Adaptive neural PD controllers for mobile manipulator trajectory tracking

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

Department of Computer Science, University of Guadalajara, Guadalajara, Jalisco, México

ABSTRACT

Artificial intelligence techniques have been used in the industry to control complex systems; among these proposals, adaptive Proportional, Integrative, Derivative (PID) 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 and an existing adaptive neuron PID controller.

INTRODUCTION

Artificial intelligence (AI) is actively present in our society; AI is used for decades in many relevant areas of our society as the industry, science, entertainment, education, and others (*Bryson, 2019*). However, it is essential to remark that interest in AI has risen in the last decade. Due to this interest, recently, many works have been reported in the literature in many research areas no name some control, internet of things, natural language processing, machine vision, medicine, robotics, security, social application, among others (*Bryson, 2019*; *Maglogiannis, Iliadis & Pimenidis, 2020*).

Proportional Integral, Derivative (PID) controllers are a well-studied kind of controller, which are among the most popular controllers in the industry, mainly for their simplicity (*Åström & Hägglund, 1995*; *Ogata, 2010*). The main drawback of PID controllers is that they are only adequate for a nominal process; they have a bad performance under

systems uncertainties in operating conditions and changing environmental conditions (*Tian, Tadé & Tang, 1999*). Even though they are usually the first approach when facing a control problem, even more than other adaptive techniques that have reported more satisfactory results for real-world problems where time delays, unmodeled dynamics and uncertainties are present (*Tahoun, 2017b*).

It is well-known that there exist techniques to improve the selection of conventional PID parameters; however, most of these techniques are offline methodologies and usually required knowledge about the model of the system, which not always is available (*Johnson & Moradi, 2006*; *Visioli, 2006*; *Ogata, 2010*). The use of artificial intelligence on PID controllers has been used as a tool to improve the performance of PID controllers adapting its parameters online, adjusting them to the changes of the system under consideration. Some of these techniques require access to the complete state of the system and information on its uncertainties and delays, and usually complex calculations (*Tahoun 2015*, *2017c*, *2020*). Among these techniques, neural networks stand out; their characteristics allow the implementation of easy, fast, and robust PID controllers known as neural PID controllers, which vary mainly on architecture and training methodology (*Rios et al., 2020b*; *Tahoun & Arafa, 2020*; *Hernandez-Barragan et al., 2020*).

Neural PID controllers learning capabilities allow them to adapt themselves during system operation to unmodeled dynamics, communication time-delays, actuator saturation, among other issues (*Ge, Zhang & Lee, 2004*; *Lopez-Franco et al., 2017*; *Sarangapani, 2018*; *Gomez-Avila, 2019*), which clearly is a better approach than fixed parameters during the whole operation. Neural adaptive PID controllers have been presented with single neuron and multilayer schemes. The single neuron controllers have three inputs, which are the proportional, derivative, and integral errors. The output of the neuron represents the control action (*Rivera-Meja, Léon-Rubio & Arzabala-Contreras, 2012*; *Jiao et al., 2018*; *Tang et al., 2020*). The multilayer controllers consist of a network with one hidden layer and one node at the output layer. In the hidden layer, three neurons represent the proportional, integral and derivative gains. The neuron of the output layer defines the control action. The inputs of this scheme can be the actual state and the reference (*Chen, He & Zhou, 2015*; *Zeng et al., 2019*). But the inputs can also include the proportional, derivative, and integral errors (*Sento & Kitjaidure, 2016*). Adaptive neural PID controllers trained with the extended Kalman filter (EKF) algorithm based algorithms have proved to show faster learning speed rates and convergence time than adaptive neural PID based on backpropagation training methods, which makes EKF training based neural PID controller more suitable for experimental and real-time tests (*Hernandez-Barragan et al., 2020*). Also, training algorithms based on Extended Kalman filter (EKF) for neural networks have proven to reliable for recurrent and feedforward neural networks for control applications, presenting real-time applications (*Haykin, 2004*; *Sanchez, Alanis & Loukianov, 2010*; *Alanis, Arana-Daniel & Lopez-Franco, 2019*; *Rios et al., 2020a*).

Besides the previously mentioned withdraws of PID controllers, a common problem is the windup effect, which is the result of accumulative error action due to the 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 & Negi, 2012*; *Hernandez-Barragan et al., 2020*), which is the reason why anti-windup strategies are important when using PID controllers *Tahoun (2017a)*. Among the proposed anti-windup strategies in the literature are limiter integrator, back-calculation, and observer approach (*Visioli, 2006*; *Kumar & Negi, 2012*; *Kheirkhahan, 2017*; *Angel, Viola & Paez, 2019*). The integral term is important because it eliminates the steady-state error that the proportional term cannot suppress with a fixed proportional gain. However, the integral action causes oscillations, overshoot, and the windup effect mainly on physical implementation.

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 (*Lin & Goldenberg, 2001*; *Li & Ge, 2017*). Among these applications: construction, health-care, nuclear reactor maintenance, manufacturing, military operations, and planetary exploration. Some of those tasks can risk human lives (*Lin & Goldenberg, 2001*; *Li & Ge, 2017*). However, such advantages come with complexity and difficulty when designing a control strategy (*Li & Ge, 2017*). When conventional PID strategies are not enough, and considering what has been previously stated, adaptive intelligent controllers appeared as plausible solutions, especially the ones based on neural networks.

The contributions of this paper are summarized as follows: In *Hernandez-Barragan et al. (2020)*, a single neuron PID controller trained with an extended Kalman filter (EKF) based algorithm and anti-windup effect is proposed, as other neural PID controllers it adjusts itself online during the operation of the system, even with changes in the nature of the problem. However, it requires and anti-wind up methodology. The contribution of this work is the proposal of adaptive neural PD controllers trained with extended Kalman Filter-based algorithms; the adaptive properties of these controllers allow them to compensate for the missing integral part, which can cause the windup effect; in this way, there is no need for an anti-windup methodology as in previous work (*Hernandez-Barragan et al., 2020*) and at the same time getting a good performance. The proposed EKF-based trained PD controllers, single neuron and multilayer adapt their weights online, eliminating the steady-state error; moreover, oscillations and overshoot are highly suppressed. The performance of the controllers is shown in simulation and experimental results on trajectory tracking tasks for a mobile manipulator robot. In simulations, the proposed controllers are compared against the conventional PID, and an existing single neuron adaptive PID (SNA-PID) controllers (*Tang et al., 2020*). In real experiments, a study on the robustness of the proposed controller under the presence of disturbances and non-modeled dynamics is presented.

The remaining of this work is organized as follows: first, a summary of the components of PID controllers and adaptive neural PD controllers. Second, the implementation of the proposed neural PD controller on mobile manipulators. Third, simulation and experimental results on a mobile manipulator robot where the performance of proposed

controllers is shown; experimental tests are implemented on a KUKA[TM][1]. Finally, conclusions are presented.

ADAPTIVE NEURAL PD CONTROLLERS

A basic PID controller consists of applying the sum of three types of control actions, proportional (P), integral (I) and derivative (D) (*Visioli, 2006*; *Temel, Yagli & Gören, 2013*). Moreover, using those control actions, simpler controllers can be obtained, namely, P, PD and PI, which may be enough for some applications, especially linear ones and under regulated conditions. Nevertheless, the PID controller is recognized as the better of them. Even with the existence of more robust control schemes reported in the literature, the popularity of PID is mainly due to its simple implementation. Inspire in this popularity; several works have been proposed to improve the performance of PID controllers. However, most of those works introduce complex methodologies.

P, PI, PD and PID controllers brief summary

P controller

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.

PI controller

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

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

Proportional Integral, Derivative 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 Fig. 1. The manual tuning of the proportional K_P, integrative K_I and derivative K_D gains represent an inconvenience of conventional PID controllers.

Adaptive neuron PD controller

The main disadvantage of conventional PD controllers is that they are not suitable for nonlinear, time-variant systems. adaptive neural controllers are an alternative to overcome this issue.
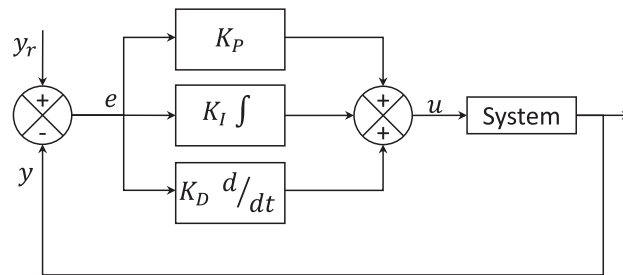
Figure 1 Control PID scheme. Full-size ⊡ DOI: 10.7717/peerj-cs.393/fig-1
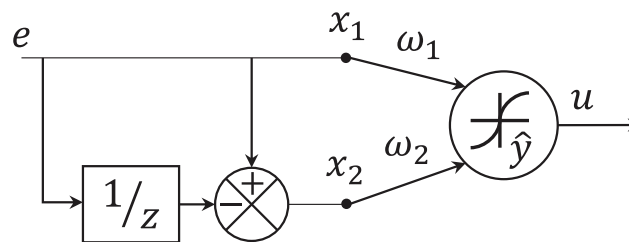

Figure 2 Adaptive single neuron PID controller. Full-size ⊡ DOI: 10.7717/peerj-cs.393/fig-2

The proposed adaptive single neuron PD (SNPD) controller is illustrated in Fig. 2. The value e represents the 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 (*Moradi, Katebi & Johnson, 2001*). The weights ω_1 and ω_2, are adapted online using the EKF algorithm. The weight ω_1 and ω_2 represents the proportional gain, and derivative gain, respectively. 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 as activation function is selected $\tanh(\cdot)$. The activation function reacts in the range $(-1, 1)$. However, the parameter α can be selected to adjust the control action, since the output of the neuron is directly the control signal $u(k) = \hat{y}(k)$.

