25th ABCM International Congress of Mechanical Engineering
October 20-25, 2019, Uberlândia, MG, Brazil

COBEM2019-0203
ADAPTIVE CONTROL APPLIED TO TRAJECTORY TRACKING OF AN AUTONOMOUS ROBOT

Arthur Henrique Iasbeck
Murilo Mendonça Venâncio
Rogério Sales Gonçalves
Federal University of Uberlândia (UFU) - Av. João Naves de Ávila, 2121 - Santa Mônica, Uberlândia - MG, 38408-100
arthuriasbeck@gmail.com | murilomend92@gmail.com | rsgoncalves@ufu.br

Abstract. *This paper aims the design of a trajectory controller for an autonomous robot through the implementation of STR (Self Tuning Regulator) control, which is an adaptive control approach. The adopted STR was designed based on the PP (Pole Placement) method, and the online parameter estimation in RLS (Recursive Least Squares) method. The loop control behavior was evaluated by both simulations and real tests. The results have indicated that the desired trajectory were tracked accordingly with the requirements for overshoot and settling time.*

Keywords: *Self Tuning Regulator, Adaptive Control, Trajectory Control, Pole Placement,Least Mean Squares*

1. INTRODUCTION

When an autonomous robot moves around a dynamic and unknown environment, there is a need for avoiding obstacles and handling unforeseen situations. Furthermore, physical limitations, actuators quality, sensors and the materials that composes the robot's structure shall be considered during the control system design. Therefore, it is desirable that the system is robust enough for handling the possible perturbations and non modeled behaviors.

This paper has as main objective the trajectory control of an autonomous robot by implementing an approach in which the controller has its parameters adjusted automatically during an online operation. This system shall guarantee a suitable tracking for a straight line as reference, even with disturbances to the system. The implementation of a compass sensor and an adaptative controller allows the robot to be guided.

The Adaptive Control is a subarea of Systems Control field which studies control designs capable to handle unknown processes and systems that varies their intrinsic characteristics over time. As the name suggests, adaptive controllers adapt their behaviour according to changes both in the controlled system dynamics and in the acting perturbation (Astrom and Wittenmark, 1995).

Adaptive controlling approaches were developed for handling processes that could not be controlled by implementing constant gain controllers. The first applications in this area appeared in the 50's, when NASA® associated researchers began the development of *autopilots* for high-performance airplanes, e.g. the X-15, and they realized that fixed gain controllers could only guarantee controllability for constant flight conditions (such as airplane velocity and its altitude) (Astrom and Wittenmark, 1995). Therefore, the first adaptive controllers were proposed for keeping the airplane control even with a dynamic environment.

Since a digital compass was used to determine the robot's orientation, a rectilinear trajectory was adopted to generate the results presented in this paper, since, in order to trace it, it would only be necessary for the readings of the compass (system output) to follow a constant reference. However, so that both the system dynamics and the performance requirements could be evaluated, a square wave of amplitude equal to 80^o instead of a single step was used as a reference.

The present work was organized as follows: in section 2 the system to be controlled is presented and its components detailed. The control scheme employed is presented in section 3. The results obtained, both through simulations and experiments using the real robot, are presented in section 4. Finally, section 5 presents the conclusions and some proposals for future work.

2. ANALYSIS OF THE CONTROLLED SYSTEM

The robot used for tests, identified as *follower* (Fig. 1) throughout this paper, was made with LEGO Mindstorms NXT® kit. It is basically assembled with standard structures, such as bars, wheels and rods. A Programmable Logic Controller (PLC) was used for managing sensor readings and servomotor actuation.

A compass sensor attached to the robot provides it orientation data. The value obtained from this device indicates

