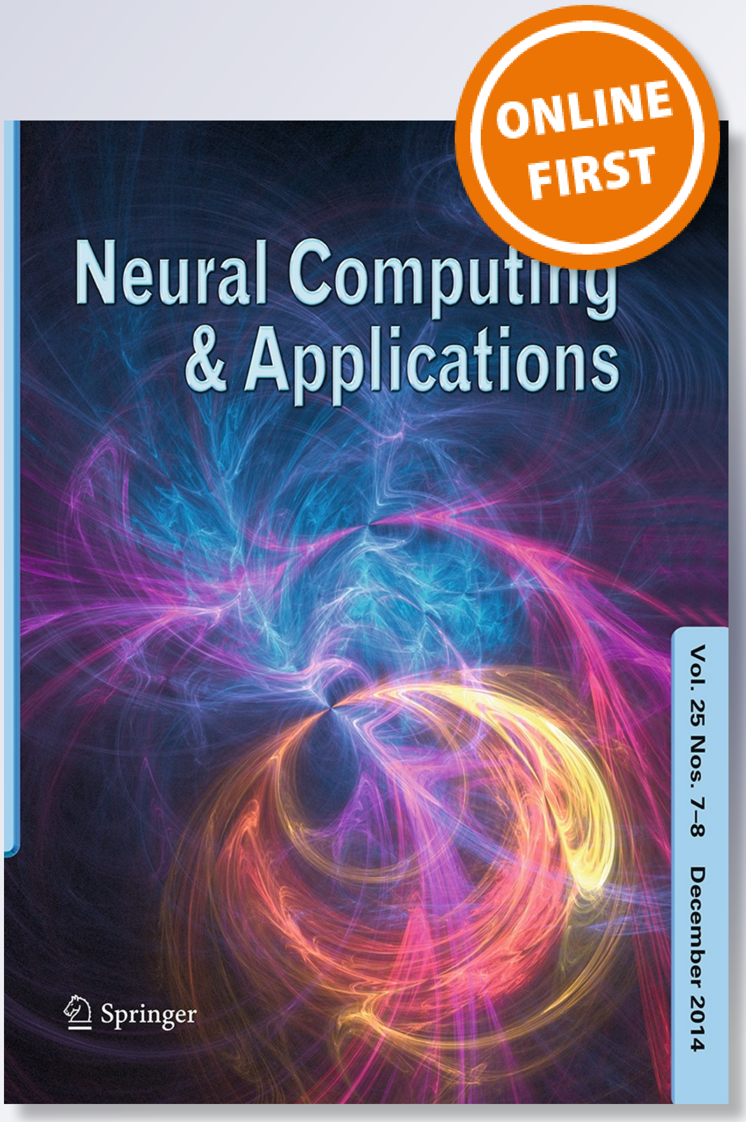
Identification and control of nonlinear dynamics of a mobile robot in discrete time using an adaptive technique based on neural PID

F. G. Rossomando & C. M. Soria

Neural Computing and Applications

ISSN 0941-0643

Neural Comput & Applic
DOI 10.1007/s00521-014-1805-8


ONLINE FIRST
Neural Computing & Applications
Springer
Vol. 25 Nos. 7–8 December 2014

Neural Comput & Applic
DOI 10.1007/s00521-014-1805-8

ORIGINAL ARTICLE

Identification and control of nonlinear dynamics of a mobile robot in discrete time using an adaptive technique based on neural PID

F. G. Rossomando · C. M. Soria

Received: 30 June 2014 / Accepted: 8 December 2014

Abstract In this work, original results, concerning the application of a discrete-time adaptive PID neural controller in mobile robots for trajectory tracking control, are reported. In this control strategy, the exact dynamical model of the robot does not need to be known, but a neural network is used to identify the dynamic model. To implement this strategy, two controllers are implemented separately: a kinematic controller and an adaptive neural PID controller. The uncertainty and variations in the robot dynamics are compensated by an adaptive neural PID controller. It is efficient and robust in order to achieve a good tracking performance. The stability of the proposed technique, based on the discrete-time Lyapunov's theory, is proven. Finally, experiments on the mobile robot have been developed to show the performance of the proposed technique, including the comparison with a classical PID.

Keywords MIMO system · Neural networks · Nonlinear control · Adaptive control

1 Introduction

Currently, classical PID control schemes are commonly used in the industry because the physical meaning of their control parameters is clear and easy to interpret. But the robotics systems are represented by nonlinear systems, such as most of the physical processes and mechanical systems. Generally, it is difficult to find the optimal set of parameters for the above systems; furthermore, in some cases they are time variant. However, the gains of these controllers are very important for determining the behavior of the control system. In recent years, various papers have proposed tuning methods [1, 2]. Additionally, some methods designed for linear systems do not have a good performance to run on nonlinear systems.

In the field of robot control, the robot's model and their parameters are not easy to obtain accurately. Some of the problems of classical PID algorithms that exist in the robot control arise when their parameters vary with time, or when the parameters do not vary at the same time or when they are set to a constant value. Therefore, advanced PID controllers have been developed quickly in the control field of robots, including a lot of algorithms with PID tuning parameters. Some of these methods are based on the fuzzy technology [3, 5–7], PID controllers based on neural networks [8, 9], or genetic algorithms [4, 10, 11], and other intelligent control theories combined with PID controllers [12–16].

In papers about PID controllers based on neural networks with applications in mobile robotics, Li et al. [17] can be mentioned, where a hybrid control algorithm was proposed including a PID control with a dynamic sliding mode and a *backstepping* kinematics control. The algorithm was applied to a non-holonomic mobile robot. The gain of the sliding mode control is adjusted by a RBF-NN network with an adaptive adjustment algorithm. The result gives two torques based on a dynamic model. The results shown are based on simulations and do not use any method to identify the model to be controlled. In addition, the complete control system is designed in the continuous domain.

In Normey-Rico [1], a PID controller applied to a non-holonomic mobile robot for trajectory tracking, is

F. G. Rossomando (✉) · C. M. Soria
INAUT - Universidad Nacional de San Juan (UNSJ - Conicet), Capital, San Juan, Argentina
e-mail: frosoma@inaut.unsj.edu.ar

C. M. Soria
e-mail: csoria@inaut.unsj.edu.ar

proposed. In such paper, the development and implementation of an adaptive PID controller on a linear model of the robot with experimental results of the proposed technique, is described. In [2], the author uses a PID controller, which is set by identifying the robot through the recursive least squares method. The experimental and simulation results are used to validate the proposed technique.

This work aims at showing the design and implementation of an adaptive neural PID controller on the nonlinear dynamics of a mobile robot. The control structure is divided into two parts: kinematic and dynamic controllers. A control technique of inverse model, which has no parameter variations, is applied to the robot kinematics. Therefore, it is not necessary to apply adaptive control techniques. On the dynamic part, the parameters may vary due to friction, mass loss, sliding, etc. In addition, the robot has a nonlinear dynamics. Thus, the adaptive neural PID controller is proposed to compensate the dynamic nonlinearities and their variations.

The control technique has the following advantages:

1. It can be applied to any MIMO nonlinear system, as in the case of a mobile robot.
2. The exact model of the process or system (in this case the robot dynamics) does not need to be known.
3. The design of a MIMO neuronal PID controller is simple, and it is based on the topology of a dynamic neural network which is adaptable.
4. The controller shown in this article is based on a control action of linear and angular velocity, and not on the torque applied to the wheels.
5. The identification of the dynamic model, the design, analysis and implementation of the controllers were made in discrete time, facilitating thus the programming in computers and digital equipment. Besides, the stability problems caused by direct implementation in discrete time of a system designed in the continuous domain are avoided.
6. A stability analysis performed in discrete time, demonstrating the convergence of the direct neural model to identify the dynamic model as well as the convergence of the proposed controller, is presented.

This article is organized as follows: Sect. 2 presents an overview of the system and shows the mathematical representation of the unicycle robot model. The kinematic control, nonlinear system identification and neural PID control are studied in Sects. 3, 4 and 5, respectively. In Sect. 6, the stability of the proposed system on a mobile robot is shown. Experimental results are presented in Sect. 7 showing the efficiency of the controllers. Finally, the conclusions are presented in Sect. 8.

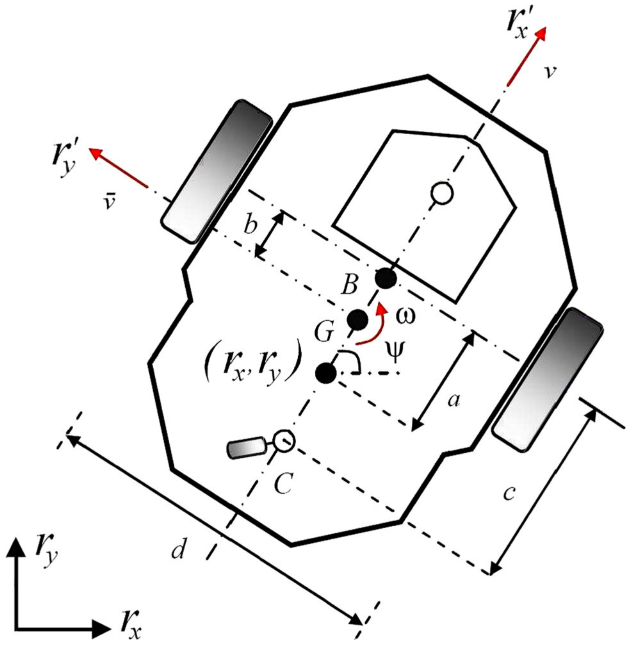


Fig. 1 Parameters of the unicycle mobile robot

2 Model of the mobile robot

2.1 Review of the mobile robotic system

In mobile robotics research, generally the task performed by the robot requires high speed and many times the mass and the friction are time-varying. Therefore, the dynamics of the robot must be considered in these cases. Furthermore, the unicycle mobile robot is non-holonomic, since it has two drive wheels that receive control commands from linear and angular velocities. In practice, the presence of non-holonomic constraints in the mechanical system prevents the robot that performs some trajectories.

