A Comparison of Adaptive PID Methodologies Controlling a DC Motor With a Varying Load

Luís Osório, Jérôme Mendes, Rui Araújo, and Tiago Matias
Institute for Systems and Robotics (ISR-UC), and Department of Electrical and Computer Engineering (DEEC-UC), University of Coimbra, Pólo II, PT-3030-290 Coimbra
lbica@isr.uc.pt, jermendes@isr.uc.pt, rui@isr.uc.pt, tmatias@isr.uc.pt

Abstract

This work addresses the problem of controlling unknown and time varying plants for industrial applications. To deal with such problem several Self-Tuning Controllers with a Proportional Integral and Derivative (PID) structure have been chosen. The selected controllers are based on different methodologies, and some use implicit identification techniques (Single Neuron and Support Vector Machine) while the others use explicit identification (Dahlin, Pole placement, Deadbeat and Ziegler-Nichols) based in the Least Squares Method. The controllers were tested on a real DC motor with a varying load. The results have shown that all the tested methods were able to properly control an unknown plant with varying dynamics.

1 Introduction

Because of its simplicity and good performance, the Proportional Integral and Derivative (PID) controller is by far the most popular feedback controller in the automatic control field. In industrial processes the classical PID controller was employed in about 90% or more of control loops [2]. Generally, engineers tune the optimal parameters of a PID controller to match the operating condition and such parameters remain fixed during the whole operation [14]. The problem when using fixed parameter controllers is that most of the processes met in industrial practice have dynamics that are not modeled or that can change over time. In such cases, the classical controller with fixed parameters may became unstable and would be required to be adequately re-tuned to retain robust control performance. To overcome this difficulty, adaptive algorithms were developed, which extends the area of real situations in which high quality control can be achieved.

According to Bobal et al. [5] the development of adaptive control started in the 1950s with simple analogue techniques since the computing equipment had not the required performance to execute the most sophisticated algorithms that were already proven in theory. Later in the 1980s, as the microprocessors became faster and cheaper, it evolved to discrete-time control and the theory developed in the early years was finally be applied. At the present there is yet much unused potential in mass applications and there are still opportunities for improvements, for streamlining in the areas of theory and application, and for increasing reliability and robustness [3]. The work of Kolavennu et al. [6] shows that in many real-world processes where a nonadaptive controller is sufficient, an adaptive controller can achieve an even better quality of control. Other example is given in [12] where the use of an adaptive controller decreased fuel consumption significantly.

Adaptive controllers follow three basic approaches: the Model Reference Adaptive Systems (MRAS), the Heuristic Approach (HA), and the Self-Tuning Controllers (STC). The MRAS controllers use one or multiple system models to determine the difference between the output of the adjustable system and the output of a reference model, and adjust the parameters of the adjustable system or generate a suitable input signal [4]. The methods based on HA do not require determining the optimum solution of a problem, ignoring whether the solution can be proven to be correct, provided that it produces a good result. Such methods are based on expert human experience [1]. STC are based on the recursive estimation of the characteristics of the system. Once the system is determined, appropriate methods can be employed to design an adequate controller [11].

The main objective of this work is to test PID algorithms that can get close to the concept of “plug and play” (algorithms that do not require information about the plant to be controlled and must be able to auto-adapt their control parameters taking in account the variations of the plant). Controllers based on MRAS require the knowledge of an approximate model of the plant to control, and HA controllers are experience-based techniques for learning the control laws, meaning that both these approaches require previous information about the plant. Thus, only controllers based in STC will be considered.


To compare the performance of the control algorithms a real experimental setup composed of two coupled DC
motors with varying load, was built and used. The paper is organized as follows. Section 2 presents the algorithms used to perform the identification and the control of the plants. Section 3 is dedicated to the analysis and discussion of the results. Finally, section 4 makes concluding remarks.

2 STC Methodologies

STC algorithms can be divided into two categories. If the identification is explicit then controllers that use the transfer function to determine the gains of the controller can be applied. This means that the identification algorithm and the controller algorithm can be chosen independently. On the other hand, implicit controllers do not translate the identification to determine the gains of the controller can be applied. This means that the identification algorithm is explicit then controllers that use the transfer function to determine the gains have been created specifically to the output of that identification algorithm. The advantage of implicit algorithms is that they require less processor time.

