Modeling and Control of an Educational Manipulator Robot Joint

J. A. B. Coelho[1(✉)], Laiany Brancalião[1,3], Mariano Alvarez[1,4], Paulo Costa[4,5], and José Gonçalves[1,2]

[1] Research Centre in Digitalization and Intelligent Robotics (CeDRI), Instituto Politécnico de Bragança, Campus de Santa Apolónia, 5300-253 Bragança, Portugal
{joaocoelho,laiany,marianoalvarez,goncalves}@ipb.pt
[2] Laboratório para a Sustentabilidade e Tecnologia em Regiões de Montanha (SusTEC), Instituto Politécnico de Bragança, Campus de Santa Apolónia, 5300-253 Bragança, Portugal
[3] University of León, Campus de Vegazana, 24071 León, Spain
[4] Faculty of Engineering, University of Porto, 4200-465 Porto, Portugal
paco@fe.up.pt
[5] INESC TEC - Institute for Systems and Computer Engineering, Technology and Science, Porto, Portugal

Abstract. Integrating physical robots in an educational context often entails acquiring expensive equipment that often operates using proprietary software. Both conditions restrict the students from exploring and fully understanding the internal operation of robots. In response to these limitations, a three-degree-of-freedom robotic manipulator, based on the "EEZYbotARM MK2" open-source design by Carlo Franciscone, is being repurposed and integrated within the SimTwo simulation environment to operate within a hardware-in-the-loop architecture. To accomplish this objective, first, an open-source Arduino-based library was developed aiming at the robot's online and offline programming akin to industrial robots. The firmware is able to communicate with the SimTwo software in which the digital twin's robot is living. The dynamic behavior of the robot's digital twin must be properly parametrized and aligned with the physical robot's dynamics. This article describes the modeling of the robot joint's actuator and its closed-loop controller formulation. The obtained results show that the dynamic behavior of the robot joint digital twin closely matches both open and closed-loop, the one of its physical counterpart.

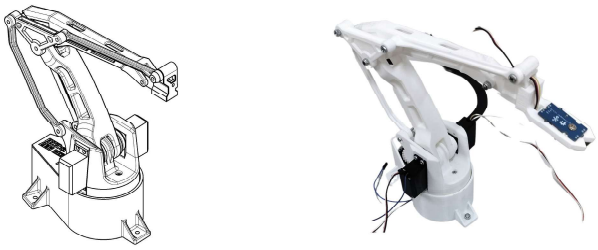
Keywords: Education · robotics · SimTwo · modeling · control · physics simulation

1 Introduction

The development of robots presents a range of complex challenges across design, hardware, and software domains. Addressing these challenges involves navigating through various iterative steps in order to achieve the intended outcomes.

R. Molina Carmona et al. (Eds.): TEEM 2024, LNET, pp. 659–668, 2025.
https://doi.org/10.1007/978-981-96-5658-5_65

This particular research endeavor delves into the realm of robotics education, instrumentation, and control, entailing the prototyping and control of an educational manipulator robot. Central to the project is the assembly and adaptation of the open-source "EEZYbotARM MK2" manipulator robot, originally conceptualized by Carlo Franciscone [1]. This robot features a three-degree-of-freedom revolute robot arm, complemented by a gripper that always remains parallel to the robot's base. This design characteristic promotes its suitability for pick-and-place applications. As depicted in Fig. 1, the 3D design of the robot arm and the physically assembled version of the prototyped robot are shown. It is worth noticing that this robot's direct and inverse kinematics have already been studied and presented in [2].



(a) 3D model. (b) Physical prototype.

Fig. 1. Technical 3D rendering and physical implementation of the "EEZYbotARM MK2" manipulator robot.

The robot's operation is driven by three 5 V servo motors, each controlling one of the three degrees of freedom within the robot's structure. While the low-cost factor of those servo motors aligns with the design requirements, the embedded controller introduces instability in position control. Furthermore, it lacks the capacity to directly determine the servo's velocity between positions without emulating it through defined positions and time stops. This control aspect holds significant importance for manipulator robot applications. Consequently, this work addresses the replacement of the MG995 servo motor controller with a customized digital controller. Moreover, this servo motor, comprising the motor and the custom-made controller, will be integrated into a rigid body dynamics simulator featuring interaction between the physical motor and its simulated counterpart with hardware-in-the-loop. The precision of the simulation is paramount to capturing the authentic dynamics of the motor and controller, facilitating accurate inter-system communication.