In this section, the model of a unicycle-type mobile robot is introduced. Figure 1 shows the parameters and variables of interest. Here, x_1 and x_2 are the linear and angular velocities of the robot, G is the center of mass of the robot, c is the position of the castor wheel, h is the point of interest with coordinates r_x, r_y on the XY plane, ψ is the robot orientation, and a is the distance between the landmark and the center point of the virtual axis that links the drive wheels. The mathematical representation of the full model [18] is given by the model: Kinematic model

$$\begin{pmatrix} \dot{r}_x(t) \\ \dot{r}_y(t) \\ \dot{\psi}(t) \end{pmatrix} = \begin{pmatrix} \cos\psi(t) & -a\sin\psi(t) \\ \sin\psi(t) & a\cos\psi(t) \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix} + \begin{pmatrix} \delta_{rx}(t) \\ \delta_{ry}(t) \\ 0 \end{pmatrix} \tag{1}$$

Dynamic model

$$\begin{pmatrix}\dot{x}_1(t)\\ \dot{x}_2(t)\end{pmatrix} = \begin{pmatrix}\frac{\vartheta_3}{\vartheta_1}x_2^2(t) - \frac{\vartheta_4}{\vartheta_1}x_1(t)\\ -\frac{\vartheta_5}{\vartheta_2}x_1(t)x_2(t) - \frac{\vartheta_6}{\vartheta_2}x_2(t)\end{pmatrix} + \begin{pmatrix}\frac{1}{\vartheta_1} & 0\\ 0 & \frac{1}{\vartheta_2}\end{pmatrix}\begin{pmatrix}u_{1\mathrm{ref}}(t)\\ u_{2\mathrm{ref}}(t)\end{pmatrix} + \begin{pmatrix}\delta_1(t)\\ \delta_2(t)\end{pmatrix} \tag{2}$$

The uncertainties parameters associated to the mobile robot are δ_{rx}, δ_{ry}, δ_1 and δ_2. The parameters δ_{rx} and δ_{ry} are functions of the sliding speed and robot orientation, δ_1 and δ_2 depend on physical parameters such as mass, momentum, wheel diameters and parameters of the servos, forces applied to the wheels, and other factors considered as disturbances.

The robot model presented in (1) and (2) is divided into a kinematics part and a dynamics part, respectively. Thus, two controllers are applied, one of them based on feedback linearization for the robot kinematics, and the other one based on an adaptable neural network (for the robot dynamics).

The discretization of the mobile robot model for the purpose of digital control is: Discrete Kinematic model

$$\begin{pmatrix}r_x(k+1)\\ r_y(k+1)\\ \psi(k+1)\end{pmatrix} = T_0\begin{pmatrix}\cos\psi(k) & -a\sin\psi(k)\\ \sin\psi(k) & a\cos\psi(k)\\ 0 & 1\end{pmatrix}\begin{pmatrix}x_1(k)\\ x_2(k)\end{pmatrix} + \begin{pmatrix}r_x(k)\\ r_y(k)\\ \psi(k)\end{pmatrix} + \begin{pmatrix}\delta_{rx}\\ \delta_{ry}\\ 0\end{pmatrix} \tag{3}$$

Model of discretized dynamics

$$\begin{pmatrix}x_1(k+1)\\ x_2(k+1)\end{pmatrix} = \begin{pmatrix}\frac{\Omega_3}{\Omega_1}x_2^2(k) + \frac{\Omega_4}{\Omega_1}x_1(k)\\ -\frac{\Omega_5}{\Omega_2}x_1(k)x_2(k) + \frac{\Omega_6}{\Omega_2}x_2(k)\end{pmatrix} + \begin{pmatrix}\frac{1}{\Omega_1} & 0\\ 0 & \frac{1}{\Omega_2}\end{pmatrix}\begin{pmatrix}u_{1\mathrm{ref}}(k)\\ u_{2\mathrm{ref}}(k)\end{pmatrix} + \begin{pmatrix}\delta_1\\ \delta_2\end{pmatrix} \tag{4}$$

where the parameters are defined as:

$$\frac{\Omega_3}{\Omega_1} = \left(\frac{\vartheta_3}{\vartheta_1}T_0\right); \quad \frac{\Omega_4}{\Omega_1} = \left(1 - \frac{\vartheta_4}{\vartheta_1}T_0\right); \quad \frac{1}{\Omega_1} = \left(\frac{1}{\vartheta_1}T_0\right);$$
$$\frac{\Omega_5}{\Omega_2} = \left(-\frac{\vartheta_5}{\vartheta_2}T_0\right); \quad \frac{\Omega_6}{\Omega_2} = \left(1 - \frac{\vartheta_6}{\vartheta_2}T_0\right); \quad \frac{1}{\Omega_2} = \left(\frac{1}{\vartheta_2}T_0\right) \tag{5}$$

where T_0 is the sampling time and k is the discrete time.

The identified parameters for the Pioneer DX2 mobile robot are: $\vartheta_1 = 0.3037$; $\vartheta_2 = 0.2768$; $\vartheta_3 = -0.0004018$; $\vartheta_4 = 0.9835$; $\vartheta_5 = -0.003818$; $\vartheta_6 = 1.0725$ and $T_0 = 0.1$ seg. These parameters were obtained in [18].

3 Kinematic controller

The controller design is based on a kinematic model of the robot. The proposed kinematic controller is:

$$\begin{pmatrix}x^c_{1\,\mathrm{ref}}(k)\\ x^c_{2\,\mathrm{ref}}(k)\end{pmatrix} = \begin{pmatrix}\frac{\cos\psi(k)}{T_0} & \frac{\sin\psi(k)}{T_0}\\ -\frac{\sin\psi(k)}{aT_0} & \frac{\cos\psi(k)}{aT_0}\end{pmatrix} \times \left[\begin{pmatrix}r_{x\,\mathrm{ref}}(k+1) + l_x\tanh\left(\frac{k_x}{l_x}\tilde{r}_x(k)\right)\\ r_{y\,\mathrm{ref}}(k+1) + l_y\tanh\left(\frac{k_y}{l_y}\tilde{r}_y(k)\right)\end{pmatrix} - \begin{pmatrix}r_x(k)\\ r_y(k)\end{pmatrix}\right] \tag{6}$$

where $\begin{pmatrix}\tilde{r}_x(k) & \tilde{r}_y(k)\end{pmatrix}^T = \begin{pmatrix}r_{x\mathrm{ref}}(k) - r_x(k) & r_{y\mathrm{ref}}(k) - r_y(k)\end{pmatrix}^T$ are the position errors and the tan$h(\cdot)$ function has been added to avoid a saturation of the control actions due to large position errors. By replacing (6) in (3) and considering a perfect velocity tracking that is $x^c_{1\mathrm{ref}}(k) \equiv x_1(k); x^c_{2\mathrm{ref}}(k) \equiv x_2(k)$, the closed-loop equation is:

$$\begin{pmatrix}\tilde{r}_x(k+1)\\ \tilde{r}_y(k+1)\end{pmatrix} - \begin{pmatrix}l_x\tanh\left(\frac{k_x}{l_x}\tilde{r}_x(k)\right)\\ l_y\tanh\left(\frac{k_y}{l_y}\tilde{r}_y(k)\right)\end{pmatrix} = \begin{pmatrix}0\\ 0\end{pmatrix} \tag{7}$$

Defining the output error vector, $\tilde{\mathbf{h}}(k) = \begin{pmatrix}\tilde{r}_x(k) & \tilde{r}_y(k)\end{pmatrix}^T = \begin{pmatrix}r_{x\,\mathrm{ref}}(k+1) & r_{y\,\mathrm{ref}}(k+1)\end{pmatrix}^T - \begin{pmatrix}r_x(k) & r_y(k)\end{pmatrix}^T$, (7) can be written as:

$$\tilde{\mathbf{h}}(k+1) = \left[l_x\tanh\left(\frac{k_x}{l_x}\tilde{r}_x(k)\right) \quad l_y\tanh\left(\frac{k_y}{l_y}\tilde{r}_y(k)\right)\right] \tag{8}$$

By taking a Lyapunov candidate $V(k) = \tilde{\mathbf{h}}(k)\tilde{\mathbf{h}}^T(k)$, and for k_x, $k_y < 1$, $k_x/l_x < 1$ and $k_y/l_y < 1$, then $\tilde{h}(k) \to 0$ for $k \to \infty$. The assumption about perfect velocity tracking will be relaxed when analyzing the stability of the whole control system.

$V(k) = \tilde{\mathbf{h}}(k)\tilde{\mathbf{h}}^T(k)$, y para $k_x, k_y < 1$, $k_x/l_x < 1$; $k_y/l_y < 1$,

entonces $\tilde{\mathbf{h}}(k) \to 0$ for $k \to \infty$.

4 Identification of the nonlinear dynamic model

To implement the PID control with variable gain, a back-propagation of the output error of the mobile robot dynamic system to the control action, is necessary. The model identification error is defined in the following way:

$$\mathbf{e}_m(k) = \mathbf{x}(k) - \hat{\mathbf{x}}(k) = (x_1(k) - \hat{x}_1(k), x_2(k) - \hat{x}_2(k))^T \tag{9}$$

The dynamics of the robot is nonlinear; it is defined in (5) and can be represented by a neural network, defined by:

$$\mathbf{x}(k+1) = \boldsymbol{\theta}_f^* \xi(\zeta(k)) + \boldsymbol{\theta}_g^* \mathbf{u}(k) + \boldsymbol{\theta}_\delta^* \chi(\upsilon(k)) + \varepsilon \tag{10}$$

where $\boldsymbol{\theta}_f^*$ $(j \times m)$, $\boldsymbol{\theta}_g^*$ $(j \times j)$ and $\boldsymbol{\theta}_\delta^*$ $(m \times j)$ are the vectors of optimal parameters of the function $f(x)$ and g of (4). The functions ξ $(m \times 1)$ and χ $(m \times 1)$ are radial basis functions related to the weights $\boldsymbol{\theta}_f$ and $\boldsymbol{\theta}_\delta$ respectively, and $\boldsymbol{\varepsilon}(m \times 1)$ is the lowest approximation error with respect to the dynamic system, where j is the number of outputs, m is the number of neurons, being:

$$\boldsymbol{\varepsilon}(k) = \left[\mathbf{f}(\mathbf{x}) - \theta_f^*(k)\boldsymbol{\xi}(\varsigma(k))\right] + \left[\mathbf{g} - \boldsymbol{\theta}_g^*\right]\mathbf{u}(k) + \left[\boldsymbol{\delta}(k) - \boldsymbol{\theta}_\delta^*(k)\chi(\upsilon(k))\right] \tag{11}$$

