Control Engineering Practice 69 (2017) 73–84

Contents lists available at ScienceDirect

Control Engineering Practice

journal homepage: www.elsevier.com/locate/conengprac

Self-tuning PID controllers in pursuit of plug and play capacity

Jérôme Mendes [*], Luís Osório, Rui Araújo

Institute of Systems and Robotics (ISR-UC), and Department of Electrical and Computer Engineering (DEEC-UC), University of Coimbra, Pólo II, PT-3030-290 Coimbra, Portugal

ARTICLE INFO

Keywords:
Adaptive PID
Control
Self-Tuning Controller (STC)
PID comparison
Coupled DC motors

ABSTRACT

This work addresses the problem of controlling unknown and time varying plants for industrial applications. The concept of "plug-and-play" was pursued using control algorithms that auto-adapt their control parameters in order to control unknown and time-varying plants. Self Tuning Controllers (STC) with PID form were studied and tested on a real process setup. The setup is composed of two coupled DC motors and a variable load. Controllers' performances were compared in order to distinguish which controllers perform better, which are easier to set up, which have a better initial response, and which enable faster reaction to plant variations and load disturbances.

1. Introduction

The classic Proportional Integral and Derivative (PID) controller is one of the most popular controllers in industry having applicability, and according Åström and Hägglund (2001) is employed in about 90% or more of the control loops. In industry, even nowadays, a common procedure is the tunning of the PID parameters in order to match with the operating conditions and these tunned parameters are kept fixed for the whole operation (Åström & Hägglund, 2001; Ziegler & Nichols, 1942). However, most of the industrial processes have dynamics that are not modeled or can be changed over time. In these cases, PID parameters may require adequate retuning in order to reach a robust control performance. To overcome this problem adaptive methodologies have been proposed. While a classic controller uses the principle of feedback to compensate the error in the same way in all situations, an adaptive controller uses the feedback to change the control law of the process (adjusting the controller factors over time to cope with the plant's variations) and only then tries to compensate the error. This ability to adapt allows to reach a larger number of real applications with a high quality control (Yildiz, Annaswamy, Yanakiev, & Kolmanovsky, 2010).

According Bobál, Böhm, Fessl, and Macháček (2005), the first applications of adaptive control methods were on 1950s using just analogue techniques, since there was not computational capacity to apply the state of the art of adaptive methods. Just in the 1980s, with the discrete-time control methods and faster microprocessors, the state of the art of adaptive methods were applied. However, even nowadays there is many unused application potential in real industrial processes where nonadaptive controllers are used with "sufficient" performance, where a better quality of control and robustness could be reached using adaptive methodologies (Astrom & Wittenmark, 1994; Kolavennu, Palanki, Cartes, & Telotte, 2008). In Kolavennu et al. (2008) a model reference adaptive controller was used and compared with a classic PID controller, and it was shown that the adaptive method reached a better performance than the classic PID controller. Both controllers were studied on the tracking of a time-varying power profile in an automobile powered by a fuel cell. In Panzani, Corno, and Savaresi (2013) a nonlinear adaptive control strategy for a motorcycle electronic throttle body is presented. The controller architecture includes a friction compensator besides a PID regulator, and the tuning of the controller relies on the detailed description of the system that has been achieved via a closed loop identification method. The experimental tests have shown the effectiveness of this adaptive control architecture. In a generic way, adaptive controllers can be divided in three classes (Bobál et al., 2005): Model Reference Adaptive Systems (MRAS), Heuristic Approach (HA), and Self-Tuning Controllers (STC). In MRAS controllers, it is determined the output error between the adjustable system and the reference model, and then an adaptation mechanism is used to tune the adjustable system or it is generated a suitable input signal (Bashivan & Fatehi, 2012; Lim, Venugopal, & Ulsoy, 2012). The control methods based on HA are designed using operators' experience (Ajiboye & Weir, 2005; Fileti, Antunes, Silva, Jr, & Pereira, 2007) and in a general way they do not require the optimal solution of a problem, just a solution that achieves good results. However, both MRAS and HA require knowledge about the process to be controlled. STC algorithms recursively estimate the

[*] Corresponding author.
E-mail addresses: jermendes@isr.uc.pt (J. Mendes), lbica@isr.uc.pt (L. Osório), rui@isr.uc.pt (R. Araújo).

http://dx.doi.org/10.1016/j.conengprac.2017.09.006
Received 29 December 2016; Received in revised form 30 June 2017; Accepted 11 September 2017
0967-0661/© 2017 Elsevier Ltd. All rights reserved.

unknown characteristics/model of the process to be controlled (Wang, Cheng, & Kong, 2011; Zhang, Ding, & Shi, 2009), and the process model is used to design a suitable controller. Taking into account the identification methods used, STC can be classified into two categories (Psichogios & Ungar, 1991): explicit STC (also known as indirect STC), and implicit STC (also known as direct STC). In explicit STC, the process model, typically in the form of a transfer function, is obtained using explicit identification methods. In implicit STC, implicit identification algorithms use the plant's relationship between the process input and output to directly calculate the parameters of the controller. Since the objective of this paper is to study controllers with PID structure which are close to the concept of "plug and play", in this paper the STC controllers will be studied.

This work covers STC algorithms with both explicit and implicit identification algorithms. In Osório, Mendes, Araújo, and Matias (2013) it is presented a study of the following Self-Tuning (STC) Controllers: SCT with implicit identification (Single Neuron Controller (Wang et al., 2011), a Least Squares with Support Vector Machine (LSSVM) controller (Wanfeng, Shengdun, & Yajing, 2008), and a LSSVM with Kernel Tuning (LSSVMKT) Controller (Ucak & Oke, 2011)), and STC with explicit identification (Dahlin PID Controller (Corripio & Tomkins, 1981), the Pole Placement Controller (Wittenmark, 1979), the Deadbeat controllers of second and third orders (Kučera, 1980), and the Ziegler–Nichols controller (Ziegler & Nichols, 1942)) using the recursive least squares method with adaptive directional forgetting (RLSMadf) identification method (Bobál et al., 2005). The present paper is an improvement of the study presented in Osório et al. (2013) which has the following contributions over (Osório et al., 2013): (1) in order to perform a better study of the controllers' performance, a new experimental setup was constructed with a much better resolution (11,000 Rotations Per Minute) and load capacity (load with the capacity to reduce the velocity of the motor up to 44% instead of up to 10% in the setup used in Osório et al. (2013)); (2) new tests, results, and conclusions were performed where for a better analysis, the initial response, the tracking response, and the load response of each controller were studied, allowing to conclude which controllers have better characteristics in different aspects like control performance, easiness to set up, which have a better initial response, and which enable faster reaction to plant variations and load disturbances; (3) to perform explicit identification, three algorithms based on the Least Squares Method (LSM) were used (Bobál et al., 2005): recursive LSM (RLSM), recursive LSM with exponential forgetting (RLSMef), and recursive LSM with adaptive directional forgetting (RLSMadf), instead of Osório et al. (2013) that used only the RLSMadf method. Thus, in the present paper, besides studying the above mentioned implicit STCs, each explicit controller (Dahlin, Pole Placement, Deadbeat of second and third orders, and Ziegler–Nichols controllers) was combined with the RLSM, RLSMef, and RLSMadf identification methods, resulting in 18 "different" controllers.

To compare the controllers' performance, a real experimental setup composed of two coupled DC motors with varying load, was built and used. Each algorithm was tested to determine its ability to perform the initial identification of the plant, to respond to variations in the reference speed, and to respond to variations in the load of the motor. Besides evaluating if these control algorithms are able to control an unknown plant, this work determines which controllers have better characteristics like control performance, easiness to set up, which have a better initial response, and which enable faster reaction to plant variations and load disturbance.

The paper is organized as follows. Section 2 presents the explicit identification algorithms and the indirect control algorithms that were tested in this work. Section 3 describes and tests direct control algorithms and their implicit identifications. Section 4 is dedicated to the analysis and discussion of the results obtained with the tests. Finally, Section 5 makes concluding remarks.

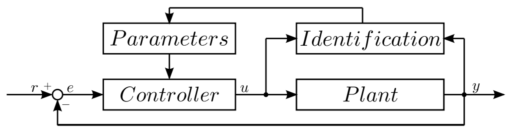


Fig. 1. Explicit STC.

2. Explicit STCs

This section describes the explicit identification algorithms and the indirect control algorithms that were compared.

The block diagram of Fig. 1 schematizes how an explicit STC performs adaptive control. The controller parameters are determined in two steps. First, the controller's and the plant's outputs are used to determine the plant's model, normally in the form of a transfer function, and then the model parameters determined in the first step are used to determine the controller parameters. These steps are repeated at each control cycle allowing the explicit STC to recursively adapt the estimated model parameters and the controller parameters. In this paper, the desired output is defined as $r(k)$.

2.1. Identification

In explicit STC controllers, it is necessary to estimate the process model, which, in this paper, is defined by a transfer function learned recursively in real-time. In Bobál et al. (2005), three different identification algorithms based on LSM can be found. In the simpler, the recursively LSM (RLSM), all the pairs of input/output data affect the identified parameters with the same weight. In the recursive LSM with exponential forgetting (RLSMef), the latest pairs of input/output data affect identified parameters more than older pairs using a forgetting coefficient. The recursive LSM with adaptive directional forgetting (RLSMadf), the most evolved of the three, uses a forgetting factor that is automatically adjusted depending on the changes of the input and output signals.

The model estimated in the LSM-based identification algorithms used in this paper (RLSM, RLSMef, and RLSMadf) are defined by the following transfer function

$$G(z) = \frac{B(z^{-1})}{A(z^{-1})} = \frac{b_1 z^{-1} + b_2 z^{-2} + \cdots + b_m z^{-m}}{1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_n z^{-n}} z^{-d}, \tag{1}$$

where $m, n \in \mathbb{N}$ are the input and output orders of the system, respectively, and $d \in \mathbb{N}$ is the time-delay. The system model is given by

$$\frac{y(k)}{u(k)} = \frac{B(z^{-1})}{A(z^{-1})},$$
$$A(z^{-1})y(k) = B(z^{-1})u(k), \tag{2}$$

where $u(\cdot) : \mathbb{N} \to \mathbb{R}$ and $y(\cdot) : \mathbb{N} \to \mathbb{R}$ are the process input and output, respectively.

The estimated output of the learned plant is given by

$$\begin{aligned}\hat{y}(k) &= \boldsymbol{\Theta}^T(k-1)\boldsymbol{\Phi}(k),\\ &= -\hat{a}_1 y(k-1) - \cdots - \hat{a}_n y(k-n)\\ &\quad + \hat{b}_1 u(k-d-1) + \cdots +\\ &\quad + \hat{b}_m u(k-d-m),\end{aligned} \tag{3}$$

where vector $\boldsymbol{\Theta}(k-1) = [\hat{a}_1, \ldots, \hat{a}_n, \hat{b}_1, \ldots, \hat{b}_m]^T$ contains the estimated plant's parameters from the previous iteration, and $\boldsymbol{\Phi}(k) = [-y(k-1), \ldots, -y(k-n), u(k-d-1), \ldots, u(k-d-m)]^T$ is the regression vector which contains information about the input and output of the plant.

