UNIVERSITY OF SALERNO

DEPARTMENT OF INFORMATION AND ELECTRICAL ENGINEERING AND APPLIED MATHEMATICS

Report Digital Control Systems Design

Control of a DC motor with state feedback

Francesco Avallone - 0622701488 - f.avallone20@studenti.unisa.it
Lorenzo Pagliara - 0622701576 - l.pagliara5@studenti.unisa.it

2021 - 2022

Indice

1 Formulation of the control problem formulation at high level

Nowadays, DC motors are widely used in many different technology applications,in particular in the robotics field for the robot's joints, but also in the locomotion field and others. In these contests, it has a significative role the definition of a control where it is possible to assign a position or velocity reference.In this document has been developed both control approaches:

- position control;
- velocity control.

For both approaches, it has been used the advanced *state feedback control* technique, starting from the state space of the mathematical models of the same process on which apply the control approaches mentioned before.

2 Mathematical model

2.1 Mathematical model for the position controller

The DC motor mathematical model in the state space form, which is obtained from the DC motor electrical and mechanical equations, is the following second-order system where the state's variables are the angular position θ and the angular velocity ω:

$$\begin{aligned} \begin{bmatrix} \dot{\theta} \\ \dot{\omega} \end{bmatrix} &= \begin{bmatrix} 0 & 1 \\ 0 & -\dfrac{1}{\tau_m} \end{bmatrix} \begin{bmatrix} \theta \\ \omega \end{bmatrix} + \begin{bmatrix} 0 \\ \dfrac{k_m}{\tau_m} \end{bmatrix} v \\ y &= \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \omega \end{bmatrix} \end{aligned} \tag{1}$$

The control input is the tension applied to the motor v which values are beetween $0V$ and $12V$, while the ouput is the angular position θ in rad.

R	3.2Ω
L	$8.2mH$
K_e	$0.85Vs/rad$
K_t	$0.85Nm/A$
b	$0.016Nms/rad$
I	$0.0059Nms^2/rad$

Tabella 1: Motor parameters.

Considering the parameters in Table 1, it is possible to obtain the following state space model:

$$\begin{aligned} \begin{bmatrix} \dot{\theta} \\ \dot{\omega} \end{bmatrix} &= \begin{bmatrix} 0 & 1 \\ 0 & -38.27 \end{bmatrix} \begin{bmatrix} \theta \\ \omega \end{bmatrix} + \begin{bmatrix} 0 \\ 45.02 \end{bmatrix} u \\ y &= \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \omega \end{bmatrix} \end{aligned} \tag{2}$$

2.2 Mathematical model for the velocity controller

In the same way, applying the Laplace trasformation and multiplying for $60/2\pi$, it has been obtained the transfer function of the DC motor. The control input is the tension v and the ouput is the angular velocity ω in RPM.

$$G(s) = \frac{\omega(s)}{v(s)} = \frac{9.5493k}{(sL+R)(sI+b)+k^2} \tag{3}$$

Considering the DC motor parameters in Table 1, it is possible to obtain the relative transfer function:

$$G(s) = \frac{\omega(s)}{v(s)} = \frac{167773.98}{s^2+392.96s+15992.15} \tag{4}$$

3 Tecniche di controllo

For both control approaches (position and speed) the state feedback technique has been used. The starting point for the application of this technique is the obtaining of a representation in the state space. In the case of the control in position the starting model already met this requirement, while for the speed control it was necessary to carry out the transfer function.

Having obtained both representations in the state space, the first step was to verify the reachability and the observability. Having made sure that both of these requirements were verified,and then the models were extended with a fictitious state, representing the full error, for the rejection of constant disturbances. In order to obtain a law of discrete control, such extended representations were then converted into models in the discrete time state space. The latter were then used to design a feedback state controller, whose gains were obtained through LQR.

Due to the non-observability of the state, for both approaches, a Luenberger observer, also obtained by the LQR technique, was made.
For a more complete view of the design steps followed, please refer to the MATLAB code, which is considered an integral part of this document.

Velocity control, **Position control**

The same control approaches were then implemented using the *direct coding*technique, the links, which reference to the source files, are written below.

Controllo in posizione direct coding, **Controllo in velocità direct coding**

4 Results obtained and discussion of performance

Both for the control in position and for the control in speed, it has been followed the model based development which previews a phase of testing for each step of development. In particular, the validation tasks were carried out:

1. Model in the Loop (MIL);
2. Software in the Loop (SIL);
3. Processor in the Loop (PIL);
4. Test sul processo reale;

It should be noted that the test phase on the real process must be preceded by the Hardware in the Loop (HIL) phase, which was not carried out due to the unavailability of the appropriate instrumentation.

4.1 Results with control in position

For the simulations of the control in place the following references were used: 0, -2π, π, 2π, 0, updated every $2s$ through a Stateflow chart.

From the various simulations corresponding to the different validation tasks it can be noticed that the system response remains almost unchanged, with a settling time less than 1 second and a behavior quite free of overshoot. The slightly more relevant peaks are obtained when the reference changes in form by a value of at least 2π. This behavior depends on the presence of the integral action, which in the absence of an anti windup scheme, with larger errors makes the control input saturated at the limit of the actuator.