Since the optimal parameter vectors are unknown, then, it is necessary to estimate their values. An estimate function based on (4) is defined as follows:

$$\hat{\mathbf{x}}(k+1) = \hat{\theta}_f(k)\boldsymbol{\xi}(\zeta(k)) + \hat{\theta}_g \mathbf{u}(k) + \hat{\theta}_\delta(k)\chi(\upsilon(k)) + \boldsymbol{\varepsilon} \tag{12}$$

From (9), (10) and (11), the identification error can be described by:

$$\mathbf{e}_m(k+1) = \tilde{\theta}_f(k)\boldsymbol{\xi}(\zeta(k)) + \tilde{\theta}_g \mathbf{u}(k) + \tilde{\theta}_\delta(k)\chi(\upsilon(k)) \tag{13}$$

where ξ and χ are the activation functions of each neuron in the hidden layer, and $\tilde{\theta}_f(k) = \left(\boldsymbol{\theta}_f^* - \hat{\theta}_f(k)\right)$;.$\tilde{\theta}_g = \left(\boldsymbol{\theta}_g^* - \hat{\theta}_g(k)\right)$; and $\tilde{\theta}_\delta(k) = \left(\boldsymbol{\theta}_\delta^* - \hat{\theta}_\delta(k)\right)$

$$\xi(\zeta(k)) = \exp\left(-\frac{(\zeta(k) - \mathbf{c}_i)^T(\zeta(k) - \mathbf{c}_i)}{2\varsigma_i^2}\right) \tag{14}$$

and

$$\chi(\boldsymbol{\upsilon}(k)) = \exp\left(-\frac{(\boldsymbol{\upsilon}(k) - \mathbf{c}_i)^T(\boldsymbol{\upsilon}(k) - \mathbf{c}_i)}{2\varsigma_i^2}\right) \tag{15}$$

where i is the ith neuron of the hidden layer, $\boldsymbol{c}_i$ is the central position of the ith neuron, and ς_i is the width factor (equal to one in this case) of the Gaussian function.

The vectors ζ and υ are regressors of the neural model, defined as:

$$\boldsymbol{\zeta}(k) = [\mathbf{x}(k), \mathbf{x}(k-1), \mathbf{x}(k-2), \mathbf{u}(k)] \tag{16}$$

and

$$\boldsymbol{\upsilon}(k) = [\mathbf{e}_m(k), \mathbf{e}_m(k-1), \mathbf{e}_m(k-2), \mathbf{u}(k)] \tag{17}$$

Consideration 1: The optimal approximation error $\boldsymbol{\varepsilon}$ is bounded by a constant $\|\boldsymbol{\varepsilon}\| \le \varepsilon_{\max}$.

Consideration 2: The control action signal $u(k)$ is limited to the maximum input $u_i(k) \le u_{i\mathrm{Max}}$ to avoid possible damage.

Theorem: *Considering the nonlinear system defined by (4), it can be approximated by the neural network (12) using a neuronal adjustment law defined by:*

$$\Delta\boldsymbol{\theta}_{fi} = -\gamma_1 e_{mi}\boldsymbol{\xi}(\boldsymbol{\zeta}(k)) \tag{18}$$

$$\Delta\boldsymbol{\theta}_{gi} = -\gamma_2 e_{im} u_i(k) \tag{19}$$

$$\Delta\boldsymbol{\theta}_{\delta i} = -\gamma_3 e_{mi}\boldsymbol{\chi}(\upsilon(k)) \tag{20}$$

Proof Defining a candidate function of Lyapunov, with V a positive definite function given by:

$$V(k) = \frac{1}{2}\sum_{i=1}^{2}\left[e_{mi}^2(k) + \gamma_1^{-1}\left(\tilde{\theta}_{fi}^T(k-1)\tilde{\theta}_{fi}(k-1)\right) + \gamma_2^{-1}\left(\tilde{\theta}_{gi}^T(k-1)\tilde{\theta}_{gi}(k-1)\right) + \gamma_3^{-1}\left(\tilde{\theta}_{\delta i}^T(k-1)\tilde{\theta}_{\delta i}(k-1)\right)\right] \tag{21}$$

Now, taking the difference in discrete time ΔV as follows,

$$\begin{aligned}\Delta V = \sum_{i=1}^{2}\Big[&\left(e_{mi}^2(k+1) - e_{mi}^2(k)\right)\\ &+ \gamma_1^{-1}\left(\tilde{\theta}_{fi}^T(k)\tilde{\theta}_{fi}(k) - \tilde{\theta}_{fi}^T(k-1)\tilde{\theta}_{fi}(k-1)\right) + \cdots\\ &+ \gamma_2^{-1}\left(\tilde{\theta}_{gi}^T(k)\tilde{\theta}_{gi}(k) - \tilde{\theta}_{gi}^T(k-1)\tilde{\theta}_{fi}(k-1)\right)\\ &+ \gamma_3^{-1}\left(\tilde{\theta}_{\delta i}^T(k)\tilde{\theta}_{\delta i}(k) - \tilde{\theta}_{\delta i}^T(k-1)\tilde{\theta}_{\delta i}(k-1)\right)\Big]\end{aligned} \tag{22}$$

Defining $\Delta\boldsymbol{\theta}_{fi}$, $\Delta\boldsymbol{\theta}_{gi}$ and $\Delta\boldsymbol{\theta}_{\delta i}$ as:

$$\begin{aligned}\Delta\boldsymbol{\theta}_{fi} &= \gamma_1^{-1}\left(\tilde{\theta}_{fi}^T(k)\tilde{\theta}_{fi}(k) - \tilde{\theta}_{fi}^T(k-1)\tilde{\theta}_{fi}(k-1)\right)\\ \Delta\boldsymbol{\theta}_{gi} &= \gamma_2^{-1}\left(\tilde{\theta}_{gi}^T(k)\tilde{\theta}_{gi}(k) - \tilde{\theta}_{gi}^T(k-1)\tilde{\theta}_{fi}(k-1)\right)\\ \Delta\boldsymbol{\theta}_{\delta i} &= \gamma_3^{-1}\left(\tilde{\theta}_{\delta i}^T(k)\tilde{\theta}_{\delta i}(k) - \tilde{\theta}_{\delta i}^T(k-1)\tilde{\theta}_{\delta i}(k-1)\right)\end{aligned} \tag{23}$$

and organizing (22), we have,

$$\begin{aligned}\Delta V &= \sum_{i=1}^{2}\left[\left(e_{mi}^2(k+1) - e_{mi}^2(k)\right) + \Delta\boldsymbol{\theta}_{fi} + \Delta\boldsymbol{\theta}_{gi} + \Delta\boldsymbol{\theta}_{\delta i}\right] = \cdots\\ &= \sum_{i=1}^{2}\left[\left((e_{mi}(k) + \Delta e_{mi}(k))^2 - e_{mi}^2(k)\right) + \Delta\boldsymbol{\theta}_{fi} + \Delta\boldsymbol{\theta}_{gi} + \Delta\boldsymbol{\theta}_{\delta i}\right] = \cdots\\ &= \sum_{i=1}^{2}\left[\left(2e_{mi}(k)\Delta e_{mi}(k) + \Delta e_{mi}^2(k)\right) + \Delta\boldsymbol{\theta}_{fi} + \Delta\boldsymbol{\theta}_{gi} + \Delta\boldsymbol{\theta}_{\delta i}\right]\end{aligned} \tag{24}$$

Next, replacing (13) into (24), ΔV is written as:

$$\Delta V = \sum_{i=1}^{2}\left[\left(2e_{mi}(k)(e_{mi}(k+1)-e_{mi}(k))+\Delta e_{mi}^2(k)\right)+\Delta e_i^2(k)+\Delta\boldsymbol{\theta}_{fi}+\Delta\boldsymbol{\theta}_{gi}+\Delta\boldsymbol{\theta}_{\delta i}\right]\Delta V$$
$$=\sum_{i=1}^{2}\left[2e_{mi}(k)\tilde{\theta}_{gi}^T u_i+2e_i\tilde{\theta}_{fi}^T\xi(\zeta(k))+2e_i\tilde{\theta}_{\delta i}^T\chi(\upsilon(k)) - e_{mi}^2(k)+\Delta e_{mi}^2(k)+\Delta\boldsymbol{\theta}_{fi}+\Delta\boldsymbol{\theta}_{gi}+\Delta\boldsymbol{\theta}_{\delta i}\right] \tag{25}$$

From (23), and rewriting $\Delta\boldsymbol{\theta}_{fi}$, it yields:

$$\Delta\boldsymbol{\theta}_{fi}=\gamma_1^{-1}\left(\tilde{\theta}_{fi}^T(k)\tilde{\theta}_{fi}(k)-\left[\tilde{\theta}_{fi}(k)-\Delta\tilde{\theta}_{fi}(k)\right]^T\left[\tilde{\theta}_{fi}(k)-\Delta\tilde{\theta}_{fi}(k)\right]\right)=\cdots$$
$$=2\gamma_1^{-1}\left(\tilde{\theta}_{fi}^T(k)\Delta\tilde{\theta}_{fi}(k)\right)-\left(\Delta\tilde{\theta}_{fi}^T(k)\Delta\tilde{\theta}_{fi}(k)\right) \tag{26}$$

Making the same analysis for $\Delta\boldsymbol{\theta}_{gi}$ and $\Delta\boldsymbol{\theta}_{\delta i}$, such variables can be expressed by:

$$\Delta\boldsymbol{\theta}_{gi}=2\gamma_2^{-1}\left(\tilde{\theta}_{gi}^T(k)\Delta\tilde{\theta}_{gi}(k)\right)-\left(\Delta\tilde{\theta}_{gi}^T(k)\Delta\tilde{\theta}_{gi}(k)\right) \tag{27}$$

$$\Delta\boldsymbol{\theta}_{\delta i}=2\gamma_3^{-1}\left(\tilde{\theta}_{\delta i}^T(k)\Delta\tilde{\theta}_{\delta i}(k)\right)-\left(\Delta\tilde{\theta}_{\delta i}^T(k)\Delta\tilde{\theta}_{\delta i}(k)\right) \tag{28}$$