The identification methods used/studied are described in the following items:

- *Recursive Least Squares Method:* In the simplest version of RLSM all the pairs of input/output data affect the identified parameters with the same weight. The vector of parameter estimations $\boldsymbol{\Theta}(k)$ is updated in each iteration, k, according to Eq. (4),

$$\boldsymbol{\Theta}(k) = \boldsymbol{\Theta}(k-1) + \frac{\boldsymbol{C}(k-1)\boldsymbol{\Phi}(k)}{1+\boldsymbol{\Phi}(k)^T\boldsymbol{C}(k-1)\boldsymbol{\Phi}(k)} \times (y(k) - \boldsymbol{\Theta}(k-1)^T\boldsymbol{\Phi}(k)), \tag{4}$$

where $\boldsymbol{C}(k)$ is the covariance matrix of the regression vector $\boldsymbol{\Phi}(k)$ which is updated at each iteration, k, using Eq. (5),

$$\boldsymbol{C}(k) = \boldsymbol{C}(k-1) - \frac{\boldsymbol{C}(k-1)\boldsymbol{\Phi}(k)\boldsymbol{\Phi}(k)^T\boldsymbol{C}(k-1)}{1+\boldsymbol{\Phi}(k)^T\boldsymbol{C}(k-1)\boldsymbol{\Phi}(k)}. \tag{5}$$

- *Recursive Least Squares Method with exponential forgetting:* The advantage of the RLSMef over the base RLSM is that the latest pairs of input/output data affect identified parameters more than older pairs. This is accomplished using a forgetting coefficient φ to decrease the influence of the older data. The vector of parameter estimations $\boldsymbol{\Theta}(k)$ is updated at each iteration, k, using Eq. (6),

$$\boldsymbol{\Theta}(k) = \boldsymbol{\Theta}(k-1) + \frac{\boldsymbol{C}(k-1)\boldsymbol{\Phi}(k)}{\varphi+\boldsymbol{\Phi}(k)^T\boldsymbol{C}(k-1)\boldsymbol{\Phi}(k)} \cdot (y(k) - \boldsymbol{\Theta}(k-1)^T\boldsymbol{\Phi}(k)), \tag{6}$$

while the covariance matrix $\boldsymbol{C}(k)$ of the regression vector $\boldsymbol{\Phi}(k)$ is updated at each iteration, k, using Eq. (7),

$$\boldsymbol{C}(k) = \frac{1}{\varphi}\left(\boldsymbol{C}(k-1) - \frac{\boldsymbol{C}(k-1)\boldsymbol{\Phi}(k)\boldsymbol{\Phi}(k)^T\boldsymbol{C}(k-1)}{\varphi+\boldsymbol{\Phi}(k)^T\boldsymbol{C}(k-1)\boldsymbol{\Phi}(k)}\right). \tag{7}$$

The forgetting factor φ should be in the range: $0 < \varphi \le 1$ and the smaller it gets, the smaller the influence the older pairs/data have on the current parameter estimation.

- *Recursive Least Squares Method with adaptive directional forgetting:* The RLSMadf is an evolved form of RLSM where the advantage of the RLSMadf over RLSM and RLSMef is its forgetting factor that gives less weight to older data compensating the possible non-uniform distribution of information of the time-varying nature of the process (Bittanti, Bolzern, & Campi, 1989) and is automatically updated at each iteration.

  In each iteration, k, the model parameters $\boldsymbol{\Theta}(k)$ are obtained by

$$\boldsymbol{\Theta}(k) = \boldsymbol{\Theta}(k-1) + \frac{\boldsymbol{C}(k-1)\boldsymbol{\Phi}(k)}{1+\xi}(y(k) - \boldsymbol{\Theta}(k-1)^T\boldsymbol{\Phi}(k)), \tag{8}$$

where $\xi = \boldsymbol{\Phi}(k)^T\boldsymbol{C}(k-1)\boldsymbol{\Phi}(k)$, and $\boldsymbol{C}(k)$ is the covariance matrix of the regression vector $\boldsymbol{\Phi}(k)$ which is updated at each iteration, k, using Eq. (9),

$$\boldsymbol{C}(k) = \boldsymbol{C}(k-1) - \frac{\boldsymbol{C}(k-1)\boldsymbol{\Phi}(k)\boldsymbol{\Phi}(k)^T\boldsymbol{C}(k-1)}{\varepsilon^{-1}+\xi}, \tag{9}$$

where $\varepsilon = \varphi(k-1) - \frac{1-\varphi(k-1)}{\xi}$ and $\varphi(k-1)$ is the forgetting factor at iteration $k-1$. The adaption of $\varphi(k)$ is performed as follows:

$$\varphi(k) = \frac{1}{1+(1+\rho)\left\{\ln(1+\xi) + \left[\frac{(\nu(k)+1)\eta}{1+\xi+\eta} - 1\right]\frac{\xi}{1+\xi}\right\}}, \tag{10}$$

where $\nu(k) = \varphi(k-1)(\nu(k-1)+1)$, $\eta = \frac{(y(k)-\boldsymbol{\Theta}^T(k-1)\boldsymbol{\Phi}(k))^2}{\lambda(k)}$, $\lambda(k) = \varphi(k-1)\left[\lambda(k-1) + \frac{(y(k)-\boldsymbol{\Theta}(k-1)^T\boldsymbol{\Phi}(k))^2}{1+\xi}\right]$, and ρ a forgetting factor.

In all the identification methods (RLSM, RLSMef, and RLSMadf) it is necessary initialize the initial estimation of the model parameters ($\Theta(0)$), and the covariance matrix ($\boldsymbol{C}(0)$) at $k = 0$. $\Theta(0)$ can be initialized using any method to extract model parameters as for example the Least Squares Method (LSM). In this paper, it is assumed to be dealing with unknown plants. Therefore, all model parameters are initialized with the value 0.1. $\boldsymbol{C}(0)$ must be defined as a diagonal matrix where its diagonal values are usually large numbers. In this paper, for example, such diagonal values were initialized to 10^3. When compared to RLSM, as explained above, RLSMef and RLSMadf have forgetting factors φ and ρ, respectively, both to be initialized in the range $]0, 1]$. In RLSMadf, the forgetting factor $\varphi(k)$ is automatically adjusted depending on the changes of the input and output signals. Variables $\lambda(k)$ and $\nu(k)$ are also automatically adjusted meaning that their initial values only have impact in the initial moments of the identification process. In any case, they should be set between zero and one.

2.2. Control algorithms

This section briefly overviews the five explicit STC controllers studied in this paper. In these controllers, $e(k)$ represents the control error which is given by $e(k) = r(k) - y(k)$, and the sampling interval is represented by T_0.

- *Dahlin PID Controller* (Corripio & Tomkins, 1981): This controller uses the transfer function defined in Eq. (1) with $n = 2$, and $m = 1$, being the estimated plant's parameters defined as $\boldsymbol{\Theta}(k-1) = [\hat{a}_1, \hat{a}_2, \hat{b}_1]^T$, the regression vector defined as $\boldsymbol{\Phi}(k) = [-y(k-1), -y(k-2), u(k-1)]^T$, and its control law given by

$$u(k) = K_p + \left\{e(k) - e(k-1) + \frac{T_0}{T_I}e(k) + \frac{T_D}{T_0}[e(k) - 2e(k-1) + e(k-2)]\right\} + u(k-1), \tag{11}$$

where K_p, T_I, T_D are the proportional gain, the integral time constant, and the differential time constant, respectively, which depend on the model parameters as follows:

$$K_p = \frac{(\hat{a}_1 + 2\hat{a}_2)\,Q}{b_1}, \tag{12}$$

$$T_I = -\frac{T_0}{\frac{1}{\hat{a}_1+2\hat{a}_2} + 1 + \frac{T_D}{T_0}}, \tag{13}$$

$$T_D = \frac{T_0\hat{a}_2 Q}{K_P\hat{b}_1}, \tag{14}$$

where $Q = 1 - e^{-\frac{T_0}{B}}$. The adjustment factor B specifies the dominant time constant of the of closed loop step response, where the smaller the positive constant B gets, quicker is the step response of closed control loop.

- *Pole Placement* (Wittenmark, 1979): For the Pole Placement controller the closed loop natural frequency (ω_n) and damping factor (ξ) must be tunned by the user in order to control a second order plant with $n = 2$ and $m = 2$. In this way, the vector of the estimated plant's parameters is defined as $\boldsymbol{\Theta}(k-1) = [\hat{a}_1, \hat{a}_2, \hat{b}_1, \hat{b}_2]^T$ and the regression vector as $\boldsymbol{\Phi}(k) = [-y(k-1), -y(k-2), u(k-1), u(k-2)]^T$. The control law is given by the following equation

$$u(k) = q_0e(k) + q_1e(k-1) + q_2e(k-2) + (1-\gamma)u(k-1) + \gamma u(k-2), \tag{15}$$

where the coefficients q_0, q_1 and q_2 are obtained by

$$q_0 = \frac{1}{\hat{b}_1}(d_1 + 1 - \hat{a}_1 - \gamma), \tag{16}$$

$$q_1 = \frac{\hat{a}_2}{\hat{b}_2} - q_2\left(\frac{\hat{b}_1}{\hat{b}_2} - \frac{\hat{a}_1}{\hat{a}_2} + 1\right), \tag{17}$$

$$q_2 = \frac{s_1}{r_1}, \tag{18}$$

where

$$d_1 = \begin{cases} -2e^{-\xi\omega_nT_0}\cos(\omega_nT_0\sqrt{1-\xi^2}), & \text{if } \xi \le 1, \\ -2e^{-\xi\omega_nT_0}\cosh(\omega_nT_0\sqrt{\xi^2-1}), & \text{if } \xi > 1, \end{cases} \tag{19}$$

$$d_2 = e^{-2\xi\omega_nT_0}, \tag{20}$$

$$r_1 = (\hat{b}_1 + \hat{b}_2)(\hat{a}_1\hat{b}_1\hat{b}_2 - \hat{a}_2\hat{b}_1^2 - \hat{b}_2^2), \tag{21}$$

$$s_1 = \hat{a}_2[(\hat{b}_1 + \hat{b}_2)(\hat{a}_1\hat{b}_2 - \hat{a}_2\hat{b}_1) + \hat{b}_2(\hat{b}_1d_2 - \hat{b}_2d_1 - \hat{b}_2)], \tag{22}$$