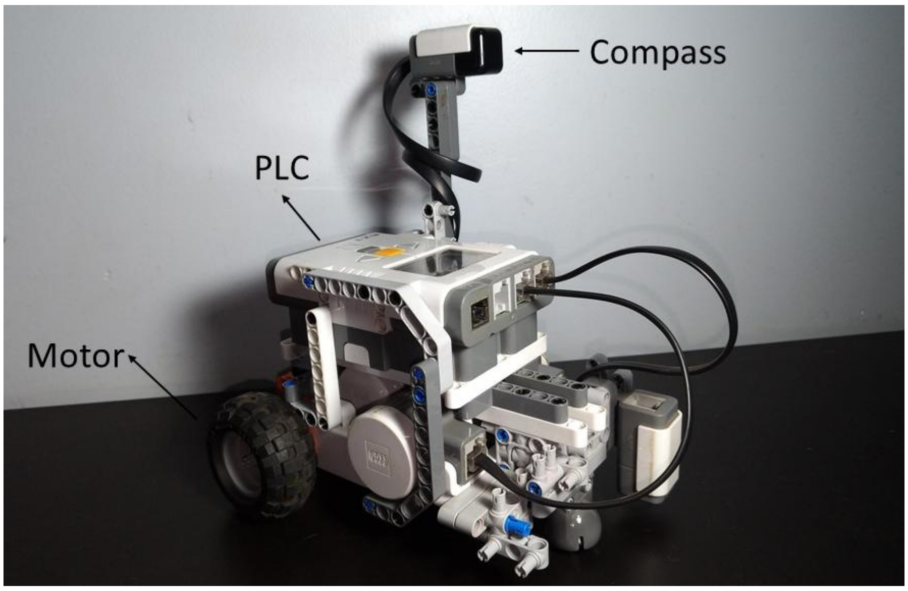

Figure 1. Autonomous robot which will follow a desired trajectory with the proposed controller in this paper.

relative angular position to the earth north. Ideally, when the sensor reference axis is aligned with north, it outputs a zero value. When the sensor is rotated of α^{o}, the outputted value will be equal to α (HiTechnic, 2018).

The PLC is responsible for processing sensor data, applying the designed control logic, and actuating the servomotors according to the desired values from control output. The embedded logic for the PLC is based in Java language (leJOS, 2019).

The servomotors provide mobility to the robot. The developed software determines the fraction of the maximum supplied power to be delivered to the servomotors. A function *setPower(int power)* is responsible for this task, where the integer value *power* defines the percentage from the maximum torque to be delivered. For a null value of *power*, there is no rotation, and for a value of 100 the rotation reaches its maximum velocity and power.

The *power* variable is a system input and should be used for controlling the system behavior. The data from the compass sensor, α, is the system output. Moreover, the *follower* has two inputs, since there are two servomotors to be controlled. However, in order to represent the system as a SISO (Single Input Single Output), the input will be considered as the difference between the *power* of both servomotors. Consequently, the *power* value for each one will be calculated as $K + u(t)$ for the left motor and $K - u(t)$ for the rigth one. The K value is a constant and $u(t)$ is the input signal (control action).

Thus, when $u(t) = 0$, both servo motors rotate with the same electrical power (K) and the *follower* moves forward. The *follower* turns to the right if $u(t) < 0$, and to the left if $u(t) > 0$, since the variation of $u(t)$ interferes the rotation speed of a servomotor in relation to another one. In addition, the intensity of the curve performed by the *follower* is directly proportional to the amplitude of $u(t)$.

3. PROPOSED CONTROLLER

3.1 Self Tuning Regulator

The proposed adaptive controller for this paper (Fig. 2) is composed of two *loops*. An internal one, in which it is implemented the closed loop control, and an external one, where the system and controllers parameters are updated. A recursive parameter estimator and a tuning controller block, allow to automatically model and design the control system, and, for this reason, this method is known as *Self Tuning Regulator*, or STR (Astrom and Wittenmark, 1995).

Both process and controller parameters are updated at each time sample, and the controller is tuned to perform a desired closed loop behavior (Astrom and Wittenmark, 1995).

Several methods are available in literature for estimation of the process parameters and for the controller design. In this paper, a combination of RLS and PP was the choice for achieving an adaptive behavior to the system control, as suggested by Astrom and Wittenmark (1995).