Then, (26), (27) and (28) are replaced into (25) where reorganizing terms, we obtain,

$$\Delta V=\sum_{i=1}^{2}\Big[-e_{mi}^2(k)+\Delta e_{mi}^2(k)+2\tilde{\theta}_{fi}^T(k)\left(e_{mi}\xi(\zeta(k))+2\gamma_1^{-1}\Delta\tilde{\theta}_{fi}(k)\right)$$
$$-2\gamma_1^{-1}\left(\Delta\tilde{\theta}_{fi}^T(k)\Delta\tilde{\theta}_{fi}(k)\right)+\cdots+2\tilde{\theta}_{gi}^T(k)\left(e_{mi}u_i(k)+\gamma_2^{-1}\Delta\tilde{\theta}_{gi}(k)\right)$$
$$-2\gamma_2^{-1}\left(\Delta\tilde{\theta}_{gi}^T(k)\Delta\tilde{\theta}_{gi}(k)\right)+\cdots+2\tilde{\theta}_{\delta i}^T(k)\left(e_{mi}\chi(\upsilon(k))+\gamma_3^{-1}\Delta\tilde{\theta}_{\delta i}(k)\right)$$
$$-2\gamma_3^{-1}\left(\Delta\tilde{\theta}_{\delta i}^T(k)\Delta\tilde{\theta}_{\delta i}(k)\right)\Big] \tag{29}$$

Replacing the adjustment laws (18), (19) and (20) with $\Delta\tilde{\theta}_{fi}$, $\Delta\tilde{\theta}_{gi}$ and $\Delta\tilde{\theta}_{\delta i}$, in (29), ΔV is represented by,

$$\Delta V=\sum_{i=1}^{2}\Big[-e_{mi}^2(k)+\Delta e_{mi}^2(k)-2\gamma_1^{-1}\left(\Delta\tilde{\theta}_{fi}^T(k)\Delta\tilde{\theta}_{fi}(k)\right)$$
$$-\cdots-2\gamma_2^{-1}\left(\Delta\tilde{\theta}_{gi}^T(k)\Delta\tilde{\theta}_{gi}(k)\right)-2\gamma_3^{-1}\left(\Delta\tilde{\theta}_{\delta i}^T(k)\Delta\tilde{\theta}_{\delta i}(k)\right)\Big] \tag{30}$$

Most terms of (30) are negative, with the exception of Δe_{mi}.. This increasing of the model error can be approximated by the following equation:

$$\Delta e_{mi}(k)=\left(\frac{\partial e_{mi}(k)}{\partial\hat{\theta}_{fi}(k)}\right)^T\Delta\hat{\theta}_{fi}(k)+\left(\frac{\partial e_{mi}(k)}{\partial\hat{\theta}_{gi}(k)}\right)^T\Delta\hat{\theta}_{gi}(k)+\left(\frac{\partial e_{mi}(k)}{\partial\hat{\theta}_{\delta i}(k)}\right)^T\Delta\hat{\theta}_{\delta i}(k) \tag{31}$$

The partial derivative of the model error depends only on the weights of the neural network and can be rewritten as:

$$\Delta e_{mi}(k)=-\left(\frac{\partial\hat{x}_i(k)}{\partial\hat{\theta}_f(k)}\right)^T\Delta\hat{\theta}_f(k)-\left(\frac{\partial\hat{x}_i(k)}{\partial\hat{\theta}_g(k)}\right)^T\Delta\hat{\theta}_g(k)-\left(\frac{\partial\hat{x}_i(k)}{\partial\hat{\theta}_\delta(k)}\right)^T\Delta\hat{\theta}_\delta(k) \tag{32}$$

Changing the values of the weights according to (18), (19) and (20); Eq. (32) can be written as:

$$\Delta e_{mi}(k)=\left(\frac{\partial\hat{x}_i(k)}{\partial\hat{\theta}_f(k)}\right)^T\gamma_1 e_{mi}\xi(\zeta(k))+\left(\frac{\partial\hat{x}_i(k)}{\partial\hat{\theta}_g(k)}\right)^T\gamma_2 e_{im}u_i(k)+\left(\frac{\partial\hat{x}_i(k)}{\partial\hat{\theta}_\delta(k)}\right)^T\gamma_3 e_{mi}\boldsymbol{\chi}(\upsilon(k)) \tag{33}$$

Considering the value of the partial derivatives of the neural network (12) and replacing in (33), it yields:

$$\Delta e_{mi}(k)=\gamma_1 e_{mi}\boldsymbol{\xi}^T(\boldsymbol{\zeta}(k))\boldsymbol{\xi}(\boldsymbol{\zeta}(k))+\gamma_2 e_{im}u_i^2(k)+\gamma_3 e_{mi}\boldsymbol{\chi}^T(\upsilon(k))\boldsymbol{\chi}(\upsilon(k)) \tag{34}$$

The increase in the model error is defined as:

$$\Delta e_{mi}(k)=\gamma e_{mi} \tag{35}$$

where

$$\gamma=\max\left(\gamma_1\|\boldsymbol{\xi}(\boldsymbol{\zeta}(k))\|^2+\gamma_2|u_i(k)|^2+\gamma_3\|\boldsymbol{\chi}(\upsilon(k))\|^2\right)$$
$$\gamma=\max\left(\gamma_1+\gamma_2|u_{iMax}|^2+\gamma_3\right)<1 \tag{36}$$

The values γ_1, γ_2, γ_3, are learning factors of the neural network ($0<\gamma_{1,2,3}<1$), and they can be arbitrarily defined.

Substituting the increment value of the model error in (30), the Lyapunov discrete difference is defined as:

$$\Delta V=\sum_{i=1}^{2}\Big[-e_{mi}^2(k)(1-\gamma^2)-2\gamma_1^{-1}\left(\Delta\tilde{\theta}_{fi}^T(k)\Delta\tilde{\theta}_{fi}(k)\right)$$
$$-\cdots-2\gamma_2^{-1}\left(\Delta\tilde{\theta}_{gi}^T(k)\Delta\tilde{\theta}_{gi}(k)\right)-2\gamma_3^{-1}\left(\Delta\tilde{\theta}_{\delta i}^T(k)\Delta\tilde{\theta}_{\delta i}(k)\right)\Big]<0 \tag{37}$$

Equation (37) shows that the identification of the robot dynamic model is stable and can be used as retro-propagation error of the neural PID controller, which will be described in the next section.

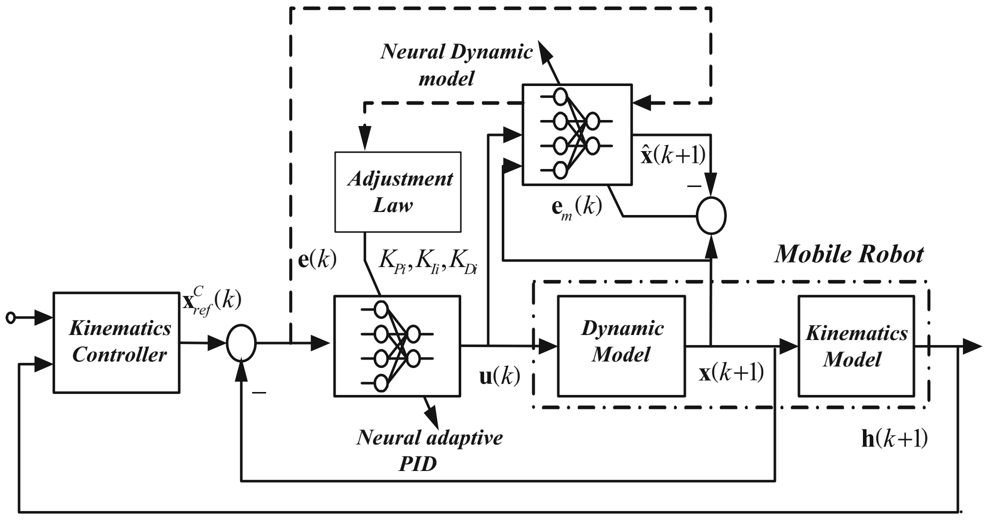


Fig. 2 Control structure of the mobile robot

5 Dynamic neural PID controller

The dynamic neural PID controller receives references of linear and angular velocities computed by a kinematic controller and gives as result two output velocities (linear and angular) that are sent to the robot servos, as it is shown in Fig. 2. Now, the error vector of output velocities is defined as:

$$\mathbf{e}(k) = (e_1(k), e_2(k))^T \tag{38}$$

where

$$e_1(k) = x^c_{1\,\text{ref}}(k) - x_1(k) \quad e_2(k) = x^c_{2\,\text{ref}}(k) - x_2(k)$$

The model of the classical PID controller in discrete time is presented by:

$$o_i(k) = K_{Pi}e_i(k) + K_{Ii}\sum_{j=1}^{k} e_i(j)T_0 + \frac{K_{Di}}{T_0}(e_i(k) - e_i(k-1)) \tag{39}$$

where the subscript i indicates reference of linear velocity ($i = 1$) or angular velocity ($i = 2$) of the mobile robot.

Considering the control action for the time instant ($k - 1$), it yields:

$$o_i(k-1) = K_{Pi}e_i(k-1) + K_{Ii}\sum_{j=1}^{k-1} e_i(j)T_0 + \frac{K_{Di}}{T_0}(e_i(k-1) - e_i(k-2)) \tag{40}$$

Next, making the difference between $o(k)$ and $o(k - 1)$ from (39) and (40), respectively, the signal $o(k)$ can be described by:

$$o_i(k) = o_i(k-1) + K_{Pi}[e_i(k) - e_i(k-1)] + K_{Ii}e_iT_0 + \frac{K_{Di}}{T_0}(e_i(k) - 2e_i(k-1) + e_i(k-2)) \tag{41}$$

Thus, the PID controller can be obtained as a function of the time instant ($k - 1$). Our goal is to get a neural network based on the self-tuning method of a PID control system, where the velocity outputs (linear and angular) can follow the references generated by a kinematic controller.