$$e(k) = y_r(k) - y(k) \tag{1}$$

$$x_1(k) = e(k) \tag{2}$$

$$x_2(k) = e(k) - e(k-1) \tag{3}$$

$$v(k) = \omega_1(k)x_1(k) + \omega_2(k)x_2(k) \tag{4}$$

$$\hat{y}(k) = \alpha \tanh(v(k)) \tag{5}$$

The proposed EKF-based training method is described in a section bellow. EKF provides faster learning rates and convergence time than backpropagation, which is crucial for online training.
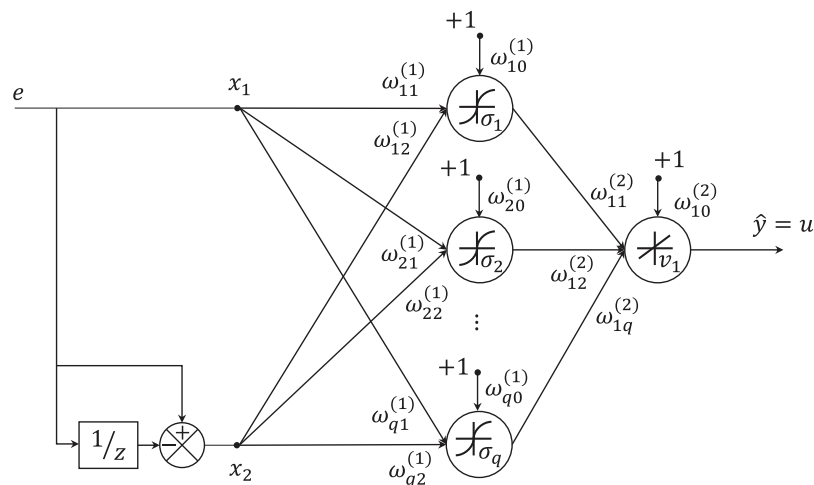
Figure 3 MLP architecture. In this case, the network has one hidden layer whose weights are denoted by $w_{ij}^{(1)}$ and the output layer has one node and its weights are represented with $w_{1j}^{(2)}$.

Full-size 🖼 DOI: 10.7717/peerj-cs.393/fig-3

Adaptive multilayer PD controller

The multilayer network PD (MNPD) scheme is shown in Fig. 3; 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 Fig. 3 with 2 input signals and q nodes in the hidden layer.

The output of the network is given by

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

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

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

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

Extended Kalman filter based training algorithm for neural networks

For training of neural networks, the weights of the network become the state to be estimated by the EKF, with the objective of reducing the neural network error, which in this case, since the output is the reference minus the system output $e(k) = y_r(k) - y(k)$, this

is because the neural network output is considered directly as the control signal $u(k)$. The fact that e is minimized means that the neural PD controller output u is working in achieving the control obective of tracking the desired reference y_r. The neural network is trained online using an extended Kalman filter-based algorithm (10–12).

$$\mathbf{K}(k) = \mathbf{P}(k)\mathbf{H}(k)\left[\mathbf{R}(k) + \mathbf{H}^{\mathrm{T}}(k)\mathbf{P}(k)\mathbf{H}(k)\right]^{-1} \tag{10}$$

$$\omega(k+1) = \omega(k) + \eta\mathbf{K}(k)\mathbf{e}(k) \tag{11}$$

$$\mathbf{P}(k+1) = \mathbf{P}(k) - \mathbf{K}(k)\mathbf{H}^{\mathrm{T}}(k)\mathbf{P}(k) + \mathbf{Q}(k) \tag{12}$$

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

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), $y_i \in \mathbb{R}$ 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 & Alanis, 2006*).

Neural network weights are initialized randomly. The Kalman filter learning rate η is selected heuristically to minimized e. It should be considered that if η is sufficiently large, the network could not converge. Conversely, if a lower η is selected, it would take longer to converge. Moreover, matrices $\mathbf{P}$, $\mathbf{Q}$ and $\mathbf{R}$ are initialized as diagonal matrices with initial values chosen heuristically. The $\mathbf{Q}$ matrix is set to deal with process noise, while the $\mathbf{R}$ matrix is set to deal with measurement noise. Metaheuristic algorithms can be used to optimize the initial values of the Kalman settings (*Villaseñor et al., 2018*).

Let us remark that matrices H, K and P are bounded (*Song & Grizzle, 1992*).

Single neuron EKF training algorithm

The EKF algorithm adjusts onlone the weights ω_1 and ω_2 for the single neuron. The single neuron scheme is composed with $n = 2$ (weights) and $m = 1$ (one output neuron); 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 Eq. (1). The matrix $\mathbf{H}$ is computed as Eq. (14).

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

Multilayer network EKF training algorithm

The EKF algorithm adjusts online the wights $\omega_{ij}^{(1)}(k)$ and $\omega_{j1}^{(2)}(k)$ for the multilayer network. The multilayer network scheme is set with n (weights) and $m = 1$

(output neuron). 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 Eq. (1). The matrix $\mathbf{H}$ can be expressed as Eq. (16).

$$\mathbf{H}(k) = \left[\frac{\partial \hat{y}(k)}{\partial w_{10}^{(1)}(k)} \quad \frac{\partial \hat{y}(k)}{\partial w_{11}^{(1)}(k)} \quad \cdots \quad \frac{\partial \hat{y}(k)}{\partial w_{1q}^{(2)}(k)} \right] \tag{15}$$

$$= \begin{bmatrix} \gamma(n_1(k))x_0(k) & \cdots & \gamma(n_1(k))x_p(k) & \gamma(n_2(k))x_0(k) & \cdots \\ & \gamma(n_q(k))x_p(k) & u_0(k) & u_1(k) & \cdots & u_q(k) \end{bmatrix} \tag{16}$$

with

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

IMPLEMENTATION TO A MOBILE MANIPULATOR FOR 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 robots increase the workspace of manipulators. However, these platforms have limited movement capabilities due to their nonholonomic kinematics constraints, *Li et al. (2016)*. On the other hand, omnidirectional mobile platforms improved the movement capabilities, allowing moving towards any position and desired orientation (*Zhang et al., 2016*; *Wu et al., 2017*; *Kundu et al., 2017*). This section introduces a kinematic model of a mobile manipulator, which consists of a robotic manipulator of n Degrees of Freedom (DOF) attached to an omnidirectional mobile platform.

The Kinematics chain of mobile manipulators is described in Fig. 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 & 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 Eq. (18).

$$^{w}\mathbf{T}_b = \begin{bmatrix} \cos(\theta_b) & -\sin(\theta_b) & 0 & x_b \\ \sin(\theta_b) & \cos(\theta_b) & 0 & y_b \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{18}$$
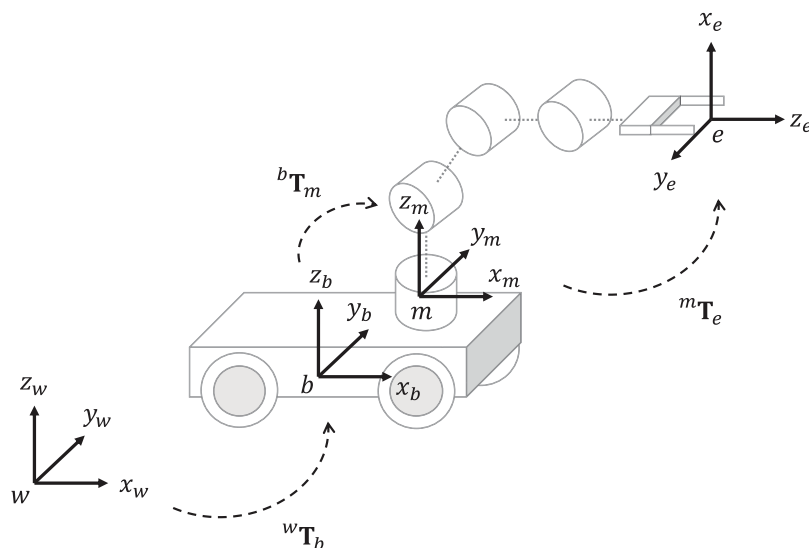
Figure 4 Kinematic chain of mobile manipulators. The transformation $^{w}\mathbf{T}_b$ is the homogeneous matrix from the world frame w to the mobile platform base frame b, $^{b}\mathbf{T}_m$ is the homogeneous matrix from b to the manipulator base frame m, $^{m}\mathbf{T}_e$ is the homogeneous matrix from m to the end-effector frame e.
Full-size ◩ DOI: 10.7717/peerj-cs.393/fig-4

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 matrix $^{b}\mathbf{T}_m$ can be described by Eq. (19).

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

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

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

$$^{w}\mathbf{T}_e(\mathbf{q}) = {}^{w}\mathbf{T}_b(\mathbf{q}_b) \, ^{b}\mathbf{T}_m \, ^{m}\mathbf{T}_e(\mathbf{q}_m) \tag{20}$$

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 **R**, and its Cartesian position is given by the vector **t**. More information about homogeneous matrices, manipulators kinematics, and forward kinematics can be found in *Spong & Vidyasagar (2008)*, *Craig (2005)* and *Sciavicco & Siciliano (2008)*.

Differential kinematics

The inverse kinematics consists in the computation of the joint variables **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 & 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 Eq. (22) gives this relationship

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

where **J** is the matrix that relates the contribution of the joint velocities **q** to the end-effector velocity $\dot{\mathbf{t}}$. The matrix **J** is called the geometric Jacobian. This Jacobian matrix can be computed as Eq. (23).

