\begin{aligned}
&\textbf{Circular trajectory}\\
x_r(k) &= 0.5,\\
y_r(k) &= 0.05\cos(0.2\,k\,\pi),\\
z_r(k) &= 0.45 + 0.05\sin(0.2\,k\,\pi).
\end{aligned}$$

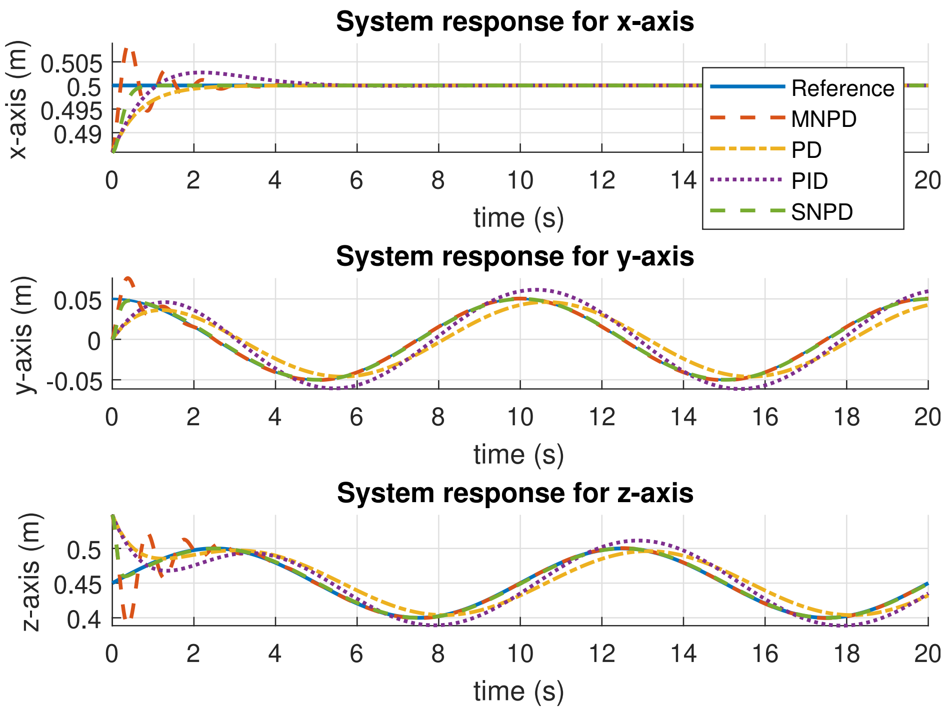
$$\begin{aligned}
&\textbf{Rose curve trajectory}\\
x_r(k) &= 0.5,\\
y_r(k) &= r(k)\cos(0.2\,k\,\pi),\\
z_r(k) &= 0.45 + r(k)\sin(0.2\,k\,\pi),\\
r(k) &= 0.035 + 0.015\cos(0.6\,k\,\pi).
\end{aligned}$$

$$\begin{aligned}
&\textbf{Trapezoidal trajectory}\\
x_r(k) &= 0.5,\\
y_r(k) &= 0.1 * k,\\
r(k) &= 0.45 + 0.08\sin(2\,y_r(k)\,\pi),\\
z_r(k) &= \begin{cases} 0.5 & \textbf{if } r(k) > 0.5\\ 0.4 & \textbf{if } r(k) < 0.4\\ r(k) & \textbf{otherwise} \end{cases}.
\end{aligned}$$

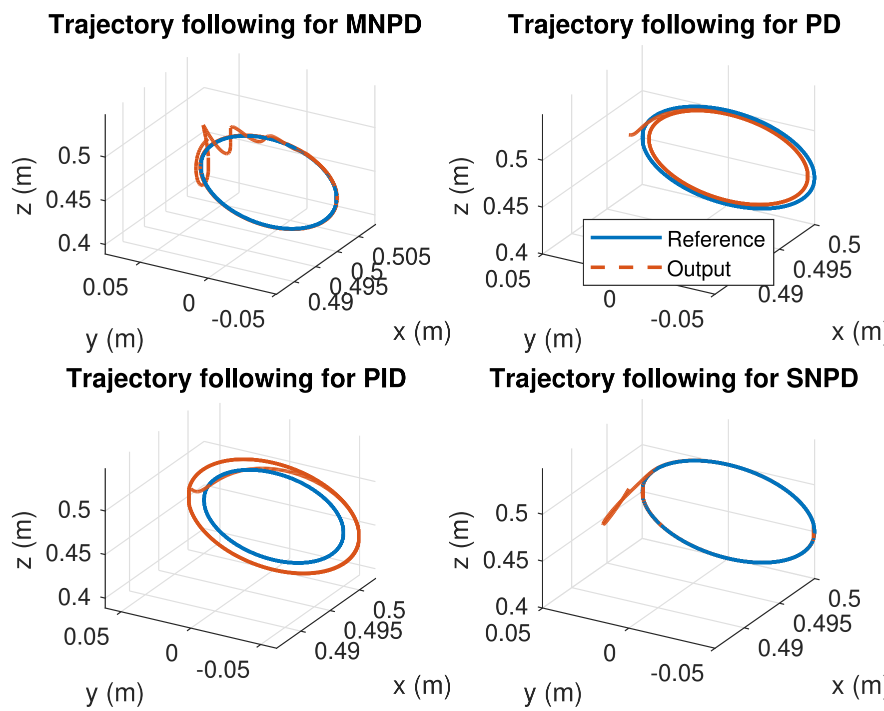
The desired position for the end-effector is defined as $\mathbf{t}(k)^* = \begin{bmatrix} x_r(k) & y_r(k) & z_r(k) \end{bmatrix}^T$. A circular and rose curve trajectories are considered for simulations. A rose curve and trapezoidal trajectories are considered for real experiments.