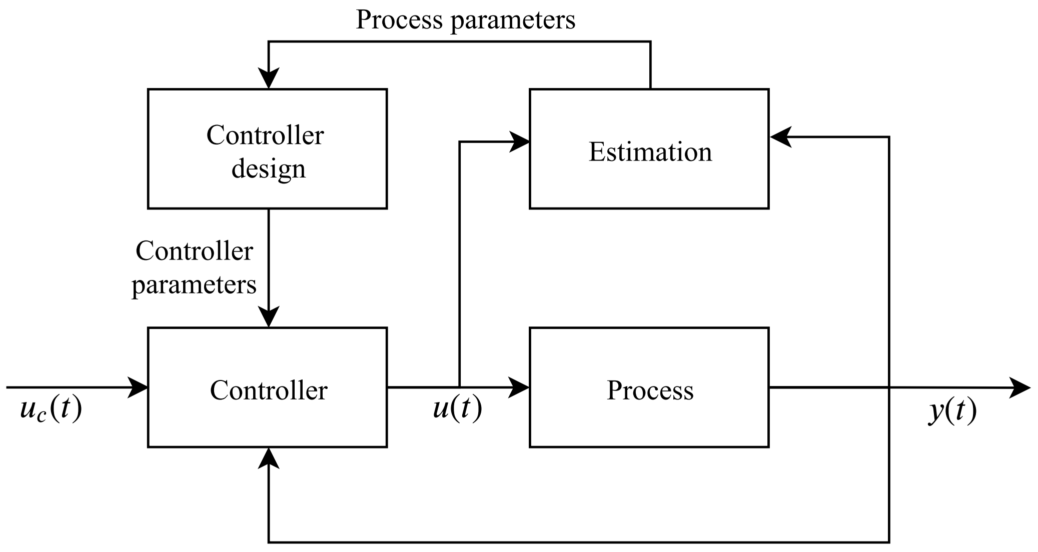

Figure 2. Control design block diagram for illustrating the STR structure. The reference signal is denoted by $u_c(t)$

As discussed earlier, adaptive controllers are typically applied for controlling time-varying dynamical systems. To represent this type of system, it is usually common to adopt models whose structure remains constant and the parameters vary in time. These parameters shall be estimated at each sampling period in order to obtain a representative model of the process dynamics (Astrom and Wittenmark, 1995).

3.2 Recursive Least Square

The RLS method was used for estimating the parameters of the process. This approach adjusts the parameters of a given model structure of the process, in a way that this model converges as close as possible of the real system behavior by using its input and output data (Astrom and Wittenmark, 1995).

The regression model, Eq. 1, is a math relation for representing experimental dataset. This equation allows to calculate an approximated $y_{est}(t)$ for the observable variable $y(t)$, from known values $\varphi(t)$ and initial unknown parameters θ. Determining θ must guarantee the convergence of $y_{est}(t)$ to the real values of $y(t)$ (Astrom and Wittenmark, 1995).

$$y_{est}(t) = \varphi_1(t)\theta_1 + \varphi_2(t)\theta_2 + \varphi_3(t)\theta_3 + \cdots + \varphi_n(t)\theta_n \tag{1}$$

The Eq. 1 can be rewritten in its matrix form, introduced in Eq. 2.

$$y_{est}(t) = \varphi^T(t)\theta \tag{2}$$

where

$$\varphi^T(t) = \begin{bmatrix}\varphi_1(t) & \varphi_2(t) & \varphi_3(t) & \cdots & \varphi_n(t)\end{bmatrix} \tag{3}$$

$$\theta = \begin{bmatrix}\theta_1 & \theta_2 & \theta_3 & \cdots & \theta_n\end{bmatrix}^T \tag{4}$$

The parameter θ can be determined by minimizing the cost function of a *least-squares loss function*, Eq. 5.

$$V(\theta, t) = \frac{1}{2}\sum_{i=1}^{t}\left(y(i) - \varphi^T(i)\theta\right)^2 \tag{5}$$

In the present work, the recursive determination of the θ parameters was applied, which implies that the application of the RLS must guarantee that $y_{est}(t)$ approximates to $y(t)$ the greater is the number of iterations. For simplification purposes, two auxiliary variables P and K, also obtained recursively, were defined. The estimation error at time t was represented by $\varepsilon(t)$. The implementation of the RLS estimator was performed using the Eqs. 6-9 (Astrom and Wittenmark, 1995).

$$\theta(t) = \theta(t-1) - K(t)\varepsilon(t) \tag{6}$$

$$\varepsilon(t) = y(t) - \varphi^T(t)\theta(t-1) \tag{7}$$

$$K(t) = P(t)\varphi(t) \tag{8}$$

$$P(t) = P(t-1) - P(t-1)\varphi(t)\left(1 + \varphi^T(t)P(t-1)\varphi(t)\right)^{-1}\varphi^T(t)P(t-1) \tag{9}$$

It is important to note that the dimensions of P and θ are directly related to the number of parameters to be estimated. These matrices will respectively have dimensions $n \times n$ and $n \times 1$ if the model has n unknown parameters. Therefore, the more complex the model, the more elements will have the $P(t)$ matrix and the more computational power will be necessary.