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

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 **t** and the desired position $\mathbf{t}^*$. This error is defined as $\mathbf{e} = \mathbf{t}^* - \mathbf{t}$. The error **e** can be mapped to the joint velocities $\dot{\mathbf{q}}$ based on the differential kinematics equation. Eq. (22) is rewritten to compute $\dot{\mathbf{q}}$ given **e** as Eq. (24).

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

where $\mathbf{J}^\dagger$ is the pseudo-inverse of **J**. The mentioned inverse kinematics approach can be defined as a first-order algorithm that allows the inversion of a motion trajectory, specified at the end-effector position into equivalent joint position and velocities (*Sciavicco & Siciliano, 2008*).

A robot system with a Jacobian matrix $\mathbf{J} \in \mathbb{R}^{3 \times n}$ where $n>3$, is considered redundant; 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 Eq. (24) can be generalized into Eq. (25).

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

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 Eq. (26).

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

Then, vector $\dot{\mathbf{q}}_0$ can be computed as Eq. (27).

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

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 & Vidyasagar (2008)*, *Craig (2005)* and *Sciavicco & 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, Katebi & Johnson, 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 Eq. (25) to control the system. This is

$$\dot{\mathbf{q}}(k) = \mathbf{J}(\mathbf{q}(k))^{\dagger}\mathbf{u}(k) + \left(\mathbf{I} - \mathbf{J}(\mathbf{q}(k))^{\dagger}\mathbf{J}(\mathbf{q}(k))\right)\dot{\mathbf{q}}_0 \tag{31}$$
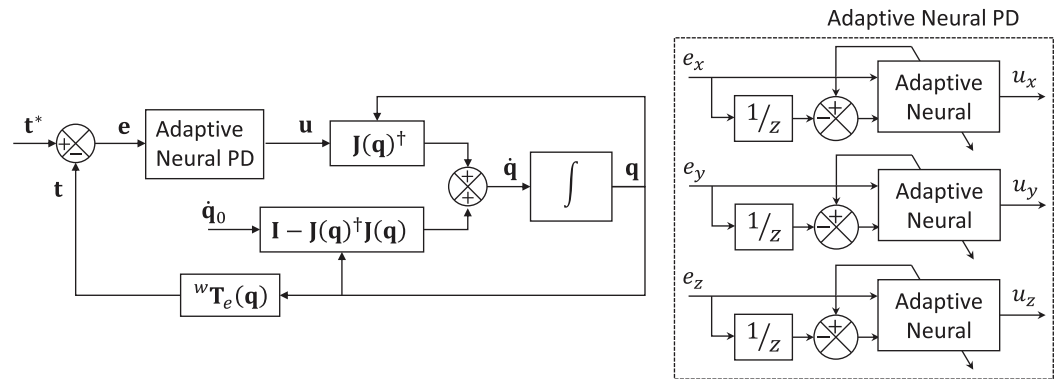
Figure 5 Adaptive neural PD control scheme for the position control of mobile manipulators. The block called Adaptive Neural, can represent the single neuron scheme or the multilayer network scheme. Full-size 🖼 DOI: 10.7717/peerj-cs.393/fig-5

Neural PD controllers implementation

Both proposed neural PD presented in previous sections are implemented on the above-described mobile manipulator. Figure 5 shows the general control scheme for both implementations.

An adaptive neural PD control module is designed to minimize the error e_x, e_y and θ_z. Each control signal (neural PD output) u_x, u_y and u_z, are compute 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 $\mathbf{q}(k)$ using Eq. (31) to control the system.

If expression Eq. (31) is multiplied by the Jacobian matrix $\mathbf{J}(\mathbf{q}(k))$, then we have

$$\mathbf{J}(\mathbf{q}(k))[\dot{\mathbf{q}}(k)] = \mathbf{J}(\mathbf{q}(k))\left[\mathbf{J}(\mathbf{q}(k))^\dagger \mathbf{u}(k) + \left(\mathbf{I} - \mathbf{J}(\mathbf{q}(k))^\dagger \mathbf{J}(\mathbf{q}(k))\right)\dot{\mathbf{q}}_0\right] \quad (32)$$

$$= \mathbf{J}(\mathbf{q}(k))\mathbf{J}(\mathbf{q}(k))^\dagger \mathbf{u}(k) + \mathbf{J}(\mathbf{q}(k))\left(\mathbf{I} - \mathbf{J}(\mathbf{q}(k))^\dagger \mathbf{J}(\mathbf{q}(k))\right)\dot{\mathbf{q}}_0 \quad (33)$$

$$= \mathbf{u}(k) + \left(\mathbf{J}(\mathbf{q}(k)) - \mathbf{J}(\mathbf{q}(k))\mathbf{J}(\mathbf{q}(k))^\dagger \mathbf{J}(\mathbf{q}(k))\right)\dot{\mathbf{q}}_0 \quad (34)$$

$$= \mathbf{u}(k) + (\mathbf{J}(\mathbf{q}(k)) - \mathbf{J}(\mathbf{q}(k)))\dot{\mathbf{q}}_0 \quad (35)$$

$$= \mathbf{u}(k) \quad (36)$$

which indicates that $\mathbf{u}(k) = \mathbf{J}(\mathbf{q}(k))\dot{\mathbf{q}}(k)$ is a solution of the differential kinematics of the system, where the matrix $\mathbf{J}(\mathbf{q}(k))$ is bounded (*Hernandez, Nuño & Alanis, 2016*). Moreover, the computed control signal of the neural controller Eq. (5) is also bounded. Considering that neural control for system Eq. (31) is a feed forward network control law, it composed a stable system (*Khalil, 2002*).

RESULTS

This section shows through simulation and experimental test the performance of the proposed controllers, the adaptive single neuron PD (SNPD) and multilayer network PD (MNPD). The controllers are compared against conventional PID controller, and an existing single neuron adaptive PID (SNA-PID) (*Tang et al., 2020*). Reference trajectories

Figure 6 Omnidirectional mobile manipulator KUKA Youbot^TM. Photo credit: Jesus Hernandez-Barragan. Full-size 🖼 DOI: 10.7717/peerj-cs.393/fig-6

are selected with different degrees of difficulty for both simulations, and experimental tests on the KUKA Youbot^TM mobile manipulator, see Fig. 6. Moreover, experiments show how the controllers behave in the presence of disturbances and non-modeled dynamics.

The KUKA Youbot^TM 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 Eq. (18). The constant transformation $^b\mathbf{T}_m$ is considered to be Eq. (37).

$$^b\mathbf{T}_m = \begin{bmatrix} 1 & 0 & 0 & 0.140 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.151 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{37}$$

The values shown in Eq. (37) were obtained based on the KUKA Youbot^TM 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 Eq. (38).

$$\mathbf{q} = \begin{bmatrix} x_b & y_b & \theta_b & \theta_1 & \theta_2 & \theta_3 & \theta_4 & \theta_5 \end{bmatrix}^T \tag{38}$$

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

For simulations and experimental test, the weights for both SNPD and MNPD controllers are set randomly in every trajectory test. The parameter setting for the EKF are:

Hernandez-Barragan et al. (2021), *PeerJ Comput. Sci.*, DOI 10.7717/peerj-cs.393

13/29

Table 1 DH table for KUKA Youbot^{TM} 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

matrices $\mathbf{P}$ and 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 $\alpha = 1$. These parameters were chosen heuristically. For PID controller, 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 PID controller are also selected heuristically. The weights for the SNA-PID controller are set randomly. Additionally, the SNA-PID learning rates are tuned to $\eta_x = \eta_y = \eta_z = 1.0 \times 10^{-5}$ and the proportional coefficient is set to $K = 2.5$. The setting for SNA-PID is tuned based on *Tang et al. (2020)*.

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

Circular trajectory
$x_r(k) = 0.5,$
$y_r(k) = 0.05 \cos(0.2\,k\,\pi),$
$z_r(k) = 0.45 + 0.05 \sin(0.2\,k\,\pi).$

Rose curve trajectory
$x_r(k) = 0.5,$
$y_r(k) = r(k) \cos(0.2\,k\,\pi),$
$z_r(k) = 0.45 + r(k) \sin(0.2\,k\,\pi),$
$r(k) = 0.035 + 0.015 \cos(0.6\,k\,\pi).$

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

Sinusoidal trajectory
$y_r(k) = 0.1 * k,$
$x_r(k) = 0.5 + 0.05 \cos(2\,y_r(k)\,\pi),$
$z_r(k) = 0.45 + 0.05 \sin(2\,y_r(k)\,\pi).$