Simulations

The first trajectory for simulation is circular. Although conventional controllers present a good response, their gains remain constant, and they cannot adapt them to changes in the system. On the other hand, the MNPD and SNPD approaches can correctly follow the reference once the weights are adapted.



(a) System response



(b) Trajectory following

Figure 7. System response and trajectory following results for the circular trajectory.

The trajectory tracking and system response results for the circular trajectory are given in Figure 7. As shown in Figure 7 (a), the settling is almost the same for all the approaches. Although MNPD presents oscillations while the weights are adapting, the neural algorithms can follow the sinusoidal trajectory better than the conventional PID and PD. This can also be seen in Figure 7 (b), where three axes are plotted at the same time. The conventional PD reports steady-state errors in the system response for the y-axis and z-axis. The PID control minimizes this error, but overshoot is presented. Figure 7 (b) shows that PID passes over the reference caused by the integral part.

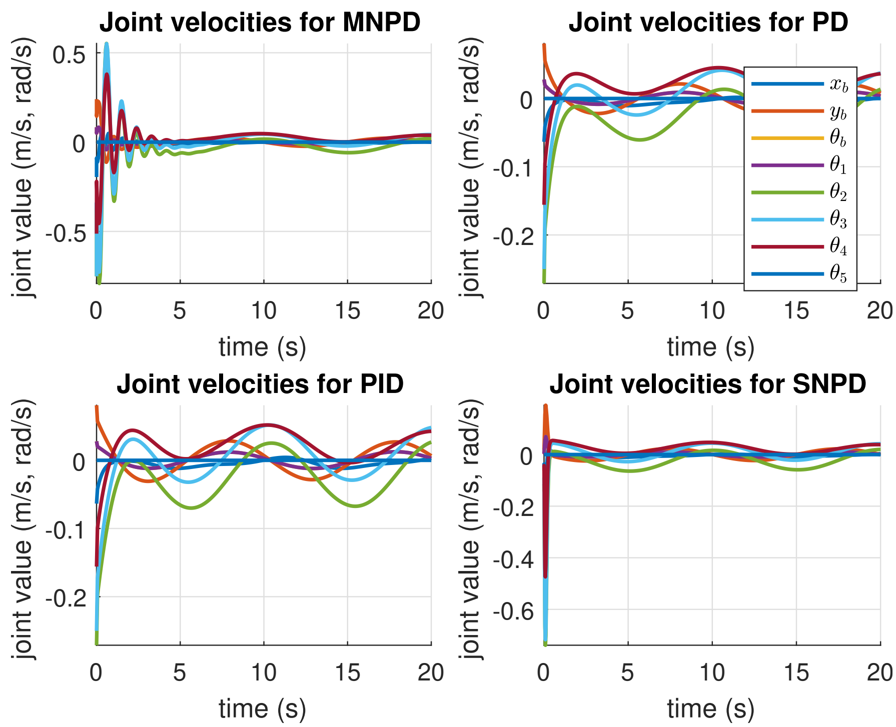


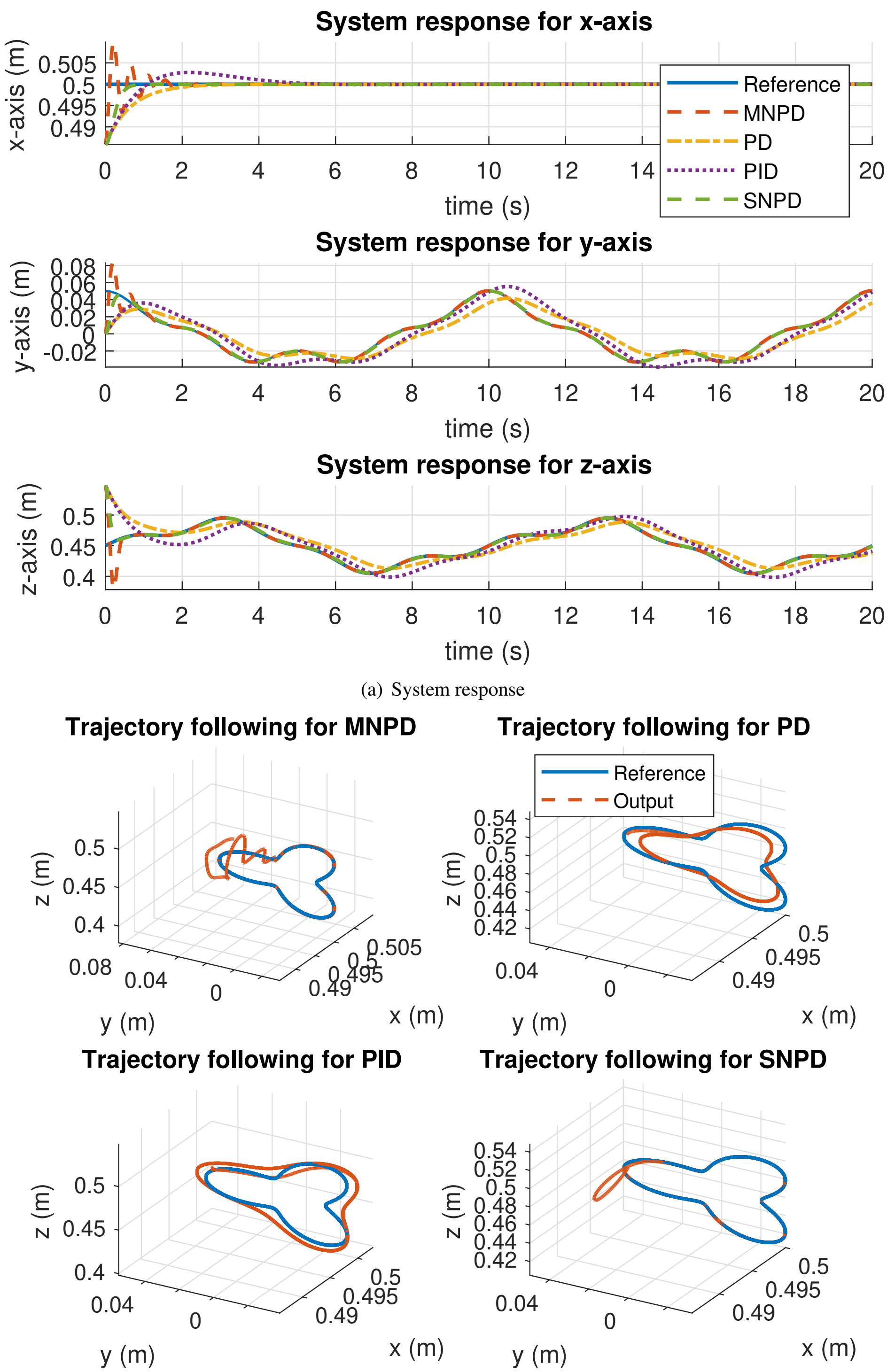
Figure 8. Velocity control signal results for the circular trajectory.