Generally a modification of the PID control parameters is made to get a more accurate control. But the classical PID controller is limited since the parameters K_{Di}, K_{Ii} *and* K_{Pi} are constant. This means that the control output is neither adaptable nor optimal. Then, an intelligent adaptive control should be used in order to adapt their control parameters for minimizing the position error. Considering the ability to learn and adaptation of a neural network, the proposed controller can solve these problems. The PID parameters can be tuned and adaptively chosen to minimize the velocity error with respect to external perturbations.

To represent a neural network with the PID controller an activation function in the output of the network must be added. In this work, a function tan$h(\cdot)$ that represents a continuous saturation, presented in (42), is applied. Therefore, the controller output is lower than the permitted maximum control action, according to the consideration 2 of the previous section:

$$\delta_i = u_{i\text{Max}} \tag{42}$$

In addition, saturation in every action of the PID controller is applied as well as the activation function of each neuron while ensuring stability, as it will be proven in Sect. 6.

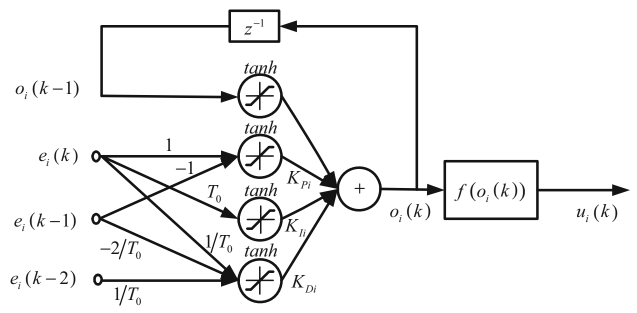


Fig. 3 Architecture of the neural PID network

Definition 1: The activation function $f(\cdot)$ is defined by:

$$f(o_i) = \delta_i \tan h\left(\frac{o_i}{\delta_i}\right) \tag{43}$$

where the controller defined in (41) is expressed in the following way:

$$\begin{aligned} o_i(k) = & f(o_i(k-1)) + K_{Pi} f(e_i(k) - e_i(k-1)) \\ & + K_{Ii} f(e_i T_0) \\ & + K_{Di} f\left(\frac{e_i(k) - 2e_i(k-1) + e_i(k-2)}{T_0}\right) \end{aligned} \tag{44}$$

Figure 3 shows the architecture of the neural PID network.

To ensure the convergence and performance of the optimization function, an index which is a function of velocity tracking errors must be defined. Such index is defined by,

$$E(k) = \frac{1}{2}\left(\mathbf{e}^T(k)\mathbf{e}(k)\right) = \frac{1}{2}\sum_{i=1}^{2} e_i^2(k) \tag{45}$$

The parameters K_{Di}, K_{Ii} *and* K_{Pi} for the neural PID controller can be adjusted using the gradient descent method, similar to the weights of a perceptron-type neural network. The rules to update the controller parameters are:

$$\begin{aligned} \mathbf{K}_{\theta i}(k+1) = \mathbf{K}_{\theta i}(k) + \Delta \mathbf{K}_{\theta i}(k) = & \begin{pmatrix} K_{Pi}(k+1) \\ K_{Ii}(k+1) \\ K_{Di}(k+1) \end{pmatrix} \\ = \begin{pmatrix} K_{Pi}(k) \\ K_{Ii}(k) \\ K_{Di}(k) \end{pmatrix} - & \begin{pmatrix} \eta_{Pi} \dfrac{\partial E(k)}{\partial K_{Pi}} \\ \eta_{Ii} \dfrac{\partial E(k)}{\partial K_{Ii}} \\ \eta_{Di} \dfrac{\partial E(k)}{\partial K_{Di}} \end{pmatrix} \end{aligned} \tag{46}$$

where the values of $\eta_{Pi,Ii,Di}$ are learning factors for the neural PID controller. From (46) using the chain rule, the following equations are obtained:

$$\begin{aligned} \frac{\partial E(k)}{\partial K_{Pi}} &= \frac{\partial E(k)}{\partial x_i(k)} \frac{\partial x_i(k)}{\partial u_i(k)} \frac{\partial u_i(k)}{\partial o_i(k)} \frac{\partial o_i(k)}{\partial K_{Pi}} \\ \frac{\partial E(k)}{\partial K_{Ii}} &= \frac{\partial E(k)}{\partial x_i(k)} \frac{\partial x_i(k)}{\partial u_i(k)} \frac{\partial u_i(k)}{\partial o_i(k)} \frac{\partial o_i(k)}{\partial K_{Ii}} \\ \frac{\partial E(k)}{\partial K_{Di}} &= \frac{\partial E(k)}{\partial x_i(k)} \frac{\partial x_i(k)}{\partial u_i(k)} \frac{\partial u_i(k)}{\partial o_i(k)} \frac{\partial o_i(k)}{\partial K_{Di}} \end{aligned} \tag{47}$$

The values of the partial derivatives of the chain rule are:

$$\frac{\partial E(k)}{\partial x_i(k)} = -e_i(k) \tag{48}$$

For $\partial x_i/\partial u_i$ the dynamics is unknown and it must be approximated by the dynamics identified by the neural network from Eq. (12) obtaining $\partial \hat{x}_i/\partial u_i$

$$\begin{aligned} \frac{\partial \hat{x}_i}{\partial u_i(k)} = & \left(\hat{\theta}_{fi}(k)\xi(\zeta(k))\left[(\zeta(k) - \mathbf{c}_i)\frac{\partial \zeta^T(k)}{\partial u_i(k)}\right]\right. \\ & \left. + \hat{\theta}_{gi}\frac{\partial u_i}{\partial u_i(k)} + \hat{\theta}_{\delta i}(k)\chi(\upsilon(k))\left[(\upsilon(k) - \mathbf{c}_i)\frac{\partial \upsilon^T(k)}{\partial u_i(k)}\right]\right) \end{aligned} \tag{49}$$

From (16) and (17) where $\partial \zeta_i^T/\partial u_i$ and $\partial \upsilon_i^T/\partial u_i$ are $[0, 0, 0, 1]^T$, respectively.

The relation $\partial u_i/\partial o_i$ is the derivative of the activation function $\tan h(\cdot)$. This partial derivative only is valid inside the linear zone of the function, where the control action does not produce levels that could cause damage or breakage of the robot dynamics. Such operation is computed by:

$$\frac{\partial u_i(k)}{\partial o_i(k)} = \frac{\partial f(o_i(k))}{\partial o_i(k)} = \left[1 - f^2(o_i(k))\right] \tag{50}$$

Now considering the partial derivatives with respect to the controller gains, defined as:

$$\begin{cases} \dfrac{\partial o_i(k)}{\partial K_{Pi}} = f(e_i(k) - e_i(k-1)) \\ \dfrac{\partial o_i(k)}{\partial K_{Ii}} = f(e_i(k)T_0) \\ \dfrac{\partial o_i(k)}{\partial K_{Di}} = f\left(\dfrac{e_i(k) - 2e_i(k-1) + e_i(k-2)}{T_0}\right) \end{cases} \tag{51}$$

Next, the expressions obtained from (46) are described as follows,

$$\begin{aligned} K_{Pi}(k+1) &= K_{Pi}(k) - \eta_{Pi} e_i(k)\left(\frac{\partial \hat{x}_i}{\partial u_i(k)}\right) f'(o_i(k)).[f(e_i(k) - e_i(k-1))] \\ K_{Ii}(k+1) &= K_{Ii}(k) - \eta_{Ii} e_i(k)\left(\frac{\partial \hat{x}_i}{\partial u_i(k)}\right) f'(o_i(k)).[f(e_i(k)T_0)] \\ K_{Di}(k+1) &= K_{Di}(k) - \eta_{Di} e_i(k)\left(\frac{\partial \hat{x}_i}{\partial u_i(k)}\right) f'(o_i(k)) \\ & \quad .\left[f\left(\frac{e_i(k) - 2e_i(k-1) + e_i(k-2)}{T_0}\right)\right] \end{aligned} \tag{52}$$

The analysis of this control technique is shown in the next section.

6 Stability analysis of the neural PID controller

First, a positive definite function for discrete time is defined,

$$V(k)=\frac{1}{2}\sum_{i=1}^{2}e_i^2(k) \tag{53}$$

The discrete difference of the candidate function is computed by,

$$\begin{aligned}\Delta V(k)&=V(k+1)-V(k)=\frac{1}{2}\sum_{i=1}^{2}\left[e_i^2(k+1)-e_i^2(k)\right]\\&=\Delta V_1(k)+\Delta V_2(k)=\cdots=\frac{1}{2}\left[e_1^2(k+1)-e_1^2(k)\right]\\&\quad+\frac{1}{2}\left[e_2^2(k+1)-e_2^2(k)\right]\end{aligned} \tag{54}$$

The value of $e(k+1)$ can be calculated by the following equation:

$$e_i(k+1)=e_i(k)+\Delta e_i(k) \tag{55}$$

where the difference of the error $\Delta e(k)$ in the learning process, is expressed by:

$$\Delta e_i(k)=\frac{\partial e_i(k)}{\partial x_i(k)}\frac{\partial x_i(k)}{\partial \mathbf{K}_{\theta i}}\Delta\mathbf{K}_{\theta i}^T \tag{56}$$

Considering (56), being $\Delta K_{\theta i}=[\Delta K_{Pi},\Delta K_{Di},\Delta K_{Ii}]$ the matrix parameters PID neural networks for the output variable x_i (linear or angular). Now, considering (55) and (56), it yields

$$\begin{aligned}\Delta V_i(k)&=e_i(k)\Delta e_i(k)+\frac{1}{2}\Delta e_i^2(k)\\&=e_i(k)\frac{\partial e_i(k)}{\partial x_i(k)}\frac{\partial x_i(k)}{\partial \mathbf{K}_{\theta i}}\Delta\mathbf{K}_{\theta i}^T\\&\quad+\frac{1}{2}\left(\frac{\partial e_i(k)}{\partial x_i(k)}\frac{\partial x_i(k)}{\partial \mathbf{K}_{\theta i}}\Delta\mathbf{K}_{\theta i}^T\right)^2\end{aligned} \tag{57}$$