Real experiments with the *follower* have indicated that the system behavior may be described by a second order transfer function. For this reason, a second order difference equation, Eq. 10, was chosen as a regression model

$$y_{est}(t) = -a_1 y_{est}(t-1) - a_2 y_{est}(t-2) + b_0 u(t-1) + b_1 u(t-2) \tag{10}$$

whose matrix representation is given by

$$y_{est}(t) = \varphi^T(t)\theta \tag{11}$$

where

$$\varphi^T(t-1) = \begin{bmatrix} -y_{est}(t-1) & -y_{est}(t-2) & u(t-1) & u(t-2) \end{bmatrix} \tag{12}$$

$$\theta = \begin{bmatrix} a_1 & a_2 & b_0 & b_1 \end{bmatrix}^T \tag{13}$$

In each iteration, the regression model parameters, a_1, a_2, b_0 e b_1, are recursively updated and adjusted for improving the overall model representation. It is important to emphasize that the corrections applied for each parameter are directly related to estimation error in a given iteration and that this error is calculated by the difference between $y_{est}(t)$ and the last experimental output $y(t)$ (Iasbeck, 2019).

3.3 Pole Placement

The Pole Placement method allows to obtain a controller that provides to the system a closed loop behavior the closest as possible of the desired specifications. This is basically done by calculating the controller parameters in a way that allocates the final system poles to reach the required performance (Astrom and Wittenmark, 1995).

The transfer function of the controlled system will be represented in this paper with the shift time operator q, defined by $q^{-1}y(t) = y(t-1)$. With this operator, it is possible to obtain a relation between the process input $u(t)$ and output $y(t)$ in time domain, from its discrete domain, only substituting z by q (Iasbeck, 2019). Consider, for example, the transfer function presented in Eq. 14. Its time domain representation can be obtained directly through q operator, as shown in Eq. 15.

$$\frac{Y(z)}{U(z)} = \frac{z^{-2}}{1 + 0.5z^{-1} + z^{-2}} \tag{14}$$

$$\frac{y(t)}{u(t)} = \frac{q^{-2}}{1 + 0.5q^{-1} + q^{-2}} \tag{15}$$

A generic linear controller, Eq. 16, is proposed in which the tuning process depends on determining $R(q)$, $S(q)$ and $T(q)$, function of q. The control reference is represented by $u_c(t)$.

$$R(q)u(t) = T(q)u_c(t) - S(q)y(t) \tag{16}$$

Considering a generic process, in which its behavior can be described by the relation $y(t)/u(t) = B(q)/A(q)$, the closed loop transfer function, that relates the system output $y(t)$ with the reference $u_c(t)$, is given by

$$\frac{y(t)}{u_c(t)} = \frac{B(q)T(q)}{A(q)R(q) + B(q)S(q)} \tag{17}$$

For determining the controller parameters, it is proposed a reference model $y_m(t)/u_c(t) = B_m(q)/A_m(q)$, which it is a transfer function that relates the desired system output $y_m(t)$ and the reference $u_c(t)$. This model represents the desired behavior and it is determined from the project requirements (Astrom and Wittenmark, 1995).

Thus, it is necessary

$$\frac{B(q)T(q)}{A(q)R(q) + B(q)S(q)} = \frac{B_m(q)}{A_m(q)} \tag{18}$$

From Eq. 18, it is possible to obtain $R(q)$, $S(q)$ and $T(q)$. The computation of the *follower* system model allowed to verify that its dynamic behaviour can be well represented by a second order transfer function, Eq. 19. The reference model considered in this work, also of second order, is presented in Eq. 20 and will be treated in detail in the next section.

$$G(z) = \frac{-0,0244z + 0,1503}{z^2 - 1,0002z - 0,0008} \Rightarrow \frac{B(q)}{A(q)} = \frac{-0,0244q + 0,1503}{q^2 - 1,0002q - 0,0008} = \frac{b_0 q + b_1}{q^2 + a_1 q + a_2} \tag{19}$$

$$\frac{B_m(q)}{A_m(q)} = \beta \frac{b_0 q + b_1}{q^2 + a_{m1} q + a_{m2}} \tag{20}$$

