CONTROL AND MODELING OF A DIRECT CURRENT MOTOR - MODELING AND ANALYSIS OF DYNAMIC SYSTEMS

Teacher: Humberto Xavier de Araújo **Henrique Nunes Poleselo** **Jesse de OS Alves**

Luis Gustavo Christensen

16 de junho de 2024

1 Introduction

The act of measuring the speed of a wheel, called odometry, is essential in the field of mobile robotics, especially nowadays, where processing power has increased, and therefore, enabling the fusion of sensors (odometry with computer vision, for example). example) for implementing more complex controllers. That said, the application in question deals with modeling and controlling the speed of a direct current motor that is coupled to a wheel. This report will present some methods used to obtain the black box model and the process of preparing the controller. Therefore, this work is separated into 8 sections where each section is divided as follows: in section 2 the project objectives will be mentioned, in section 3 the prototype is presented. In section 4 the modeling process is explained, in section 5 the design of the system controller. In section 6 the results obtained will be presented and then the project conclusions in section 7.

2 Objectives

Despite using black box model methods, it is known that the variable angular velocity of a direct current motor is a stable system. Therefore, the system will naturally reach its steady state for a step input. The objective of the application in question is to make the direct current motor reach this angular speed in a shorter time using PID control to consequently reduce the error.

3 Problem description

To build the prototype, only a direct current motor was used, an Arduino UNO that has the ATMega328 microcontroller, an infrared sensor (LM393) with a phototransistor that generates an interruption, a *encoder* disk and a bridge H to generate the current required for motor control. The motor is powered by the H bridge, which receives a PWM signal (*Pulse Width Modulation*) from the Arduino and transmits it to the motor with the necessary current. To obtain the angular velocity curves of the motor, the *encoder* disk was coupled to the motor shaft and the infrared sensor was positioned so that it detected and generated an interruption every time the electromagnetic wave beam passed through a disc hole. This way, it is possible to measure the frequency with which the disc rotates and, using the radius of the wheel, determine its angular velocity over time. These measurements carried out with the sensor were validated by marking the wheel and determining the linear speed, which was later divided by the wheel radius, and it was noticed that the sensor presents distortions for high rotations (close to 120 PWM bits), in addition to present relatively noisy measurements.

To read the speed, it was done as described above and the assembly can be seen in figure 1:

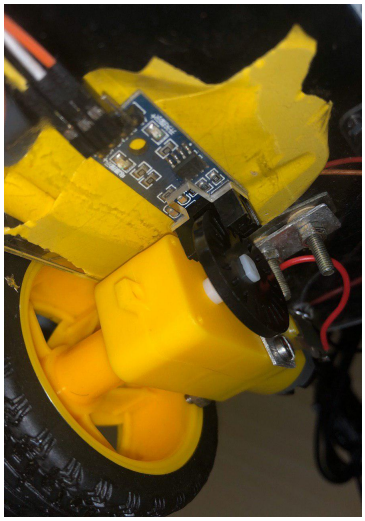

Figure 1: Assembly for reading wheel speed.

The wheels were attached to a base and could be used as a robot in the future. In section 4, the engine modeling is done with the robot suspended, therefore neglecting dynamic friction and considering only the static friction of the engine itself and the wheel. The final assembly can be seen in figure 2.

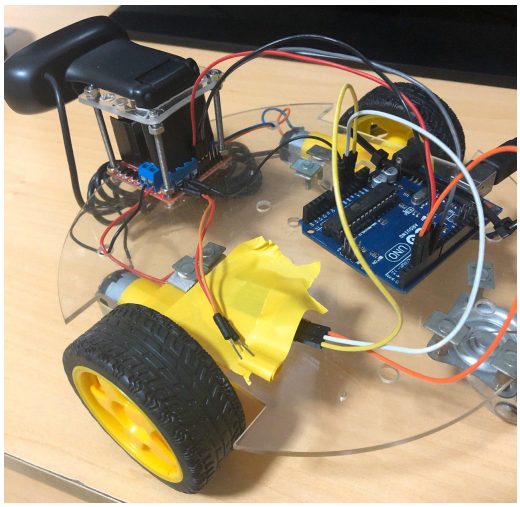

Figure 2: Assembled prototype.