The physics simulator software SimTwo [3] requires critical parameters to model a DC motor accurately. In particular, as will be shown ahead, it is important to derive the coil's resistance and inductance. Moreover, viscous friction, static friction, and both torque and electromotive motor constants must be devised. Those parameters are obtained by measuring state variables such as

voltage, electric current, and rotor speed. To establish an accurate motor model and develop a digital controller based on this model, the encoder "E6A2-CW5C" was employed to monitor the motor's position and velocity. The resulting model will enable precise parameter input into the SimTwo software and support the creation of a customized controller tailored to the motor's dynamics.

After this introductory section, Sect. 2 explains the process for acquiring motor data, computing motor coefficients for the simulator, and deriving the transfer function that characterizes the motor's dynamics for the controller. In Sect. 3, the findings regarding the determination of the motor's transfer functions and the response of the PI controller to a specified input are presented. Finally, the conclusions drawn in Sect. 4 summarise the results of this article.

2 Materials and Methods

The following section outlines the materials and methodologies employed in prototyping the system as introduced in Sect. 1. It describes the data acquisition process based on the use of an external encoder to facilitate reading the motor's output velocity and position. Additionally, it involves calculating the model's parameters to enable the simulation of an accurate representation of the motor used and the development of the PI controller based on the derived motor dynamics.

2.1 Data Acquisition

As indicated in Sect. 1, acquiring the motor's velocity and position relies on a rotary encoder. In preparation for the experiments, the servo motor underwent dismantling, with the subsequent removal of its controller and potentiometer. The motor's gears remained unaltered as the encoder was connected to the output shaft. For the connection of the encoder to the motor, a custom-designed support was created using 3D printing technology. This support holds the servo motor and the encoder in a fixed position, safeguarding them against external forces and minimizing potential disturbances during the data acquisition process. In addition to securing both the encoder and the motor, a fitting was devised to connect the motor's gear output shaft to the encoder shaft. This component, made of PLA and incorporating rubber within its structure, provides the necessary flexibility to compensate for any misalignment between the shafts and, for this reason, reduce friction. Figure 2 depicts the configuration of the data acquisition structure used to capture the encoder's output.

In order to analyze the motor's operational characteristics within both transient and steady-state conditions, a voltage-controlling driver was employed, and an Arduino Mega 2560 served as the microcontroller to capture sensor data and handle the driver. This methodology involves using external interrupt-driven procedures with reference to the timestamp of the preceding interrupt and a counter that progresses in alignment with the signal response from both operational phases. This framework facilitates the determination of the velocity's magnitude and rotation direction.

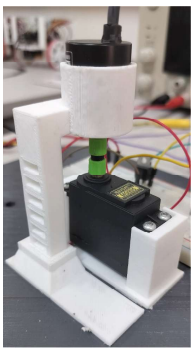


Fig. 2. Custom-designed 3D-printed support structure for mounting the encoder and servo motor.

In order to calculate the velocity, the dual-phase signals of the encoder (A and B) are utilized to determine the direction of the rotation. Specifically, a change in signal A leads to an adjustment in the counter, either upward or downward, depending on the status of signal B, and vice versa. The phase lead or lag between signals A and B is used to compute the rotation direction.

The time interval between consecutive changes in state for both phases, denoted as Δt (in microseconds), was determined as the difference between the last recorded external interruption time and the last sampled time. Subsequently, this time difference was converted to seconds. The rotational velocity of the motor, ω_r (in radians per second), was derived from the encoder counts and the number of encoder pulses per revolution. The encoder resolution (N) was adjusted to a value of 400 due to the implementation of two phases and the counter being updated for both rising and falling edges. The velocity was computed using the provided equation:

$$\omega_r = \begin{cases} \frac{(C \times 2\pi)}{\mathrm{N} \times \Delta t} & \text{if } \Delta t > 0 \\ 0 & \text{if } \Delta t = 0 \end{cases} \tag{1}$$

2.2 Electrical Motor Model

Mathematical models are essential for simulation and controller design since they provide a precise and systematic representation of complex systems that can be used to analyze and predict system behavior under various conditions. They must be able to describe the dynamic relationship between the system's input and output signals according to some internal state space [4]. Moreover, having a mathematical model of the system can lead to a more systematic controller design. In practice, mathematical models can be obtained by many different methods. Some, derived from the first principle, are based on the physical laws that govern the underlying system's dynamic behavior. Other, derived only from observed data, lead to black-box type of models. The latter ignores any internal

mechanism, and both model structure and parameters are derived from data analysis methods.

In this work, a mathematical model must be derived to include the DC motor dynamics in the SimTwo software. Moreover, due to the format required by the software, the model must be based on physical principles. In particular, the electrical motor must be described by the electromechanical lumped parameter model presented in Fig. 3.