In the figures 1, 2, 4 are reported the relative outputs in the MIL, SIL, PIL phase. In addition, in the figures 3 and 5 are reported the relative execution time of SIL and PIL phase. As can be expected, SIL validation took time considerably lower than PIL.

Particular attention should be paid to the execution of the control algorithm on the physical engine. The responses of the auto-generated algorithm and the direct coding algorithm respectively are given in Figure 6 e in Figure 7. From these figures it is possible to notice that the motor response has a slightly oscillatory behavior around the reference value. This depends on the physical resolution of the motor that does not allow to obtain an angular position perfectly coinciding with the reference.

4.2 Risultati con controllo in velocità

The following references were used for speed control simulations: 0, 80, 125, 0, -80, -125, updated every $2s$ through a Stateflow chart.

Also in this case in the simulations correspondent the answer of the system remains almost unchanged, with times of settling of approximately half second and a behavior totally free of overshoot.

In the Figures 8, 9, 11 are reported the relative motor outputs in the MIL, SIL, PIL validation phases. In addition, in the figures 10 and 12 are reported the relative execution times of the SIL and PIL validation. As can be expected, SIL validation took time considerably lower than PIL.

Particular attention should be paid to the execution of the control algorithm on the physical engine. The responses to the auto-generated algorithm and the direct coding algorithm respectively are given in Figure 13 and in Figure 14. From the figures it can be noticed that the response of the motor has of the continuous peaks due to the error of the estimate of the speed because of the loss of ticks of the encoder.