4 Modeling

4.1 Static Model

To have a good starting point for modeling the system, the system's static behavior curve was obtained. Such a curve is important to determine the range of PWM values in which the system responds in the most linear way, thus facilitating process control. To determine the operating range of the PWM, different values were applied using the microcontroller (from 0 to 255, 8-bit range) and it was analyzed that the motor's dead zone was located in the range of 0 to 60 bits of the PWM. , therefore, it was assumed that the operating range for the controller would be at least 60 PWM bits. The static model was proposed with PWM variation between 75 and 255 bits as this is the range in which the motor operates safely.

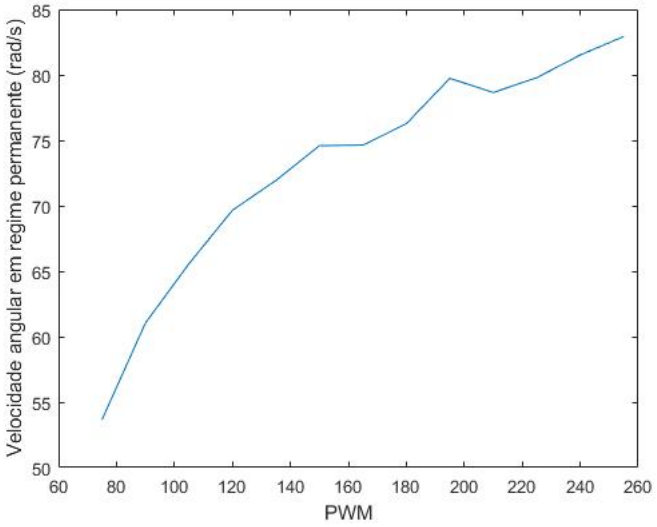

Figure 3: Static model of the DC motor.

From figure 3 it is possible to notice that in the range between 90 and 145 PWM there is an approximately linear behavior, so it is desirable to work with values within this range.

4.2 Dynamic Model

The dynamic model presents the relationship between the input and output of the system in the transient and permanent regime. Such behavior is obtained from a step input, which in the application is a "DC level"of the PWM, therefore voltage. As in the static model, the input will be the PWM value and the output will be the angular speed of the wheel. 120 was chosen as the PWM value since it is in the range in which the system showed linear behavior in the static model. In Figure 4, using the logic mentioned in section 3, the instantaneous angular speed of the wheel was obtained with a sampling time of $50ms$.

50. 5 Figure 4: Dynamic model of the DC motor.

In figure 4 it is possible to see a dead zone until the system responds, this phenomenon is due to the moving average filter applied later to the acquired speed data. There was a need to use filtering as the sensor does not offer much robustness at high speeds. As in the control it was decided not to use the moving average filter, it is possible to ignore the dead zone in question. However, different models were obtained for comparison purposes. The step input used in figure 4 was 120 bits PWM and as it is possible to infer from the graph, $y(\infty) = 69.66rad/s$ (steady regime), therefore consistent with the static model proposed in section 4.1 .

4.3 Two-Parameter Model

The first model proposed was the two-parameter model, which uses the gain (K) and the time constant (T) as follows [1]:

$$G(s) = \frac{K}{T \cdot s + 1}. \tag{1}$$

As it is known that $y(\infty) = 69.66rad/s$ and the input is a 120-bit PWM, which is equivalent to $u(t)$ with an amplitude of $4.92V$, therefore the value of K is 14.16, which is the division between $y(\infty)$ and the step input. To find the time constant, the area method between the straight line representing the steady state minus the transient regime curve [2] was used. With the area value obtained, the value of T is found by $T = Ao/k$, where Ao is the area and k the gain, therefore $T = 0.2571$. Then:

$$G(s) = \frac{14.16}{0.2571 \cdot s + 1}. \tag{2}$$

4.4 Three-Parameter Model

In the three-parameter model, in addition to using the gain and time constant, the system delay L was taken into account, in order to obtain the following model:

$$G(s) = \frac{K \cdot e^{-L \cdot s}}{T \cdot s + 1}. \tag{3}$$

The gain remains the same $K = 14.1585$ and the delay value is equal to the dead time value $L = 0.153s$. With the delay value and the gain value, the time constant is obtained in a similar way to that obtained with the two-parameter model, however considering the delay:

$$T = \frac{Ao}{K} - L \tag{4}$$

In this way, $T = 0.1045s$ was obtained. Then the model looked like this:

$$G(s) = \frac{14.1585 \cdot e^{-0.153 \cdot s}}{0.1045 \cdot s + 1} \tag{5}$$

Using the linear approximation of Padé(0,1) the following transfer function was obtained:

$$G(s) = \frac{14.1585}{0.0159 \cdot s^{2\,!} + 0.2571 \cdot s + 1} \tag{6}$$

4.5 System Identification Toolbox

In addition to the models obtained above, we used MATLAB's *toolbox System Identification*, which uses the output and input data to obtain the model.

With 2 poles and infinite zeros:

$$G(s) = \frac{1186}{s^2 + 16.49 \cdot s + 84.71} \tag{7}$$

With 1 pole and infinite zeros:

$$G(s) = \frac{65.36}{s + 4.656} \tag{8}$$

Using the function *step* to the transfer functions (2), (6), (7) and (8) the step response of $4.92V$ was obtained:

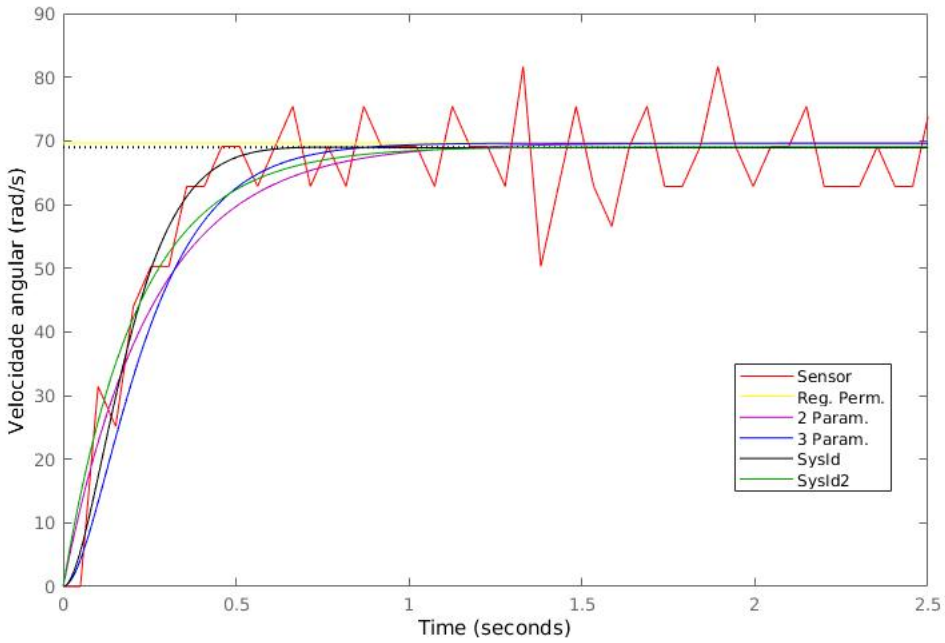

Figure 5: Graph for different approximations of the engine transfer function.

It is worth noting that the moving average filter was removed in order to eliminate the delay and be able to compare the curves more reliably. A good metric to choose the best transfer functions would be to calculate the error between the experimental curve (red) and the respective approximate functions and see which one has the smallest error. Unfortunately, there was no time during the experiment carried out for this, so the metric was used to determine which curves seem to come closest to the red one, which are the pink and blue curves, respectively, approximations (2) and (3). To validate the choices made, all approximate functions were simulated using the tool *Simulink* and using Routh's criterion for stability, the gains Kp, Ki and Kd were tested to see how the systems behaved in closed loop. Ultimately, $Simulink$ itself has a shortcut to the tool *PID Tuner*, which seeks gains that provide robustness to the system (noise rejection and reference tracking), therefore using the tool we found ourselves that the chosen models were able to provide a good response, as will be seen later.