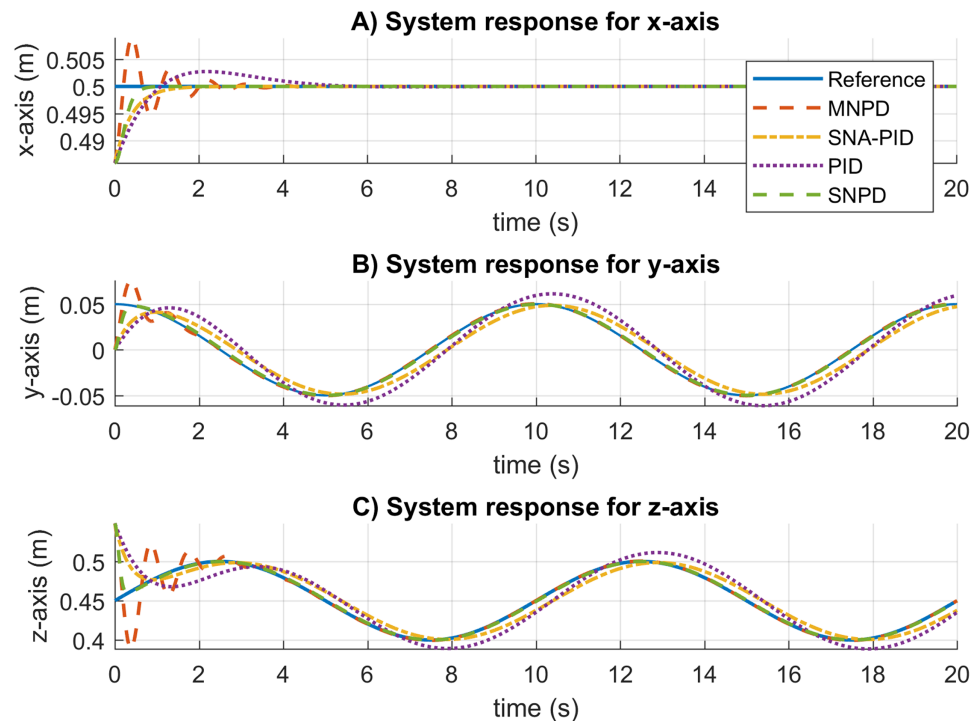**Figure 7 System response for circular trajectory.** (A–C) System responses for the x-axis, y-axis, and z-axis, respectively. Full-size ☑ DOI: 10.7717/peerj-cs.393/fig-7

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

Simulations

The first trajectory for simulation is the circular. Although conventional PID controller presents a good response, their gains remain constant, and they cannot adapt to changes in the system operating conditions. On the other hand, the MNPD, SNA-PID, and SNPD approaches can correctly follow the reference once the weights are adjusted.

The system response results for the circular trajectory are given in Fig. 7. The settling time is almost the same for all the approaches. The conventional PID presents overshoot and steady-state errors. The SNA-PID controller suppresses overshoot, but it has small steady-state errors. Although MNPD presents oscillations while the weights are adapting, the neural algorithms can follow the sinusoidal trajectory better than the PID and SNA-PID. Moreover, the SNPD controller does not suffer from overshoot.

The root mean square (RMS) and median absolute deviation (MAD) for the circular trajectory are presented in Table 2. The adaptive approaches present the best results, which are highlighted in bold. In this case, the SNPD control scheme reportesd the smallest RMS results in general.

In Fig. 8, the trajectory tracking and velocity control signals for the circular trajectory are presented. From Figs. 8A to 8D, the PID control passes over the reference caused by the integral part, and MNPD presents oscillations as expected from the system response.

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}
	SNA-PID	8.1755×10^{-4}	9.0910×10^{-3}	9.9509×10^{-3}
	PID	1.0547×10^{-3}	1.2872×10^{-2}	1.3803×10^{-2}
	SNPD	$\mathbf{7.8284} \times 10^{-4}$	$\mathbf{2.1269} \times 10^{-3}$	$\mathbf{3.5693} \times 10^{-3}$
MAD	MNPD	1.3391×10^{-4}	$\mathbf{5.5760} \times 10^{-4}$	1.3227×10^{-3}
	SNA-PID	1.8123×10^{-4}	8.0183×10^{-3}	8.0622×10^{-3}
	PID	2.1417×10^{-4}	1.1419×10^{-2}	1.1686×10^{-2}
	SNPD	$\mathbf{1.2753} \times 10^{-4}$	6.0505×10^{-4}	$\mathbf{6.4863} \times 10^{-4}$

However, MNPD, SNA-PID, and SNPD controllers follow the trajectory correctly. As seen from Figs 8E to 8H, at the first steps adaptive weights compute bigger control signals than PID. However, it is necessary to reach the reference with a small tracking error.The adaptation ability of both MNPD and SNPD is also shown.

The same gains and parameters for the four approaches are set for a new desired trajectory to be tested, and the results are shown in Fig. 9. Similar results can be seen in the system response. The settling time is similar, and the MNPD present oscillations during the adaptations of its weights. In this case, PID and SNA-PID controllers present steady-state errors. However, SNA-PID performs better than PID. Moreover, The PID controller also has overshoot. The best results are given by SNPD and MNPD.

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

In Fig. 10, the trajectory following, and the velocity control signals for the rose curve trajectory are reported. From Figs. 8A to 8D can be noticed that the adapting approaches outperform the conventional PID controller. The SNA-PID controller shows a small steady-state error, while MNPD and SNPD report the smallest. The SNA-PID control improved the performance of PID, but it is needed to tune the proportional coefficient K to improve the performance. As can be seen from Figs. 8E to 8H, at the beginning of the trajectory, the weights adaptation of the MNPD and the SNPD compute bigger control signals than SNA-PID and PID; however, this is necessary to reach the reference with a small tracking error.

Experiments

Real experiments are carried out in two parts. In the first part, the proposed SNPD controller is compared against the classical PID. In this case, two trajectories are tested for comparison purposes. In the second part, the robustness of the SNPD controller is tested under the presence of disturbances and non-modeled dynamics.

In the first part of the experiments, the adaptive SNPD and MNPD controllers performed similarly in simulations. However, MNPD shows oscillations during the
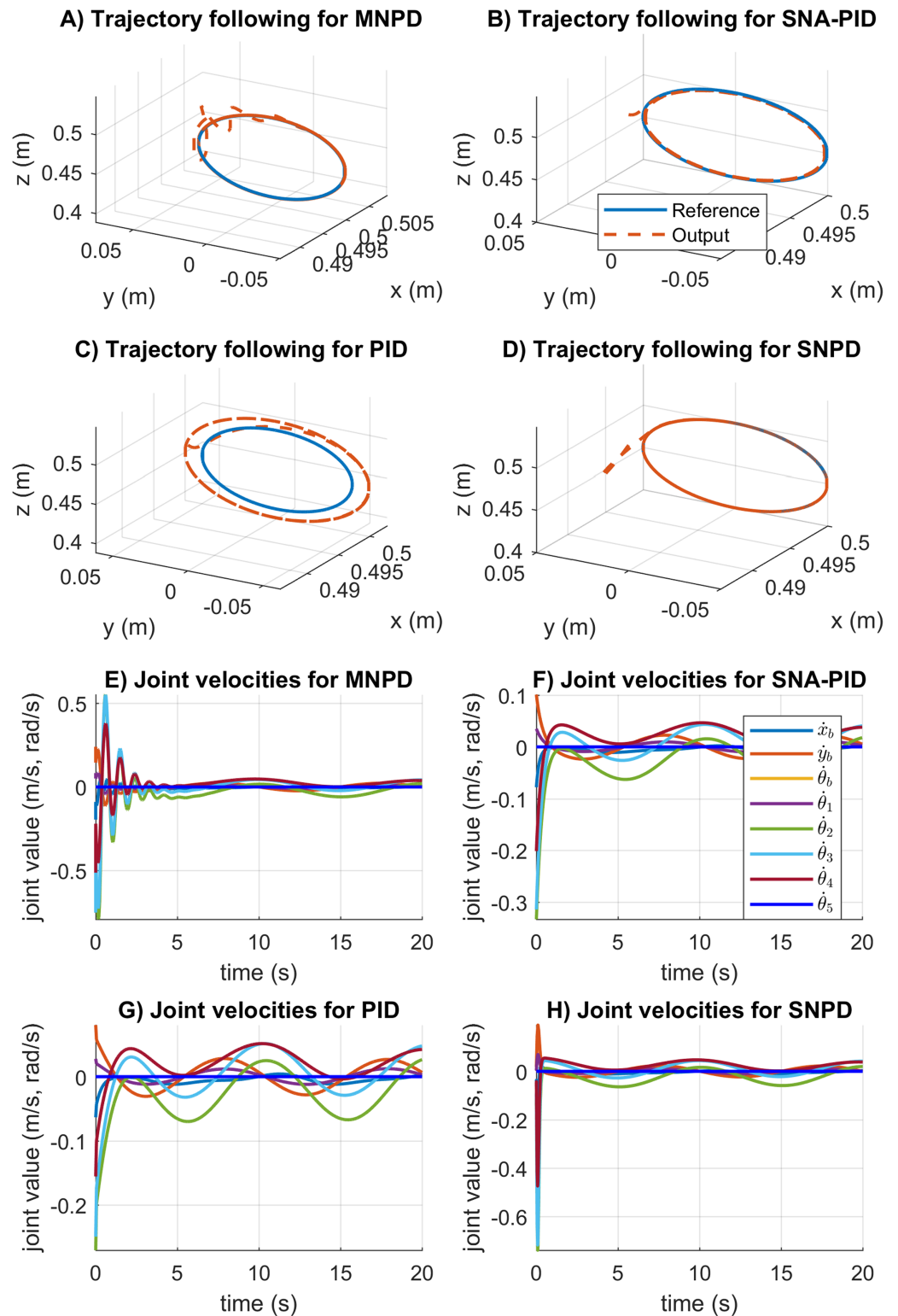
Figure 8 Trajectory following and velocity control signal results for the circular trajectory.
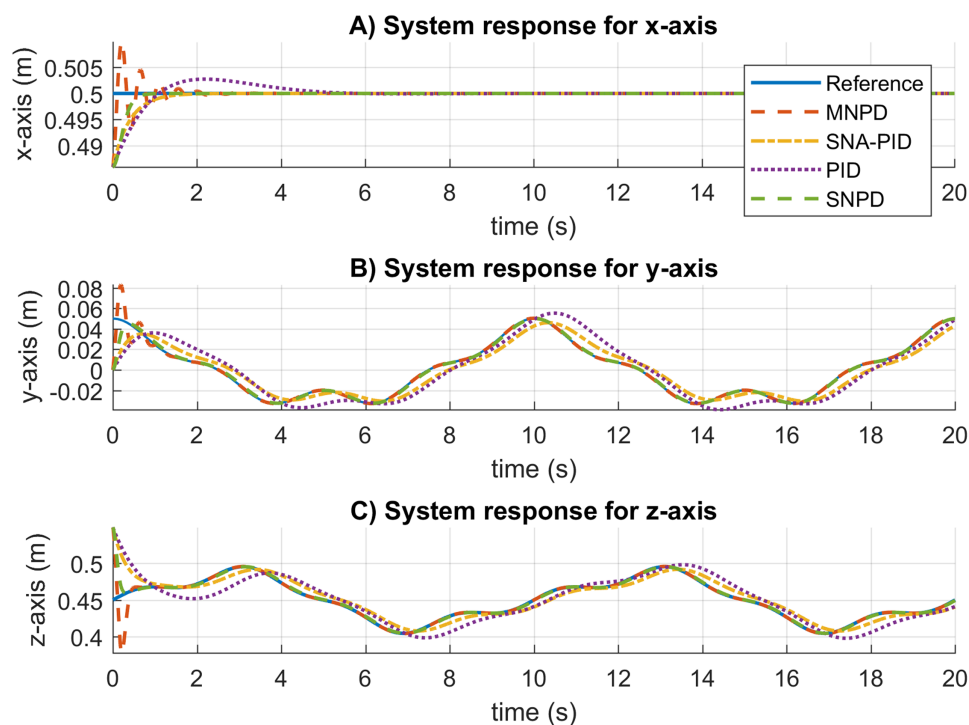(A–D) The trajectory following results. (E–H) The velocity control signal results.