A Simulations

A.1 Position control

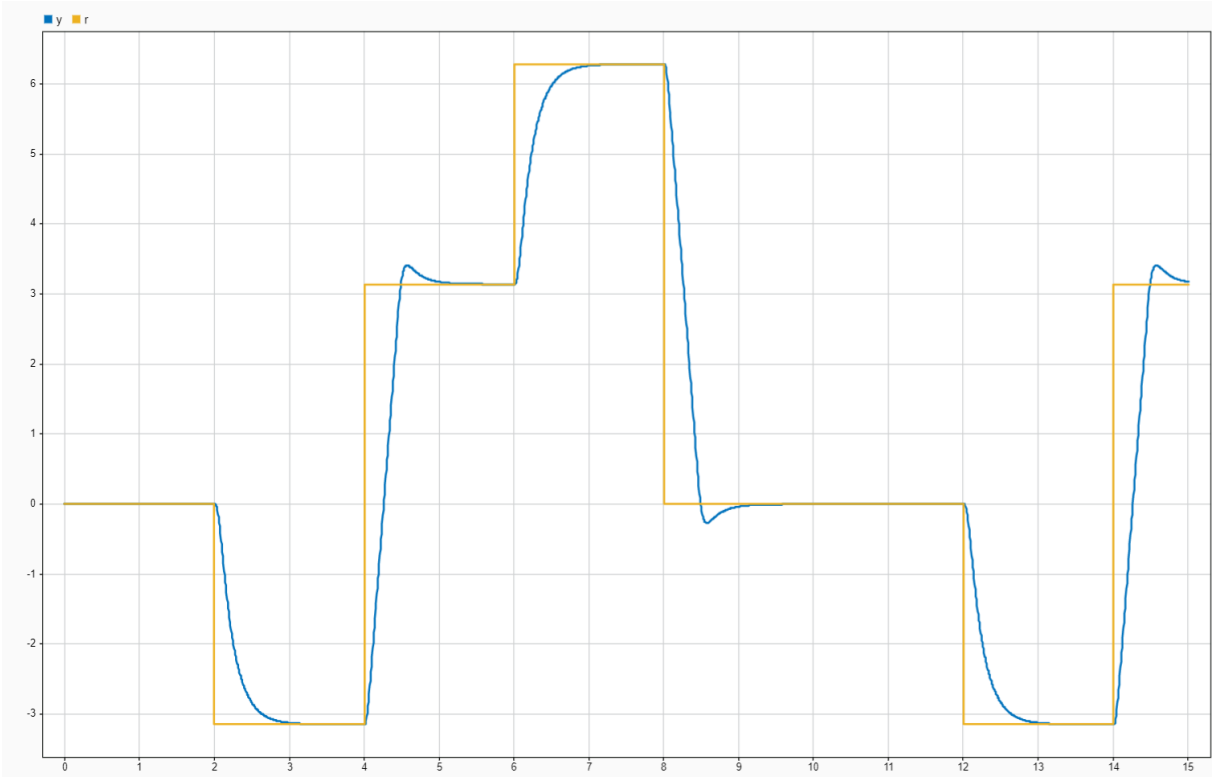

Figura 1: MIL Simulation with position control

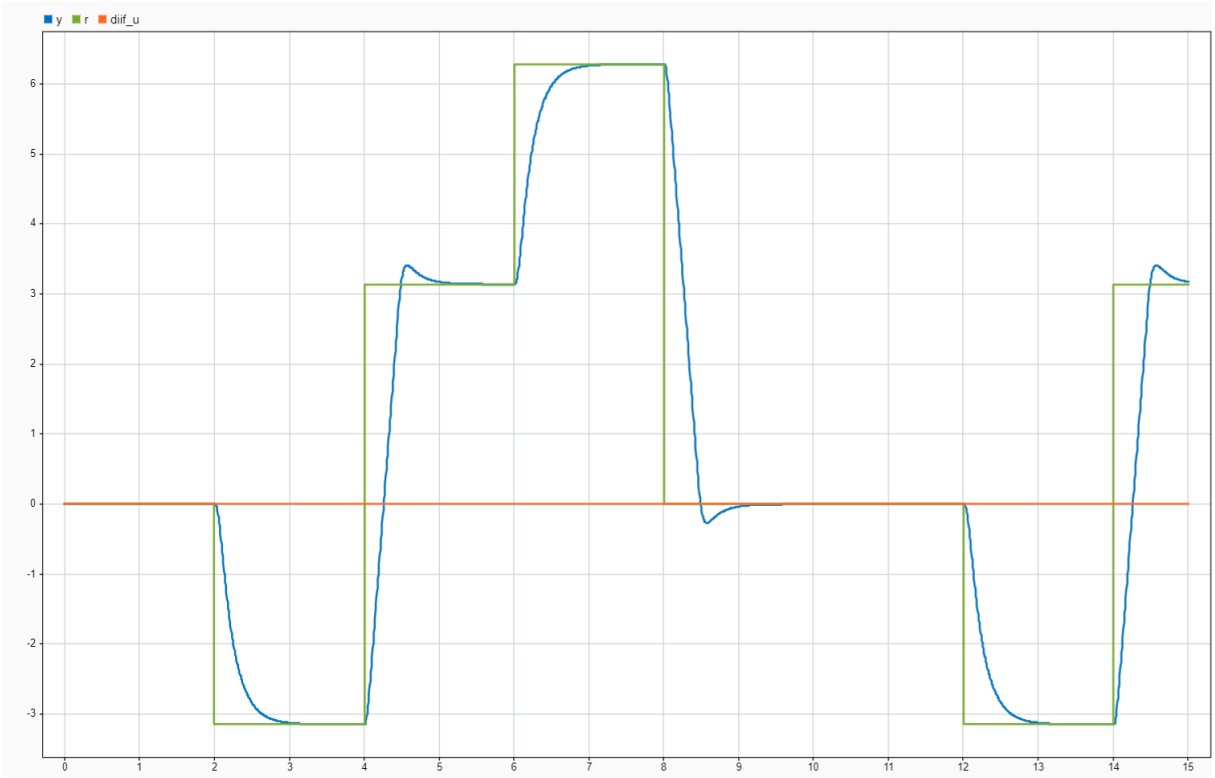

Figura 2: SIL Simulation with position control

Code Execution Profiling Report for Position_State_Feedback_SIL_Simulation/State Feedback Controller (SIL)

The code execution profiling report provides metrics based on data collected from a SIL or PIL execution. Execution times are calculated from data recorded by instrumentation probes added to the SIL or PIL test harness or inside the code generated for each component. See Code Execution Profiling for more information.

1. Summary

Total time	388678
Unit of time	ns
Command	report(executionProfile, 'Units', 'seconds', 'ScaleFactor', '1e-09', 'NumericFormat', '%0.0f');
Timer frequency (ticks per second)	2.901e+09
Profiling data created	28-May-2022 15:01:10

2. Profiled Sections of Code

Section	Maximum Execution Time in ns	Average Execution Time in ns	Maximum Self Time in ns	Average Self Time in ns	Calls	
Controller_Observer__initialize	7427	7427	7427	7427	1	
Controller_Observer_Model_Init	34	34	34	34	1	
Controller_Observer_Model [0.005 0]	80637	127	80637	127	3001	

3. CPU Utilization [hide]

Task	Average CPU Utilization	Maximum CPU Utilization
Controller_Observer_Model [0.005 0]	0.002541%	1.613%
Overall CPU Utilization	0.002541%	1.613%