In this paper, the controller must be created specifically to the output of that identification algorithm. The advantage of implicit algorithms is that they require less processor time.

2.1 Explicit Identification for STCs

When using explicit STCs, it is necessary to estimate the plant’s transfer function in real time. If this is performed recursively it allows the model of the plant to adapt whenever the real plant’s dynamics change. In [5] the LSM identification algorithm with adaptive directional forgetting (LSMadf) is presented, which uses a forgetting factor that is automatically adjusted depending on the changes of the input and output signals.

The methods based on LSM perform discrete on-line explicit identification of a plant producing a transfer function of the form

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

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. Thus,

\[ A(z^{-1}) y(k) = B(z^{-1}) u(k), \]  

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 identified plant is given by

\[ \hat{y}(k) = \Theta^T(k-1) \Phi(k) = \hat{a}_1 y(k-1) - \hat{a}_n y(k-n) + b_1 u(k-d-1) + \ldots + b_m u(k-d-m), \]  

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

- **Least Squares Method With Adaptive Directional Forgetting [5]**:

The LSMadf is an evolved form of LSM where a forgetting factor is used to give less weight to older data, and this forgetting factor is automatically updated at each iteration. In this method the vector of parameter estimations is updated at each iteration, \( k \), using equation (4)

\[ \Theta(k) = \Theta(k-1) + \frac{C(k-1) \Phi(k)}{1 + \xi} (y(k) - \Theta(k-1)^T \Phi(k)), \]  

where \( \xi = \Phi(k)^T C(k-1) \Phi(k) \), and \( C(k) \) is the covariance matrix of the regression vector \( \Phi(k) \) which is updated at each iteration, \( k \), using equation (5)

\[ C(k) = C(k-1) - \frac{C(k-1) \Phi(k) \Phi(k)^T C(k-1)}{\varepsilon - 1 + \xi}, \]  

where \( \varepsilon = \varphi(k-1) - \frac{\varepsilon(k-1)}{\xi} \) and \( \varphi(k-1) \) is the forgetting factor at iteration \( k-1 \).

The adaptation of \( \varphi \) is performed as follows:

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

where \( \nu(k) = \varphi(k - 1) \nu(k - 1) + 1 \), \( \eta = \frac{\varphi(k - 1) \nu(k - 1)}{1 + \xi} \), \( \lambda(k) = \varphi(k - 1) \left[ \lambda(k-1) + \frac{(y(k) - \Theta(k-1)^T \Phi(k))^2}{1 + \xi} \right] \), and \( \rho \) is positive constant.

In LSMadf, the forgetting factor \( \varphi(k) \) and the variables \( \lambda(k) \) and \( \nu(k) \) are automatically adjusted, so the initial values of these variables do not have much impact in the identification process. In any case, they should be set between zero and one.

2.2 Control Algorithms for Explicit Identification

A brief overview of the five tested STC controllers is presented in the following items:

- **Dahlin PID Controller [8]**:

This algorithm is based on a transfer function with the form of (1) with \( n = 2 \) and \( m = 1 \). Thus, the estimation vector is \( \Theta(k-1) = [\hat{a}_1, \hat{a}_2, \hat{b}_1]^T \) and the regression vector is \( \Phi(k) = [-y(k-1), -y(k-2), \hat{u}(k-1)]^T \). The control law of the Dahlin’s algorithm is given by

\[ u(k) = K_P + \left\{ e(k) - e(k-1) + \frac{T_I}{T_D} e(k) + \frac{T_D}{T_I} \right\} u(k-1), \]  

where \( T_0 \) is the sampling interval, and \( K_P, T_I, T_D \) are the proportional gain, the integral time constant, and the differential time constant, respectively, which depend of the model parameters as follows:

\[ K_P = \frac{\lambda_1 + 2\lambda_2 Q}{b_1}, \]  

\[ T_I = -\frac{T_0}{\lambda_1 + 2\lambda_2 + 1 + \frac{T_D}{T_I}}, \]  

\[ T_D = \frac{T_D \lambda_1 \lambda_2 Q}{K_P b_1}, \]  

where \( Q = 1 - e^{-\frac{T_D}{T_I}} \) and \( B \) is a positive constant. In this algorithm, \( B \) is an adjustment factor that specifies the dominant time constant of the transfer function according to changes made to the process output of a closed control loop. The smaller the \( B \) gets, the quicker the step response of the closed control loop becomes.
• Pole Placement [13]:

This Pole Placement algorithm requires that the user adjusts the natural frequency ($\omega_n$) and damping factor ($\xi$) to control a second order plant with $n = 2$ and $m = 2$ which means that this algorithm’s estimation vector is $\Theta(k-1) = [a_1, a_2, b_1, b_2]^T$ and the regression vector is $\Phi(k) = [-y(k-1), -y(k-2), u(k-1), u(k-2)]^T$.

The control law is given by

$$u(k) = q_0 e(k) + q_1 e(k-1) + q_2 e(k-2) + (1 - \gamma)u(k-1) + \gamma u(k-2), \quad (11)$$

where the coefficients $q_0$, $q_1$ and $q_2$ can be calculated by

$$q_0 = \frac{1}{b_1} (d_1 + 1 - a_1 - \gamma), \quad (12)$$

$$q_1 = \frac{a_2}{b_2} - q_2 \left( \frac{b_1}{b_2} - \frac{a_1}{a_2} + 1 \right), \quad (13)$$

$$q_2 = \frac{a_1}{r_1}, \quad (14)$$

where

$$d_1 = \begin{cases} -2e^{-\xi \omega_n T_0} \cos(\omega_n T_0 \sqrt{1 - \xi^2}), & \text{if } \xi \leq 1, \\ -2e^{-\xi \omega_n T_0} \cosh(\omega_n T_0 \sqrt{\xi^2 - 1}), & \text{if } \xi > 1, \end{cases} \quad (15)$$

$$d_2 = e^{-2\xi \omega_n T_0}, \quad (16)$$

$$r_1 = (b_1 + b_2)(a_1 b_1 b_2 - a_3 b_2^2 - b_3^2), \quad (17)$$

$$s_1 = a_2[(b_1 + b_2)(a_1 b_2 - a_3 b_1) + b_2(b_1 d_2 - b_2 d_1 - b_3)], \quad (18)$$

$$\gamma = \frac{q_2}{a_2}, \quad (19)$$

and $T_0$ is the sampling interval.

• Deadbeat Controller of Second Order (DB2) [7]:

This controller is based on a second order plant with $n = 2$ and $m = 2$ which means that this algorithm’s estimation vector is $\Theta(k-1) = [a_1, a_2, b_1, b_2]^T$ and the regression vector is $\Phi(k) = [-y(k-1), -y(k-2), u(k-1), u(k-2)]^T$. The control law is given by

$$u(k) = r_0 r(k) - q_0 y(k) - q_1 y(k-1) - p_1 u(k-1). \quad (20)$$

where the controller’s coefficients $q_0$, $q_1$ and $p_1$ are given by

$$\begin{bmatrix} p_1 \\ q_0 \\ q_1 \\ p_1 \end{bmatrix} = \begin{bmatrix} 1 & b_1 & 0 & 0 \\ a_1 & b_2 & b_1 & 0 \\ a_2 & 0 & b_2 & 0 \\ -a_1 & -a_2 & 0 & 0 \end{bmatrix}^{-1} \begin{bmatrix} -a_1 \\ -a_2 \\ 0 \end{bmatrix}, \quad (21)$$

and $r_0 = 1/(b_1 + b_2)$.