Figure 9 **System response results for the rose curve trajectory.** (A–C) System responses for the x-axis, y-axis, and z-axis, respectively. Full-size ⬇ DOI: 10.7717/peerj-cs.393/fig-9

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	**6.9811 × 10⁻⁴**	**2.1346 × 10⁻³**	4.4524 × 10⁻³
	SNA-PID	8.1755 × 10⁻⁴	7.9247 × 10⁻³	9.0976 × 10⁻³
	PID	1.0547 × 10⁻³	1.1191 × 10⁻²	1.2804 × 10⁻²
	SNPD	7.7519 × 10⁻⁴	2.1415 × 10⁻³	**3.5652 × 10⁻³**
MAD	MNPD	**8.7982 × 10⁻⁵**	**3.6619 × 10⁻⁴**	**4.9729 × 10⁻⁴**
	SNA-PID	1.8123 × 10⁻⁴	6.8568 × 10⁻³	7.1826 × 10⁻³
	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

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. Thus, 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.

In Fig. 11, the system response for both approaches is shown. 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
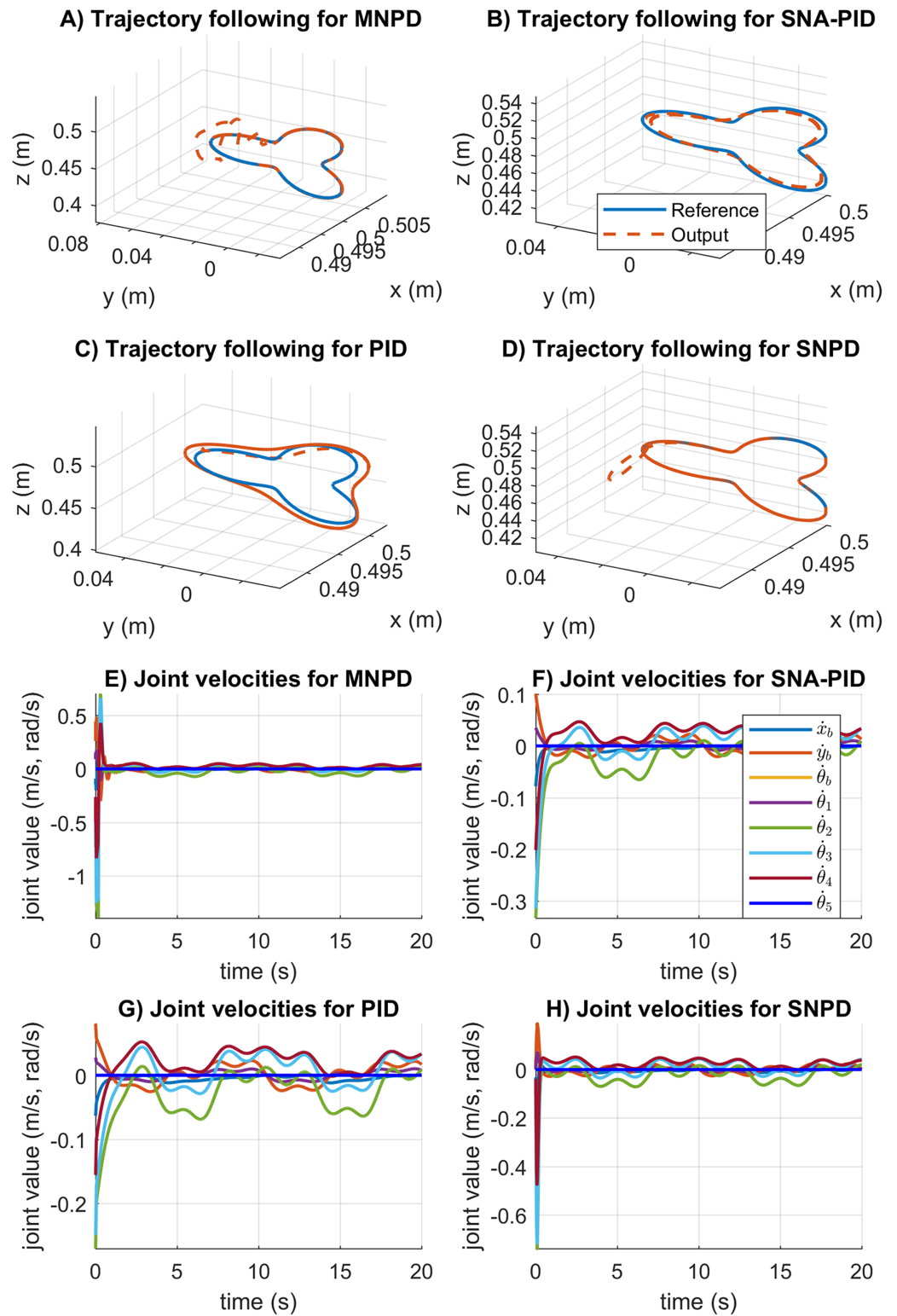
Figure 10 Trajectory following and velocity control signal results for the rose curve trajectory. (A–D) The trajectory following results. (E–H) The velocity control signal results.

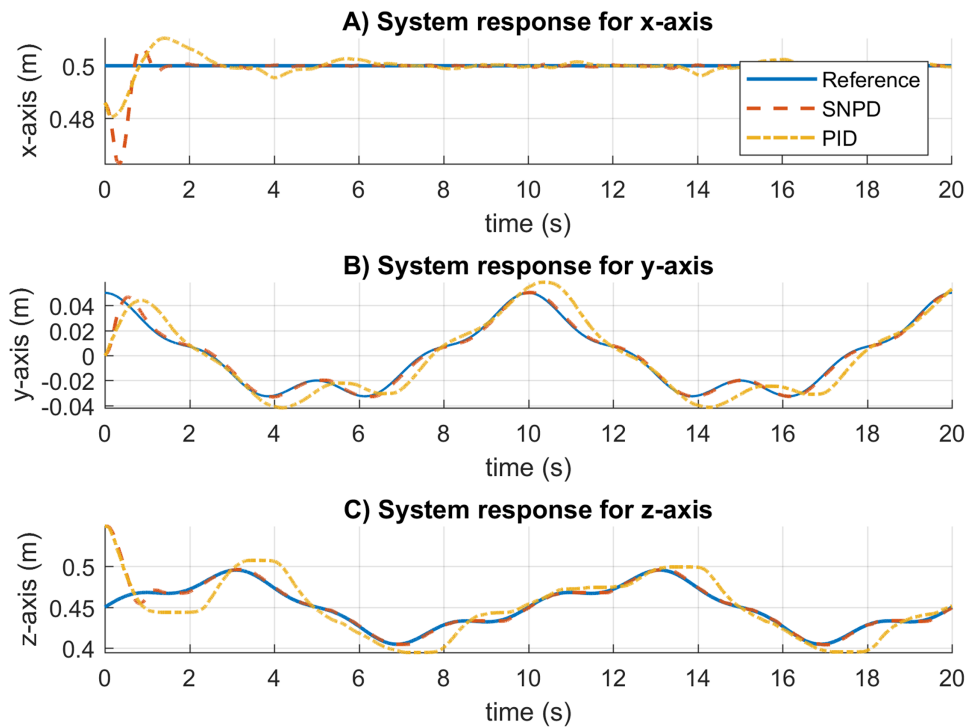Full-size ◢ DOI: 10.7717/peerj-cs.393/fig-10

Figure 11 System response results for the rose curve trajectory in real experiments. (A–C) System responses for the x-axis, y-axis, and z-axis, respectively. Full-size ☒ DOI: 10.7717/peerj-cs.393/fig-11

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}	**3.3248×10^{-3}**	**8.4861×10^{-3}**
	PID	**2.9319×10^{-3}**	1.7032×10^{-2}	2.1101×10^{-2}
MAD	SNPD	**9.5960×10^{-4}**	**1.1829×10^{-3}**	**2.0750×10^{-3}**
	PID	1.3942×10^{-3}	1.2963×10^{-2}	1.6109×10^{-2}

time. In contrast, using the same parameters as in simulation, the SNPD was able to adapt and showed a shorter settling time.