In 8, the velocity control signals for the circular trajectory are presented. At first steps, adaptive weights compute bigger control signals than PD and PID results. However, it is necessary to reach the reference with a small tracking error. Conversely, the adaptation ability of both MNPD and SNPD is shown.

Table 2. Simulation results for the circular trajectory. The best results are highlighted in bold.

Measure	Method	e_x	e_y	e_z
RMS	MNPD	8.6035×10^{-4}	2.5297×10^{-3}	6.7063×10^{-3}
	PD	1.0546×10^{-3}	1.4023×10^{-2}	1.4686×10^{-2}
	PID	1.0547×10^{-3}	1.2872×10^{-2}	1.3803×10^{-2}
	SNPD	**7.8284**$\times10^{-4}$	**2.1269**$\times10^{-3}$	**3.5693**$\times10^{-3}$
MAD	MNPD	1.3391×10^{-4}	**5.5760**$\times10^{-4}$	1.3227×10^{-3}
	PD	2.9518×10^{-4}	1.2545×10^{-2}	1.2406×10^{-2}
	PID	2.1417×10^{-4}	1.1419×10^{-2}	1.1686×10^{-2}
	SNPD	**1.2753**$\times10^{-4}$	6.0505×10^{-4}	**6.4863**$\times10^{-4}$

The Root Mean Square (RMS) and the Median Absolute Deviation (MAD) for the circular trajectory are shown in Table 2. As can be seen, the adaptive approaches present the best results, which are highlighted in bold. In this case, the SNPD control scheme reported the smallest RMS results in general.



(a) System response

(b) Trajectory following

Figure 9. System response and trajectory following results for the rose curve trajectory.

Using the same gains and parameters for the four approaches, a new trajectory is tested, and the system response and trajectory following results are shown in Figure 9. Similar results can be seen in the system response (Figure 9 (a)); the settling time is the same, and the MNPD present oscillations during the adaptations of its weights. However, in Figure 9 (b), it can be seen that the adapting approaches outperform the conventional controllers. The PD controller shows the biggest steady-state error, while

MNPD and SNPD report the smallest. The PID control improved the performance of PD, but it is needed to tune its gains to improve the performance.