$$\gamma = q_2\frac{\hat{b}_2}{\hat{a}_2}. \tag{23}$$

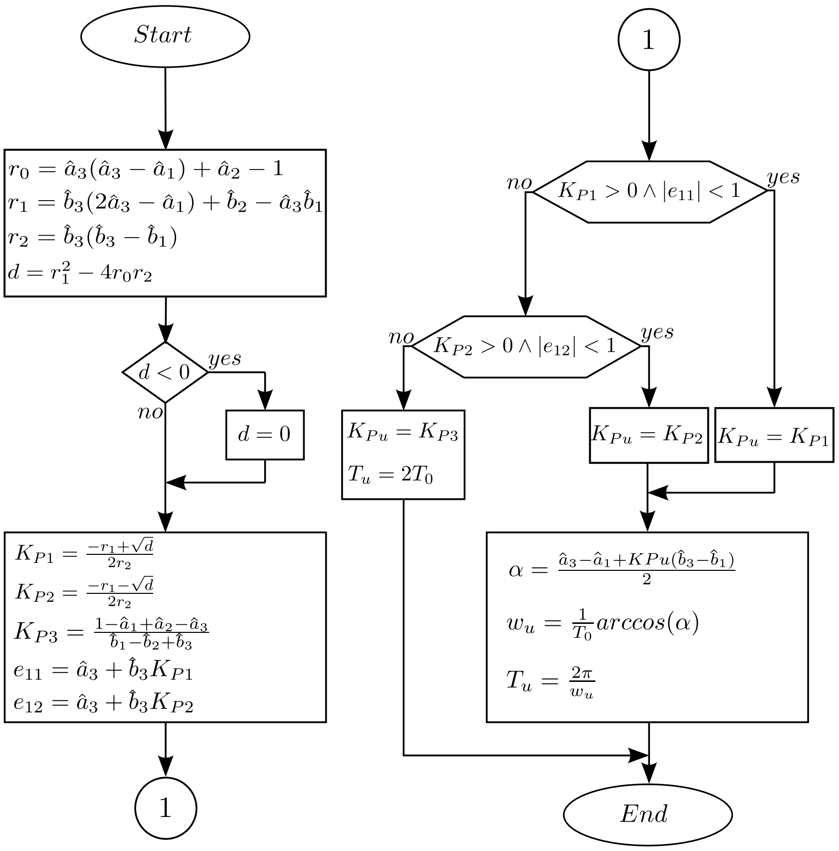

Start
$r_0 = \hat{a}_3(\hat{a}_3 - \hat{a}_1) + \hat{a}_2 - 1$
$r_1 = \hat{b}_3(2\hat{a}_3 - \hat{a}_1) + \hat{b}_2 - \hat{a}_3\hat{b}_1$
$r_2 = \hat{b}_3(\hat{b}_3 - \hat{b}_1)$
$d = r_1^2 - 4r_0r_2$
$d < 0$
yes
no
$d = 0$
$K_{P1} = \frac{-r_1+\sqrt{d}}{2r_2}$
$K_{P2} = \frac{-r_1-\sqrt{d}}{2r_2}$
$K_{P3} = \frac{1-\hat{a}_1+\hat{a}_2-\hat{a}_3}{\hat{b}_1-\hat{b}_2+\hat{b}_3}$
$e_{11} = \hat{a}_3 + \hat{b}_3K_{P1}$
$e_{12} = \hat{a}_3 + \hat{b}_3K_{P2}$
1
1
no
$K_{P1} > 0 \wedge |e_{11}| < 1$
yes
no
$K_{P2} > 0 \wedge |e_{12}| < 1$
yes
$K_{Pu} = K_{P3}$
$T_u = 2T_0$
$K_{Pu} = K_{P2}$
$K_{Pu} = K_{P1}$
$\alpha = \frac{\hat{a}_3-\hat{a}_1+KPu(\hat{b}_3-\hat{b}_1)}{2}$
$w_u = \frac{1}{T_0}arccos(\alpha)$
$T_u = \frac{2\pi}{w_u}$
End

Fig. 2. Ziegler–Nichols tunning method to obtain the ultimate proportional gain K_{P_u} and the ultimate period of oscillations T_u.

- *Deadbeat Controller of Second Order* (Kučera, 1980): This controller uses a second order plant with $n = 2$ and $m = 2$, being the vector of the estimated plant's parameters defined as $\boldsymbol{\Theta}(k-1) = [\hat{a}_1, \hat{a}_2, \hat{b}_1, \hat{b}_2]^T$ and the regression vector as $\boldsymbol{\Phi}(k) = [-y(k-1), -y(k-2), u(k-1), u(k-2)]^T$. Its control law is given by

$$u(k) = r_0r(k) - q_0y(k) - q_1y(k-1) - p_1u(k-1), \tag{24}$$

where the controller's coefficients r_0, q_0, q_1 and p_1 are obtained by $r_0 = 1/(\hat{b}_1 + \hat{b}_2)$, and

$$\begin{bmatrix} p_1 \\ q_0 \\ q_1 \end{bmatrix} = \begin{bmatrix} 1 & \hat{b}_1 & 0 \\ \hat{a}_1 & \hat{b}_2 & \hat{b}_1 \\ \hat{a}_2 & 0 & \hat{b}_2 \end{bmatrix}^{-1} \begin{bmatrix} -\hat{a}_1 \\ -\hat{a}_2 \\ 0 \end{bmatrix}. \tag{25}$$

- *Deadbeat Controller of Third Order* (Kučera, 1980): This controller uses a third order plant model with $n = 3$ and $m = 3$, being the vector of the estimated plant's parameters defined as $\boldsymbol{\Theta}(k-1) = [\hat{a}_1, \hat{a}_2, \hat{a}_3, \hat{b}_1, \hat{b}_2, \hat{b}_3]^T$, and the regression vector as $\boldsymbol{\Phi}(k) = [-y(k-1), -y(k-2), -y(k-3), u(k-1), u(k-2), u(k-3)]^T$. Its control law is given by

$$u(k) = r_0r(k) - q_0y(k) - q_1y(k-1) - q_2y(k-2) - p_1u(k-1) - p_2u(k-2), \tag{26}$$

where the controller's coefficients r_0, p_1, p_2, q_0, q_1 and q_2 are given by $r_0 = 1/(\hat{b}_1 + \hat{b}_2 + \hat{b}_3)$, and

$$\begin{bmatrix} p_1 \\ p_2 \\ q_0 \\ q_1 \\ q_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & \hat{b}_1 & 0 & 0 \\ \hat{a}_1 & 1 & \hat{b}_2 & \hat{b}_1 & 0 \\ \hat{a}_2 & \hat{a}_1 & \hat{b}_3 & \hat{b}_2 & \hat{b}_1 \\ \hat{a}_3 & \hat{a}_2 & 0 & \hat{b}_3 & \hat{b}_2 \\ 0 & \hat{a}_3 & 0 & 0 & \hat{b}_3 \end{bmatrix}^{-1} \begin{bmatrix} -\hat{a}_1 \\ -\hat{a}_2 \\ -\hat{a}_3 \\ 0 \\ 0 \end{bmatrix}. \tag{27}$$

- *Ziegler–Nichols* (Ziegler & Nichols, 1942): Regardless of the fact that this tunning procedure was designed 70 years ago, it is still a good option, as will be seen in the results. This controller uses a third order plant model with $n = 3$ and $m = 3$, being the vector of the estimated plant's parameters defined as $\boldsymbol{\Theta}(k-1) = [\hat{a}_1, \hat{a}_2, \hat{a}_3, \hat{b}_1, \hat{b}_2, \hat{b}_3]^T$ and the regression vector as $\boldsymbol{\Phi}(k) = [-y(k-1), -y(k-2), -y(k-3), u(k-1), u(k-2), u(k-3)]^T$. Its control law is given by

$$u(k) = q_0e(k) + q_1e(k-1) + q_2e(k-2) + u(k-1), \tag{28}$$

where the controller's coefficients q_0, q_1 and q_2 are given by

$$q_0 = K_P\left(1 + \frac{T_0}{T_I} + \frac{T_D}{T_0}\right), \tag{29}$$

$$q_1 = -K_P\left(1 + 2\frac{T_D}{T_0}\right), \tag{30}$$

$$q_2 = K_P\frac{T_D}{T_0}, \tag{31}$$

where the proportional gain is $K_P = 0.6K_{P_u}$, the integral time constant is $T_I = 0.5T_u$, and the differential time constant is $T_D = 0.125T_u$. The ultimate proportional gain K_{P_u} and the ultimate period of oscillations T_u must be obtained since this control method is based on the Ziegler–Nichols algorithm. Fig. 2 defines how obtain K_{P_u} and T_u.

Excluding the variables from the identification methods, Deadbeat (second and third orders) and Ziegler–Nichols controllers do not have any variable to be initialized. Dahlin controller has the parameter B to tune the responsiveness of the controller. A value of $B = 2$ was used in this paper. The pole Placement controller has the natural frequency ω_n, and the damping factor ξ to be tuned. In this paper these parameters were defined as $\omega_n = 2/T_0$, and $\xi = 1.2$.

3. Implicit STCs

This section describes the direct control algorithms (and their implicit identification) that were studied.

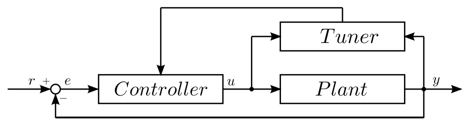


Fig. 3. Implicit STC.

The block diagram of Fig. 3 schematizes how an implicit STC performs adaptive control. The controller parameters are computed in just one step: the controller's and the plant's outputs are used to directly determine the parameters of the controller. On each control cycle the control parameters are recalculated allowing the implicit STC to adapt to changes in the plant's dynamics.