Assuming that the controller poles do not cancel any of the system zeroes, and PP (Astrom and Wittenmark, 1995) method is utilized to determine $R(q)$, $S(q)$ and $T(q)$, it follows that (Iasbeck, 2019)

$$R(q) = q + r_1 \tag{21}$$

$$S(q) = s_0 q + s_1 \tag{22}$$

$$T(q) = \beta(q + a_0) \tag{23}$$

where

$$r_1 = \frac{a_{m2} + a_{m1}a_0 - a_2 - \frac{b_0}{b_1} a_{m2} a_0 - \frac{b_1}{b_0}(a_{m1} - a_1 + a_0)}{a_1 - \frac{b_0}{b_1} a_2 - \frac{b_1}{b_0}} \tag{24}$$

$$s_0 = \frac{a_{m1} + a_0 - a_1 - r_1}{b_0} \tag{25}$$

$$s_1 = \frac{a_{m2} a_0 - a_2 r_1}{b_1} \tag{26}$$

$$\beta = \frac{1 + a_{m1} + a_{m2}}{b_0 + b_1} \tag{27}$$

It should be noted that the controller parameters are determined from the process parameters to be controlled a_1, a_2, b_0 and b_1, obtained from the RLS method, and the reference model parameters, a_{m1} and a_{m2}, defined from the performance requirements (Iasbeck, 2019).

It can be shown that the order of the controller obtained by using the PP is always one order lower than the controlled process (Iasbeck, 2019). For this reason, a first-order controller was proposed in the present work. However, it is important to emphasize that this is a disadvantage of PP, since, for higher order systems, it would require a relatively high order controller, which can be difficult to be implemented.

3.4 Determining the reference model

In the present work, it was adopted a second order reference model of the form

$$G(s) = \frac{w_n^2}{s^2 + 2\xi w_n s + w_n^2} \tag{28}$$

The natural frequency w_n and the damping coefficient ξ of the reference model are determined by the performance requirements, using the Eqs. 29 and 30. The overshoot percentage and the settling time defined by the controller designer are represented by M_p and t_a.

$$\xi = \sqrt{\frac{ln^2 M_p}{ln^2 M_p + \pi^2}} \tag{29}$$

$$w_n = \frac{4}{\xi t_a} \tag{30}$$

From ξ and w_n values, the poles of $G(s)$ are obtained. The discretization allows to define the characteristic equation for the model in the z domain, as shown in Eq. 31 (Ogata, 1987).

$$D(z) = (z - z_1)(z - z_2) = z^2 + a_{m1} z + a_{m2} \tag{31}$$

where $a_{m1} = -(z_1 + z_2)$ and $a_{m2} = z_1 \cdot z_2$.

As previously seen, it is possible to directly relate the z domain to the discrete time domain by using the q operator. Thus, from $D(z)$ it is possible to determine $A_m(q)$, Eq. 32.

$$A_m(q) = q^2 + a_{m1} q + a_{m2} \tag{32}$$

Finally, $B_m(q)$ is defined from $B(q)$, and a factor β, Eq. 27, is included so that the static gain of the reference model is unitary.

Considering, as an example, the performance requirements $M_p = 0.1\%$ and $t_a = 1\ s$. Applying these values in Eqs. 29 and 30, are obtained $\xi = 0.91$ and $w_n = 4.35\ rad/s$. From Eq. 28 follows that

$$G(s) = \frac{18.9}{s^2 + 7.917s + 18.9} \tag{33}$$

From the discretization of $G(s)$ by applying the Bilinear Transformation, considering a sampling time equals to 0.05 seconds (the same one adopted throughout this paper), it follows that $z_{1,2} = 0.8171 \pm 0.0736i$ (Ogata, 1987). Thus, $D(z) = z^2 - 1.634z + 0.6731$ and implies that $A_m(q) = q^2 + a_{m1}q + a_{m2}$, where $a_{m1} = -1.634$ and $a_{m2} = 0.6731$. Finally, assuming that $B_m(q) = \beta B(q)$, the reference model is determined.

4. RESULTS ANALYSIS

The STR method was implemented in a simulation environment, using Matlab® software. The results are shown in Fig. 3, and the performance requirements are overshoot of 0.1% and settling time of 1s. Initially, the system output does not converge to the expected value, since the process parameters are being recursively updated and converging to the real values. After approximately 3 seconds, the convergence is reached and the output approaches the desired output.

The Fig. 4 shows the experimental results obtained from the trajectory control implemented for the *follower*. The performance requirements were the same considered in the simulation. It is important to notice the proximity between both real and simulation results (Fig. 3).