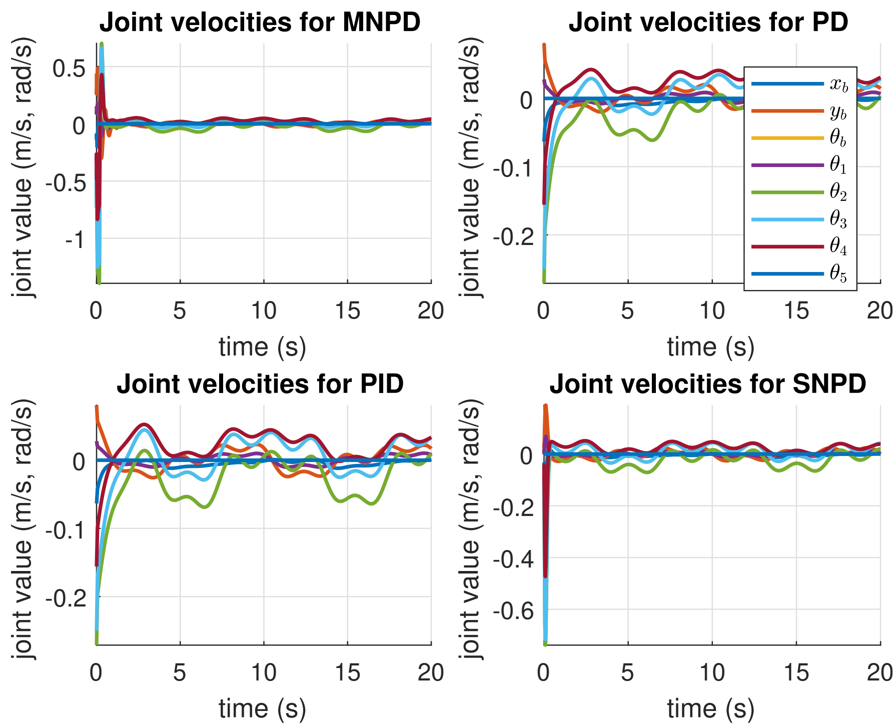


Figure 10. Velocity control signal results for the rose curve trajectory.

In Figure 10, the velocity control signals for the rose curve trajectory are reported. Similarly to the previous trajectory, at the beginning of the trajectory, the weights adaptation of the MNPD and the SNPD compute bigger control signals than PD and PID results, which are necessary to reach the reference with a small tracking error.

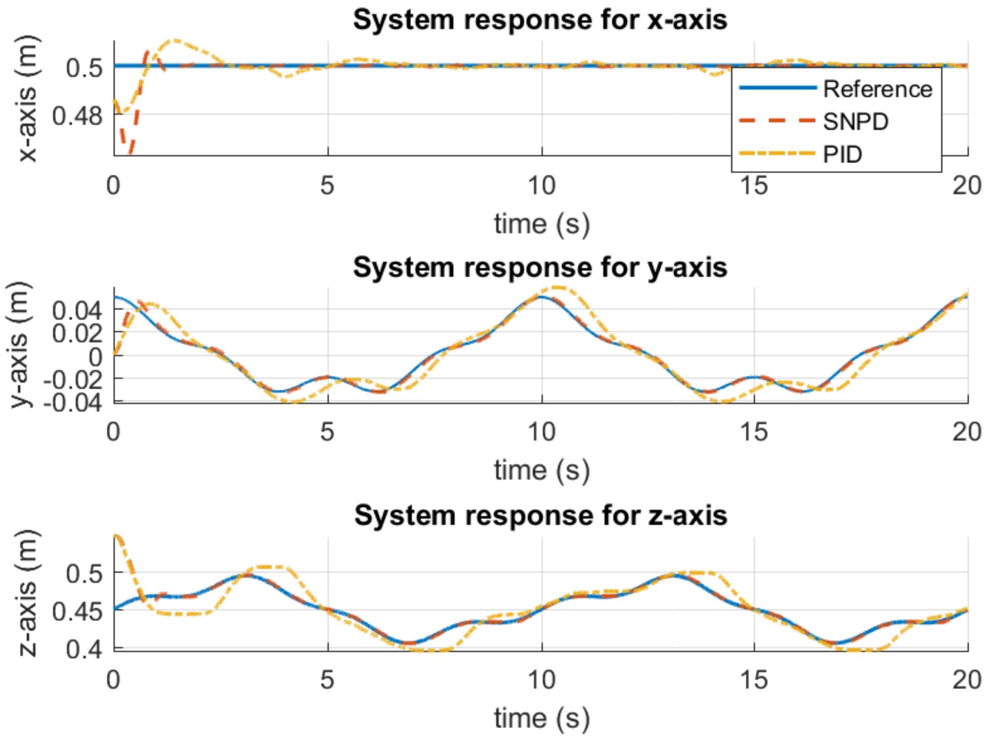
Table 3. Simulation results for the rose curve trajectory. The best results are highlighted in bold.

Measure	Method	e_x	e_y	e_z
RMS	MNPD	$\mathbf{6.9811}\times10^{-4}$	$\mathbf{2.1346}\times10^{-3}$	4.4524×10^{-3}
	PD	1.0546×10^{-3}	1.1447×10^{-2}	1.2728×10^{-2}
	PID	1.0547×10^{-3}	1.1191×10^{-2}	1.2804×10^{-2}
	SNPD	7.7519×10^{-4}	2.1415×10^{-3}	$\mathbf{3.5652}\times10^{-3}$
MAD	MNPD	$\mathbf{8.7982}\times10^{-5}$	$\mathbf{3.6619}\times10^{-4}$	$\mathbf{4.9729}\times10^{-3}$
	PD	2.9520×10^{-4}	9.9740×10^{-3}	1.0730×10^{-2}
	PID	2.1404×10^{-4}	9.1301×10^{-3}	9.9235×10^{-3}
	SNPD	1.2586×10^{-4}	5.8940×10^{-4}	6.4607×10^{-4}