Table 4 reports the RMS and MAD results for the rose curve trajectory in real experiments. The SNPD scheme has been demonstrated to have better results than conventional PID with the smallest RMS and MAD results in general.

The trajectory following and velocity control signals for the rose curve trajectory are illustrated in Fig. 12. In Figs. 12A and 12B, 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. In Figs. 12C and 12D, adaptive SNPD computes bigger control signals than PID. However, this demonstrates that SNPD is adjusting itself to reject perturbation and changes during experimental tests.
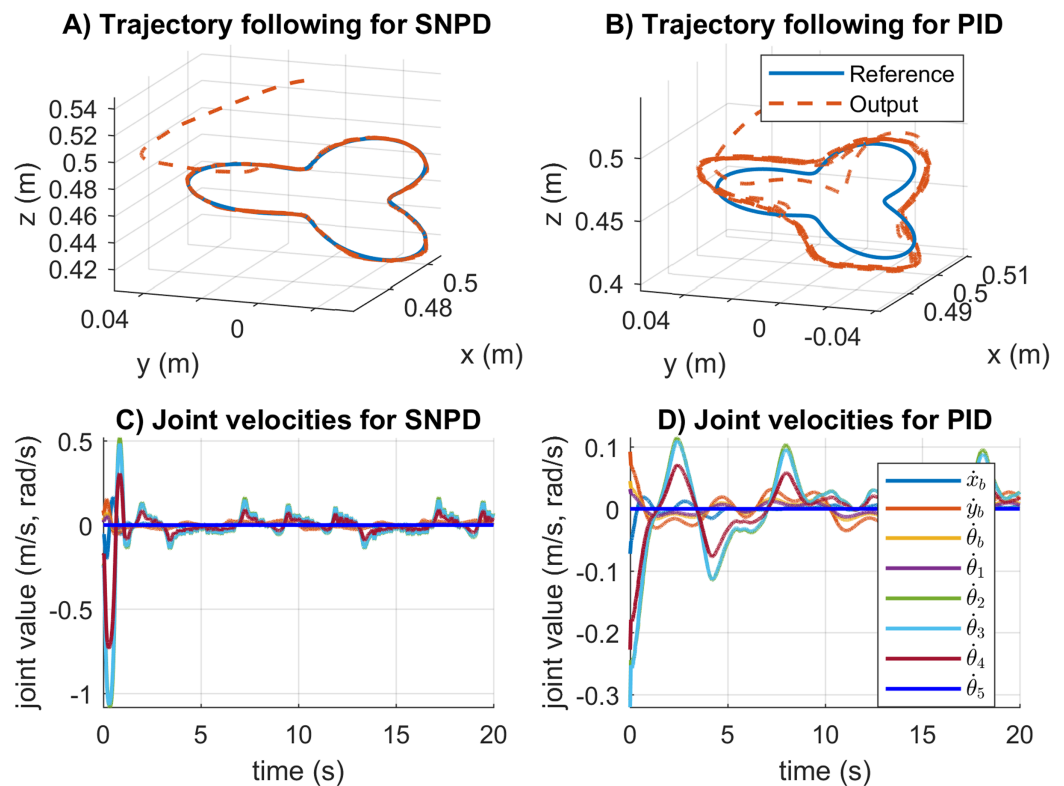
Figure 12 Trajectory following and velocity control signal results for the rose curve trajectory in real experiments. (A–B) The trajectory following results. (C–D) The velocity control signal results.

Full-size 🖼 DOI: 10.7717/peerj-cs.393/fig-12

A new trajectory is tested, and the system response results are shown in Fig. 13. The SNPD control performed better than PID for the results for the trapezoidal trajectory. The results exhibited the adaptation ability of the SNPD, while PID control requires the tune of its gains.

The trajectory following and velocity control signals results for the trapezoidal trajectory are given in Fig. 14. In Figs. 14A and 14B, the PID scheme reported bigger tracking error than SNPD controller. From Figs. 14C and 14D, 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.

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.

In the second part of the experiments, three tests were performed. In Test 1, a trajectory tracking on a smooth floor is considered. This test represents an ideal environment. In Test 2, the same trajectory tracking is performed on a rough floor. This test represents the presence of disturbances. Finally, in Test 3, the same trajectory tracking is performed on the uneven floor with a load on board of 1.5 kg. This last test represents non-modeled dynamics. Moreover, since the experiments were carried out on a real robot, there was measurement noise.
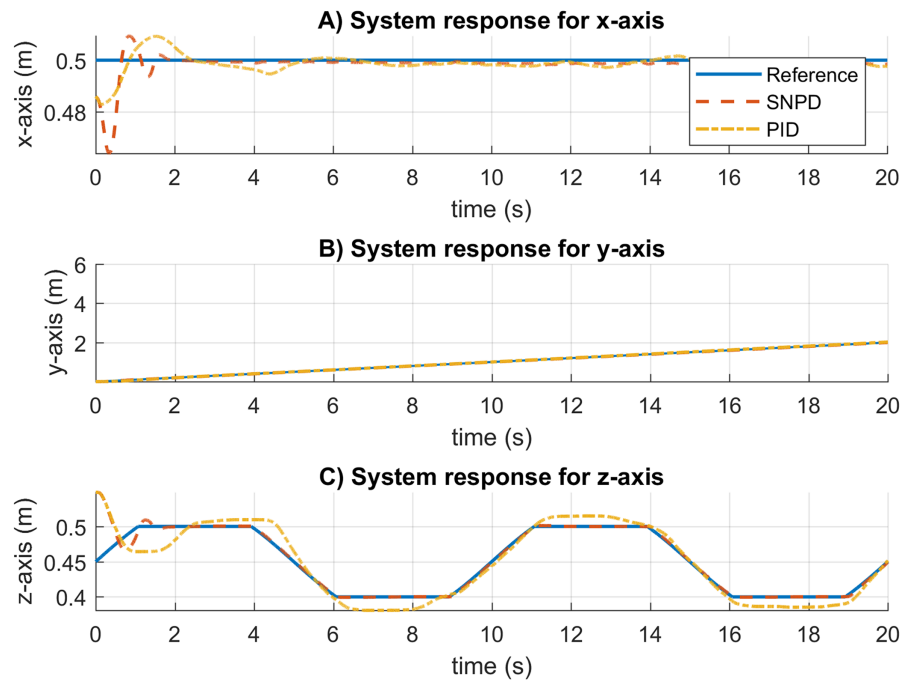
Figure 13 System response results for the trapezoidal trajectory in real experiments. (A–C) System responses for the x-axis, y-axis, and z-axis, respectively. Full-size DOI: 10.7717/peerj-cs.393/fig-13
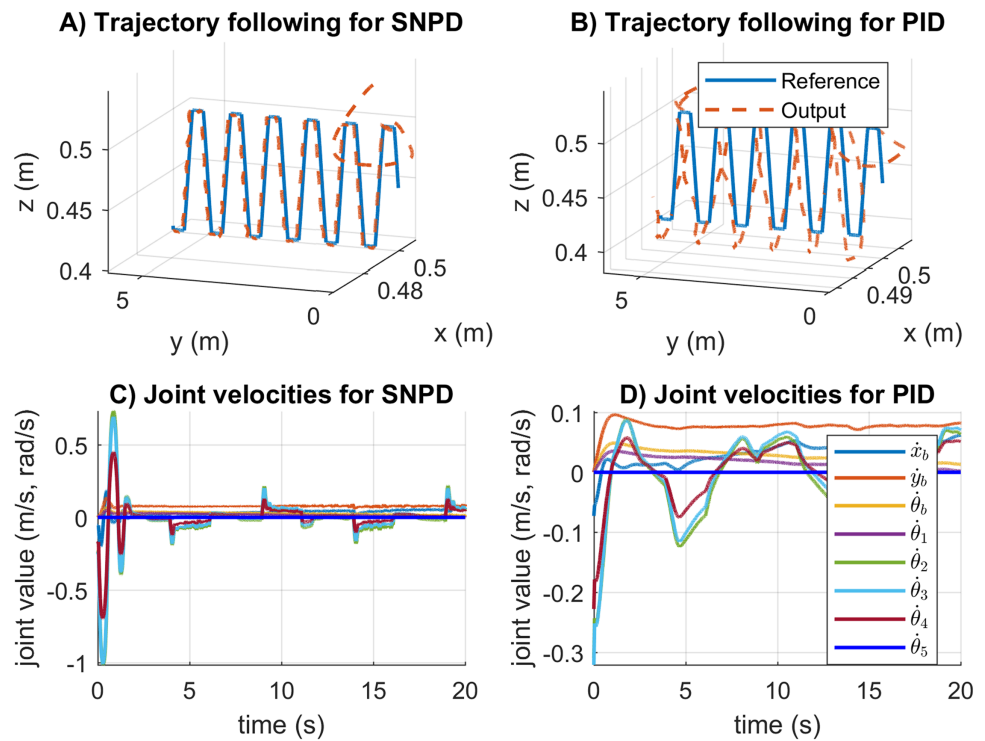

Figure 14 Trajectory following and velocity control signal results for the trapezoidal trajectory in real experiments. (A–B) The trajectory following results. (C–D) The velocity control signal results. Full-size DOI: 10.7717/peerj-cs.393/fig-14