- *Single Neuron* (Wang et al., 2011): The Single Neuron controller is a self adaptive controller with a PID structure and requires small computation effort. The single neuron can attain self-learning and self-adaptation by adjusting connection weights so as to adjust PID parameters instantly. The control law is given by

$$u(k) = u(k-1) + K_P x_1(k) + K_I x_2(k) + K_D x_3(k), \tag{32}$$

where

$$x_1(k) = e(k) = r(k) - y(k), \tag{33}$$

$$\begin{aligned} x_2(k) &= \Delta e(k) = e(k) - e(k-1) = x_1(k) - x_1(k-1), \\ &= \Delta x_1(k), \end{aligned} \tag{34}$$

$$\begin{aligned} x_3(k) &= \Delta^2 e(k) = e(k) - 2e(k-1) + e(k-2), \\ &= x_2(k) - x_2(k-1) = \Delta x_2(k). \end{aligned} \tag{35}$$

The proportional gain K_P, the integral gain K_I, and the differential gain K_D are given by

$$K_P = K\bar{w}_1(k), \tag{36}$$

$$K_I = K\bar{w}_2(k), \tag{37}$$

$$K_D = K\bar{w}_3(k), \tag{38}$$

where K is a positive scale parameter which allows to increase/decrease the responsiveness of the controller. The coefficients $\bar{w}_i(k)$ are given by

$$\bar{w}_i(k) = \frac{w_i(k)}{\sum_{i=1}^{3} |w_i(k)|}, \tag{39}$$

and are obtained through normalization of the weight coefficients

$$w_i(k) = w_i(k-1) + \eta_i K e(k) x_i(k-1) \mathrm{sgn}\left(\frac{\partial y(k)}{\partial i^*(k-1)}\right), \tag{40}$$

where η_i is the learning rate of the weight coefficients $w_i(k)$, and $\mathrm{sgn}(\cdot)$ is a signal function. The current reference of the single neuron $i^*(k)$ is given by

$$i^*(k) = i^*(k-1) + K \sum_{i=1}^{3} \bar{w}_i(k) x_i(k), \tag{41}$$

and $\partial y(k)/\partial i^*(k-1) = (y(k) - y(k-1))/(i^*(k-1) - i^*(k-2))$.

- *Least Squares Support Vector Machine (LSSVM)* (Wanfeng et al., 2008): This controller has a PID structure and its parameters are tunned/adjusted by an online implicit identification approach using the gradient information of LSSVM. The main objective of regression estimations is to approximate a function from a set of training samples. Its control law is given by

$$u(k) = u(k-1) + K_P(k) xc_1(k) + K_I(k) xc_2(k) + K_D(k) xc_3(k), \tag{42}$$

where,

$$xc_1(k) = e(k) - e(k-1), \tag{43}$$

$$xc_2(k) = e(k), \tag{44}$$

$$xc_3(k) = e(k) - 2e(k-1) + e(k-2), \tag{45}$$

$$e(k) = r(k) - y(k). \tag{46}$$

The proportional gain $K_P(k+1)$, the integral gain $K_I(k+1)$, and the derivative gain $K_D(k+1)$ are given by

$$K_P(k+1) = K_P(k) + \Delta K_P(k), \tag{47}$$

$$K_I(k+1) = K_I(k) + \Delta K_I(k), \tag{48}$$

$$K_D(k+1) = K_D(k) + \Delta K_D(k), \tag{49}$$

where

$$\Delta K_P(k) = \eta e(k) \frac{\partial \hat{y}}{\partial u}(k) xc_1(k), \tag{50}$$

$$\Delta K_I(k) = \eta e(k) \frac{\partial \hat{y}}{\partial u}(k) xc_2(k), \tag{51}$$

$$\Delta K_D(k) = \eta e(k) \frac{\partial \hat{y}}{\partial u}(k) xc_3(k), \tag{52}$$

where $0 < \eta < 1$ is the learning rate,

$$\frac{\partial \hat{y}}{\partial u}(k) = \sum_{i=k-L}^{k-1} \frac{\alpha_{i-k+L+1}(k)(u(k) - u(i)) K(\boldsymbol{x}(k), \boldsymbol{x}(i))}{\sigma^2}, \tag{53}$$

where the support vectors from the last L iterations are used, and sample $u(i)$ is the command signal stored at component $n+1$ of the support vector at instant i, $x(i)$ (55). Choosing a larger L will make the identification process more stable but will also make it react slower when variations in the plant occur.

$$K(\boldsymbol{x}(i), \boldsymbol{x}(j)) = \exp\left(\frac{-\|\boldsymbol{x}(i) - \boldsymbol{x}(j)\|^2}{\sigma^2}\right), \tag{54}$$

is the RBF used in the kernel function of the LSSVM, σ is the bandwidth of the RBF, $\boldsymbol{x}(k)$ is the state vector at iteration k

$$\begin{aligned} \boldsymbol{x}(k) &= [x_1(k), \dots, x_{m+n}(k)]^T, \\ &= [y(k), \dots, y(k-n+1), u(k), \dots, u(k-m+1)]^T, \end{aligned} \tag{55}$$

m and n are the number of samples of the control signal and the output, respectively, stored in each support vector,

$$\boldsymbol{\alpha}(k) = [\alpha_1(k), \dots, \alpha_L(k)]^T = \boldsymbol{U}(k)(\boldsymbol{Y}(k) - \boldsymbol{1}_v b(k)), \tag{56}$$

$$b(k) = \frac{\boldsymbol{1}_v^T \boldsymbol{U}(k) \boldsymbol{Y}(k)}{\boldsymbol{1}_v^T \boldsymbol{U}(k) \boldsymbol{1}_v}, \tag{57}$$

and $\boldsymbol{1}_v = [1, \dots, 1]_{L\times 1}$, $\boldsymbol{Y}(k) = [y(k), \dots, y(k-L+1)]^T$,

$$\boldsymbol{U}(k) = \begin{bmatrix} \boldsymbol{A}(k) & \boldsymbol{H} \\ \boldsymbol{H}^T & h \end{bmatrix}^{-1}, \tag{58}$$

$$\begin{aligned} \boldsymbol{H} = &[K(\boldsymbol{x}(k-L), \boldsymbol{x}(k-1)), \dots, \\ &K(\boldsymbol{x}(k-L), \boldsymbol{x}(k-L+1))]^T, \end{aligned} \tag{59}$$

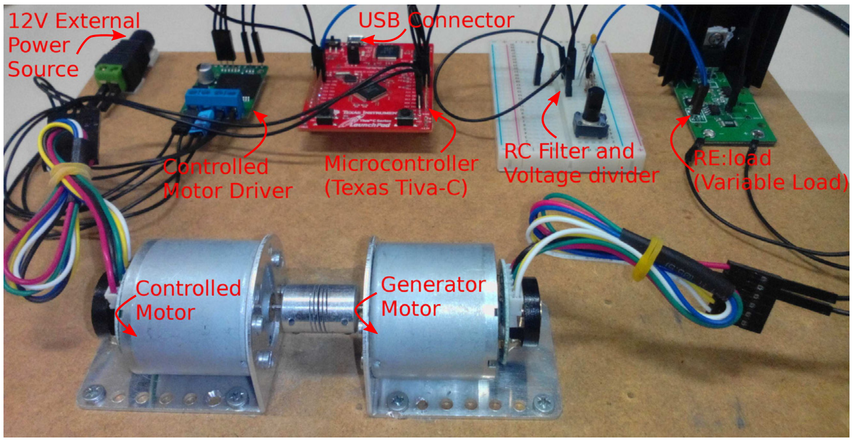
$$h = K(\boldsymbol{x}(k-L), \boldsymbol{x}(k-L)) + C^{-1}, \tag{60}$$

and $\boldsymbol{A}(k)$ is given by (61).

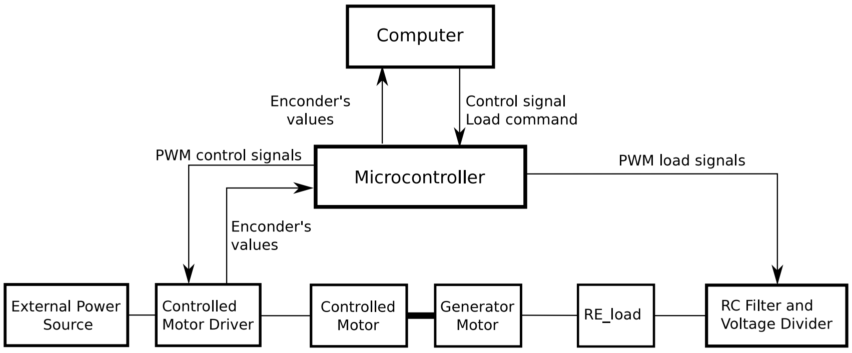
$$\boldsymbol{A}(k) = \begin{bmatrix} K(\boldsymbol{x}(k-1), \boldsymbol{x}(k-1)) + C^{-1} & \cdots & K(\boldsymbol{x}(k-L+1), \boldsymbol{x}(k-1)) \\ \vdots & \ddots & \vdots \\ K(\boldsymbol{x}(k-1), \boldsymbol{x}(k-L+1)) & \cdots & K(\boldsymbol{x}(k-L+1), \boldsymbol{x}(k-L+1)) + C^{-1} \end{bmatrix}. \tag{61}$$

C is a positive regularization factor, such that C^{-1} is added to the elements of the main diagonal of $\boldsymbol{A}(k)$, and if C is small, then possible outliers points are deemphasized. Before performing the inversion when calculating matrix $\boldsymbol{U}(k)$, it is possible to store matrix $\boldsymbol{A}(k)$, and use it to determine matrix $\boldsymbol{U}(k+1)$ as follows:

$$\boldsymbol{U}(k+1) = \begin{bmatrix} q & \boldsymbol{Q}^T \\ \boldsymbol{Q} & \boldsymbol{A}(k) \end{bmatrix}^{-1}. \tag{62}$$



(a) Photo of the experimental setup of the two coupled DC motors.



(b) Diagram of the experimental setup of the two coupled DC motors.

Fig. 4. The experimental scheme of the two coupled DC motors.

In this way it is only necessary to determine the factor q and the vector $\boldsymbol{Q}$ from Eqs. (63) and (64) respectively

$$q = K(\boldsymbol{x}(k), \boldsymbol{x}(k)) + C^{-1}, \tag{63}$$

$$\boldsymbol{Q} = [K(\boldsymbol{x}(k), \boldsymbol{x}(k-1)), \dots, K(\boldsymbol{x}(k), \boldsymbol{x}(k-L+1))]^T, \tag{64}$$

to obtain the next iteration's matrix $\boldsymbol{U}(k+1)$, saving the trouble of calculating the matrix $\boldsymbol{A}(k)$ at each iteration (in the first iteration it is still necessary to determine $\boldsymbol{A}(k)$ using Eq. (61).