5 Controller Project

As two approaches were chosen for the system, two controllers are proposed. Therefore, the closed-loop transfer function of approximation (2) was calculated:

$$\frac{Y(s)}{R(s)} = \frac{s^2 \cdot 14.1585 \cdot Kd + s \cdot 14.1585 \cdot Kp + 14.1585 \cdot Ki}{s^2 \cdot (0.2517 + 14.1585 \cdot Kd) + s \cdot (14.1585 \cdot Kp + 1) + 14.1585 \cdot Ki} \tag{9}$$

Applying Routh's criterion to obtain the range in which the controller gains generate a stable system, $Kd > -0.0178$, $Kp > -0.0706$ and $Ki > 0$ were obtained. Knowing the range of values in which the gains must be contained. As mentioned previously, *PID Tuner* was simulated together with *Simulink* to obtain gains with a certain robustness for the controller, such gains being: $Kp = 0.094472$, $Ki = 0.734462$ and $Kd = -0.000882$ where all gains comply with the conditions imposed by the Routh criterion, therefore these were used in the PID project. Therefore, the transfer function was as follows:

$$\frac{Y(s)}{R(s)} = \frac{s^2 \cdot -0.0125 + s \cdot 1.3376 + 10.3989}{s^2 \cdot 0.2392 + s \cdot 2.3376 + 10.3989} \tag{10}$$

The transfer function obtained is the curve as shown in figure 6:

..

Tabela 1: Your caption.

50. 5

Figure 6: System response (2) in closed loop with PID controller in Simulink.

When carrying out the procedure analogous to the last one but with the three-parameter model, it was observed that MATLAB's *PID Tuner* returned PID gain values that resulted in a physically unfeasible control signal (saturation), when removing the gain derivative it was observed that the control signal became viable to be generated, therefore it was decided to use a PI controller in this case. By closing the PI controller loop with the three-parameter model (6), the following transfer function was obtained:

$$\frac{Y(s)}{R(s)} = \frac{s \cdot 14.1585 \cdot Kp + 14.1585 \cdot Ki}{s^3 \cdot 0.016 s^2 \cdot 0.26 + s \cdot (14.1585 \cdot Kp + 1) + 14.1585 \cdot Ki} \tag{11}$$

Performing Routh's criterion again, the following ranges of gains were obtained $Kp > -0.07$ and $0 < Ki < Kp + 17.26$. As the gains that the *PID Tuner* tool returned are contained in this range, they were used with $Kp = 0.1341$ and $Ki = 0.7349$. In this way, the following closed-loop transfer function was obtained:

$$\frac{Y(s)}{R(s)} = \frac{s \cdot 1.8987 + 10.4051}{s^3 \cdot 0.016 + s^2 \cdot 0.26 + s \cdot 2.8987 + 10.4051} \tag{12}$$

The transfer function obtained generated the following curve:

50. 5

Figure 7: System response (6) in closed loop with PI controller in Simulink

The "optimal"gains returned by *PID Tuner* were: (consequently those used in the experiments)

Model	Kp	Ki	Kd	Controller
(2)	0.094471	0.73446	-0.000882	1
(6)	0.134078,	0.73489	0	2

Frame 1

6 Experimental results

To better evaluate the proposed controllers, it is interesting to evaluate which controller provides the lowest error, therefore a good metric to use is *Integral of the Square of the Error* or ISE, it is basically squaring each element of the error vector then add them [3].