• Deadbeat Controller of Third Order (DB3) [7]:

For Deadbeat control on a third order system with $n = 3$ and $m = 3$, the estimation vector is $\Theta(k-1) = [a_1, a_2, a_3, b_1, b_2, b_3]^T$, and the regression vector is $\Phi(k) = [-y(k-1), -y(k-2), -y(k-3), u(k-1), u(k-2), u(k-3)]^T$. The control law is given by

$$u(k) = r_0 r(k) - q_0 y(k) - q_1 y(k-1) - q_2 y(k-2) - p_1 u(k-1) - p_2 u(k-2). \quad (22)$$

where the controller’s coefficients $p_1, p_2, q_0, q_1$ and $q_2$ are given by

$$\begin{bmatrix} p_1 \\ p_2 \\ q_0 \\ q_1 \\ q_2 \end{bmatrix} = \begin{bmatrix} 1 & b_1 & 0 & 0 & 0 \\ a_1 & b_2 & b_1 & 0 & 0 \\ a_2 & a_3 & b_3 & b_2 & b_1 \\ a_3 & 0 & b_3 & b_2 & b_1 \\ 0 & a_3 & 0 & b_3 & 0 \end{bmatrix}^{-1} \begin{bmatrix} -a_1 \\ -a_2 \\ -a_3 \\ 0 \end{bmatrix}, \quad (23)$$

and $r_0 = 1/(b_1 + b_2 + b_3)$.

• Ziegler-Nichols with Forward Rectangular Discretization (ZN) [14]:

The experimental tuning of parameters for a continuous-time PID controller designed by Ziegler and Nichols 70 years ago is still a good option. The algorithm is based on a third order system with $n = 3$ and $m = 3$. Thus, the estimation vector is $\Theta(k-1) = [a_1, a_2, a_3, b_1, b_2, b_3]^T$ and the regression vector is $\Phi(k) = [-y(k-1), -y(k-2), -y(k-3), u(k-1), u(k-2), u(k-3)]^T$. The control law is given by

$$u(k) = q_0 e(k) + q_1 e(k-1) + q_2 e(k-2) + u(k-1), \quad (24)$$

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

$$q_0 = K_p \left( 1 + \frac{T_0}{T_1} + \frac{T_D}{T_0} \right), \quad (25)$$

$$q_1 = -K_p \left( 1 + \frac{T_D}{T_0} \right), \quad (26)$$

$$q_2 = K_p \frac{T_D}{T_0}, \quad (27)$$

where the proportional gain is $K_p = 0.6 K_{pr}$, the integral time constant is $T_I = 0.5 T_i$, and the differential time constant is $T_D = 0.125 T_i$. This is a Ziegler-Nichols based algorithm, thus it is required to determine the ultimate proportional gain $K_{pr}$ and the ultimate period of oscillations $T_u$. Figure 1 explains how these parameters can be calculated.

2.3 Implicit STC

A brief overview of the three implicit STC controllers tested is presented in the following sections.

• Single Neuron (SN) [11]:

The Single Neuron algorithm here described is a self adaptive PID controller that has a simple structure and requires few computation effort. The control law is given by

$$u(k) = u(k-1) + K_p x_1(k) + K_I x_2(k) + K_D x_3(k), \quad (28)$$

where

$$x_1(k) = e(k), \quad x_2(k) = \Delta e(k), \quad x_3(k) = \Delta^2 e(k). \quad (29)$$

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

$$K_p = K \pi_i(k), \quad K_I = K \pi_i(k), \quad K_D = K \pi_i(k), \quad (30)$$

where $K$ is a positive scale parameter that can be increased/decreased to adjust the responsiveness of the controller. The coefficients $\pi_i(k)$ are given by

$$\pi_i(k) = \frac{u_i(k)}{\sum_{i=1}^{n} |u_i(k)|}, \quad (31)$$
and are obtained through normalization of the weight coefficients
\[ w_i(k) = w_i(k-1) + \eta_i K e(k)x_i(k-1)\operatorname{sgn}(\frac{\partial y(k)}{\partial i^*(k)}), \]
where \( \eta_i \) is the learning rate of the weight coefficient \( w_i(k) \), and \( \operatorname{sgn}(\cdot) \) is a signal function. The current reference of the single neuron \( i^*(k) \) is given by
\[ i^*(k) = i^*(k-1) + K \sum_{i=1}^{3} \tilde{w}_i(k)x_i(k). \]
and \( \frac{\partial y(k)}{\partial i^*(k)} = (y(k) - y(k-1))/(i^*(k) - i^*(k-1)). \)