Figura 3: Execution times of the SIL simulation with position control

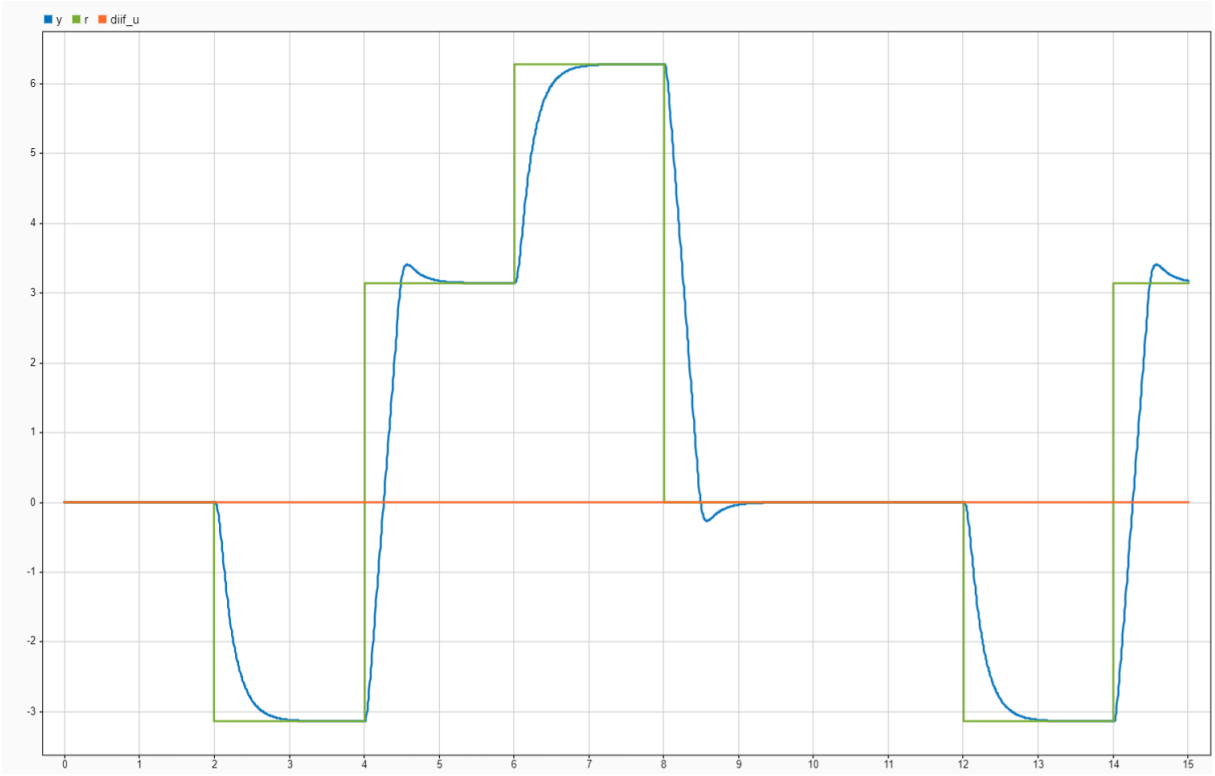

Figura 4: PIL Simulation with position control

Code Execution Profiling Report for Position_State_Feedback_PIL_Simulation/State Feedback Controller (PIL)

The code execution profiling report provides metrics based on data collected from a SIL or PIL execution. Execution times are calculated from data recorded by instrumentation probes added to the SIL or PIL test harness or inside the code generated for each component. See Code Execution Profiling for more information.

1. Summary

Total time	78735952
Unit of time	ns
Command	report(executionProfile, 'Units', 'seconds', 'ScaleFactor', '1e-09', 'NumericFormat', '%0.0f');
Timer frequency (ticks per second)	8.4e+07
Profiling data created	28-May-2022 09:55:10

2. Profiled Sections of Code

Section	Maximum Execution Time in ns	Average Execution Time in ns	Maximum Self Time in ns	Average Self Time in ns	Calls	
Controller_Observer__initialize	1500	1500	1500	1500	1	
Controller_Observer_Model_Init	1738	1738	1738	1738	1	
Controller_Observer_Model [0.005 0]	27893	26235	27893	26235	3001	

3. CPU Utilization [hide]

Task	Average CPU Utilization	Maximum CPU Utilization
Controller_Observer_Model [0.005 0]	0.5247%	0.5579%
Overall CPU Utilization	0.5247%	0.5579%