PWM (bits)	ISE (rad$^2/s^2$)	Controller	Model
120	$9.3345 * 10^4$	No	X
75	$8.7627 * 10^4$	No	X
120	$8.8662 * 10^4$	1	(2)
75	$1.0561 * 10^4$	1	(2)
120	$1.0534 * 10^5$	2	(6)
75	$9.9548 * 10^3$	2	(6)

Table 2

From Table 1 it can be inferred that the best controller was the PID with the three-parameter model, as it presented the lowest error. It is interesting to note that for high speed (high PWM) the measurements begin to be very inconsistency and therefore even with the controller the error does not decrease, this is mainly due to the sensor.

To try to better evaluate the controller, a disturbance was added to the system in order to remove it from its reference, again using ISE as a metric:

PWM (bits)	ISE (rad$^2/s^2$)	Controller
120	$1.2112 * 10^5$	1
75	$2.6285 * 10^4$	1
120	$1.3101 * 10^5$	2
75	$2.4900 * 10^4$	2

Table 3

Another metric to analyze whether the presence of the controller in this system is to draw graphs of the motor speed with the wheel when: there is no controller, with the controller, with the controller with external disturbance, as was done for the 120-bit PWM, ie reference of $59.66 rad/s$:

50. 5

Figure 8: System response without controller, with controller and with controller + disturbance

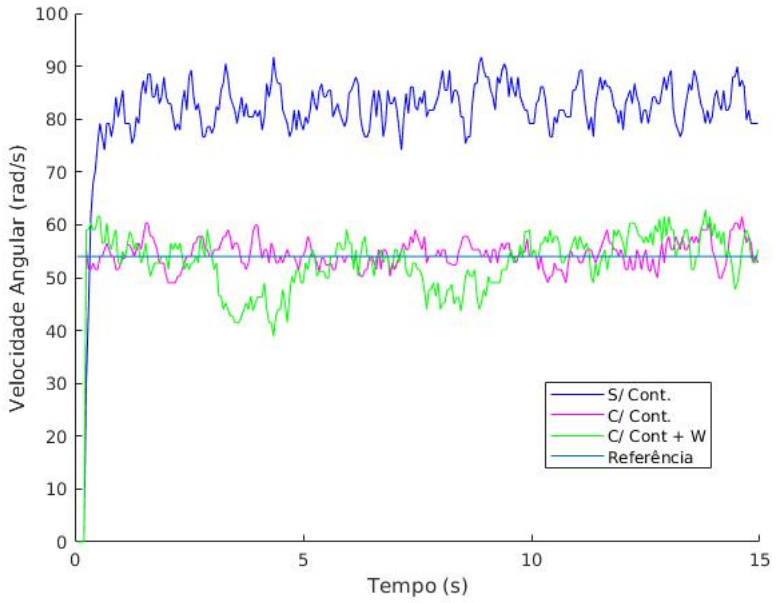

Figure 9: System response without controller, with controller and with controller + disturbance

When the wheel speed is lower (54 rad/s), the sensor is able to work in a better range, so it can be inferred from figure 9 that the pink curve actually stays around the reference (blue straight line), and even with disturbances to the system (green curve) it remains around the reference. It can be seen that the lack of a controller actually makes a difference, as the blue curve is completely outside the reference.

With the higher speed (with 69.66 rad/s) or 120 bits of PWM it is clear that there are many errors associated with the measurements, resulting in the reference speed being completely different from the real speed, whether with the controller or without.

7 Conclusion

After reading the experimental data and obtaining two models, together with the PWM variation between the values of 120 and 75 bits, it is clear that the larger the PWM signal, the greater the errors, i.e., at high speeds, problems occur. accuracy in sensor reading. In addition to the fact that the two-parameter model with a 75-bit PWM provides the lowest quadratic error and this was due to the fact that it is at lower speeds and the gain Kd is different from zero (only in the two-parameter model), making it respond faster.

The codes for both the analytical calculations and the implementation of the controller logic on Arduino can be found at the following link: `https://github.com/hpoleselo/ssfr`

1

Control System Design by Karl Johan Aestroem

MATLAB Source Code, Own Authorship. https://github.com/hpoleselo/SSFR/tree/master/PlantModel/WheelModel.

Dorf, RC; Bishop, R.h. Modern Control Systems. 5th edition