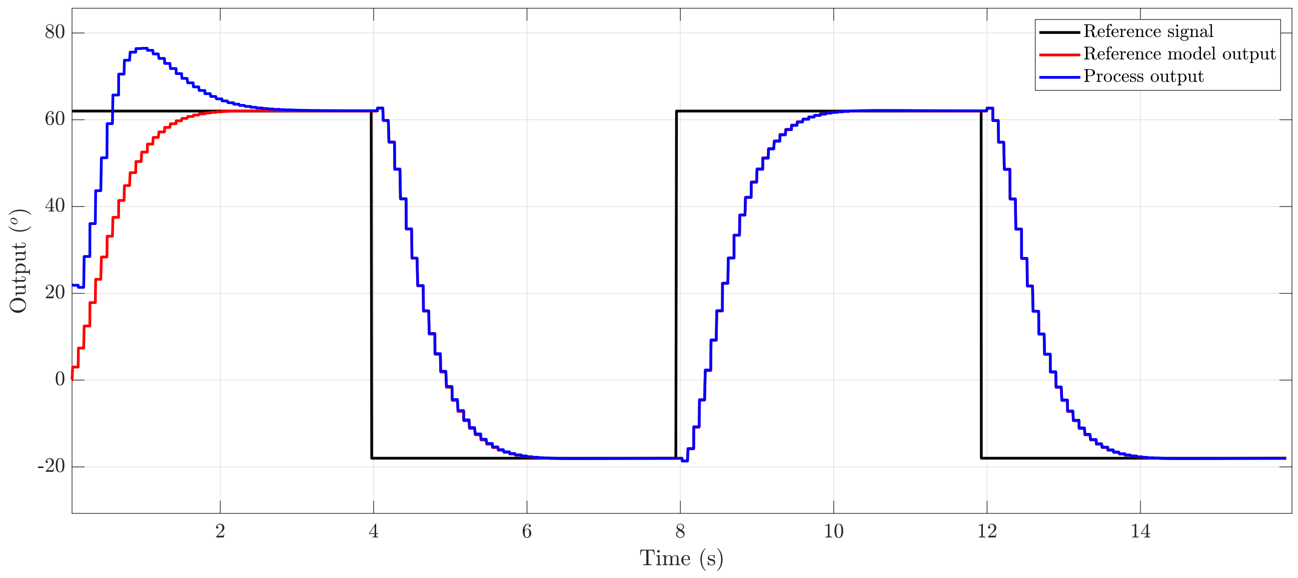

Figure 3. Simulation results from the STR implementation for trajectory control.

The experiments allowed to verify that the implementation of the proposed adaptive controller ensured that the performance requirements were met even in the presence of uncertainties in the system model. For example, the variation of the battery charge in the *follower* was not considered in the modelling and ended up causing loss of power in the motors. However, after the implementation of the control loop, this loss was compensated by the application of control signals of greater amplitude.

It can be noticed from analyzing the graphs in Fig. 5 that the system exhibited a behaviour very similar to that of the reference model despite the variation in the battery charge. In addition, the evaluation of the graphs introduced in Fig. 6 shows that the *follower* battery discharge has been compensated by the application of more intense control action. This behaviour shows the adaptability of the proposed controller.

A video that shows the automatic tuning of the proposed controller (with a variable reference) can be accessed in `https://www.youtube.com/watch?v=HHvO09RgzqI`. In this other link `https://www.youtube.com/watch?v=ZVC4HjIPwM4` there is a video that shows the *follower* tracking a rectilinear trajectory (constant reference) with the application of disturbances. Besides that, the code developed to generate the results presented in this section can be accessed in `https://gitlab.com/arthuriasbeck/iasbeck-tcc-adaptative-control`.