Figura 5: Execution times of the PIL simulation with position control

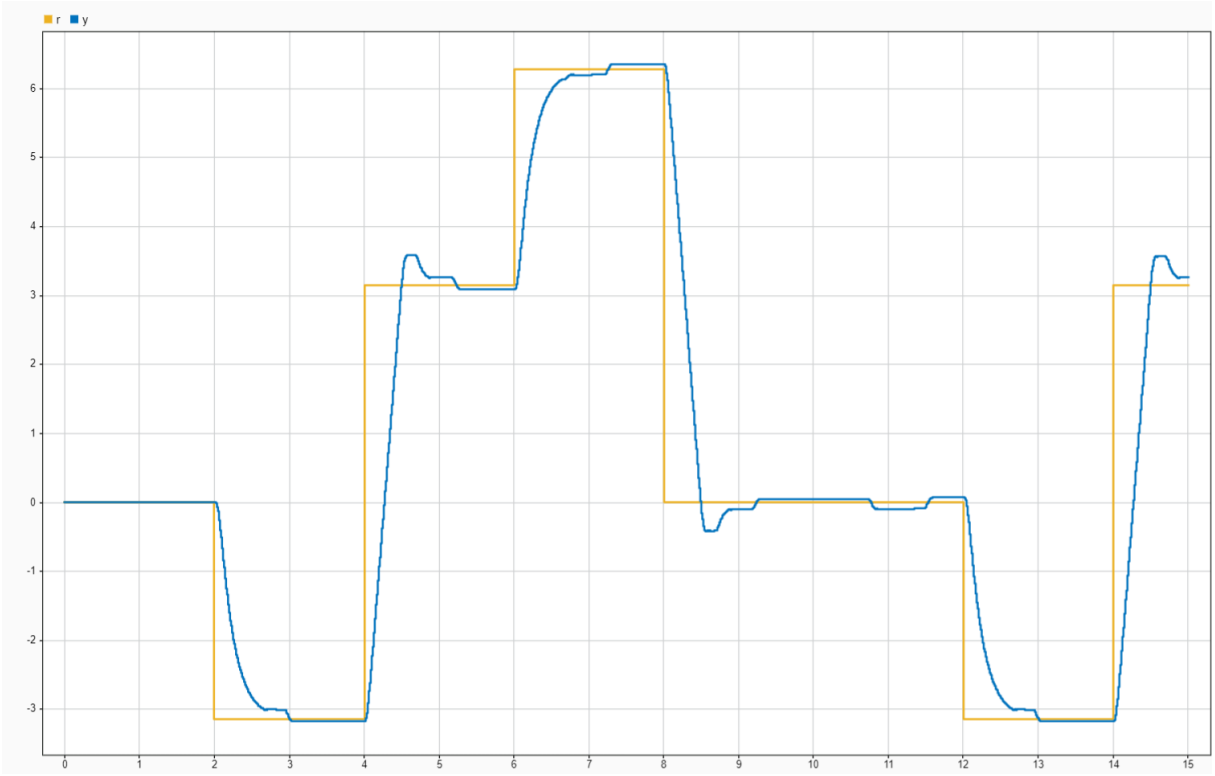

Figura 6: Execution on the real motor of the position control, with the auto-generated code

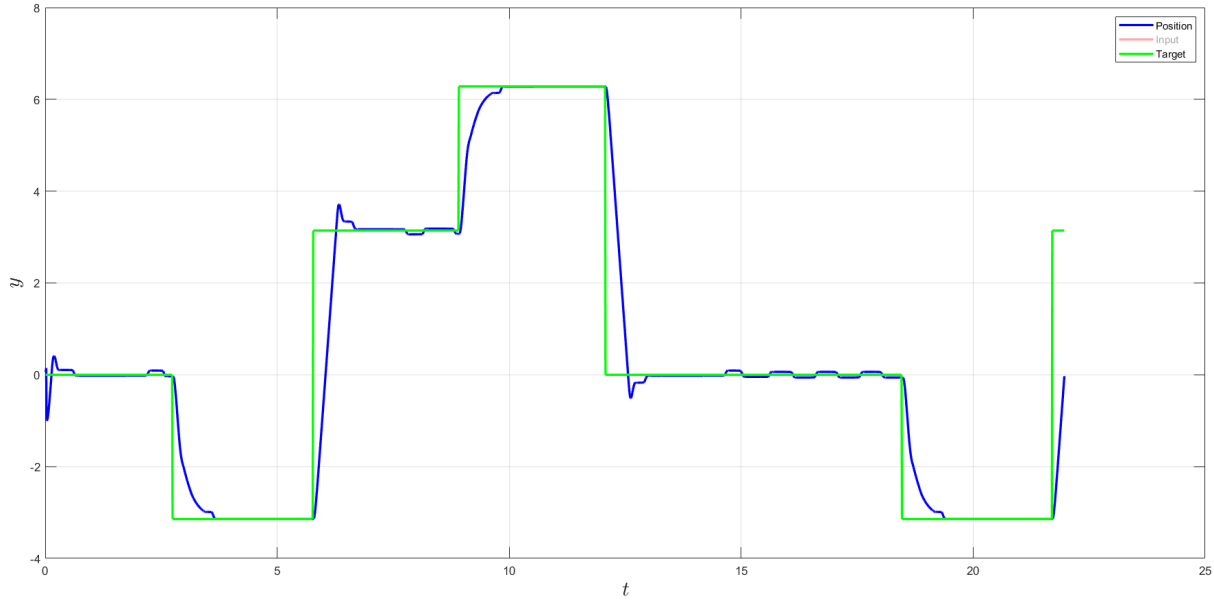

Figura 7: Execution on the real motor of the position control, with the direct coding approach