Replacing the value of $\Delta K_{\theta i}$ in (57), we have:

$$\begin{aligned}\Delta V_i(k)&=e_i(k)\frac{\partial e_i(k)}{\partial x_i(k)}\frac{\partial x_i(k)}{\partial \mathbf{K}_{\theta i}}\eta_{\theta i}e_i(k)\left(\frac{\partial e_i(k)}{\partial \mathbf{K}_{\theta i}}\right)^T\\&\quad+\frac{1}{2}\left(\frac{\partial e_i(k)}{\partial x_i(k)}\frac{\partial x_i(k)}{\partial \mathbf{K}_{\theta i}}\eta_{\theta i}e_i(k)\left(\frac{\partial e_i(k)}{\partial \mathbf{K}_{\theta i}}\right)^T\right)^2\end{aligned} \tag{58}$$

Re-arranging (58), it yields:

$$\begin{aligned}\Delta V_i(k)&=-\eta_{\theta i}e_i^2(k)\left(\frac{\partial e_i(k)}{\partial x_i(k)}\right)^2\frac{\partial x_i(k)}{\partial \mathbf{K}_{\theta i}}\left(\frac{\partial x_i(k)}{\partial \mathbf{K}_{\theta i}}\right)^T\\&\quad+\frac{1}{2}\eta_{\theta i}^2e_i^2(k)\left(\frac{\partial e_i(k)}{\partial x_i(k)}\right)^4\left(\frac{\partial x_i(k)}{\partial \mathbf{K}_{\theta i}}\left(\frac{\partial x_i(k)}{\partial \mathbf{K}_{\theta i}}\right)^T\right)^4\end{aligned} \tag{59}$$

If (59) is analyzed to assure $\Delta V_i(k)$ lower than zero, we get,

$$\begin{aligned}\Delta V_i(k)&=\left[-1+\frac{1}{2}\eta_{\theta i}\left(\frac{\partial e_i(k)}{\partial x_i(k)}\right)^2\left\|\frac{\partial x_i(k)}{\partial \mathbf{K}_{\theta i}}\right\|^2\right]\\&\quad\times\eta_{\theta i}e_i^2(k)\left(\frac{\partial e_i(k)}{\partial x_i(k)}\right)^2\left\|\frac{\partial x_i(k)}{\partial \mathbf{K}_{\theta i}}\right\|^2\le 0\end{aligned} \tag{60}$$

From (60), the following condition must be satisfied:

$$\frac{1}{2}|\eta_{\theta i}|\left|\left(\frac{\partial e_i(k)}{\partial x_i(k)}\right)\right|^2\left\|\frac{\partial x_i(k)}{\partial \mathbf{K}_{\theta i}}\right\|^2<1 \tag{61}$$

where $\eta_{Pi,Ii,Di}<1$ and

$$\begin{cases}\left(\dfrac{\partial e_i(k)}{\partial x_i(k)}\right)^2=1\\ \left\|\dfrac{\partial x_i(k)}{\partial \mathbf{K}_{\theta i}}\right\|^2=\left|\dfrac{\partial x_i(k)}{\partial u_i}\right|^2\left|\dfrac{\partial u_i(k)}{\partial o_i(k)}\right|^2\left\|\dfrac{\partial o_i(k)}{\partial \mathbf{K}_{\theta i}}\right\|^2\end{cases} \tag{62}$$

Being

$$\begin{cases}\left|\dfrac{\partial x_i(k)}{\partial u_i}\right|^2=\left|\dfrac{\partial \hat{x}_i}{\partial u_i(k)}\right|^2\le|\delta_i|^2\\ \left|\dfrac{\partial u_i(k)}{\partial o_i(k)}\right|^2=|f'(o_i(k))|^2\le 1\\ \left\|\dfrac{\partial o_i(k)}{\partial \mathbf{K}_{\theta i}}\right\|^2=(f(e_i(k)-e_i(k-1)))^2+(f(e_i(k)T_0))^2\\ +\left(f\left(\dfrac{e_i(k)-2e_i(k-1)+e_i(k-2)}{T_0}\right)\right)^2\le 3|\delta_i|^2\end{cases} \tag{63}$$

According to (62) and (63),

$$\begin{aligned}&\eta_{\theta i}3|\delta_{i\mathrm{Max}}|^4<1\\&\frac{1}{3|\delta_{i\mathrm{Max}}|^4}>\eta_{\theta i}>0\end{aligned} \tag{64}$$

Now, the error obtained by the difference between the trajectory followed by the robot and the desired trajectory $\tilde{h}(k)$ is considered. If a perfect speed tracking is not considered (see Sect. 3), (7) can be written as,

$$\begin{aligned}&\begin{pmatrix}\tilde{r}_x(k+1)\\ \tilde{r}_y(k+1)\end{pmatrix}-\begin{pmatrix}l_x&0\\0&l_y\end{pmatrix}\begin{pmatrix}\tanh\left(\frac{k_x}{l_x}\tilde{r}_x(k)\right)\\ \tanh\left(\frac{k_y}{l_y}\tilde{r}_y(k)\right)\end{pmatrix}\\&=\begin{pmatrix}\varepsilon_1(k)\\ \varepsilon_2(k)\end{pmatrix}\end{aligned} \tag{65}$$

where the error vector $\boldsymbol{\varepsilon}(k) = [\varepsilon_1(k) \quad \varepsilon_2(k)]^T$ depends on the tracking error of velocity defined as $\mathbf{He}(k)$, being:

$$\boldsymbol{\varepsilon}(k) = \begin{pmatrix} \varepsilon_1(k) \\ \varepsilon_2(k) \end{pmatrix} = T_0 \begin{pmatrix} \cos\psi(k) & -a\sin\psi(k) \\ \sin\psi(k) & a\cos\psi(k) \end{pmatrix} \begin{pmatrix} e_1(k) \\ e_2(k) \end{pmatrix} = \mathbf{He}(k) \tag{66}$$

Rewriting (66), for small control errors $L(\tilde{\mathbf{h}}(k)) \approx \mathbf{K}_{xy}\tilde{\mathbf{h}}(k)$, being $\mathbf{K}_{xy} = \mathrm{diag}(k_x, k_y)$.

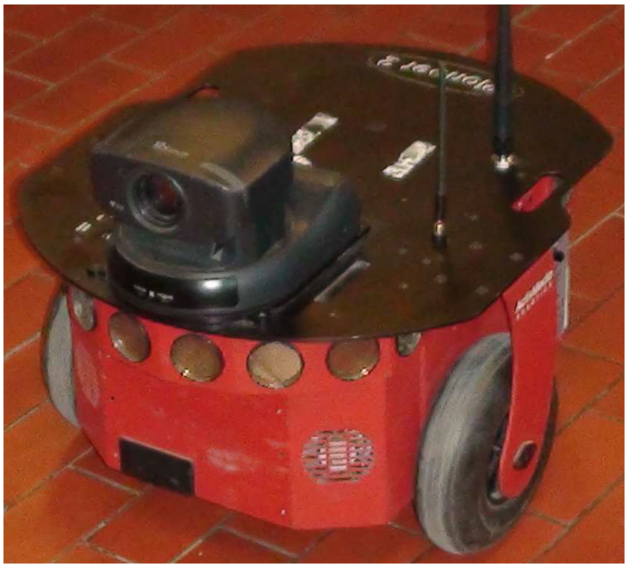


Fig. 4 Pioneer 2DX mobile robot

$$\tilde{\mathbf{h}}(k+1) - \mathbf{K}_{xy}\tilde{\mathbf{h}}(k) = \boldsymbol{\varepsilon}(k) \tag{67}$$

Considering the Lyapunov candidate function:

$$V(k) = \frac{1}{2}\tilde{\mathbf{h}}^T(k)\tilde{\mathbf{h}}(k) > 0 \tag{68}$$

The discrete difference is established by:

$$\begin{aligned} \Delta V(k) &= V(k+1) - V(k) \\ &= \tilde{\mathbf{h}}^T(k+1)\tilde{\mathbf{h}}(k+1) - \tilde{\mathbf{h}}^T(k)\tilde{\mathbf{h}}(k) = \cdots \\ &= \mathbf{e}^T(k)\mathbf{H}^T\mathbf{He}(k) - 2\mathbf{e}^T(k)\mathbf{H}^T\mathbf{K}_{xy}\tilde{\mathbf{h}}(k) \\ &\quad + \tilde{\mathbf{h}}^T(k)\mathbf{K}_{xy}^T\mathbf{K}_{xy}\tilde{\mathbf{h}}(k) - \tilde{\mathbf{h}}^T(k)\tilde{\mathbf{h}}(k) \end{aligned} \tag{69}$$

The sufficient condition for $\Delta V \le 0$ is the following one:

$$\left\|\mathbf{I} - \mathbf{K}_{xy}^2\right\| \|\tilde{\mathbf{h}}(k)\|^2 \ge \|\mathbf{H}\|^2\|\mathbf{e}(k)\|^2 + 2\|\mathbf{H}\|\left\|\mathbf{K}_{xy}\right\|\|\mathbf{e}(k)\|\|\tilde{\mathbf{h}}(k)\| \tag{70}$$

Calculating the roots of (70)

$$\begin{aligned} \|\tilde{\mathbf{h}}\| &> \frac{-2\|\mathbf{H}\|\left\|\mathbf{K}_{xy}\right\|\|\mathbf{e}(k)\|}{2\left\|\mathbf{I} - \mathbf{K}_{xy}^2\right\|} \\ &\pm \frac{\sqrt{4\|\mathbf{H}\|^2\left\|\mathbf{K}_{xy}\right\|^2\|\mathbf{e}(k)\|^2 + 4\left\|\mathbf{I} - \mathbf{K}_{xy}^2\right\|\|\mathbf{e}(k)\|^2\|\mathbf{H}\|^2}}{2\left\|\mathbf{I} - \mathbf{K}_{xy}^2\right\|} \\ &= \cdots = \frac{-2\|\mathbf{H}\|\left\|\mathbf{K}_{xy}\right\|\|\mathbf{e}(k)\| \pm 2\|\mathbf{H}\|\|\mathbf{e}(k)\|}{2\left\|\mathbf{I} - \mathbf{K}_{xy}^2\right\|} \end{aligned} \tag{71}$$