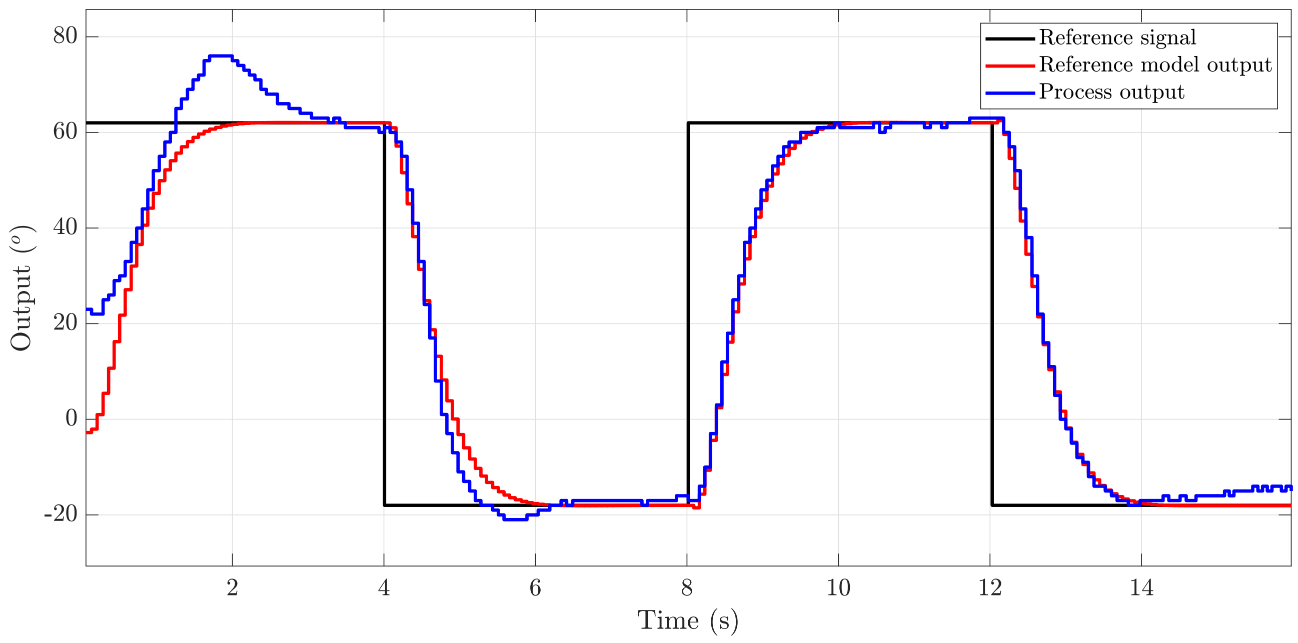

Figure 4. Real experiments results from the STR implementation for trajectory control.

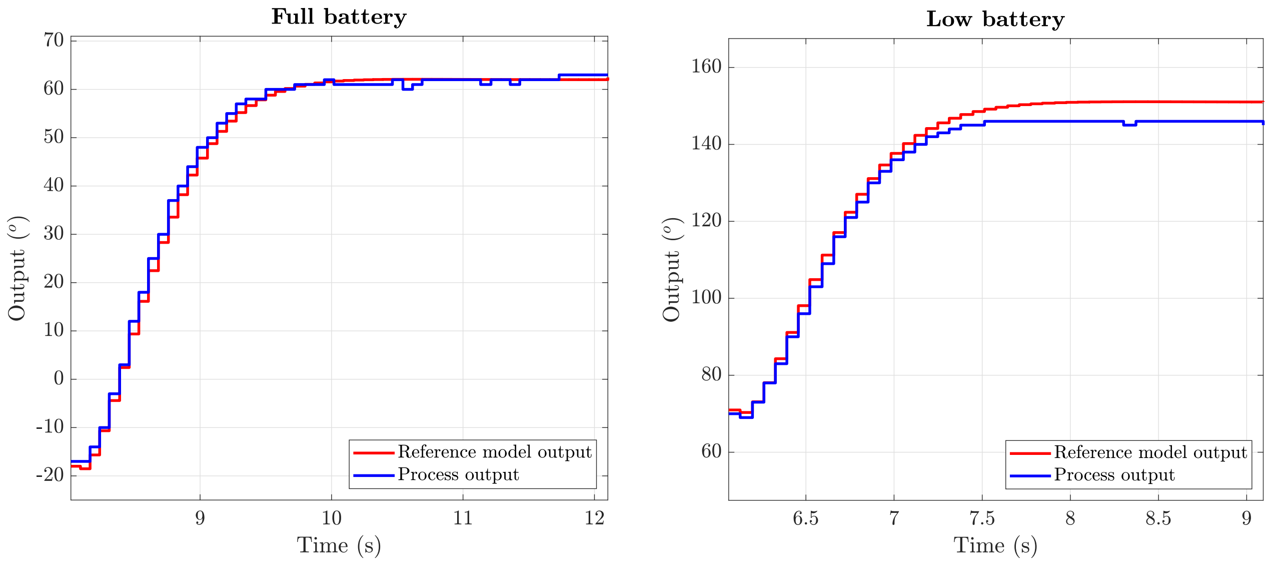

Figure 5. Evaluation of the influence of the battery charge level on the process output.

5. CONCLUSIONS

Matlab® simulations allowed to do a first evaluation of the system behaviour before proceeding with the experiments in the real robot. Both situtations demonstrated the STR implementation efficacy and its adapting aspect.