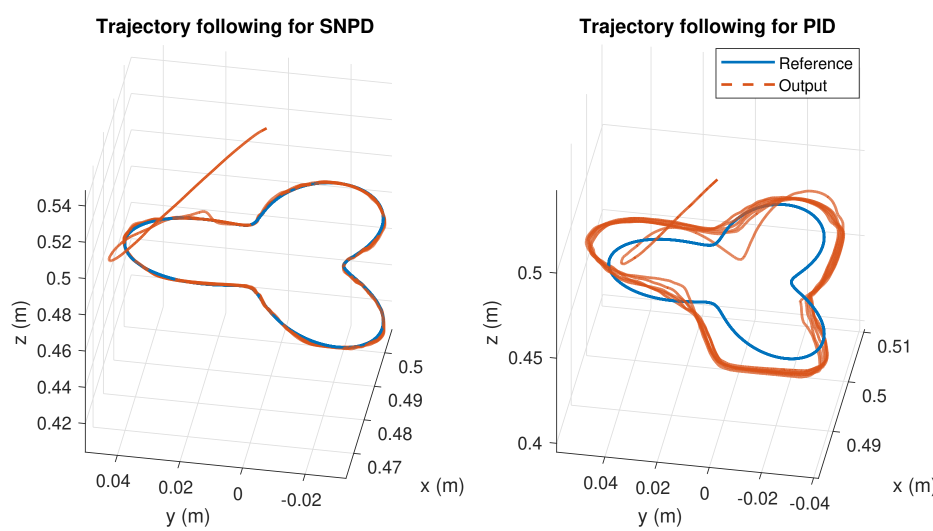
Table 3 shown the RMS and MAD results for the rose curve trajectory. The adaptive scheme has demonstrated to have better results than conventional PID and PD controllers. In this case, the MNPD controller shows the smallest RMS results in general.

Experiments

For real-time experiments, two trajectories were tested. The adaptive SNPD and MNPD controllers performed similarly in simulations. However, MNPD shows oscillations during the adaptations of its weights at the beginning. These oscillations can be eliminated if pre-trained weights are used instead of initializing them randomly every time. For this reason, it is considered to compare the SNPD controller to the PID controller since PID performed better than PD. Moreover, the same gains and parameters used for simulation were used for real-time experiments. The weights in the SNPD were randomly initialized.



(a) System response



(b) Trajectory following

Figure 11. System response and trajectory following results for the rose curve trajectory in real experiments.

In Figure 11, the system response and trajectory following for both approaches are shown. As can be seen in Figure 11 (a), the real system is not the same in simulation, and the gains of the conventional PID must be tuned again. Otherwise, it will not be able to follow the trajectory correctly and present a longer settling time. In contrast, using the same parameters as in simulation, the SNPD was able to adapt and showed shorter settling time. In Figure 11 (b) the response for the rose curve trajectory is shown. As can be seen, PID cannot follow the trajectory correctly, and it is confirmed in Table 4.