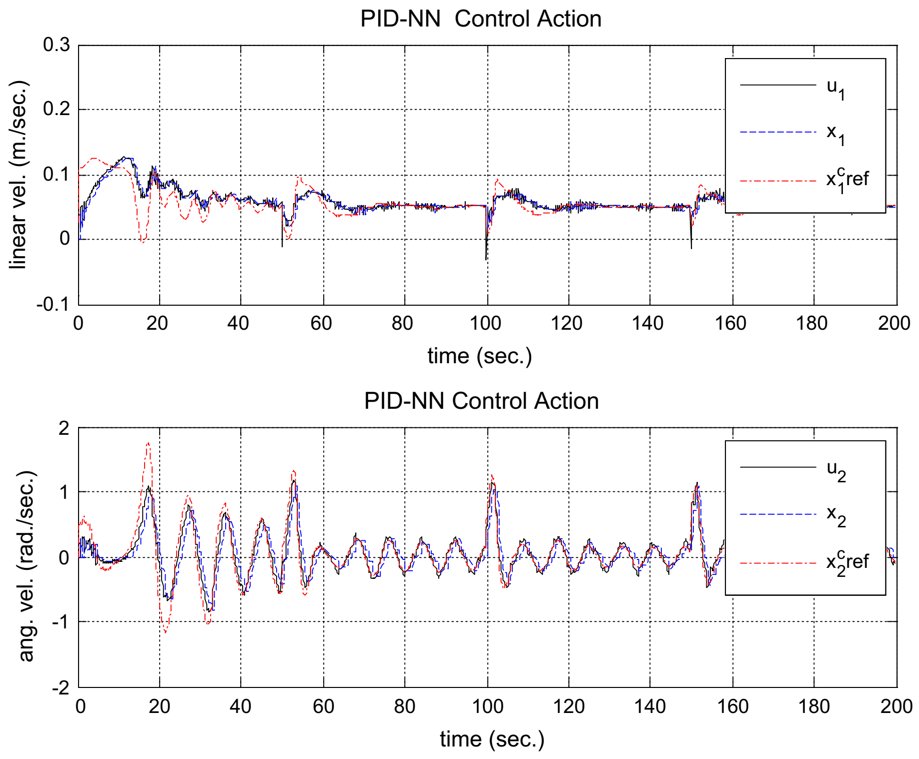


Fig. 5 Control actions, outputs of the mobile robot and kinematics references

A sufficient condition for asymptotic stability reaching $\Delta V \leq 0$ can be expressed as,

$$\|\tilde{\mathbf{h}}\| \geq \left(\left(1 - \|\mathbf{K}_{xy}\|\right)\|\mathbf{H}\|\,\|\mathbf{e}(k)\|\right) \Big/ \left\|\mathbf{I} - \mathbf{K}_{xy}^2\right\| \qquad (72)$$

This implies that $\Delta V \leq 0$, which assures that $\tilde{h} \to B_\delta$.

Besides, the result obtained allows affirming that the position error is bounded ultimately depending on the approximation error of the neural PID controller.

Making the analysis for big errors, we have,

$$L(\tilde{\mathbf{h}}(k)) = \begin{pmatrix} l_x \tanh\left(\frac{k_x}{l_x}\tilde{r}_x(k)\right) \\ l_y \tanh\left(\frac{k_y}{l_y}\tilde{r}_y(k)\right) \end{pmatrix} = \begin{pmatrix} l_x \\ l_y \end{pmatrix} \qquad (73)$$

Now considering (73) and (69), the discrete difference can be expressed as:

$$\begin{aligned}\Delta V(k) &= V(k+1) - V(k) \\ &= \tilde{\mathbf{h}}^T(k+1)\tilde{\mathbf{h}}(k+1) - \tilde{\mathbf{h}}^T(k)\tilde{\mathbf{h}}(k) = \cdots \\ &= e^T(k)\mathbf{H}^T\mathbf{H}\mathbf{e}(k) - 2\mathbf{e}^T(k)\mathbf{H}^T\mathbf{L} + \mathbf{L}^T\mathbf{L} \\ &\quad - \tilde{\mathbf{h}}^T(k)\tilde{\mathbf{h}}(k) \end{aligned} \qquad (74)$$

The sufficient condition for $\Delta V \leq 0$ is

$$\|\tilde{\mathbf{h}}(k)\|^2 \geq \|\mathbf{H}\|^2\|\mathbf{e}(k)\|^2 + 2\|\mathbf{H}\|\|\mathbf{L}\|\|\mathbf{e}(k)\| + \|\mathbf{L}\|^2 \qquad (75)$$

From (75), the following condition is obtained:

$$\|\tilde{\mathbf{h}}(k)\| \geq \|\mathbf{H}\|\|\mathbf{e}(k)\| + \|\mathbf{L}\| \qquad (76)$$

Last equation determines the size of the error vector about the position control $\tilde{\mathbf{h}}(k)$. The errors of (72) and (76) allow establishing that the errors are bounded by the approximation error of the adaptable neural PID controller.

7 Experimental results

The Pioneer 2DX mobile robot has a PC Pentium III on board, running at a frequency of 800 MHz with 512 Mb of

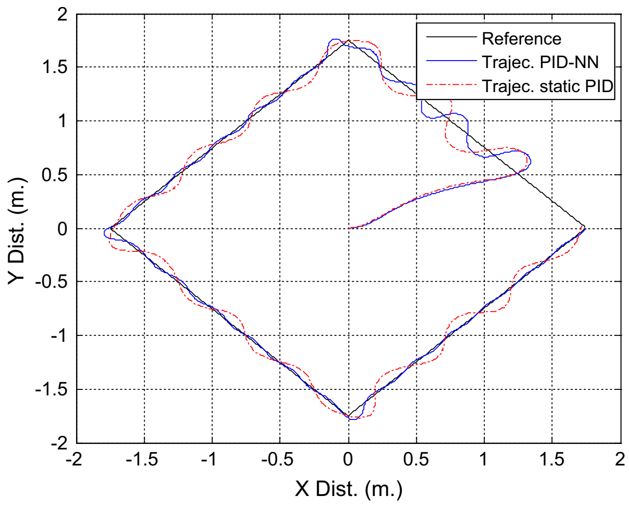


Fig. 7 Trajectory followed by the mobile robot including a PID-NN controller (*solid line*) compared with the trajectory made by the robot including a conventional static PID controller (*dashed line*)

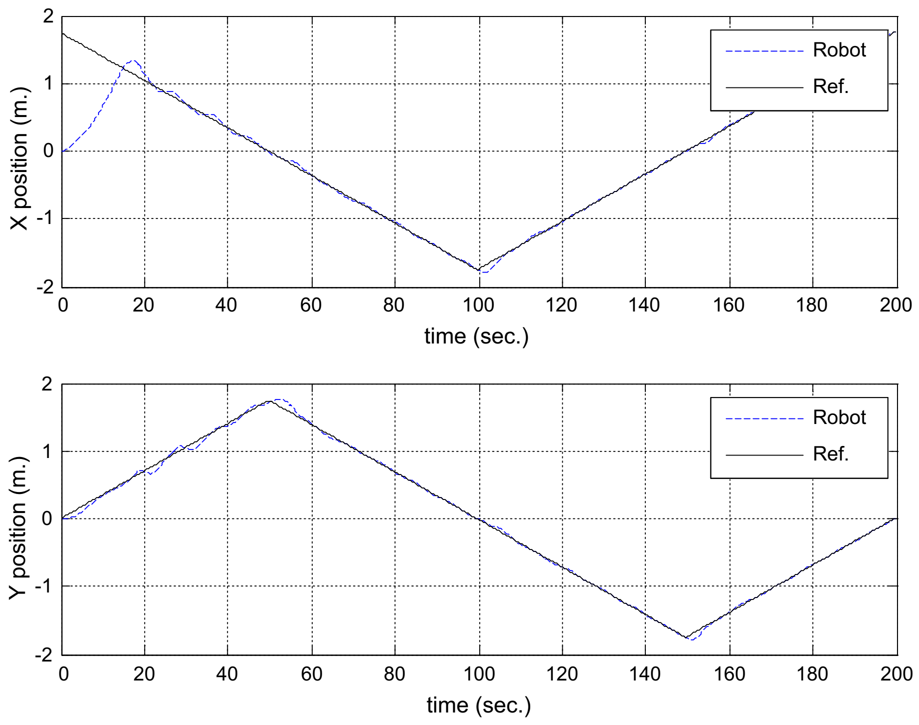


Fig. 6 Reference signals of position and mobile robot position

RAM memory (Fig. 4). The proposed control algorithm is applied to the robot that accepts linear and angular velocities as the reference input signal.

For the experiment, the PID-NN controller was started up with gains obtained from previous experiments using other trajectories. In the experiment shown in this paper, the gain parameters are adjusted on-line by the learning algorithm, evaluating its evolution as a function of time.

During the experiment, the robot should follow a square path of 1.75 m per side. Figure 5 shows the control actions applied to the robot. In the first moments, it is possible to appreciate variations since the gain parameters of the PID controller are being adjusted.

In Fig. 6, the r_x and r_y references, and the path followed by the mobile robot are shown.

Figure 7 shows the trajectory followed by the mobile robot on the X–Y plane. At the beginning of the trajectory, the error is greater due to the lack of adjustment of the PID controller parameters, but it is equal to the error of the classical PID controller without auto-tuning. Throughout the experiment, the trajectory followed by the robot is very close to the reference trajectory. Figure 8 shows the evolution of the tracking error of the trajectory. In the first moments, the error is similar to the one caused by a conventional classical PID controller, but during the next few seconds the error decreases significantly with respect to the classical PID controller, even tending the error to zero. In addition, the evolution of the tuning gains of the PID-NN controller is shown in Fig. 9. In last figure, a comparison of the output velocities between the network identified by the dynamic neural model and the real dynamic of the mobile robot is made.