A.2 Velocity control

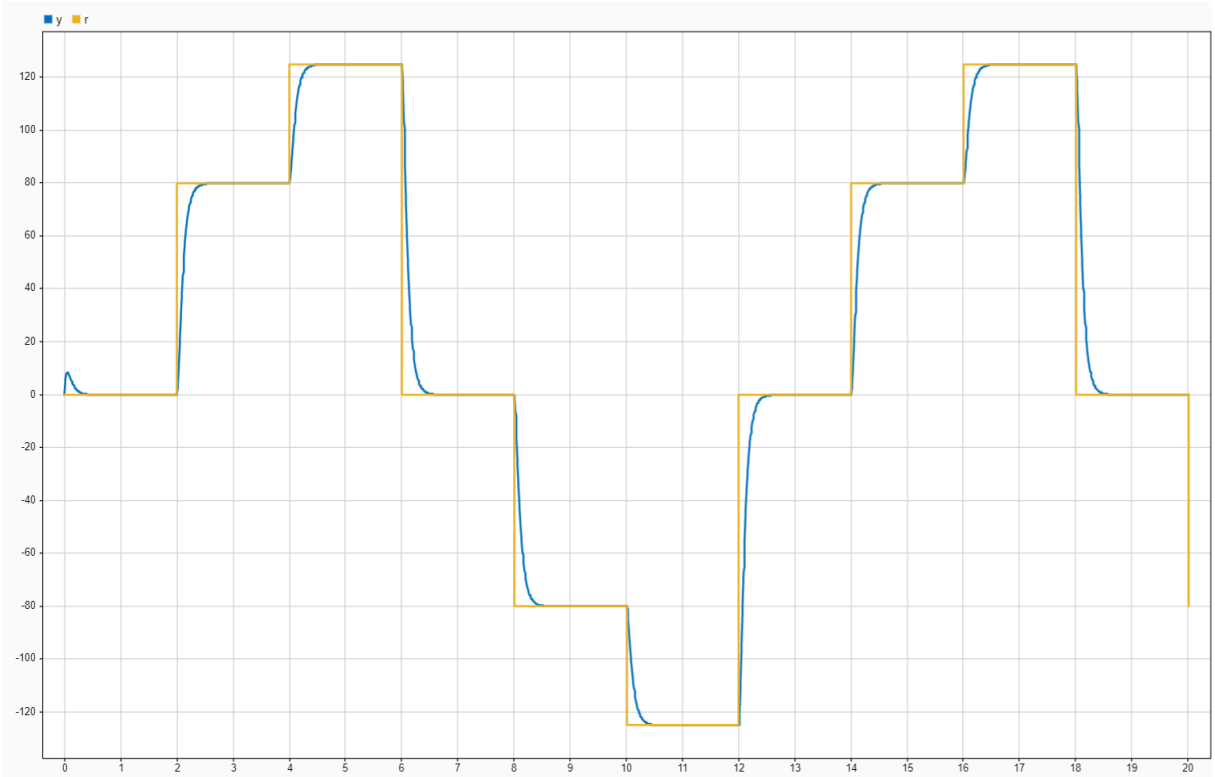

Figura 8: MIL simulation for velocity control.

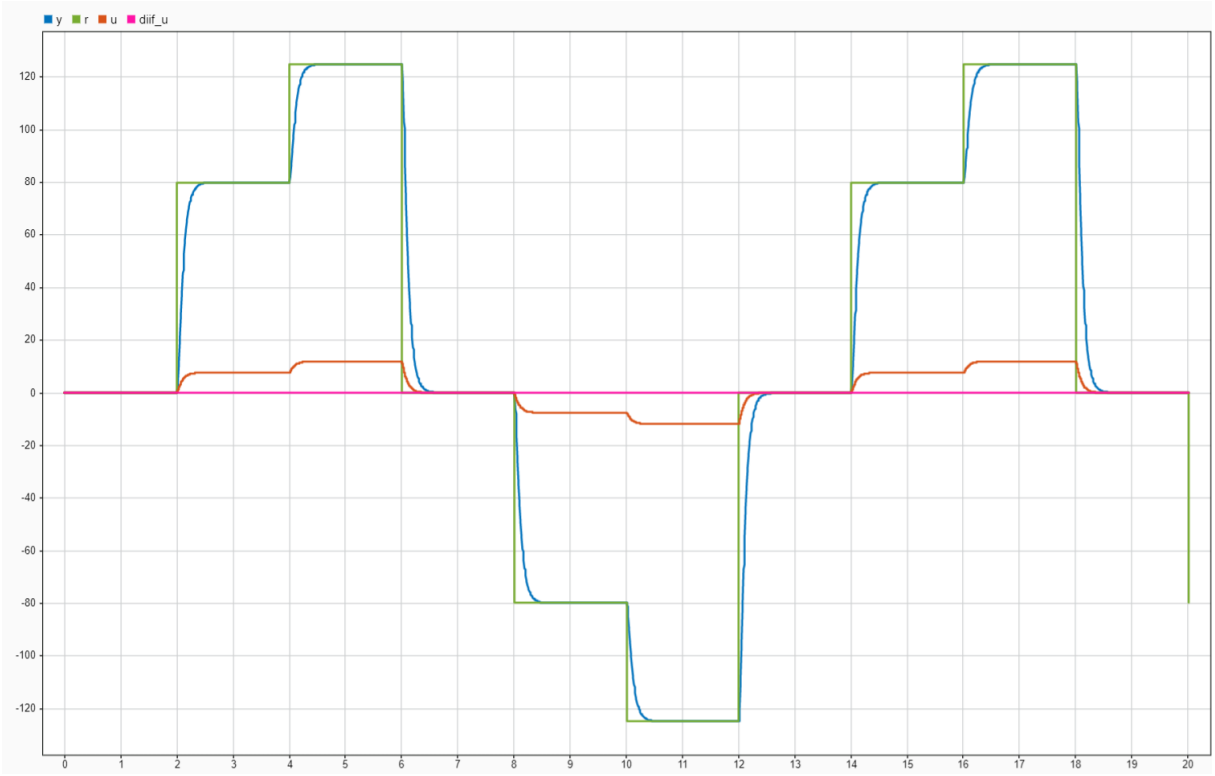

Figura 9: SIL simulation with velocity control.

Code Execution Profiling Report for Speed_State_Feedback_SIL_Simulation/State Feedback Controller (SIL)

The code execution profiling report provides metrics based on data collected from a SIL or PIL execution. Execution times are calculated from data recorded by instrumentation probes added to the SIL or PIL test harness or inside the code generated for each component. See Code Execution Profiling for more information.