Hernandez-Barragan et al. (2021), *PeerJ Comput. Sci.*, DOI 10.7717/peerj-cs.393

22/29

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}	$\mathbf{3.4122 \times 10^{-3}}$	$\mathbf{8.1596 \times 10^{-3}}$
	PID	$\mathbf{3.0025 \times 10^{-3}}$	5.5684×10^{-2}	1.8825×10^{-2}
MAD	SNPD	$\mathbf{8.8564 \times 10^{-4}}$	$\mathbf{1.3090 \times 10^{-3}}$	$\mathbf{2.0708 \times 10^{-3}}$
	PID	1.7514×10^{-3}	3.5206×10^{-2}	1.5032×10^{-2}
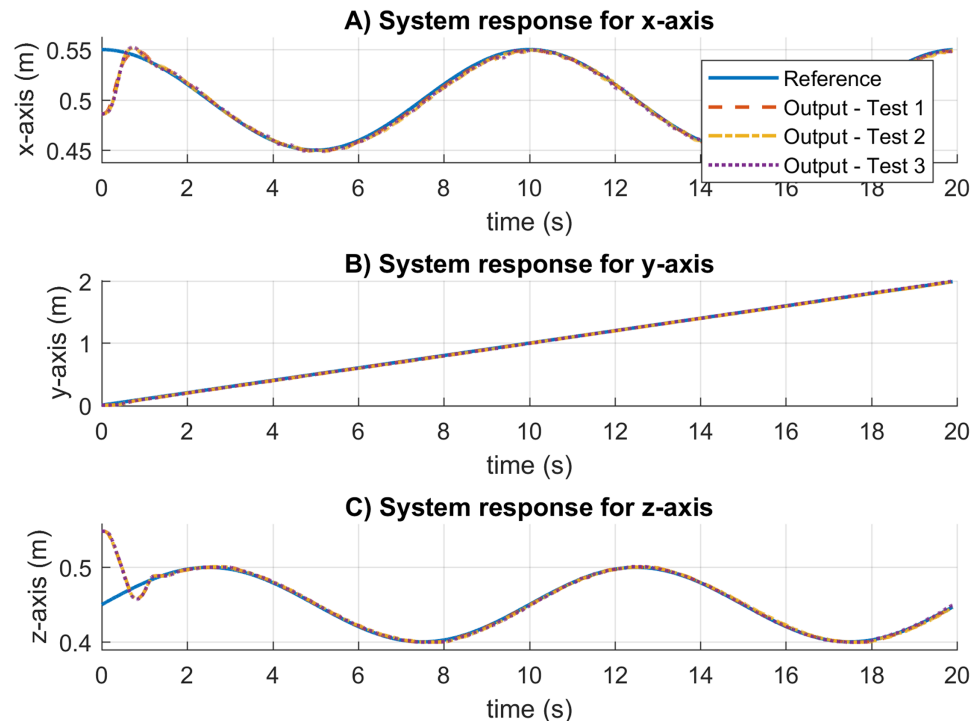
Figure 15 System response results for the sinusoidal trajectory in real experiments. (A–C) System responses for the x-axis, y-axis, and z-axis, respectively. Full-size ◪ DOI: 10.7717/peerj-cs.393/fig-15

The SNPD controller is considered for this test under the sinusoidal trajectory. The results for the system response are provided in Fig. 15. The system response results are very similar for the three tests. This indicates that no adjustment to the controller parameters is required to handle uncertainties in unmodeled changes in system dynamics. Moreover, there is no presence of steady-state errors, and the controller reaches the references within an adequate settling time.

The trajectory following and velocity control signals for these tests are illustrated in Fig. 16. In Fig. 16A, the results show the correct trajectory following as expected from the system response results. As can be seen in Figs. 16C and 16D, the SNPD computes bigger control signals than the results in Fig. 16B. This shows that the weights are dynamically adapted to reject disturbances and changes in dynamics during the experimental test
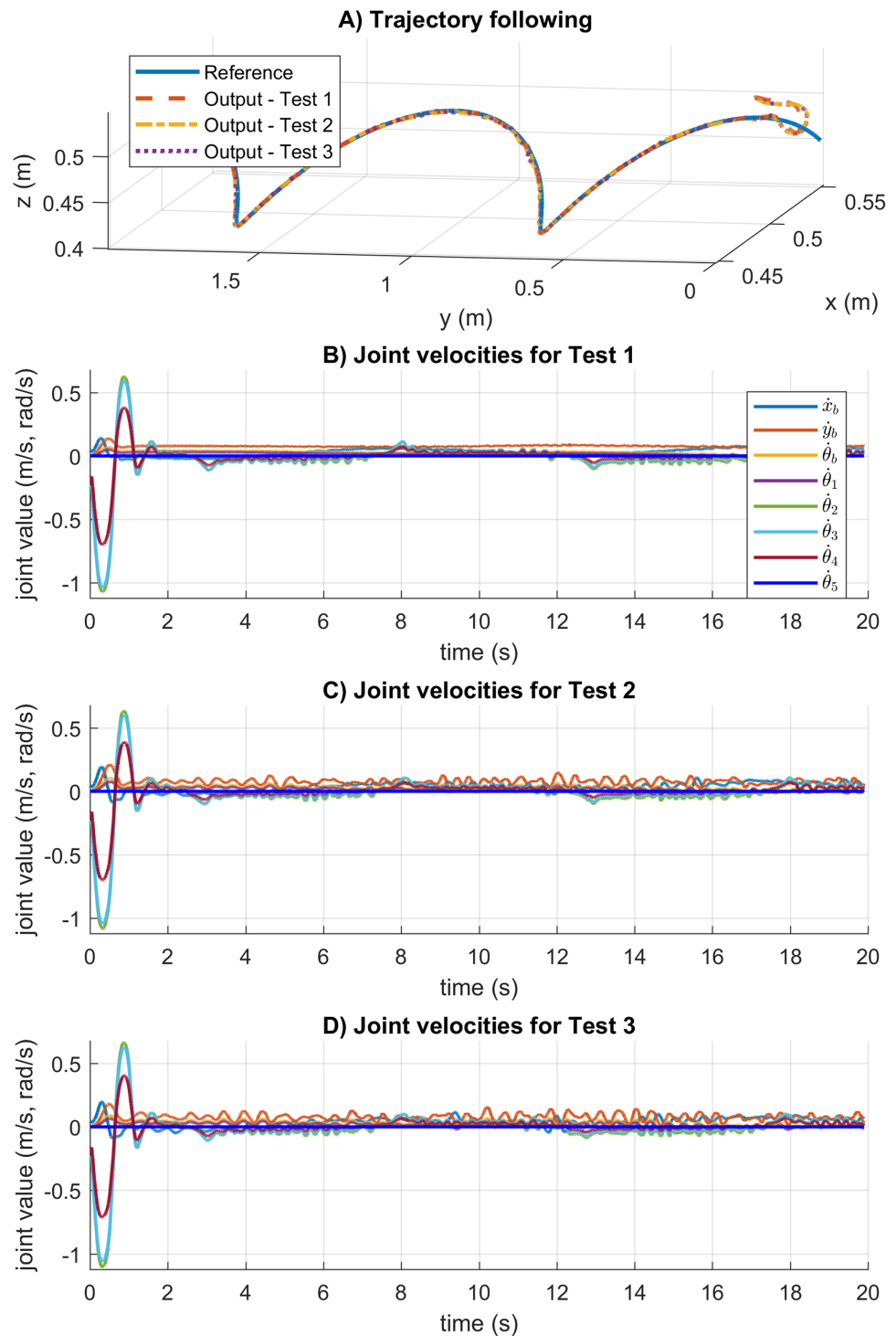
Figure 16 Trajectory following and velocity control signal results for the sinusoidal trajectory in real experiments. (A) The trajectory following results. (B–D) The velocity control signal results.

Full-size ☒ DOI: 10.7717/peerj-cs.393/fig-16

Table 6 Experimental results for the sinusoidal trajectory in real experiments.

Measure	Test	e_x	e_y	e_z
RMS	Test 1	7.7389×10^{-3}	5.0422×10^{-3}	1.2476×10^{-2}
	Test 2	8.2315×10^{-3}	7.2873×10^{-3}	1.2537×10^{-2}
	Test 3	8.2503×10^{-3}	6.8967×10^{-3}	1.2678×10^{-2}
MAD	Test 1	2.4092×10^{-3}	1.9660×10^{-3}	4.1409×10^{-3}
	Test 2	2.9295×10^{-3}	2.5150×10^{-3}	4.1397×10^{-3}
	Test 3	2.8684×10^{-3}	4.1865×10^{-3}	4.1863×10^{-3}

Finally, the Table 6 shows the RMS and MAD results for the sinusoidal trajectory. The SNPD scheme has been demonstrated to have better results than conventional PID with the smallest RMS and MAD results in general. There is not a big difference between the RMS and MAD results, which indicates that the proposed SNPD controller is robust to unmodeled dynamics and disturbances.

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 was considered for trajectory tracking of the KUKA Youbot mobile manipulator. Simulation and real experiments were performed to compare the classical PID controller against the proposals. An existing adaptive single neuron controller (SNA-PID) is also considered for comparison. Simulation and experiment results reported that conventional PID control reported overshoot and steady-state errors. The SNA-PID controller highly reduced the overshoot, but it presented small steady-state errors. 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. Finally, the presented approach considers the

system kinematics to compute a motion trajectory, specified at the end-effector position into equivalent joint position and velocities. As future work, the system dynamics can be considered to compute a motion trajectory in terms of position, velocities, and acceleration based on second-order algorithms.

ADDITIONAL INFORMATION AND DECLARATIONS

Competing Interests
Alma Yolanda Alanis Garcia is an Academic Editor for PeerJ.