- *Least Squares Support Vector Machine with Kernel Tuning (LSSVMKT) (Ucak & Oke, 2011)*: This controller is based on the previous one (LSSVM) where the main advance of LSSVMKT over the LSSVM controller is its ability to automatically adjust the kernel bandwidth (σ). The kernel bandwidth is adapted using

$$\sigma(k+1) = \sigma(k) + \Delta\sigma(k), \tag{65}$$

where

$$\Delta\sigma(k) = \eta(k)\hat{e}_m(k)\frac{\partial\hat{y}(k)}{\partial\sigma(k)}, \tag{66}$$

$$\frac{\partial\hat{y}(k)}{\partial\sigma(k)} = \sum_{i=k-L}^{k-1}\left\{\frac{\boldsymbol{\alpha}_{i-k+L+1}(k)K(\boldsymbol{x}(k),\boldsymbol{x}(i))}{\sigma(k)^3}\times (\boldsymbol{x}(k)-\boldsymbol{x}(i))^T\,(\boldsymbol{x}(k)-\boldsymbol{x}(i))\right\}, \tag{67}$$

$$\hat{e}_m(k) = y(k) - \hat{y}(k), \tag{68}$$

$$\hat{y}(k+1) = \sum_{i=k-L}^{k-1}\boldsymbol{\alpha}_{i-k+L+1}(k)K(\boldsymbol{x}(k),\boldsymbol{x}(i)) + b(k). \tag{69}$$

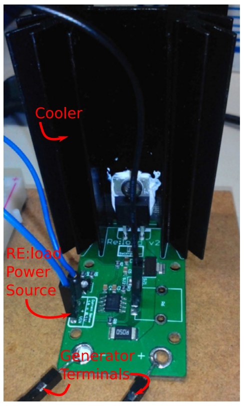
In all implicit controllers (Single Neuron, LSSVM, and LSSVMKT) it is necessary to initialize the initial estimation of the control parameters (K_P, K_I, and K_D). In this paper, these parameters were all initialized with the value 0.1, since it is assumed that the methods are dealing with unknown plants. On the Single Neuron controller, the parameter K must be tunned taking into account the responsiveness of the controller. In the paper this parameter was set as $K = 0.9$. The LSSVM and LSSVMKT controllers have a bigger complexity with six parameters which in this paper were defined as $\eta = 0.9$, $C = 320$, $\sigma = 0.9$, and $L = m = n = 30$.

4. Results and discussion

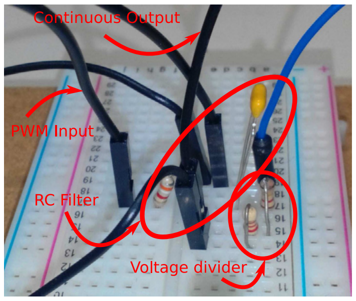
This section discusses the results obtained by the control algorithms present in this paper. Each explicit controller (Dahlin, Pole Placement, Deadbeat of second and third orders, and Ziegler–Nichols controllers) was combined with RLSM, RLSMef, and RLSMadf identification methods, resulting 15 "different" controllers. These controllers and all the implicit controllers (Single Neuron, LSSVM, and LSSVMKT) were compared using a real experimental setup composed of two coupled DC motors with varying load.

4.1. Plant

A real plant composed by two motors which are coupled by their shafts (Fig. 4) was used to test the control algorithms. The first motor is controlled by Pulse Width Modulation (PWM) and it makes the second motor rotate and behave as a generator. On the other hand, this generator is connected to a variable/controllable electrical load which acts as an energy sink where the amount of dissipated energy can be varied. The selected motors have an encoder with 64 Counts Per Revolution (CPR) and are rated for 11000 Rotations Per Minute at 12 [V] (free run). The velocity units are [pp/(100 ms)] (pulses per 100 milliseconds). The controlled motor is powered by a motor driver



(a) Photo of Re:load.



(b) Photo of the RC filter and the voltage divider.

Fig. 5. Photos of the Re:load, and the RC filter and voltage divider.

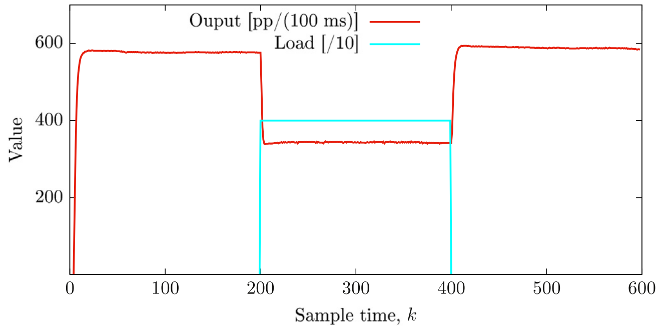


Fig. 6. Response of the plant when provided with a constant control signal ($u(k) = 2000$), and a turn on/off of the load. At instant $k = 200$ the maximum load value is applied and it returns to normal (without load) at instant $k = 400$.

that drains energy from an external power source and receives PWM control signals. The motor that works as a generator is connected to a "variable load" that is also controlled using PWM signals. Managing the plant, there is a Texas Instruments Tiva C micro-controller (μC). This μC establishes serial communication with a Computer where the control algorithms are running, receives through serial communication the PWM values to control both the PWM of the motor and the PWM of the load, delivers PWM signals both to the motor driver and the load, reads the motor's encoder and finally sends back to the computer the encoder's value using serial communication. The sampling period is 100 milliseconds.

An important feature of this plant is its "variable load", which is achieved using an electronic circuit from Arachnid Labs named Re:load (Fig. 5(a)) with a simple modification to allow it to be controlled by PWM. To be able to control the load from the computer it was needed to remove the potentiometer from the Re:load and deliver an analog voltage to the circuit. Because the used μC only has digital and PWM outputs it was needed to build an RC filter and a voltage divider (Fig. 5(b)) to be able to deliver a continuous voltage to the circuit's op-amp using a PWM output. This plant allows the control/command of both the motor and of the "heaviness" of the load by sending integer commands in the range from 0 to 4096.

Fig. 6 shows the effect of the variable load with its maximum value on the rotation velocity of the motor when it is fed a constant control signal ($u(k) = 2000$). The load is turned on at instant $k = 200$ and turned off at instant $k = 400$.

4.2. Results

The tests consisted on running all the control algorithms during $k = 1300$ sample times (130 s). The motor always started at rest and without load. The load was turned on to its maximum value at $k = 600$, and was turned off at $k = 1000$. This test allowed to verify the ability of each algorithm to control an unknown plant, and then comparing how each controller could cope with variations on the reference and on the system's characteristics (varying load). Note that, because the controller is dealing with unknowns plants, all model parameters ($\boldsymbol{\Theta}(k) = [\hat{a}_1, \ldots, \hat{a}_n, \hat{b}_1, \ldots, \hat{b}_m]$) of each explicit controller (Dahlin, Pole Placement, Deadbeat of Second Order, Deadbeat of Third Order, and Ziegler–Nichols) were initialized with the value 0.1. For the same above reasons, the control parameters (K_P, K_I and K_D) of each implicit controller was also initialized with the value 0.1.

The controllers have been made in the Scilab environment running on a PC with 3.40 GHz CPU with 4 cores and 8GB RAM, being the computational time at each sample time k of each controller (except LSSVM, and LSSVMKT controllers) less than 1 milliseconds, and for LSSVM, and LSSVMKT was in average of 4 and 5 milliseconds, respectively, with a maximum of 7 and 6 milliseconds, respectively.

The performance of the controllers are compared using two different statistical indexes, the Integral Time-weighted Absolute Error (ITAE), and the Root Mean Square Error (RMSE), which are defined as follows:

$$\text{ITAE} = \sum_{k=1}^{N} k|e(k)|, \tag{70}$$

$$\text{RMSE} = \sqrt{\frac{\left(\sum_{k=1}^{N} e(k)^2\right)}{N}}, \tag{71}$$

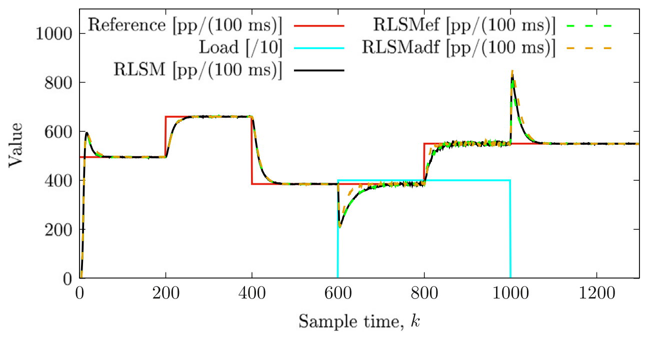
where N is the number of samples, and $e(k)$ is the output error at the iteration k.

Figs. 7 and 8 show the results of all control algorithms, and Table 1 summarizes the obtained results complementing the visual inspection by using ITAE, and RMSE numerical indexes (Eqs. (70)–(71)). Additionally, for a better analysis, the following sections will analyze the initial response ($0 \leq k \leq 150$) (Section 4.2.1), the tracking response ($151 \leq k \leq 550$) (Section 4.2.2), and the load response ($551 \leq k \leq 1150$) (Section 4.2.3) of each controller. Each controller was ranked by receiving a score from 1 to 18 for each of the ITAE and RMSE numerical indexes based on its performance (the best received 1 and the worst received 18). Then, for each controller, the sum of both indexes yields a global number of points for the controller that was used to rank/score the controllers, and the controller which had the least number of summed points was considered the best controller. For all the tables in this paper, in each column, the numbers in parentheses (if any) represent the score attributed to the controller, based on ranking by the criterion on that column, i.e., ranking by the numerical values outside the parenthesis on that column. In Tables 1–4, in the cases where one performance index has the same value for more than one controller, then the corresponding scores assigned on such index are the same. In cases where controllers have the same total score (same summed points over both indexes), it was defined to give the same total score to such controllers.

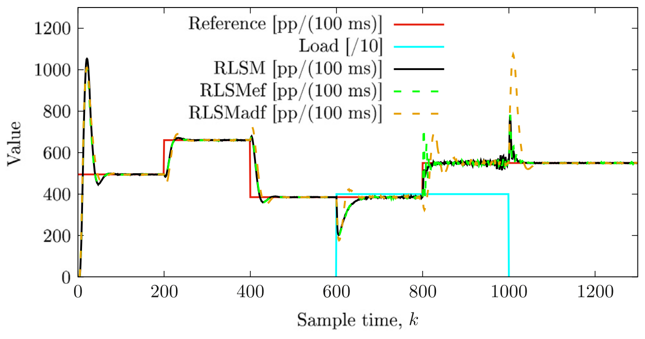
4.2.1. Initial response analysis

Table 2 shows the numeric performance index results of the initial response of all controller for instants $0 \leq k \leq 150$. Note that, as mentioned in Section 4.2, all model parameters ($\boldsymbol{\Theta}(k)$) of each explicit controller and control parameters (K_P, K_I, and K_D) of each implicit controller were initialized with the value of 0.1, representing the absence of knowledge about the model of the plant to be controlled.