1. Summary

Total time	449037
Unit of time	ns
Command	report(executionProfile, 'Units', 'seconds', 'ScaleFactor', '1e-09', 'NumericFormat', '%0.0f');
Timer frequency (ticks per second)	2.901e+09
Profiling data created	28-May-2022 15:08:11

2. Profiled Sections of Code

Section	Maximum Execution Time in ns	Average Execution Time in ns	Maximum Self Time in ns	Average Self Time in ns	Calls	
Controller_Observer__initialize	26182	26182	26182	26182	1	
Controller_Observer_Model_Init	18	18	18	18	1	
Controller_Observer_Model [0.005 0]	9215	106	9215	106	4001	

3. CPU Utilization [hide]

Task	Average CPU Utilization	Maximum CPU Utilization
Controller_Observer_Model [0.005 0]	0.002114%	0.1843%
Overall CPU Utilization	0.002114%	0.1843%

Figura 10: Execution times of SIL simulation with velocity control

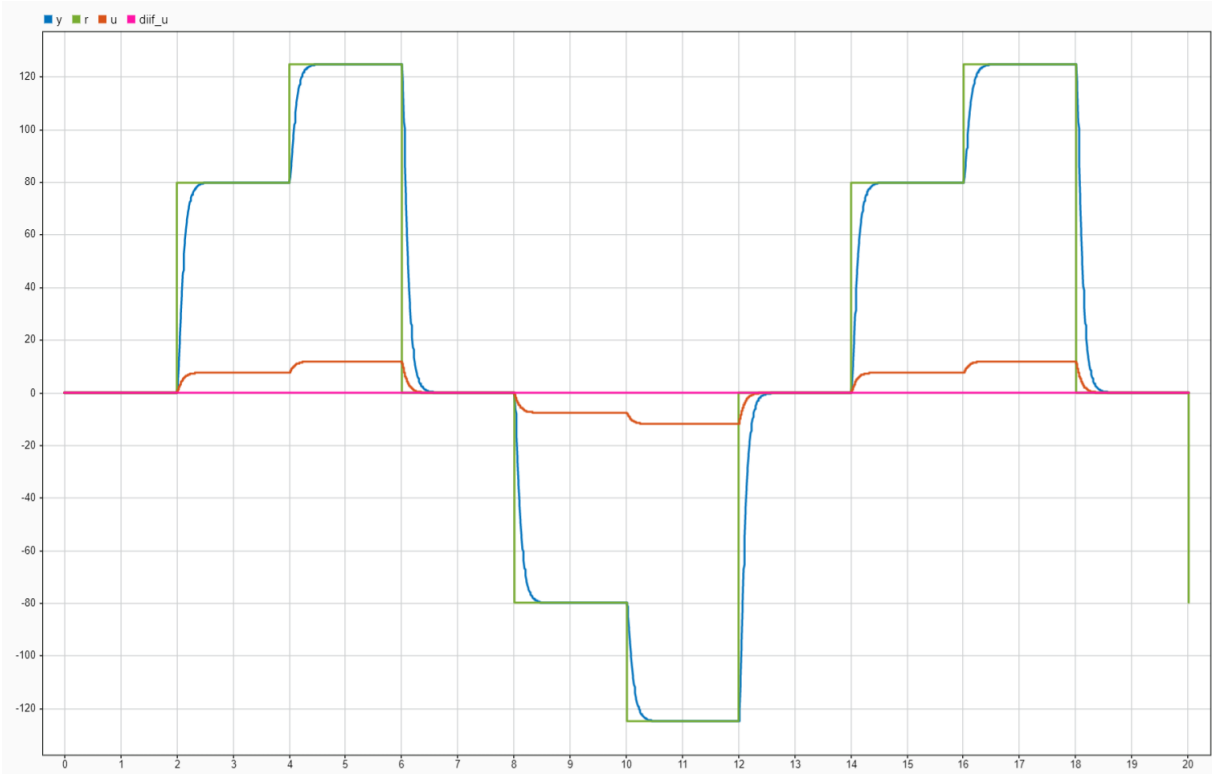

Figura 11: PIL simulation with control velocity.

Code Execution Profiling Report for Speed_State_Feedback_PIL_Simulation/State Feedback Controller (PIL)

The code execution profiling report provides metrics based on data collected from a SIL or PIL execution. Execution times are calculated from data recorded by instrumentation probes added to the SIL or PIL test harness or inside the code generated for each component. See Code Execution Profiling for more information.

1. Summary

Total time	112392690
Unit of time	ns
Command	report(executionProfile, 'Units', 'seconds', 'ScaleFactor', '1e-09', 'NumericFormat', '%0.0f');
Timer frequency (ticks per second)	8.4e+07
Profiling data created	28-May-2022 10:28:04

2. Profiled Sections of Code

Section	Maximum Execution Time in ns	Average Execution Time in ns	Maximum Self Time in ns	Average Self Time in ns	Calls
Controller_Observer__initialize	1500	1500	1500	1500	1
Controller_Observer_Model_Init	1738	1738	1738	1738	1
Controller_Observer_Model [0.005 0]	29524	28090	29524	28090	4001

3. CPU Utilization [hide]

Task	Average CPU Utilization	Maximum CPU Utilization
Controller_Observer_Model [0.005 0]	0.5618%	0.5905%
Overall CPU Utilization	0.5618%	0.5905%

Figura 12: Execution times of PIL simulation with velocity control.