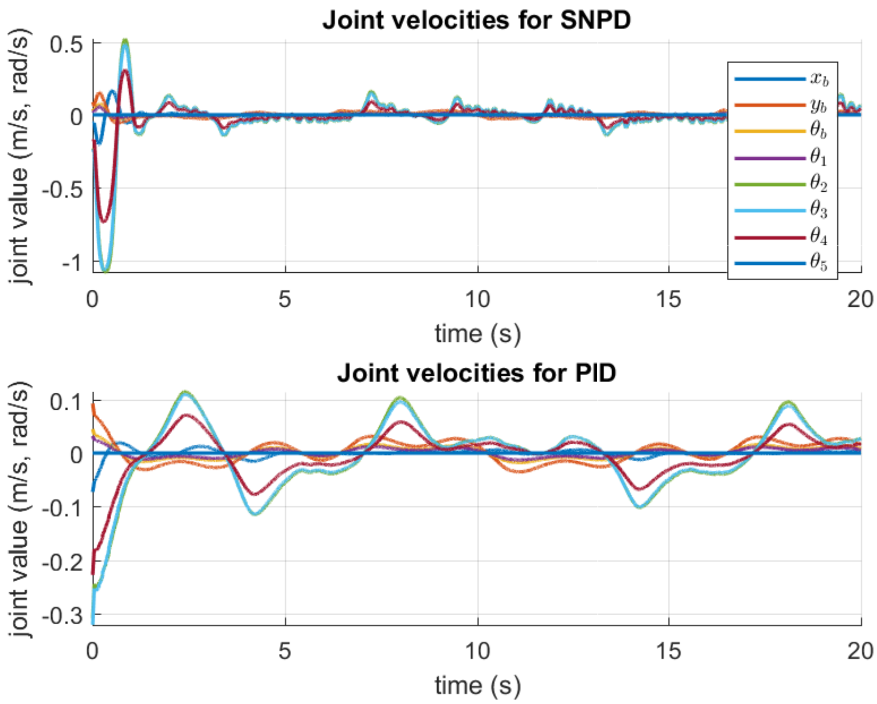


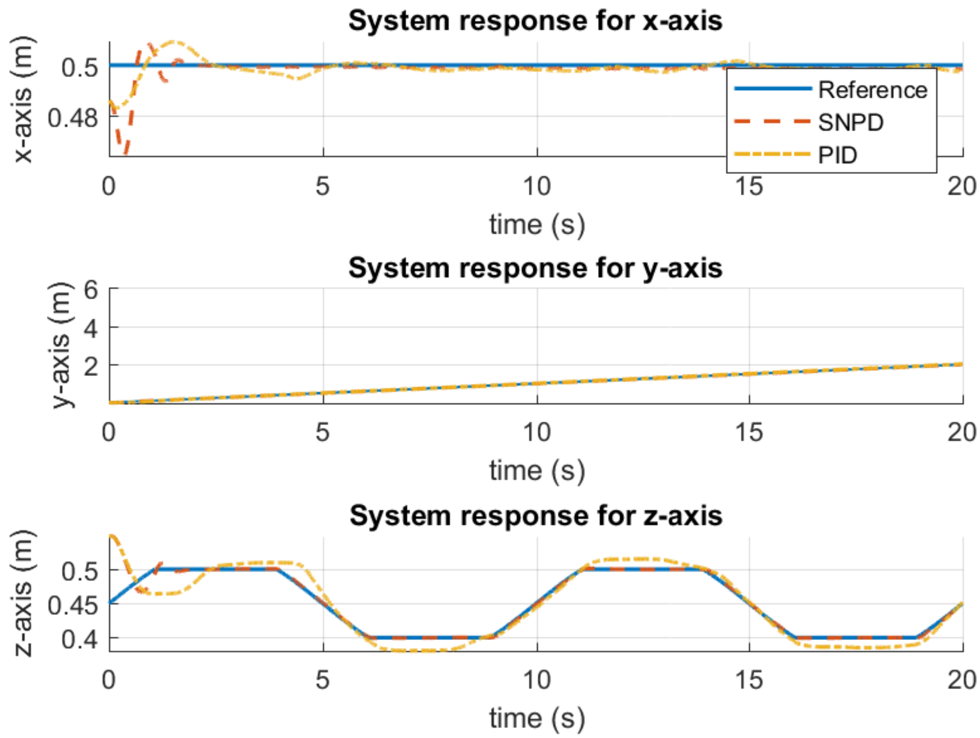
Figure 12. Velocity control signal results for the rose curve trajectory in real experiments.

The velocity control signals for the rose curve trajectory are illustrated in Figure 12. Once again, adaptive SNPD computes bigger control signals than PID. However, this demonstrates that SNPD is adjusting itself to reject perturbation and changes during experimental tests.

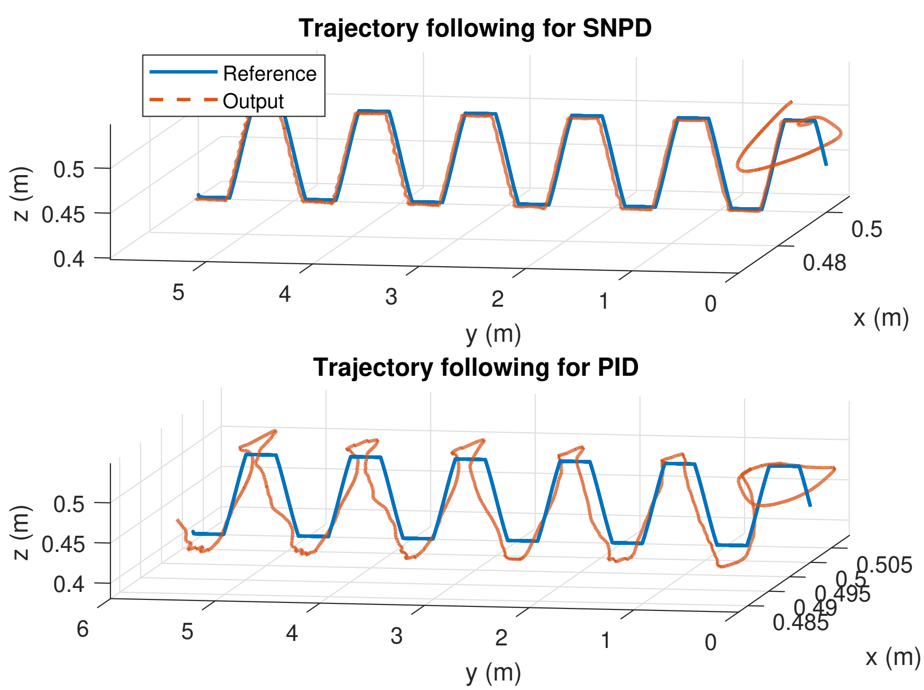
Table 4. Experimental results for the rose curve trajectory. The best results are highlighted in bold.

Measure	Method	e_x	e_y	e_z
RMS	SNPD	3.7452×10^{-3}	$\mathbf{3.3248}\times10^{-3}$	$\mathbf{8.4861}\times10^{-3}$
	PID	$\mathbf{2.9319}\times10^{-3}$	1.7032×10^{-2}	2.1101×10^{-2}
MAD	SNPD	$\mathbf{9.5960}\times10^{-4}$	$\mathbf{1.1829}\times10^{-3}$	$\mathbf{2.0750}\times10^{-3}$
	PID	1.3942×10^{-3}	1.2963×10^{-2}	1.6109×10^{-2}

Table 3 reported the RMS and MAD results for the rose curve trajectory in real experiments. The SNPD scheme has demonstrated to have better results than conventional PID with the smallest RMS and MAD results in general.



(a) System response



(b) Trajectory following

Figure 13. System response and trajectory following results for the trapezoidal trajectory in real experiments.

A new trajectory is tested, and the system response and trajectory following results are shown in Figure 13. As can be seen, the SNPD control presents better than PID for the results for the trapezoidal trajectory. Figure 13 (a) shows the system response, where it is exhibited the adaptation ability of the SNPD, while PID control requires the tune of its gains. The PID scheme reported bigger tracking error that are presented in Figure 13 (b).