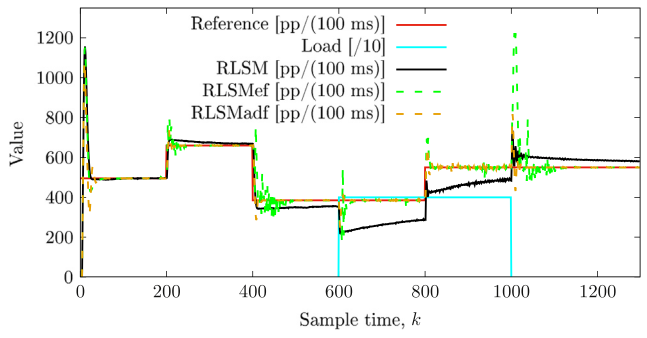
In this particular section, for a better analysis, different statistical indexes were used instead of the ITAI, and RSME. To analyze the initial response of all controllers, the percentage of overshoot and



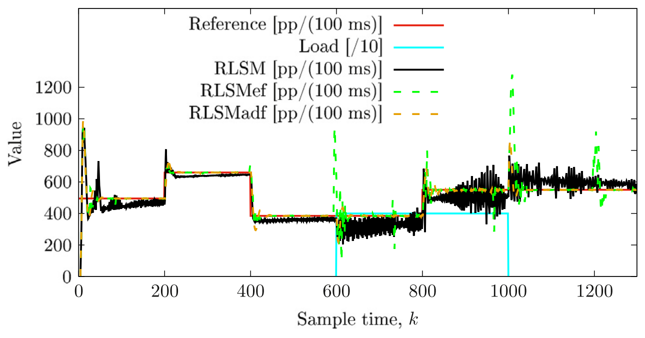
(a) Dahlin's control results.



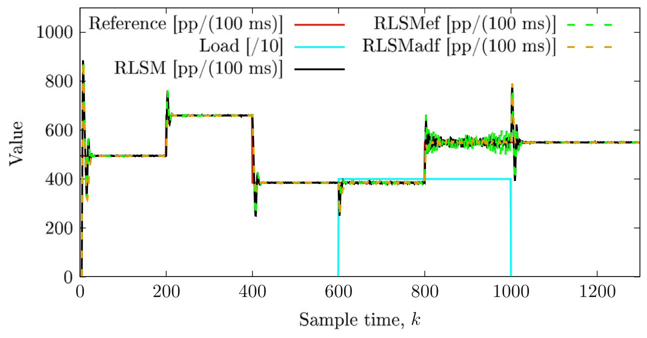
(b) Pole Placement control results.



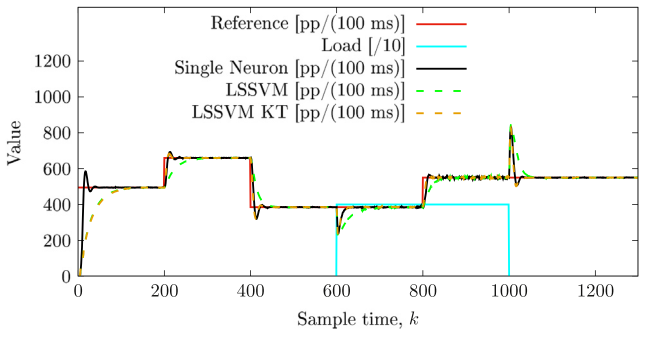
(c) Deadbeat of second order control results.



(d) Deadbeat of third order control results.

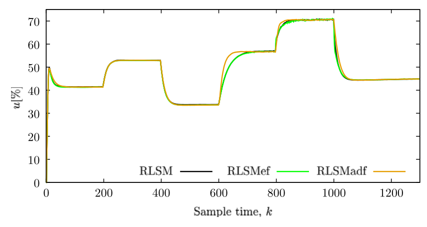


(e) Ziegler–Nichols control results.

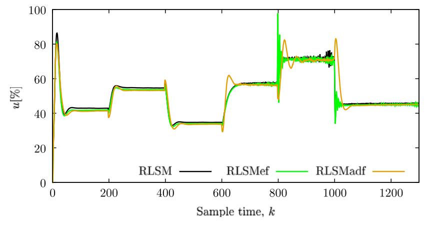


(f) Control results of Single Neuron, LSSVM, and LSSVMKT.

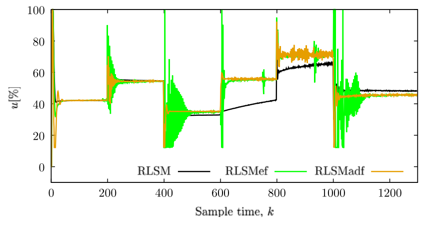
Fig. 7. Results of the test with all the algorithms controlling a real DC Motor with a varying load.



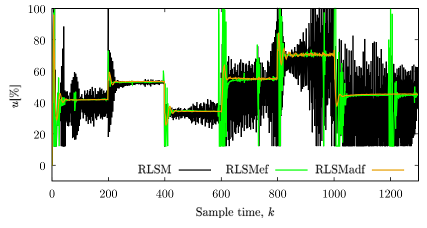
(a) Dahlin's controller.



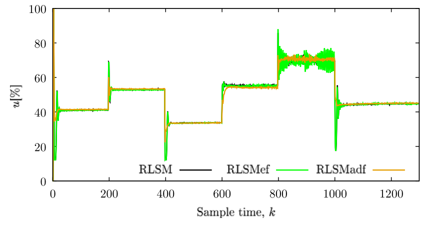
(b) Pole Placement controller.



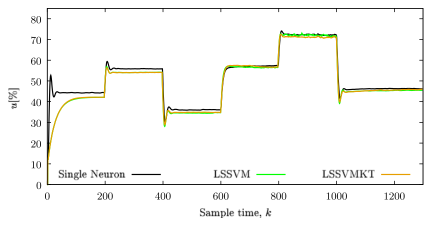
(c) Deadbeat of second order controller.



(d) Deadbeat of third order controller.



(e) Ziegler–Nichols controller.



(f) Single Neuron, LSSVM, and LSSVMKT controllers.

Fig. 8. Command signal of all algorithms controlling a real DC Motor with a varying load.

Table 1
Statistical comparison between the control algorithms studied in this article, considering all the time interval in each test. For each column, the numbers in parentheses (if any) represent the attributed score based on ranking on the criterion of that column. For each controller, the values presented on the "Points" column are the sum of the scores attributed on the respective ITAE and RMSE indexes.

		ITAE	RMSE	Points
Dahlin	RLSM	1.41×10^7 (13)	56.46 (8)	21 (9)
	RLSMef	1.38×10^7 (12)	**55.28** (7)	19 (7)
	RLSMadf	$\mathbf{1.36 \times 10^7}$ (11)	58.40 (9)	20 (8)
Pole Placement	RLSM	$\mathbf{0.83 \times 10^7}$ (9)	75.60 (14)	23 (11)
	RLSMef	0.90×10^7 (10)	**72.94** (12)	22 (10)
	RLSMadf	2.13×10^7 (16)	98.97(18)	34 (15)
Deadbeat 2nd	RLSM	4.60×10^7 (18)	88.65(17)	35 (16)
	RLSMef	1.56×10^7 (14)	83.73(15)	29 (12)
	RLSMadf	$\mathbf{0.49 \times 10^7}$ (2)	**49.89** (5)	**7(3)**
Deadbeat 3rd	RLSM	3.62×10^7 (17)	73.10(13)	30 (13)
	RLSMef	1.66×10^7 (15)	83.93(16)	31 (14)
	RLSMadf	$\mathbf{0.74 \times 10^7}$ (8)	**51.58** (6)	14 (5)
Ziegler–Nichols	RLSM	0.51×10^7 (3)	35.36(1)	**4 (2)**
	RLSMef	0.69×10^7 (7)	36.78 (3)	10 (4)
	RLSMadf	$\mathbf{0.35 \times 10^7}$ (1)	**35.45** (2)	**3 (1)**
Single Neuron	✕	$\mathbf{0.64 \times 10^7}$ (6)	**47.66** (4)	10 (4)
LSSVM	✕	$\mathbf{0.63 \times 10^7}$ (5)	**61.79** (10)	15 (6)
LSSVMKT	✕	$\mathbf{0.59 \times 10^7}$ (4)	**61.91** (11)	15 (6)

Table 2
Statistical comparison between the control algorithms for their initial response ($0 \leq k \leq 150$). For each column, the numbers in parentheses (if any) represent the attributed score based on ranking on the criterion of that column. The values presented on the "Points" column are the sum of the scores attributed on the respective Overshoot and Settling time indexes.

		Overshoot	Settling time	Points
Dahlin	RLSM	20 (5)	57 (6)	16 (5)
	RLSMef	**19.19** (4)	**56** (5)	**13 (2)**
	RLSMadf	20 (5)	74 (10)	20 (8)
Pole Placement	RLSM	112.72 (14)	85 (12)	40 (15)
	RLSMef	**103.63** (12)	**83** (44)	35 (12)
	RLSMadf	104.04 (13)	105 (15)	41 (17)
Deadbeat 2nd	RLSM	133.73 (17)	104 (14)	48 (18)
	RLSMef	132.72 (16)	62 (8)	40 (16)
	RLSMadf	**117.78** (15)	**58** (7)	37 (14)
Deadbeat 3rd	RLSM	**90.70** (9)	- (18)	33 (13)
	RLSMef	92.52 (10)	97 (13)	37 (11)
	RLSMadf	100.20 (11	**73** (9)	31 (10)
Ziegler–Nichols	RLSM	78.79 (7)	**34** (1)	15 (4)
	RLSMef	79.80 (8)	36 (2)	18 (7)
	RLSMadf	**78.58** (6)	45 (3)	**15(3)**
Single Neuron	✕	**18.18** (3)	**55** (4)	**10 (1)**
LSSVM	✕	**0** (1)	**148** (16)	18 (6)
LSSVMKT	✕	**0.02** (2)	**138** (17)	21 (9)

settling time performance indexes, were used. The percentage of overshoot is the maximum value minus the reference value divided by the reference value. The settling time is the amount of time taken for a response reach and remain within some error band around the setpoint. In this paper, the error band is defined as 1% of the setpoint value (i.e., 5 [pp/(100 ms)]). For this section, it was considered that the percentage of overshoot has a bigger influence to reach a better performance than the settling time, being chosen as the total number of points the sum of twice of the points of percentage of overshoot and the points of settling time.

As can be seen in Fig. 7(f) and Table 2, the overshoots of the LSSVM and LSSVMKT controllers are near zero however their settling times are much larger when compared to the other controllers. From Table 2, by the statistical indexes used, the best initial response is given by the Single Neuron controller, that has a small overshoot (18.18%) and a fast settling time (Fig. 7(f)). The Dahlin controller with RLSMef, and the

Table 3
Statistical comparison between the control algorithms for their tracking response ($151 \leq k \leq 550$). For each column, the numbers in parentheses (if any) represent the attributed score based on ranking on the criterion of that column. The values presented on the "Points" column are the sum of the scores attributed on the respective ITAE and RMSE indexes.