- Least Squares Support Vector Machine [10]:
In the Least Squares Support Vector Machine (LSSVM) adaptive PID Controller, the PID parameters are adjusted using the gradient information of LSSVM to perform online implicit identification. The control law of this method is given by
\[ u(k) = u(k-1) + K_P x_1(k) + K_I x_2(k) + K_D x_3(k), \]
where,
\[ x_1(k) = \Delta e(k), \quad x_2(k) = e(k), \quad x_3(k) = \Delta^2 e(k). \]
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
\[ K_P(k+1) = K_P(k) + \Delta K_P(k), \]
\[ K_I(k+1) = K_I(k) + \Delta K_I(k), \]
\[ K_D(k+1) = K_D(k) + \Delta K_D(k), \]
where
\[ \Delta K_P(k) = \eta e(k) \frac{\partial y(k)}{\partial u}(k)x_1(k), \]
\[ \Delta K_I(k) = \eta e(k) \frac{\partial y(k)}{\partial u}(k)x_2(k), \]
\[ \Delta K_D(k) = \eta e(k) \frac{\partial y(k)}{\partial u}(k)x_3(k), \]
where \( 0 < \eta < 1 \) is the learning rate,
\[ \frac{\partial y}{\partial u}(k) = \sum_{i=k-L}^{k-1} \alpha_i(k) (u(k) - x_{i+1}(k)) K(x(k), x(i)) \]
where \( L \) is the size of the sliding window,
\[ K(x(i), x(j)) = \exp \left( -\frac{||x(i) - x(j)||^2}{\sigma^2} \right), \]
is the RBF used in the kernel function of the LSSVM, and \( \sigma \) is the bandwidth of the RBF,
\[ x(k) = [u(k), ..., u(k-m), y(k), ..., y(k-n)]^T, \]
and
\[ \alpha(k) = U(k)(Y(k) - 1, b(k)), \]
where \( \alpha_i(k) \) is the \( i \)-th element of vector \( \alpha(k) \), and \( x_{i+1}(k) \) is the \( (i+1) \)-th element of vector \( x(k) \),
\[ b(k) = 1^T U(k) Y(k), \]
where \( 1_v = [1, ..., 1]_1 \times L, Y(k) = [y(k), ..., y(k-L + 1)]^T \),
\[ U(k) = \begin{bmatrix} A(k) & H^T \\ h \end{bmatrix}^{-1}, \]
\[ H = [K(x(k-L), x(k-1)), ..., K(x(k-L), x(k-L+1))] \]
\[ A(k) = K(x(k-L), x(k-L+1)) \]
where \( h = K(x(k-L), x(k-L)) + C^{-1} \), and \( A(k) \) is given by (54). \( C \) is a positive regularization factor, and if its value is low, then the outlier points are deemphasized.

- Least Squares Support Vector Machine with Kernel Tuning [9]:
The Least Squares Support Vector Machine with Kernel Tuning (LSSVMKT) adaptive PID controller is an evolution of the LSSVM controller. The main difference is the ability to adjust the LSSVMKT kernel bandwidth (\( \sigma \)) as follows:
\[ \sigma(k+1) = \sigma(k) + \Delta \sigma(k), \]
\[ \Delta \sigma(k) = \eta(k) \hat{e}_m(k) \frac{\partial y(k)}{\partial \sigma(k)}, \]
\[ \frac{\partial y(k)}{\partial \sigma(k)} = \sum_{i=k-L}^{k-1} \left\{ \frac{\alpha_i(k) K(x(k), x(i)) (x(k) - x(i))}{\sigma(k)^3} - x(i)^T (x(k) - x(i)) \right\}, \]
\[ \hat{e}_m(k) = y(k) - \hat{y}(k), \]
\[ \hat{y}(k+1) = \sum_{i=k-L}^{k-1} \alpha_i(k) K(x(k), x(i)) + b(k). \]

3 Results and Discussion
This section discusses the results obtained when the adaptive algorithms were set to control a real plant. The performances of the controllers are compared using four different statistical indices, the Integral Absolute Error (IAE), the Integral Time-weighted Absolute Error (ITAE),
\[ A(k) = \begin{bmatrix} K(x(k - 1), x(k - 1)) + C^{-1} & \cdots & K(x(k - L + 1), x(k - 1)) \\ \vdots & \ddots & \vdots \\ K(x(k - 1), x(k - L + 1)) & \cdots & K(x(k - L + 1), x(k - L + 1)) + C^{-1} \end{bmatrix} \]

(54)

\[ IAE = \sum_{k=1}^{N} |e(k)|, \quad ITAE = \sum_{k=1}^{N} k|e(k)|, \]

\[ ISE = \sum_{k=1}^{N} e(k)^2, \quad RMS = \sqrt{\frac{\sum_{k=1}^{N} e(k)^2}{N}}, \]

(54)

where \( N \) is the number of samples (time instants).