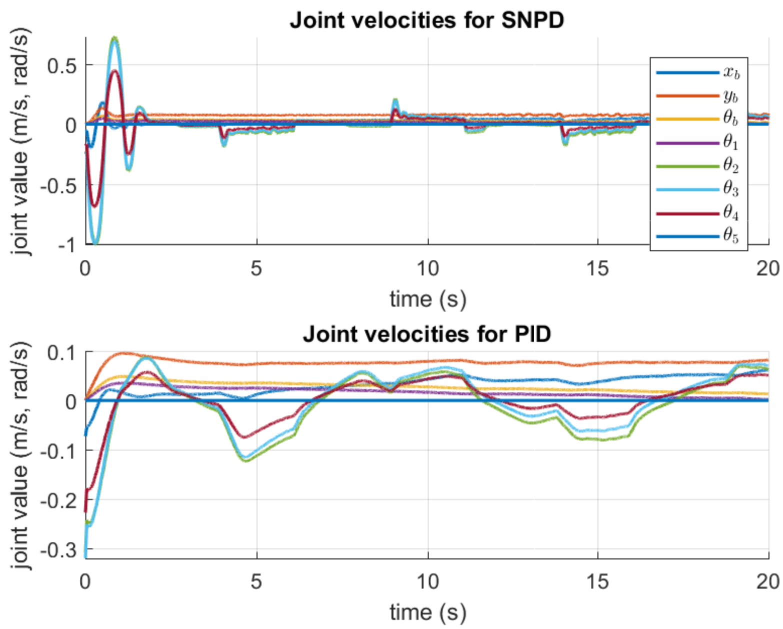


Figure 14. Velocity control signal results for the trapezoidal trajectory.

The velocity control signals results for the trapezoidal trajectory are given in Figure 14 . It is clear that bigger control action is required to be able to follow the trajectory with minimum error tracking. This is achieved with the online adaptation of SNPD controller.

Table 5. Experimental results for the trapezoidal trajectory. The best results are highlighted in bold.

Measure	Method	e_x	e_y	e_z
RMS	SNPD	3.8064×10^{-3}	$\mathbf{3.4122}\times10^{-3}$	$\mathbf{8.1596}\times10^{-3}$
	PID	$\mathbf{3.0025}\times10^{-3}$	5.5684×10^{-2}	1.8825×10^{-2}
MAD	SNPD	$\mathbf{8.8564}\times10^{-4}$	$\mathbf{1.3090}\times10^{-3}$	$\mathbf{2.0708}\times10^{-3}$
	PID	1.7514×10^{-3}	3.5206×10^{-2}	1.5032×10^{-2}

Finally, table 5 reported the RMS and MAD results for the trapezoidal trajectory in real experiments. The SNPD scheme outperformed the PID controller with the smallest RMS and MAD results in general.

CONCLUSIONS

In this work, an adaptive single neuron PD (SNPD) and multilayer network PD (MNPD) controllers trained with the EKF algorithm were proposed. The performance of these approaches were considered for trajectory tracking of the KUKA Youbot ™ mobile manipulator. Simulation and real experiments were performed to compare the classical PD and PID controllers against the proposals. Simulation and experiment results reported that PD control presented steady-state errors, while PID control overcomes this inconvenience but with overshoot results. In contrast, the adaptive neural PD controllers eliminated the steady-state error and highly suppressed the overshoot in general. Moreover, adaptive PD schemes show better settling time and high performance with smaller tracking results. The results also showed that even without an integral part, the PD neural controllers trained with extended Kalman filter offer better overall performance than a conventional PID. They present a small overshoot, and the offset is reduced. Additionally, the experimental results indicate that the SNPD controller shows a superior system response under perturbations and changes during the operation that the PID controller. The conventional PID controller requires the tuning of its gains to improve the performance. The SNPD controller shows better performance than MNPD, mainly due to more weights present in MNPD. It is shown that they present similar settling times, and the oscillations present with MNPD can be eliminated if trained weights are used instead of initializing them randomly every time. However, it was exposed that this is unnecessary, and both approaches exhibit good adaptation to uncertainties in the system. One of the main reasons for PI, PD, and PID controllers' success is their implementation simplicity. Some works have been proposed to deal with the drawbacks of the conventional PID, adding in some cases a fair complexity at implementation time. The proposed adaptive neural PD controllers are easy to implement, having good performances.

ACKNOWLEDGMENTS

The authors thank the support of Council of Sciences and Technology (CONACYT), Mexico, for financially supporting the following projects: CB-256769, CB-258068 and PN-2016-4107.

REFERENCES

Alanis, A., Arana-Daniel, N., and Lopez-Franco, C. (2019). *Artificial Neural Networks for Engineering Applications*. Elsevier Science.

Angel, L., Viola, J., and Paez, M. (2019). Evaluation of the windup effect in a practical pid controller for the speed control of a dc-motor system. In *2019 IEEE 4th Colombian Conference on Automatic Control (CCAC)*, pages 1–6.

Åström, K. J. and Hägglund, T. (1995). *PID controllers: theory, design, and tuning*, volume 2. Instrument society of America Research Triangle Park, NC.

Bryson, J. (2019). *The Past Decade and Future of AI's Impact on Society*, volume 11. Turner, Spain.

Ge, S. S., Zhang, J., and Lee, T. H. (2004). Adaptive neural network control for a class of mimo nonlinear systems with disturbances in discrete-time. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 34(4):1630–1645.

Gomez-Avila, J. (2019). Adaptive pid controller using a multilayer perceptron trained with the extended kalman filter for an unmanned aerial vehicle. In *Artificial Neural Networks for Engineering Applications*, pages 55–63. Elsevier.