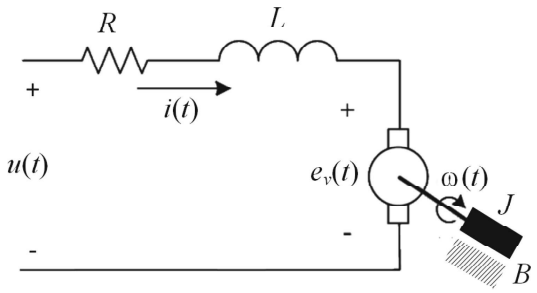


Fig. 3. Lumped parameter model for a permanent magnet DC motor.

As can be seen, the electrical behavior is captured by the series of three lumped elements. The first is a resistance with value R, the second is an inductor with inductance L, and finally, the third is a voltage source that represents the counter-electromotive force generated by the movement of the rotor. On the mechanical side, the diagram includes the rotor's inertia moment, J, and a friction coefficient, B.

From Kirchhoff's voltage law, the relation between the input voltage, $u(t)$, and electrical current, $i(t)$, is given by the following differential equation:

$$u(t) = Ri(t) + L\frac{di(t)}{dt} + e_V(t) \tag{2}$$

Within a given operating point, the counter-electromotive force is linearly proportional to the rotor speed, $\omega(t)$. Assuming K_e as the proportionality constant, then:

$$e_V(t) = K_e\omega(t) \tag{3}$$

On the other hand, mechanical behavior can be derived from Newton's second law:

$$T_M - T_B - T_Q = J\frac{d\omega(t)}{dt} \tag{4}$$

where the motor torque, T_M, is a linear function of the electrical current, $i(t)$, according to $T_M = K_M i(t)$ for any positive constant K_M. The braking torque, T_B, depends on the friction coefficient, B, and is proportional to the rotator's

speed, ω, by $T_B = B\omega(t)$. A constant torque, T_Q, due to other unaccounted factors, is also considered. In this reference frame,

$$K_M i(t) - B\omega(t) - T_Q = J\frac{d\omega(t)}{dt} \tag{5}$$

Combining (2) and (5) lead to the following second order differential equation:

$$\frac{LJ}{K_M}\frac{d^2\omega(t)}{dt^2} + \left(\frac{RJ}{K_M} + \frac{LB}{K_M}\right)\frac{d\omega(t)}{dt} + \left(\frac{RB}{K_M} + K_e\right)\omega(t) + \frac{RT_Q}{K_M} = u(t) \tag{6}$$

Now, neglecting the effect of the inductance and assuming that $K_e = K_M = K$, the previous expression leads to the following first-order differential equation:

$$\frac{RJ}{K}\frac{d\omega(t)}{dt} + \left(\frac{RB}{K} + K\right)\omega(t) + \frac{RT_Q}{K} = u(t) \tag{7}$$

To describe this dynamic behavior in SimTwo, it is necessary to determine the parameters R, B, K, and T_Q. For this purpose, a series of tests were conducted in which the electric current and the rotational speed of the motor at steady state were measured for different voltage values. The results obtained are presented in Table 1.

Table 1. Steady-state current and angular velocity for different motor voltages.

Voltage (V)	Current (A)	Angular Velocity (rad/s)
1	0.03914	0.55665
1.5	0.0429	0.98467
2	0.0497	1.38156
2.5	0.05502	1.7766
3	0.05765	2.20709
3.5	0.06337	2.61674
4	0.06616	2.98489
4.5	0.06962	3.29374
5	0.07476	3.70856

For the steady-state behavior, the following system of equations is derived:

$$\begin{cases} u(t) = Ri(t) + K\omega(t) \\ 0 = Ki(t) - B\omega(t) - Tq \end{cases} \tag{8}$$

Based on the results from Table 1, the parameters R and K of Eq. (8) are obtained, in the least squares sense, by solving the Eq.:

$$\boldsymbol{\theta} = \left(\mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{u} \tag{9}$$

where $\mathbf{H}$ is the matrix of regressor vectors, $\mathbf{u}$ is the vector associated with the voltage values, and $\theta = \left[R\,K\right]^T$. The results obtained yielded $R = 7.29\ \Omega$ and $K = 1.2$. The same procedure was carried out for the second equation represented in (8), resulting in $B = 0.0133$ and $T_Q = 0.0396$. The value of the inertia coefficient, J, was determined from the DC gain of the system using:

$$J = \frac{1}{9.374}\frac{K}{R} = 0.0176 \tag{10}$$

Considering the system as first-order, experimental tests were conducted to assess the dynamic behavior. A set of measurements was performed by exciting the motor with a voltage $u(t)$, which has been forced to be the pseudo-random binary sequence [5] shown in Fig. 4.