3.1 Plant

A system composed of two motors, a shaft coupler, a motor driver, a relay, two lamps, a programmable logic controller (PLC), a computer (running Scilab) and a power source was used to test the control algorithms. The computer and the PLC were connected using the OPC (OLE (Object Linking and Embedding) for Process Control) communication protocol. Figure 2 outlines the connections between all the components of the setup. One of the motors receives command signals, and the other works as a generator. The control signal can be varied in the interval from 0 to 100 (percentage), which corresponds to a variation from 0 to 12 Volts. The lamps are connected to the terminals of the generator and since they consume energy, they increase its load. The relay is used to turn on/off the lamps/load. The tests consisted of running all the control algorithms during 100 seconds with a sampling interval of 250 milliseconds. The motor always started in rest and was set to achieve a reference speed of 100 \( \text{pp/(0.25 seg)} \) (pulses per 250 milliseconds). After 20 seconds the reference speed changed to 120 \( \text{pp/(0.25 seg)} \), and at 60 seconds it changed again to 90 \( \text{pp/(0.25 seg)} \). The relay was turned on at 40 seconds (increasing the load of the generator), and was turned off at 80 seconds.

3.2 Control Algorithms Comparison

Figure 3 shows the output speed of the real DC motor under the control of the studied control algorithms. It shows that all the controllers were able to properly follow reference changes and that they were able to compensate variations on the load of the motor. Since all the controllers performed similarly the IAE, ITAE, ISE, RMS numerical indices, eqs. (54), were used to compare the controllers performances.

Table 1 presents the results of the application of these indices for all control algorithms. Each controller received a score for each numerical index based on its performance (the best received 1 and the worst received 8) and the best controller was the one which summed least points. With just 5 points, the Deadbeat controller of second order achieved the best score. Figures 4(a) and 4(b) shown the
results of the Deadbeat controller of second order. Figure 4(a) shows how the output of the plant and the control signal change when the reference changes, and when a variation on the motor load is introduced. Figure 4(b) shows the time evolution of the plant’s estimated parameters.

Besides controller performance, simplicity of tuning is another important feature that was pursued. The explicit identification algorithms LSMadf have two variables that need to be tuned, the initial gain of the covariance matrix, and the forgetting factor \( \rho \). Neither of them is much sensitive and a satisfactory tuning of these variables is easy to obtain. The Deadbeat algorithms (of second and third orders) and Ziegler-Nichols do not have any variable to be adjusted (obviously the variables from the explicit identifications still need to be adjusted), which means they are easier to install. 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 algorithm has two variables that need to be adjusted, the natural frequency \( \omega_n \), and the damping factor \( \xi \), which makes it a bit more challenging for the installer. The algorithms LSSVM and LSSVMKT revealed to be the most difficult to adjust. Not only both algorithms have six variables that need to be adjusted (which means that the installer needs to have a deeper understanding of the controller) but the calibration of these variables also revealed to be more sensitive and difficult.

4 Conclusions

In this work, several adaptive PID controllers, STCs with a PID structure, that can be used to control unknown plants in industry were tested and compared. The controllers were tested on a real DC motor with a varying load, and their performance was mathematically analyzed. The tested algorithms were STCs with either implicit or explicit identification (the later requiring independent identification algorithms). The employed explicit identification method was the LSMadf, and had a good performance. Among the control algorithms, the one which performed better was the Deadbeat of second order, followed by the Dahlin’s controller, and the third best was the LSSVRKT. Besides having the best performance, the Deadbeat of second order and Dahlin, were also very easy to tune to a satisfactory performance. The LSSVMKT was much more difficult to tune.

Acknowledgment

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).

References