Haykin, S. (2004). *Kalman Filtering and Neural Networks*. Adaptive and Cognitive Dynamic Systems: Signal Processing, Learning, Communications and Control. Wiley.

Hernandez-Barragan, J., Rios, J. D., Alanis, A. Y., Lopez-Franco, C., Gomez-Avila, J., and Arana-Daniel, N. (2020). Adaptive single neuron anti-windup pid controller based on the extended kalman filter algorithm. *Electronics*, 9(4):636.

J.Craig, J. (2005). *Introduction to Robotics Mechanics and Control 3rd edition*. Pearson Education, Inc., 3 edition.

Johnson, M. and Moradi, M. (2006). *PID Control: New Identification and Design Methods*. Probability and its applications. Springer London.

Kheirkhahan, P. (2017). Robust anti-windup control design for pid controllers. In *2017 17th International Conference on Control, Automation and Systems (ICCAS)*, pages 1622–1627.

Kumar, S. and Negi, R. (2012). A comparative study of pid tuning methods using anti-windup controller. In *2012 2nd International Conference on Power, Control and Embedded Systems*, pages 1–4.

Kundu, A. S., Mazumder, O., Dhar, A., Lenka, P. K., and Bhaumik, S. (2017). Scanning camera and augmented reality based localization of omnidirectional robot for indoor application. *Procedia Computer Science*, 105:27 – 33. 2016 IEEE International Symposium on Robotics and Intelligent Sensors, IRIS 2016, 17-20 December 2016, Tokyo, Japan.

Li, Z. and Ge, S. (2017). *Fundamentals in Modeling and Control of Mobile Manipulators*. Automation and Control Engineering. Taylor & Francis Group.

Li, Z., Yang, C., Su, C., Deng, J., and Zhang, W. (2016). Vision-based model predictive control for steering of a nonholonomic mobile robot. *IEEE Transactions on Control Systems Technology*, 24(2):553–564.

Lopez-Franco, C., Gomez-Avila, J., Alanis, A. Y., Arana-Daniel, N., and Villaseñor, C. (2017). Visual servoing for an autonomous hexarotor using a neural network based pid controller. *Sensors*, 17(8):1865.
Lopez-Franco, C., Hernandez-Barragan, J., Alanis, A. Y., and Arana-Daniel, N. (2018). A soft computing approach for inverse kinematics of robot manipulators. *Engineering Applications of Artificial Intelligence*, 74:104 – 120.
Maglogiannis, I., Iliadis, L., and Pimenidis, E. (2020). *Artificial Intelligence Applications and Innovations: 16th IFIP WG 12.5 International Conference, AIAI 2020, Neos Marmaras, Greece, June 5–7, 2020, Proceedings, Part I*. IFIP Advances in Information and Communication Technology. Springer International Publishing.
Moradi, M. H., Katebi, M. R., and Johnson, M. A. (2001). Predictive pid control: a new algorithm. In *IECON'01. 27th Annual Conference of the IEEE Industrial Electronics Society (Cat. No.37243)*, volume 1, pages 764–769 vol.1.
Ogata, K. (2010). *Modern Control Engineering*. Instrumentation and controls series. Prentice Hall.
Rios, J., Alanis, A., Arana-Daniel, N., Lopez-Franco, C., and Sanchez, E. (2020). *Neural Networks Modeling and Control: Applications for Unknown Nonlinear Delayed Systems in Discrete Time*. Elsevier Science.
Sanchez, E., Alanis, A., and Loukianov, A. (2010). *Discrete-Time High Order Neural Control: Trained with Kalman Filtering*. Studies in Computational Intelligence. Springer Berlin Heidelberg.
Sanchez, E. N. and Alanis, A. Y. (2006). Redes neuronales: conceptos fundamentales y aplicaciones a control automático. *Cinvestav Unidad Guadalara. Editorial Prentice Hall*.
Sarangapani, J. (2018). *Neural Network Control of Nonlinear Discrete-Time Systems*. Automation and Control Engineering. CRC Press.
Sciavicco, L. and Siciliano, B. (2008). *Robotics - Modelling, Planning and Control*. Advanced Textbooks in Control and Signal Processing. Springer, 2nd printing. edition.
Sheng Lin and Goldenberg, A. A. (2001). Neural-network control of mobile manipulators. 12(5):1121–1133.
Spong, M. W. and Vidyasagar, M. (2008). *Robot dynamics and control*. John Wiley & Sons.
Temel, S., Yağli, S., and Gören, S. (2013). P, pd, pi, pid controllers. *Middle East Technical University, Electrical and Electronics Engineering Department*.
Tian, Y.-C., Tadé, M. O., and Tang, J. (1999). A nonlinear pid controller with applications. *IFAC Proceedings Volumes*, 32(2):2657 – 2661. 14th IFAC World Congress 1999, Beijing, Chia, 5-9 July.
Visioli, A. (2006). *Practical PID Control*. Advances in Industrial Control. Springer London.
Wu, J., Lv, C., Zhao, L., Li, R., and Wang, G. (2017). Design and implementation of an omnidirectional mobile robot platform with unified i/o interfaces. In *2017 IEEE International Conference on Mechatronics and Automation (ICMA)*, pages 410–415.
Zhang, G., Qin, W., Qin, Q., He, B., and Liu, G. (2016). Varying gain mpc for consensus tracking with application to formation control of omnidirectional mobile robots. In *2016 12th World Congress on Intelligent Control and Automation (WCICA)*, pages 2957–2962.