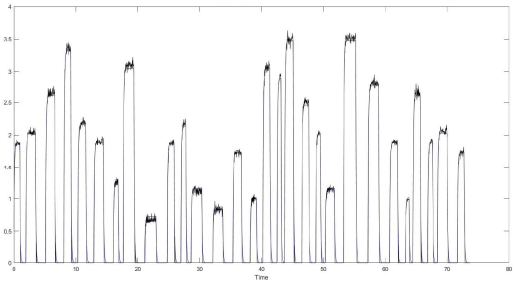


Fig. 4. DC motor response to a PRBS signal.

An iterative system identification method was employed to derive the model parameters of a canonical first-order transfer function. This approach has led to:

$$H(s) = \frac{9.374}{s + 12.7} \tag{11}$$

2.3 PI Controller

Resorting to the transfer function obtained in the previous section, a proportional-integral (PI) controller was designed to regulate velocity [6,7]. The PI controller adheres to the following equation in the Laplace domain:

$$PI = K_c\frac{T_i s + 1}{T_i s} \tag{12}$$

Here, K_c represents the inverse of the gain value governing the relationship between angular velocity and counter electromotive force, as well as the motor's torque gain. Meanwhile, T_i denotes the coefficient associated with the open-loop

response time to achieve 60% of its steady-state value. In addition to implementing the controller, a feed-forward component was incorporated to enhance response time to the reference value provided to the controller. The gain of this feed-forward component was stipulated as 35% of the overall value of K_c. Subsequently, the controller was tested in MATLAB, and its outcomes were evaluated. Additionally, a saturation block was added to the output of the controller, given that the motor is restrained to voltage values ranging from −5 V to 5 V. Figure 5 depicts the PI controller model implemented in Simulink.

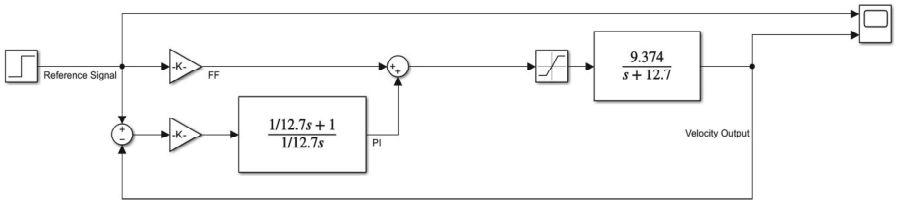


Fig. 5. Implementation of the controller of the motor composed by the PI controller block and the feed-forward block in the Laplace domain.

The closed-loop simulation depicted in Fig. 5 illustrates the coordinated functioning of the controller and motor, where a step response in time is utilized with a predefined reference velocity of 2 rad/s. This reference value serves as input to the controller, determining the necessary voltage for the motor to achieve the desired velocity. Post-controller output, a saturation block is employed to confine voltage values within the predefined minimum and maximum limits. The resulting voltage is then directed to a block representing the motor's transfer function model. This block produces the motor's velocity, which, in turn, is fed back to the controller. The controller repetitively calculates the velocity error, thereby updating the voltage values.

As seen in Fig. 6, the controller produced satisfying results with a 200 millisecond response time to reach the desired reference velocity with a slight overshoot during its transience response. Given the satisfying results produced by this controller, the controller was then transformed to the discrete mode in order for it to be implemented in the microcontroller's hardware. The simplified expression of the controller is given by the equation below, where the first term is the expression of the PI controller while the term that is summed to it is the feed-forward constant.

$$G(s) = \frac{0.1067\,s + 1.355}{0.07874\,s} + 0.47418 \tag{13}$$

To obtain a discrete-time equation from the previous expression, the Tustin approximation method was employed [8]. Considering a 30 Hz sampling frequency, the controller transfer function in the z-domain takes the following form:

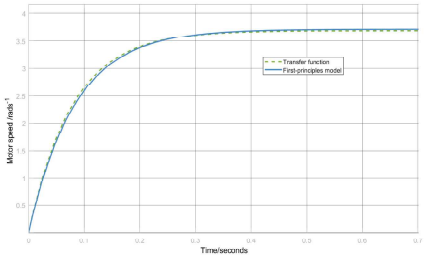
$$G(z) = \frac{1.639z - 1.071}{z - 1} + 0.47418 \tag{14}$$

The discrete-time controller was implemented in Simulink, and a one-second simulation was performed. The performance between the continuous and discrete domain controllers was then compared. Further details on the obtained results can be found ahead in the following section.