The adaptable neural PID controller is robust with respect to the modeling errors. In addition, it is effective in the rejection of disturbances without producing any constant error caused by the parameter uncertainties or external disturbances. On the other hand, the classical PID controller is vulnerable to changes in the dynamics and uncertainties of the model, since the classic PID controller

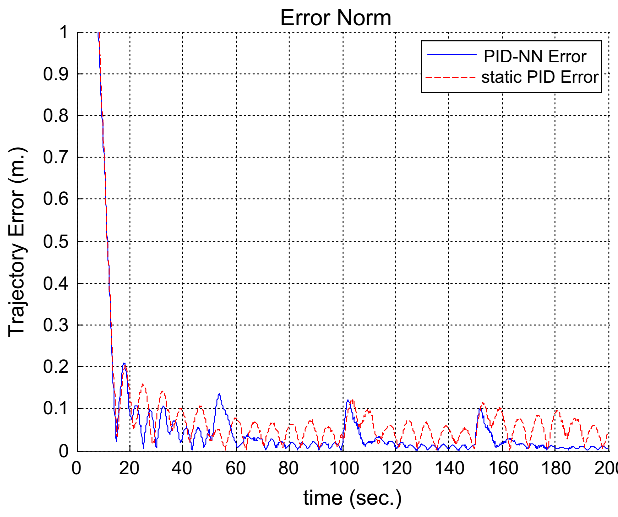


Fig. 8 Instantaneous quadratic error of the robot position. Robot with PID-NN (*solid line*) and robot with classical PID (*dashed line*)

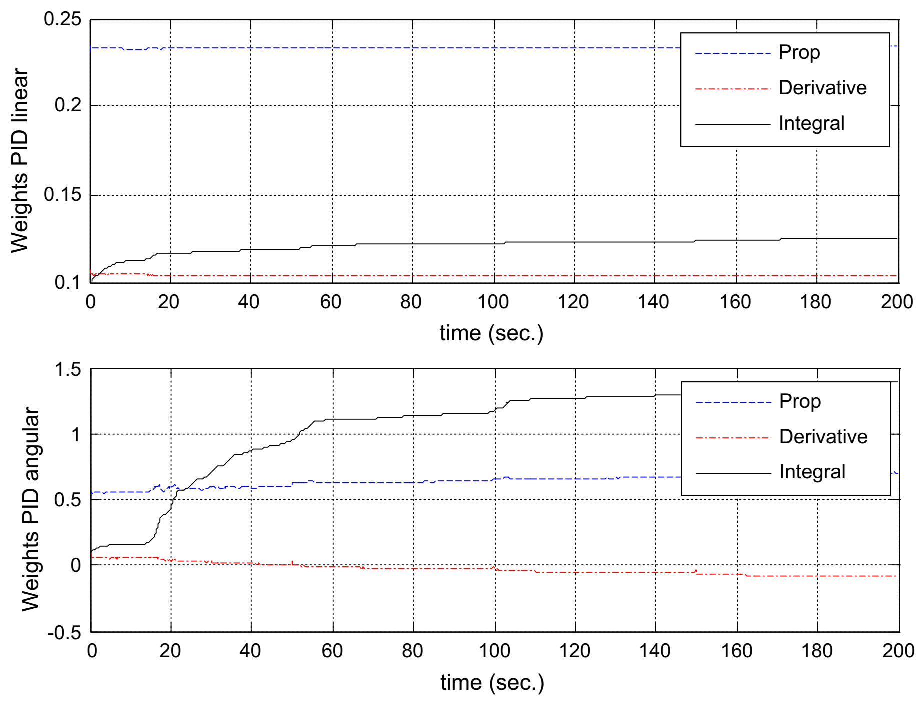


Fig. 9 Evolution of the gain parameters of the neural PID network during the experiment

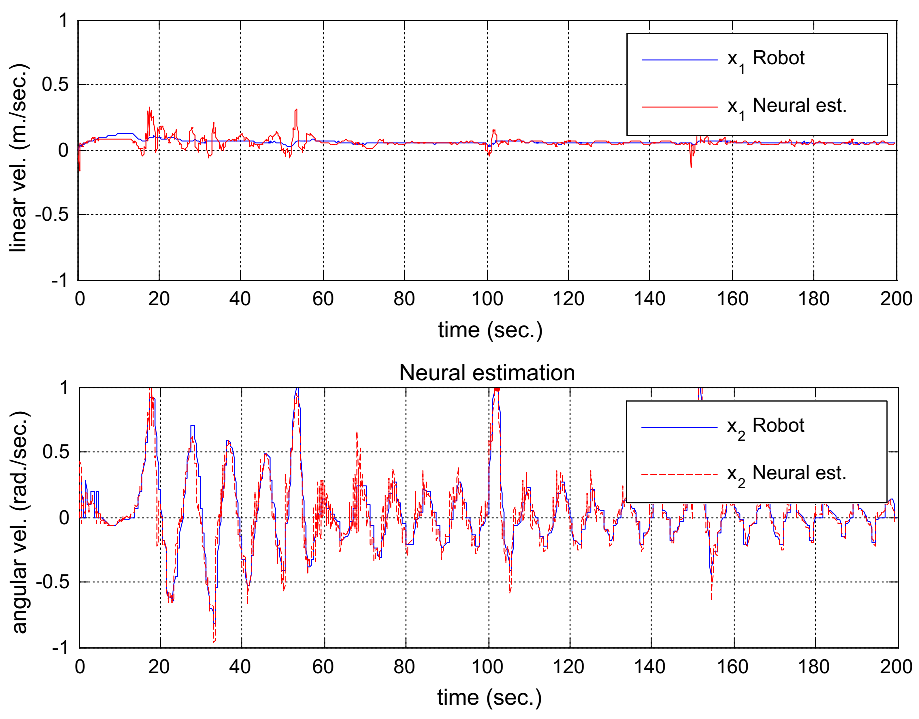


Fig. 10 Behavior of the dynamic identified model of the mobile robot compared with the real dynamics of the robot

is designed for a linearized dynamic model of the robot. The control law developed in this paper for a nonlinear system does not need the dynamic model of the robot. It is well known that the mathematical representation of a dynamic model is not concerned with perfect accuracy, but the response of the adaptive neural PID controller in front of these uncertainties and nonlinearities of the model as well as possible variations in the robot dynamics, increases the system robustness. The overall stability of the closed-loop system was demonstrated analytically through the Lyapunov stability theory (Sect. 6). This control approach can be considered as a general solution for the control of nonlinear systems, and in particular for robotic systems, when the dynamics is variable or there are uncertainties in the model (Fig. 10).

This area of research is very large, and the issues of modeling, stability, convergence and robustness analysis of learning systems remain open to further investigation to design controllers increasingly accurate.

8 Conclusions

In this paper, an adaptive neural PID controller for trajectory tracking of a unicycle-type mobile robot has been proposed. The controller generates velocity commands with minimal error for the nonlinear dynamics of the robot. In addition, it is important to remark that the nonlinear dynamics does not need to be known a priori, but it must be approximated (or identified), feeding back the errors in order to set the controller gain parameters. It was proven that the tracking errors are bounded and that their limits are calculated depending on the approximation error of the neural PID controller.

The adaptive neural PID controller can adjust their parameters to reduce the error caused by differences in the dynamics of the robot.

Experimental results have shown a good performance of the proposed neural PID controller and its adaptation to the dynamics of real robots. Besides, the proposed controller can be applied to any process with nonlinear dynamics.

References

1. Normey-Rico EJ, Alcalá I, Gómez-Ortega J, Camacho EF (2001) Mobile robot path tracking using a robust PID controller. Control Eng Pract 9(11):1209–1214
2. Padhy PK, Sasaki T, Nakamura S, Hashimoto H (2010) Modelling and position control of mobile robot. Advanced motion control, 2010 11th IEEE international workshop on, pp 100–105, March (2010)
3. Esfandyari M, Fanaei MA, Zohreie H (2013) Adaptive fuzzy tuning of PID controllers. J Neural Comput Appl 23(1):19–28
4. Jin J, Su Y (2005) Improved adaptive genetic algorithm. Comput Eng Appl 29(3):64–70
5. Precup R-E, Preitl S, Faur G (2003) PI predictive fuzzy controllers for electrical drive speed control: methods and software for stable development. Comput Ind 52:253–270
6. Ding Y, Ying H, Shao S (2003) Typical Takagi–Sugeno PI and PD fuzzy controllers: analytical structures and stability analysis. Inf Sci 151:245–262

7. Ahn KK, Truong DQ (2009) Online tuning fuzzy PID controller using robust extended Kalman filter. J Process Control 19:1011–1023
8. Lu W, Yang JH, Liu XD (2012) The PID controller based on the artificial neural network and the differential evolution algorithm. J Comput 7(10):2368–2375
9. Mahmud K (2013) Neural network based PID control analysis. In: Global high tech congress on electronics (GHTCE), 2013 IEEE, pp 141, 145, 17–19 Nov 2013
10. Gan SC, Yang PX (2005) PID self-tuning based on fuzzy genetic algorithm. J N China Electric Power Univ 32(5):43–46
11. Mahony TO, Downing CJ, Fatla K (2000) Genetic algorithm for PID parameter optimization: minimizing error criteria. In: Process control and instrumentation, 26–28 July 2000, pp 148–153. University of Stracthclyde (2000)
12. He G, Tan G (2007) An optimal nonlinear PID controller based on ant algorithm. Programmable controller & factory automation, pp 99–105
13. Kennedy J, Eberhart RC (2001) Swarm intelligence. Morgan Kaufmann Publishers, Los Altos
14. Gaing ZL (2004) A particle swarm optimization approach for optimum design of PID Controller in AVR system. IEEE Trans Energy Convers 19(2):384–391
15. Shi Y, Eberhart R (1998) A modified particle swarm optimizer. In: Proceedings IEEE international conference evolution computer, Anchorage, AK, pp 69–73
16. Rossomando FG, Soria C, Carelli R (2013) Adaptive neural sliding mode compensator for a class of nonlinear systems with unmodeled uncertainties. Eng Appl Artif Intell 26(10):2251–2259
17. Li Y, Wang Z, Zhu L (2010) Adaptive neural network PID sliding mode dynamic control of nonholonomic mobile robot. In: IEEE international conference on information and automation (ICIA), pp 753–757
18. Carelli, R. and De La Cruz C. Dynamic Modeling and Centralized Formation Control of Mobile Robots. In: 32nd Annual conference of the IEEE industrial electronics society IECON, Paris (2006)