Author Contributions
- Jesus Hernandez-Barragan conceived and designed the experiments, performed the experiments, performed the computation work, prepared figures and/or tables, authored or reviewed drafts of the paper, and approved the final draft.
- Jorge D. Rios conceived and designed the experiments, performed the experiments, prepared figures and/or tables, authored or reviewed drafts of the paper, and approved the final draft.
- Javier Gomez-Avila conceived and designed the experiments, performed the experiments, performed the computation work, prepared figures and/or tables, authored or reviewed drafts of the paper, and approved the final draft.
- Nancy Arana-Daniel analyzed the data, authored or reviewed drafts of the paper, supervision and project administration, and approved the final draft.
- Carlos Lopez-Franco analyzed the data, authored or reviewed drafts of the paper, supervision and project administration, and approved the final draft.
- Alma Y. Alanis analyzed the data, authored or reviewed drafts of the paper, supervision and project administration, and approved the final draft.

Data Availability
The following information was supplied regarding data availability:
Raw data and code are available at GitHub:
https://github.com/JesusHernandezBarragan/Adaptive-neural-PD-controllers-for-mobile-manipulator-trajectory-tracking.git.

REFERENCES

Alanis A, Arana-Daniel N, Lopez-Franco C. 2019. *Artificial neural networks for engineering applications.* Amsterdam: Elsevier Science.

Angel L, Viola J, 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.* Piscataway: IEEE, 1–6.

Åström KJ, Hägglund T. 1995. *PID controllers: theory, design, and tuning.* Vol. 2. Durham: Instrument society of America Research Triangle Park.

Bryson J. 2019. *The past decade and future of AI's impact on society.* Vol. 11. Madrid: Turner.

Chen Y-m, He Y-l, Zhou M-f. 2015. Decentralized PID neural network control for a quadrotor helicopter subjected to wind disturbance. *Journal of Central South University* **22(1)**:168–179 DOI 10.1007/s11771-015-2507-9.

Craig J. 2005. *Introduction to robotics mechanics and control.* Third Edition. London: Pearson Education, Inc.

Ge SS, Zhang J, Lee TH. 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 DOI 10.1109/TSMCB.2004.826827.

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.* Amsterdam: Elsevier, 55–63.

Haykin S. 2004. Kalman filtering and neural networks. In: *Adaptive and cognitive dynamic systems: signal processing, learning, communications and control.* New York: Wiley.

Hernandez T, Nuño E, Alanis AY. 2016. Teleoperation of mobile manipulators with non-holonomic restrictions. In: *2016 IEEE 13th International Conference on Networking, Sensing, and Control (ICNSC).* Piscataway: IEEE, 1–6.

Hernandez-Barragan J, Rios JD, Alanis AY, Lopez-Franco C, Gomez-Avila J, Arana-Daniel N. 2020. Adaptive single neuron anti-windup pid controller based on the extended kalman filter algorithm. *Electronics* **9(4)**:636 DOI 10.3390/electronics9040636.

Jiao J, Chen J, Qiao Y, Wang W, Wang C, Gu L. 2018. Single neuron pid control of agricultural robot steering system based on online identification. In: *2018 IEEE Fourth International Conference on Big Data Computing Service and Applications (BigDataService).* Piscataway: IEEE, 193–199.

Johnson M, Moradi M. 2006. *PID control: new identification and design methods—probability and its applications.* London: Springer.

Khalil HK. 2002. *Nonlinear systems; 3rd ed—the book can be consulted by contacting: PH-AID: Wallet, Lionel.* Upper Saddle: River, Prentice-Hall.

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

Kumar S, 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.* 1–4.

Kundu AS, Mazumder O, Dhar A, Lenka PK, Bhaumik S. 2017. Scanning camera and augmented reality based localization of omnidirectional robot for indoor application. *Procedia Computer Science* **105**:27–33.

Li Z, Ge S. 2017. *Fundamentals in modeling and control of mobile manipulators—automation and control engineering.* Abingdon: Taylor & Francis Group.

Li Z, Yang C, Su C, Deng J, 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.

Lin S, Goldenberg AA. 2001. Neural-network control of mobile manipulators. *IEEE Transactions on Neural Networks* **12(5)**:1121–1133.

Lopez-Franco C, Gomez-Avila J, Alanis AY, Arana-Daniel N, Villaseñor C. 2017. Visual servoing for an autonomous hexarotor using a neural network based pid controller. *Sensors* **17(8)**:1865 DOI 10.3390/s17081865.

Lopez-Franco C, Hernandez-Barragan J, Alanis AY, Arana-Daniel N. 2018. A soft computing approach for inverse kinematics of robot manipulators. *Engineering Applications of Artificial Intelligence* **74(4)**:104–120 DOI 10.1016/j.engappai.2018.06.001.

Maglogiannis I, Iliadis L, 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. In: *IFIP Advances in Information and Communication Technology.* Berlin: Springer International Publishing.

Moradi MH, Katebi MR, Johnson MA. 2001. Predictive pid control: a new algorithm. *IECON'01. 27th Annual Conference of the IEEE Industrial Electronics Society* **1**:764–769.

Ogata K. 2010. *Modern control engineering—instrumentation and controls series.* Upper Saddle: River, Prentice-Hall.

Rios JD, Alanis AY, Arana-Daniel N, Lopez-Franco C. 2020b. Real-time neural observer-based controller for unknown nonlinear discrete delayed systems. *International Journal of Robust and Nonlinear Control* **30(18)**:8402–8429 DOI 10.1002/rnc.5250.

Rios J, Alanis A, Arana-Daniel N, Lopez-Franco C, Sanchez E. 2020a. *Neural networks modeling and control: applications for unknown nonlinear delayed systems in discrete time.* Amsterdam: Elsevier Science.

Rivera-Meja J, Léon-Rubio A, Arzabala-Contreras E. 2012. Pid based on a single artificial neural network algorithm for intelligent sensors. *Journal of applied research and technology* **10(2)**:262–282.

Sanchez EN, Alanis AY. 2006. *Redes neuronales: conceptos fundamentales y aplicaciones a control automático.* Upper Saddle River: Cinvestav Unidad Guadalara. Editorial Prentice Hall.

Sanchez E, Alanis A, Loukianov A. 2010. Discrete-time high order neural control: trained with kalman filtering. In: *Studies in computational intelligence.* Berlin: Springer.

Sarangapani J. 2018. *Neural network control of nonlinear discrete-time systems—automation and control engineering.* Boca Raton: CRC Press.

Sciavicco L, Siciliano B. 2008. Robotics—modelling, planning and control. In: *Advanced Textbooks in Control and Signal Processing.* Second Edition. Berlin: Springer.

Sento A, Kitjaidure Y. 2016. Neural network controller based on pid using an extended kalman filter algorithm for multi-variable non-linear control system. In: *2016 Eighth International Conference on Advanced Computational Intelligence (ICACI).* 302–309.

Song Y, Grizzle JW. 1992. The extended kalman filter as a local asymptotic observer for nonlinear discrete-time systems. In: *1992 American Control Conference.* 3365–3369.

Spong MW, Vidyasagar M. 2008. *Robot dynamics and control.* Hoboken: John Wiley & Sons.

Tahoun A. 2015. Adaptive stabilization of neutral systems with nonlinear perturbations and mixed time-varying delays. *International Journal of Adaptive Control and Signal Processing* **29(10)**:1328–1340 DOI 10.1002/acs.2545.

Tahoun A. 2017a. Anti-windup adaptive pid control design for a class of uncertain chaotic systems with input saturation. *ISA transactions* **66(25)**:176–184 DOI 10.1016/j.isatra.2016.10.002.

Tahoun A. 2017b. A new online delay estimation-based robust adaptive stabilizer for multi-input neutral systems with unknown actuator nonlinearities. *ISA transactions* **70(11)**:139–148 DOI 10.1016/j.isatra.2017.07.012.

Tahoun AH. 2017c. Less-conservative robust adaptive control of neutral systems with mixed time-delays. *International Journal of Systems Science* **48(4)**:675–685 DOI 10.1080/00207721.2016.1206987.

Tahoun AH. 2020. Fault-tolerant control for a class of quantised networked control of nonlinear systems with unknown time-varying sensor faults. *International Journal of Control* **93(3)**:619–628 DOI 10.1080/00207179.2018.1484170.

Tahoun A, Arafa M. 2020. A new unmatched-disturbances compensation and fault-tolerant control for partially known nonlinear singular systems. *ISA Transactions* **104**:310–320.

Tang W, Wang L, Gu J, Gu Y. 2020. Single neural adaptive pid control for small uav micro-turbojet engine. *Sensors* **20(2)**:345 DOI 10.3390/s20020345.

Temel S, Yagli S, Gören S. 2013. *P, pd, pi, pid controllers.* Ankara: Middle East Technical University, Electrical and Electronics Engineering Department.

Tian Y-C, Tadé MO, Tang J. 1999. A nonlinear pid controller with applications. *IFAC Proceedings* **32(2)**:2657–2661.

Villaseñor C, Rios JD, Arana-Daniel N, Alanis AY, Lopez-Franco C, Hernandez-Vargas EA. 2018. Germinal center optimization applied to neural inverse optimal control for an all-terrain tracked robot. *Applied Sciences* **8(1)**:31.

Visioli A. 2006. *Practical PID control: advances in industrial control.* London: Springer.

Wu J, Lv C, Zhao L, Li R, 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).* Piscataway: IEEE, 410–415.

Zeng G-Q, Xie X-Q, Chen M-R, Weng J. 2019. Adaptive population extremal optimization-based PID neural network for multivariable nonlinear control systems. *Swarm and Evolutionary Computation* **44(4)**:320–334 DOI 10.1016/j.swevo.2018.04.008.

Zhang G, Qin W, Qin Q, He B, 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).* 2957–2962.