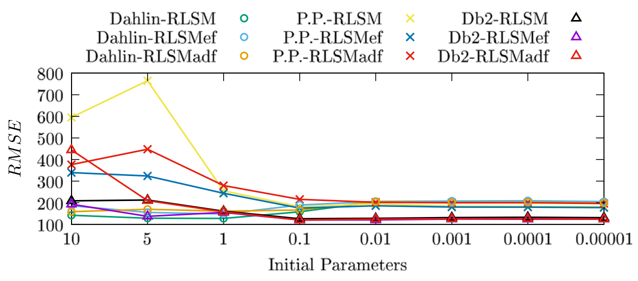
		ITAE	RMSE	Points
Dahlin	RLSM	$\mathbf{1.35 \times 10^6}$ (13)	**50.91** (13)	26 (10)
	RLSMef	1.36×10^6 (14)	50.17(14)	28 (12)
	RLSMadf	1.41×10^6 (15)	51.35(15)	30 (14)
Pole Placement	RLSM	$\mathbf{1.26 \times 10^6}$ (10)	**54.90** (16)	26 (10)
	RLSMef	1.30×10^6 (11)	56.39(17)	28 (12)
	RLSMadf	1.82×10^6 (16)	69.95(18)	34 (15)
Deadbeat 2nd	RLSM	2.49×10^6 (18)	38.85(11)	29 (13)
	RLSMef	1.33×10^6 (12)	45.49(12)	24 (9)
	RLSMadf	$\mathbf{0.48 \times 10^6}$ (4)	**29.84** (4)	8 (4)
Deadbeat 3rd	RLSM	2.14×10^6 (17)	37.61(10)	27 (11)
	RLSMef	$\mathbf{0.57 \times 10^6}$ (7)	33.35(8)	15 (7)
	RLSMadf	0.68×10^6 (8)	**32.53** (7)	15 (7)
Ziegler–Nichols	RLSM	0.43×10^6 (2)	26.71(2)	**4 (2)**
	RLSMef	0.44×10^6 (3)	26.74(3)	**6 (3)**
	RLSMadf	$\mathbf{0.34 \times 10^6}$ (1)	**24.28** (1)	**2 (1)**
Single Neuron	✕	$\mathbf{0.69 \times 10^6}$ (9)	**36.66** (9)	18 (8)
LSSVM	✕	$\mathbf{0.53 \times 10^6}$ (5)	**30.86** (5)	10 (5)
LSSVMKT	✕	$\mathbf{0.54 \times 10^6}$ (6)	**31.13** (6)	12 (6)

Ziegler–Nichols controller with RLSMadf were the second and third best controllers, respectively. The Deadbeat of third order controller with RLSM does not have a value in Table 2 because it is considered by the results (Fig. 7(d)) that this controller has not stabilized the output of the system in the reference value.

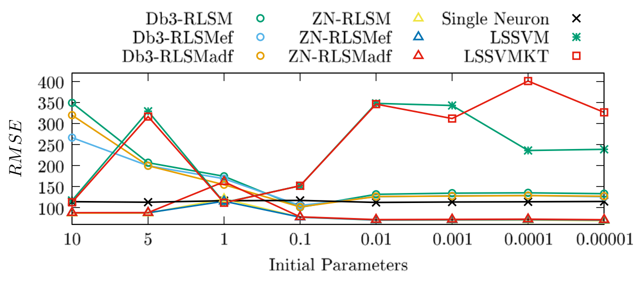
In order to study the performance of the initial response of the controllers for different initial parameters values ($\boldsymbol{\Theta}(k)$ for the explicit controllers and K_P, K_I, and K_D for the implicit controllers), Fig. 9 presents the $RMSE$ values of the initial response ($0 \leq k \leq 150$) of all control algorithms using the following different initial parameters values: 10, 5, 1, 0.1, 0.01, 0.001, 0.0001, and 0.00001. In Fig. 9(a) can be seen that Dahlin, Pole Placement, and Deadbeat of second order controllers' performance is not significantly influenced by low initial values (i.e., $\leq$ 0.1); and that for higher values than 0.1 these controllers have different behaviors: all combinations of Pole Placement controller and Deadbeat of second order controller using RLSMadf have significantly decreased their performance, and the remain controllers are not significantly influenced (note that all combinations of Dahlin controller presented better performance with higher initial values). In Fig. 9(b) can be seen that the Deadbeat of third order, and Ziegler–Nichols controllers' performance is not significantly influenced by low initial values (i.e., $\leq$ 0.1); and that for higher values than 0.1 all combinations of Deadbeat of third order controllers have significantly decreased their performance, and all combinations of Ziegler–Nichols controllers are not significantly influenced (however Ziegler–Nichols using RLSMadf decreased its performance slightly). Single Neuron controller's performance is very similar for all values; and the initial response of LSSVM and LSSVMKT controllers is very depending of the initial controller values. In conclusion, based on the results presented in Fig. 9 which may vary plant to plant, it is recommended choose low values (i.e., ≤ 0.1) for the initial model parameters ($\boldsymbol{\Theta}(k)$) of all explicit controllers and for the initial control parameters (K_P, K_I, and K_D) of Single Neuron controller; and for LSSVM and LSSVMKT controllers, that their initial responses showed be significantly influenced by the initial controller parameters, is recommended the 0.1 or 1 values.

4.2.2. *Tracking analysis*

Table 3 shows the numeric results of the tracking response of all the controllers on the instants $151 \leq k \leq 550$. In these instants, the reference suffered two changes as can be seen in Fig. 7. From Table 3, it is seen that the best tracking response is given by the Ziegler–Nichols controller



(a) Dahlin controller results using RLSM (Dahlin-RLSM), RLSMef (Dahlin-RLSMef), and RLSMadf (Dahlin-RLSMadf); Pole Placement controller results using RLSM (P.P.-RLSM), RLSMef (P.P.-RLSMef), and RLSMadf (P.P.-RLSMadf); and Deadbeat of second order controller results using RLSM (Db2-RLSM), RLSMef (Db2-RLSMef), and RLSMadf (Db2-RLSMadf).



(b) Deadbeat of third order controller results using RLSM (Db3-RLSM), RLSMef (Db3-RLSMef), and RLSMadf (Db3-RLSMadf); Ziegler–Nichols controller results using RLSM (ZN-RLSM), RLSMef (ZN-RLSMef), and RLSMadf (ZN-RLSMadf), Single Neuron, LSSVM, and LSSVMKT controllers results.

Fig. 9. $RMSE$ values of the initial response ($0 \le k \le 150$) of all control algorithms for different initial parameters values ($\Theta(k)$) of each explicit controller and K_P, K_I, and K_D of each implicit controller: 10, 5, 1, 0.1, 0.01, 0.001, 0.0001, and 0.00001.

Table 4
Statistical comparison between the control algorithms for disturbance rejection capability ($551 \le k \le 1150$). For each column, the numbers in parentheses (if any) represent the attributed score based on ranking on the criterion of that column. The values presented on the "Points" column are the sum of the scores attributed on the respective ITAE and RMSE indexes.

		ITAE	RMSE	Points
Dahlin	RLSM	3.78×10^6 (12)	51.93(12)	24 (11)
	RLSMef	$\mathbf{3.67 \times 10^6}$ (11)	**49.56** (11)	22 (10)
	RLSMadf	4.10×10^6 (13)	54.58(13)	26 (12)
Pole Placement	RLSM	$\mathbf{1.60 \times 10^6}$ (3)	**34.41** (7)	10 (4)
	RLSMef	1.82×10^6 (7)	37.65(10)	17 (8)
	RLSMadf	6.69×10^6 (16)	90.46(16)	32 (15)
Deadbeat 2nd	RLSM	13.39×10^6 (18)	95.10 (17)	35 (17)
	RLSMef	5.45×10^6 (14)	85.66 (15)	29 (13)
	RLSMadf	$\mathbf{1.47 \times 10^6}$ (2)	**25.86** (3)	**5**(**2**)
Deadbeat 3rd	RLSM	11.52×10^6 (17)	75.43(14)	31 (14)
	RLSMef	5.50×10^6 (15)	99.54(18)	33 (16)
	RLSMadf	$\mathbf{2.17 \times 10^6}$ (9)	**37.52** (9)	18 (9)
Ziegler–Nichols	RLSM	1.63×10^6 (5)	23.48(2)	**7**(3)
	RLSMef	2.35×10^6 (10)	26.42(4)	14 (6)
	RLSMadf	$\mathbf{1.01 \times 10^6}$ (1)	**22.60** (1)	**2** (**1**)
Single Neuron	✕	$\mathbf{1.83 \times 10^6}$ (8)	**36.18** (8)	16 (7)
LSSVM	✕	$\mathbf{1.81 \times 10^6}$ (6)	**33.41** (5)	11 (5)
LSSVMKT	✕	$\mathbf{1.63 \times 10^6}$ (4)	**33.43** (6)	10 (4)

using RLSMadf, with fast and stable responses to the reference changes. The Ziegler–Nichols controllers with RLSM and RLSMef were the second and third best controllers, respectively.

4.2.3. Load response analysis

Table 4 shows the numeric results of the disturbance rejection capability for all controllers for the instants $551 \le k \le 1150$. From Table 4, the best disturbance rejection capability is given by the Ziegler–Nichols controller using RLSMadf, having a fast and stable response. The Deadbeat controller of second order with RLSMadf and the Ziegler–Nichols with RLSMadf were the second and third best controllers, respectively. Note that the Ziegler–Nichols controller using RLSM and RLSMadf has shown a good response both in tracking of the reference with step changes (Section 4.2.2), and in the presence of disturbances (load).

4.2.4. Controllers set up

This section does an analysis of the simplicity of set up of the controllers. If an algorithm has many variables that need to be adjusted or if these variables need to be set up very accurately, then such algorithm will demand more effort from the controller installer than a simpler controller.

The simpler identification algorithm to be set up is the RLSM, meaning it is the simpler identification algorithm to install, having just the initial covariance matrix to be set up. The RLSMef has two variables that require to be set up, the initial gain covariance matrix, and the forgetting factor φ. The explicit identification algorithm RLSMadf has two variables that need to be set up, the initial covariance matrix, and the forgetting factor ρ. RLSMadf also has another two variables that need to be initialized (ν, and λ), but they are automatically tunned/adapted during online operation, thus significantly decreasing the importance on their specific initial values. In the RLSMef and RLSMadf neither of the variables that are required to be set up is much sensitive to its value and it is easy to choose values to obtain a satisfactory performance.

The deadbeat of second and third orders, and the Ziegler–Nichols controllers are the easier controllers to apply since, excluding the variables from the identification methods, they do not have variables to be tuned. The Dahlin and Single Neuron algorithms, both have a scale parameter to increase/decrease the responsiveness of the controller which is also easy to adjust. The Pole Placement controller has also two variables, the natural frequency ω_n and the damping factor ξ, to be tunned, which require some control engineering knowledge to be defined. LSSVM and LSSVMKT controllers are the most difficult to tune, due to their significant high number of parameters to be tunned (six parameters), and by the fact that the tunning of these parameters has shown, in the experimental tests, to be more sensitive requiring some understanding of these specific controllers.