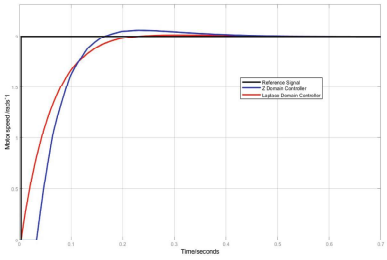
3 Results

In the previous section, two distinct methods were developed to replicate the motor's behavior: the first-principles model and the system-identified model using pseudo-random binary search sampling. The comparison between the transfer functions and the model in the time domain, as outlined in Eq. (7), was simulated, yielding the results presented in Fig. 6a. This figure demonstrates the close similarity between both transfer functions.

Figure 6b depicts the comparison results between the reference velocity and the actual motor output for evaluating the controller's performance. The figure demonstrates a satisfactory response time and also highlights that the discrete-time controller closely matches the performance of its continuous counterpart, albeit with some deviations attributable to the selected sample frequency and phase lag due to the sample and hold.



(a) Comparison between the transfer function derived from the first-principles model and the transfer function based on the system identification method.



(b) Comparison between the controller's response in the continuous-time and the discrete-time domain to a step reference signal.

Fig. 6. Comparison of transfer functions representing the motor's model and controller's response in different domains.

4 Conclusions

In this study, advancements were achieved in the prototyping and control of the "EEZYbotARM MK2" educational manipulator robot. Central to the research was the development and implementation of an improved control system with enhanced responsiveness compared to the original one. The process began with

creating an accurate mathematical model of the DC motor based on fundamental physical principles. This model was validated through experimental data obtained from interfacing with a commercial encoder. By capturing precise measurements of the motor's position and velocity, the model developed a well-tuned controller.

The PI controller, designed to improve velocity control, demonstrated robust simulation performance. It achieved a response time of 200 milliseconds with minimal overshoot, reflecting its effectiveness in maintaining desired velocity setpoints. The controller's design included a feed-forward component to enhance response time, which was successfully implemented and tested in both continuous and discrete domains. The conversion of the continuous-time controller to a discrete-time format confirmed that the discrete controller closely approximated the performance of its continuous counterpart, validating the feasibility of hardware implementation.

In future work, the controller design within the microcontroller is planned to be executed, and the calculated parameters for the motor model will be integrated into the SimTwo software. Furthermore, it is intended to incorporate hardware-in-the-loop testing to assess the correlation between the physics simulator and the actual motor model. This step is deemed crucial for the final applications.

References

1. Franciscone, C.: (2018). EEZYbotARM MK2. Retrieved from http://www.eezyrobots.it/eba_mk2.html
2. Coelho, J.A.B., Brancalião, L., Alvarez, M., Costa, P., Gonçalves, J.: Prototyping and control of an educational manipulator robot. In: 2024 10th International Conference on Control, Decision and Information Technologies (CoDIT), Vallette, Malta, pp. 1814–1819 (2024). https://doi.org/10.1109/CoDIT62066.2024.10708583
3. Costa, P., Gonçalves, J., Lima, J., Malheiros, P.: Simtwo realistic simulator: a tool for the development and validation of robot software. Theory Appl. Math. Comput. Sci. **1**, 17–33 (2011)
4. Gonçalves, J., Lima, J., Costa, P.G.: DC motors modeling resorting to a simple setup and estimation procedure. In: Moreira, A.P., Matos, A., Veiga, G. (eds.) CONTROLO'2014 – Proceedings of the 11th Portuguese Conference on Automatic Control. LNEE, vol. 321, pp. 441–447. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-10380-8_42
5. Alam, M.S., Tokhi, M.O.: System identification of a twin rotor multi-input multi-output system using adaptive filters with pseudo random binary input. Dhaka Univ. J. Sci. **57**(2), 131–136 (2008)
6. Franklin, G.F., Powell, J.D., Workman, M.L.: Digital Control of Dynamic Systems-Third Edition (2022)
7. Ibrahim, M.A., Hamoodi, A.N., Salih, B.M.: PI controller for DC motor speed realized with simulink and practical measurements. Int. J. Power Electr. Drive Syst. (IJPEDS) **11**(1), 26–119 (2020)
8. Malinen, J.: Tustin's method for final state approximation of conservative dynamical systems. IFAC Proc. **44**(1) (2011). (IFAC-PapersOnline) 18. 10.3182/20110828-6-IT-1002.01445