The simulations and the real implementation confirmed that the proposed adaptive controller with STR has the potential to perform trajectory tracking for an autonomous robot, even with unknown system parameters.

The implemented controller was capable to adapt to the process dynamics, in which the performance requirements were closely reached to those specified. The experiments with the real robot validated the simulation results.

It is intended to address in future work other Adaptive Control schemes, based on Reference Models and Intelligent Systems. In this last case, it is proposed the use of Neural Networks for the implementation of the adaptive control loop, since this approach presents good performance when implemented in the modelling and control of nonlinear systems (Ge *et al.*, 2002).

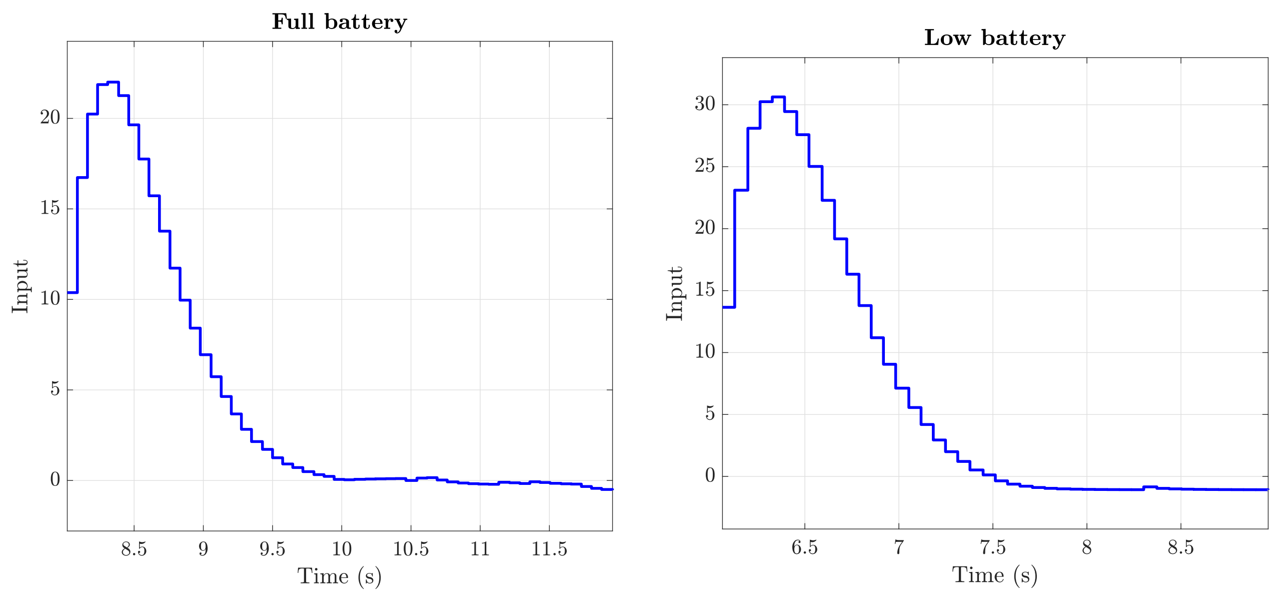

Figure 6. Evaluation of the influence of the battery charge level on the control action.

6. REFERENCES

Astrom, K.J. and Wittenmark, B., 1995. *Adaptative Control*. Addison Wesley, 2nd edition.

Ge, S.S., Hang, C.C., Lee, T.H. and Zhang, T., 2002. *Stable Adaptative Neural Network Control*. Springer Science + Business Media.

HiTechnic, 2018. “Hitechnic nxt compass sensor for lego mindstorms nxt”. `http://www.hitechnic.com/cgi-bin/commerce.cgi?preadd=action&key=NMC1034`. [Accessed 21 October 2018].

Iasbeck, A.H., 2019. “Controle adaptativo empregado no rastreamento de trajetória de um robô autônomo”. Monografia (Bacharel em Engenharia Mecatrônica), UFU (Universidade Federal de Uberlândia), Uberlândia, Brasil.

leJOS, 2019. “The lejos nxj tutorial”. `http://www.lejos.org/nxt/nxj/tutorial/index.htm`. [Accessed 2 March 2019].

Ogata, K., 1987. *Discrete-Time Control Systems*. Prentice-Hall International Editions. Prentice-Hall.

7. RESPONSIBILITY NOTICE

The authors are the only responsible for the printed material included in this paper.