4.2.5. General discussion

Table 5 summarizes all the previous results (Tables 1–4) and the total sum of the points obtained in respective tables. In Table 5, all the performance indexes are scores attributed in the previous Tables 1–4, and in the cases where controllers have the same total score (same summed scores), it was defined that the best controller is the one that has the largest number of best performance indexes. If the number of best performance indexes is the same, then it is assigned the same total score. As can be seen in Table 5, the best controller was the Ziegler–Nichols with the RLSMadf identification method, followed by the Ziegler–Nichols with RLSM, and Ziegler–Nichols with RLSMef. Note that the Ziegler–Nichols controller is the oldest controller (1942) presented in this paper, and surprisingly, when combined with adaptive the RLSM, RLSMef, and RLSMadf identification methods, it was the best controller even when compared with the more recent Single Neuron (2011), LSSVM (2008), and LSSVMKT (2011) controllers. Moreover, the Dahlin controller (also an old controller; year 1981) combined with adaptive RLSM, RLSMef, and RLSMadf identification methods had good results having smooth initial and tracking responses. The authors also consider that, if the overshoot is an important factor to take into account, then the Single Neuron controller can be a solution with a small percentage of overshoot (18.18%), or even the LSSVM with no overshoot can be a

Table 5
Final comparison of all controllers. The values presented on "Table 1", "Table 2", "Table 3", and "Table 4" columns are the scores obtained on the "Points" columns of Tables 1, 2, 3, and 4, respectively. The values presented on "Total" column are the sum of the scores attributed on the respective "Table 1", "Table 2", "Table 3", and "Table 4" columns. The numbers in parentheses represent the attributed score based on ranking on the "Total" criterion.

		Points				
		Table 1	Table 2	Table 3	Table 4	Total
Dahlin	RLSM	9	5	10	11	35 (10)
	RLSMef	7	**2**	12	10	31 (9)
	RLSMadf	8	8	14	12	42 (12)
Pole Placement	RLSM	11	15	10	4	40 (11)
	RLSMef	10	12	12	8	42 (12)
	RLSMadf	15	17	15	15	62 (16)
Deadbeat 2nd	RLSM	16	18	13	17	64 (17)
	RLSMef	12	16	9	13	50 (14)
	RLSMadf	**3**	14	4	**2**	23 (6)
Deadbeat 3rd	RLSM	13	13	11	14	51 (15)
	RLSMef	14	11	7	16	48 (13)
	RLSMadf	5	10	7	9	31 (8)
Ziegler–Nichols	RLSM	**2**	4	**2**	**3**	**11 (2)**
	RLSMef	4	7	**3**	6	**20 (3)**
	RLSMadf	**1**	**3**	**1**	**1**	**6 (1)**
Single Neuron	✗	4	**1**	8	7	20 (4)
LSSVM	✗	6	6	5	5	22 (5)
LSSVMKT	✗	6	9	6	4	25 (7)

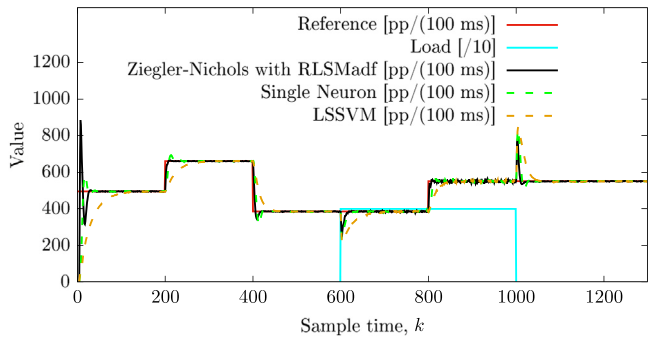


Fig. 10. Result of Ziegler–Nichols with RLSMadf, Single Neuron and LSSVM controllers.

solution. Fig. 10 shows the best controllers (considered by the authors) for the following scenarios: Ziegler–Nichols with RLSMadf for a fast and stable tracking response and disturbance rejection capability if the overshoot is not an important factor to take into account; Single Neuron for a fast and stable tracking response and disturbance rejection capability considering the possibility of appearance of small overshoots; and LSSVM for a smooth tracking response and disturbance rejection capability if it is considered that overshoots should not appear in the response.

The study presented in this paper does not consider the control saturation problem. However, there are some existing methods dealing this subject, as for example Sun, Fang, Chen, and Lu (2017), that can be considered for the controllers used on this paper.

5. Conclusions

In this paper, it was studied several Self-Tuning Controllers (STC) with Proportional Integral and Derivative (PID) form with the goal of applying them to control unknown plants and plants with variations in their dynamics. Explicit STC (also known as indirect STC), and implicit STC (also known as direct STC) were studied. All model parameters of each explicit controller and control parameters of each implicit controller were initialized with the value 0.1 to represent the absence of knowledge about the model of the plant to be controlled. These tests allowed to determine which algorithms have better control performance, which are easier to set up, which have a better initial response, and which enable faster reaction to plant variations and disturbances. The controllers were tested on a real setup composed of two coupled DC motors with a variable load in order to study the capability of the controllers to cope with unknown and time varying plants. The employed explicit identification methods were the base RLSM, RLSMef, and RLSMadf, and a comparison on their performance was presented. Analyses and comparisons regarding the control performance, and the simplicity of set up, of the presented controllers were made. Among the control algorithms, the one which performed better was the Ziegler–Nichols with the RLSMadf identification method. Besides having the best performance, Ziegler–Nichols is also very easy to set up to a good performance. Taking into account the overshoot, Single Neuron can be a solution because had a small percentage of overshoot (18.18%) or even LSSVM with no overshoot. It is worth to note that the Ziegler–Nichols, the oldest control algorithm analyzed in this paper, when combined with the presented identification methods, was evaluated as the best performing controller.

The control of a real system, with time-varying characteristics, and unknown assumed model, was performed using STCs with a PID structure. There are several algorithms that are able to perform this task, and it is up to the installer to choose which is the most suitable for the system at hands. The installer can choose characteristics like simplicity of set up, quick response to plant variations, no overshoot depending on the needs.

Acknowledgments

This work was supported by Project SCIAD "Self-Learning Industrial Control Systems Through Process Data" (reference: SCIAD/2011/21531) co-financed by QREN, in the framework of the "Mais Centro - Regional Operational Program of the Centro", and by the European Union through the European Regional Development Fund (ERDF). Jérôme Mendes was supported by Fundação para a Ciência e a Tecnologia (FCT) under grant SFRH/BPD/99708/2014.

References

Ajiboye, A. B., & Weir, R. F. F. (2005). A heuristic fuzzy logic approach to emg pattern recognition for multifunctional prosthesis control. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, *13*, 280–291.

Åström, K. J., & Hägglund, T. (2001). The future of PID control. *Control Engineering Practice*, *9*, 1163–1175.

Astrom, K. J., & Wittenmark, B. (1994). *Adaptive control* (2nd ed.). Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc..

Bashivan, P., & Fatehi, A. (2012). Improved switching for multiple model adaptive controller in noisy environment. *Journal of Process Control*, *22*, 390–396.

Bittanti, S., Bolzern, P., & Campi, M. (1989). Adaptive identification via prediction-error directional-forgetting factor: Convergence analysis. *International Journal of Control*, *50*, 2407–2421.

Bobál, V., Böhm, J., Fessl, J., & Macháček, J. (2005). *Digital self-tuning controllers: Algorithms, implementation and applications*. London, UK: Springer.

Corripio, A. B., & Tomkins, P. M. (1981). Industrial application of self-tuning feedback control algorithm. *ISA Transactions*, *20*, 3–10.

Fileti, A., Antunes, A., Silva, F., Jr, V. S., & Pereira, J. (2007). Experimental investigations on fuzzy logic for process control. *Control Engineering Practice*, *15*, 1149–1160.

Kolavennu, P. K., Palanki, S., Cartes, D. A., & Telotte, J. C. (2008). Adaptive controller for tracking power profile in a fuel cell powered automobile. *Journal of Process Control*, *18*, 558–567.

Kučera, V. (1980). A dead-beat servo problem. *International Journal of Control*, *32*, 107–113.

Lim, Y., Venugopal, R., & Ulsoy, A. G. (2012). Auto-tuning and adaptive control of sheet metal forming. *Control Engineering Practice*, *20*, 156–164.

Osório, L., Mendes, J., Araújo, R., & Matias, T. (2013). A comparison of adaptive PID methodologies controlling a DC motor with a varying load. In *Proc. 18th IEEE international conference on emerging technologies and factory automation*. ETFA 2013, (pp. 1–8). Cagliari, Italy: IEEE.

Panzani, G., Corno, M., & Savaresi, S. M. (2013). On adaptive electronic throttle control for sport motorcycles. *Control Engineering Practice*, *21*, 42–53.

Psichogios, D. C., & Ungar, L. H. (1991). Direct and indirect model based control using artificial neural networks. *Industrial and Engineering Chemistry Research*, *30*, 2564–2573.

Sun, N., Fang, Y., Chen, H., & Lu, B. (2017). Amplitude-saturated nonlinear output feedback antiswing control for underactuated cranes with double-pendulum cargo dynamics. *IEEE Transactions on Industrial Electronics*, *64*, 2135–2146.

Ucak, K., & Oke, G. (2011). Adaptive PID controller based on online LSSVR with kernel tuning. In *2011 international symposium on innovations in intelligent systems and applications, INISTA 2011* (pp. 241–247).

Wanfeng, S., Shengdun, Z., & Yajing, S. (2008). Adaptive PID controller based on online lssvm identification. In *IEEE/ASME international conference on advanced intelligent mechatronics, AIM 2008* (pp. 694–698).

Wang, M., Cheng, G., & Kong, X. (2011). A single neuron self-adaptive PID controller of brushless DC motor. In *2011 third international conference on measuring technology and mechatronics automation, ICMTMA, Vol. 1* (pp. 262–266).

Wittenmark, B. (1979). *Self-tuning PID-controllers based on pole placement*. Department of Automatic Control, Lund Institute of Technology.

Yildiz, Y., Annaswamy, A. M., Yanakiev, D., & Kolmanovsky, I. (2010). Spark ignition engine fuel-to-air ratio control: An adaptive control approach. *Control Engineering Practice*, *18*, 1369–1378.

Zhang, J., Ding, F., & Shi, Y. (2009). Self-tuning control based on multi-innovation stochastic gradient parameter estimation. *Systems & Control Letters*, *58*, 69–75.

Ziegler, J. G., & Nichols, N. B. (1942). Optimum settings for automatic controllers. *Transactions of ASME*, *64*, 759–768.