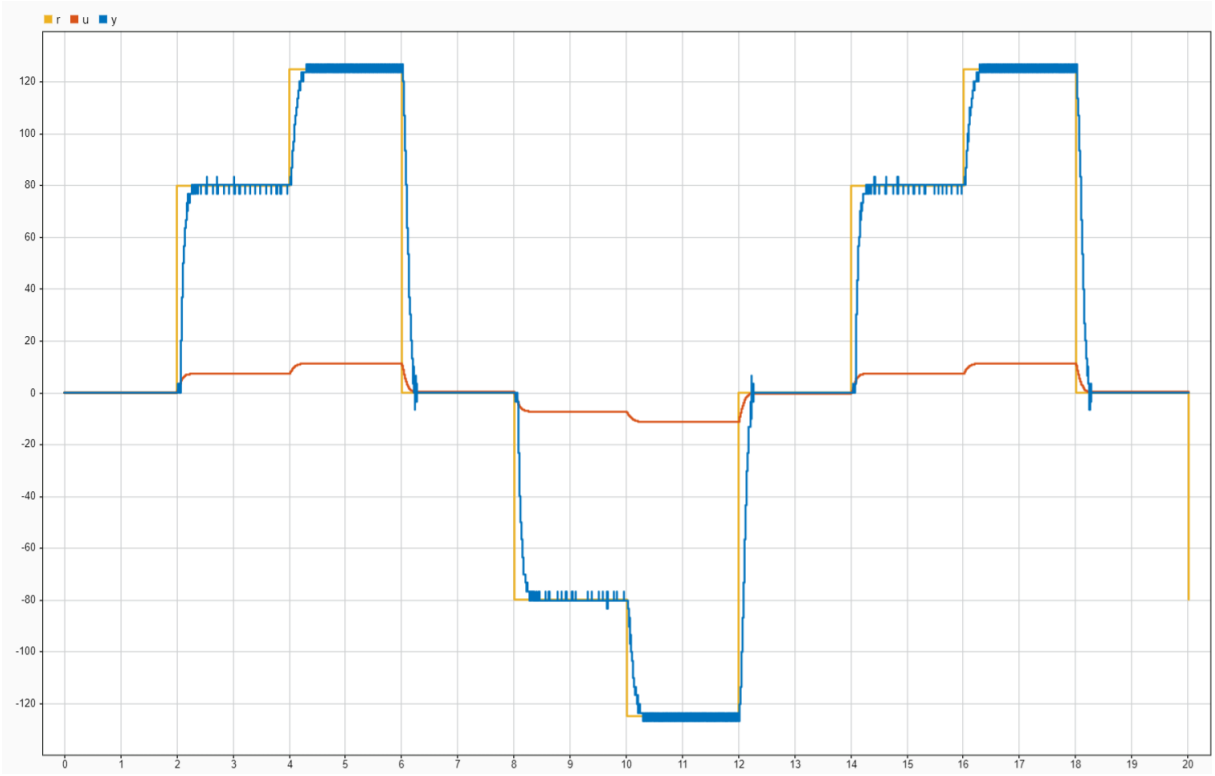

Figura 13: Execution on the real motor of the velocity control, with the auto-generated code

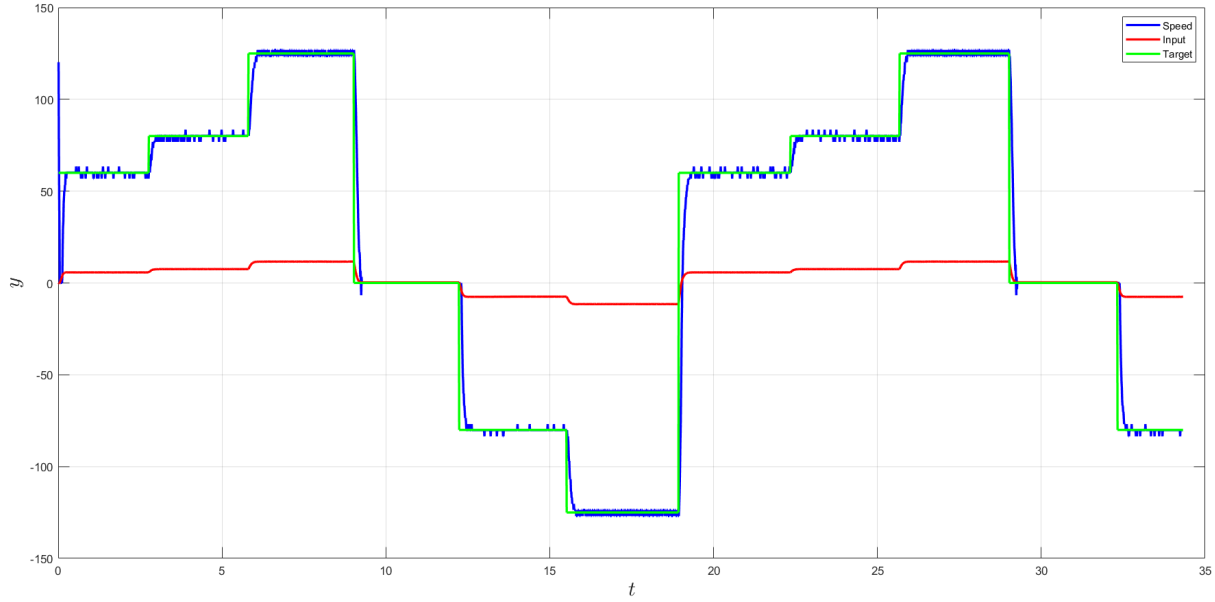

Figura 14: Execution on the real motor of the velocity control, with the direct coding approach