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 Modello matematico per controllo in velocità

Analogamente, applicando la trasformata di Laplace alle equazioni del motore e moltiplicando per $60/2\pi$, è stata ottenuta la funzione di trasferimento del motore DC, in cui l'ingresso di controllo è la tensione fornita al motore v e l'uscita è la velocità angolare ω, espressa in RPM:

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

Considerando i parametri del motore in dotazione in tabella 1, si ottiene la seguente funzione di trasferimento:

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

3 Tecniche di controllo

Per entrambi gli approcci di controllo (posizione e velocità) è stata utilizzata la tecnica della *retroazione di stato*. Il punto di partenza per l'applicazione di tale tecnica è l'ottenimento di una rappresentazione nello spazio di stato. Per il caso del controllo in posizione il modello di partenza già rispettava tale requisito, mentre per il controllo in velocità è stato necessario effettuare la realizzazione della funzione di trasferimento.

Ottenute entrambe le rappresentazioni nello spazio di stato, il primo passo effettuato è stato quello di verificarne la raggiungibilità e l'osservabilità. Accertatosi che entrambe verificassero tali requisiti, si è poi proceduto con l'estendere i modelli con uno stato fittizio, rappresentante l'integrale dell'errore, per la reiezione di disturbi costanti. Al fine di ottenere una legge di controllo discreta, tali rappresentazioni estese sono poi state convertite in modelli nello spazio di stato a tempo discreto. Quest'ultimi, infine, sono stati poi utilizzati per la progettazione di un controllore feedback state, i cui guadagni sono stati ottenuti mediante LQR.

Data la non osservabilità dello stato, per ambedue gli approcci, è stato realizzato un osservatore di Luenberger, anch'esso ottenuto mediante la tecnica dell'LQR.

Per una visione più completa riguardo i passi progettuali seguiti, si rimanda al codice MATLAB, considerato parte integrante di questo documento.

Controllo in velocità, **Controllo in posizione**

Gli stessi approcci di controllo sono poi stati implementati utilizzando la tecnica del *direct coding*, di cui di seguito sono riportati i link ai file sorgenti.

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

4 Risultati ottenuti e discussione delle performance

Sia per il controllo in posizione che per il controllo in velocità, è stato seguito il modello di sviluppo model based a V che prevede una fase di testing ad ogni step di sviluppo. In particolare sono stati effettuati i task di validazione:

1. Model in the Loop (MIL);
2. Software in the Loop (SIL);

3. Processor in the Loop (PIL);

4. Test sul processo reale;

È doveroso sottolineare che la fase di test sul processo reale deve essere preceduta dalla fase Hardware in the Loop (HIL), la quale non è stata effettuata per via dell'indisponibilità dell'opportuna strumentazione.

4.1 Risultati con controllo in posizione

Per le simulazioni del controllo in posizione sono stati utilizzati i seguenti riferimenti: 0, -2π, π, 2π, 0, aggiornati ogni $2s$ mediante uno Stateflow chart.

Dalle diverse simulazioni corrispondenti ai diversi task di validazione si può notare che la risposta del sistema resta pressoché invariata, con tempi di assestamento inferiori al secondo e un comportamento abbastanza privo di sovra-elongazione. I picchi un po' più rilevanti si ottengono quando il riferimento cambia in modulo di un valore almeno pari a 2π. Tale comportamento dipende dalla presenza dell'azione integrale, che in assenza di uno schema anti windup, con errori più grandi fa saturare l'ingresso di controllo al limite dell'attuatore.

Nelle Figure 1, 2, 4 sono riportate rispettivamente le risposte del motore nella fase di validazione MIL, SIL, PIL. Inoltre, nelle Figure 3 e 5 sono riportati rispettivamente i tempi di esecuzione relativi alla validazione SIL e PIL. Come lecito aspettarsi, la validazione SIL ha richiesto tempi notevolmente inferiori a quella PIL.

Un'attenzione particolare va rivolta all'esecuzione dell'algoritmo di controllo sul motore fisico. Le risposte rispettivamente dell'algoritmo auto generato e di quello ottenuto mediante direct coding sono riportate in Figura 6 e in Figura 7. Dalle figure si può notare che la risposta del motore ha un comportamento leggermente oscillatorio attorno al valore del riferimento. Ciò dipende dalla risoluzione fisica del motore che non consente di ottenere una posizione angolare perfettamente coincidente con il riferimento.

4.2 Risultati con controllo in velocità

Per le simulazioni del controllo in velocità sono stati utilizzati i seguenti riferimenti: 0, 80, 125, 0, -80, -125, aggiornati ogni $2s$ mediante uno Stateflow chart.

Anche in questo caso nelle simulazioni corrispondenti ai diversi task di validazione la risposta del sistema resta pressoché invariata, con tempi di assestamento di circa mezzo secondo e un comportamento totalmente privo di sovra-elongazione.

Nelle Figure 8, 9, 11 sono riportate rispettivamente le risposte del motore nella fase di validazione MIL, SIL, PIL. Inoltre, nelle Figure 10 e 12 sono riportati rispettivamente i tempi di esecuzione relativi alla validazione SIL e PIL. Come lecito aspettarsi anche in questo, la validazione SIL ha richiesto tempi notevolmente inferiori a quella PIL.

Un'attenzione particolare va rivolta all'esecuzione dell'algoritmo di controllo sul motore fisico. Le risposte rispettivamente dell'algoritmo auto generato e di quello ottenuto mediante direct coding sono riportate in Figura 13 e in Figura 14. Dalle figure si può notare che la risposta del motore ha dei picchi continui dovuti all'errore della stima della velocità a causa della perdita di ticks dell'encoder.

A Simulazioni

A.1 Controllo in posizione

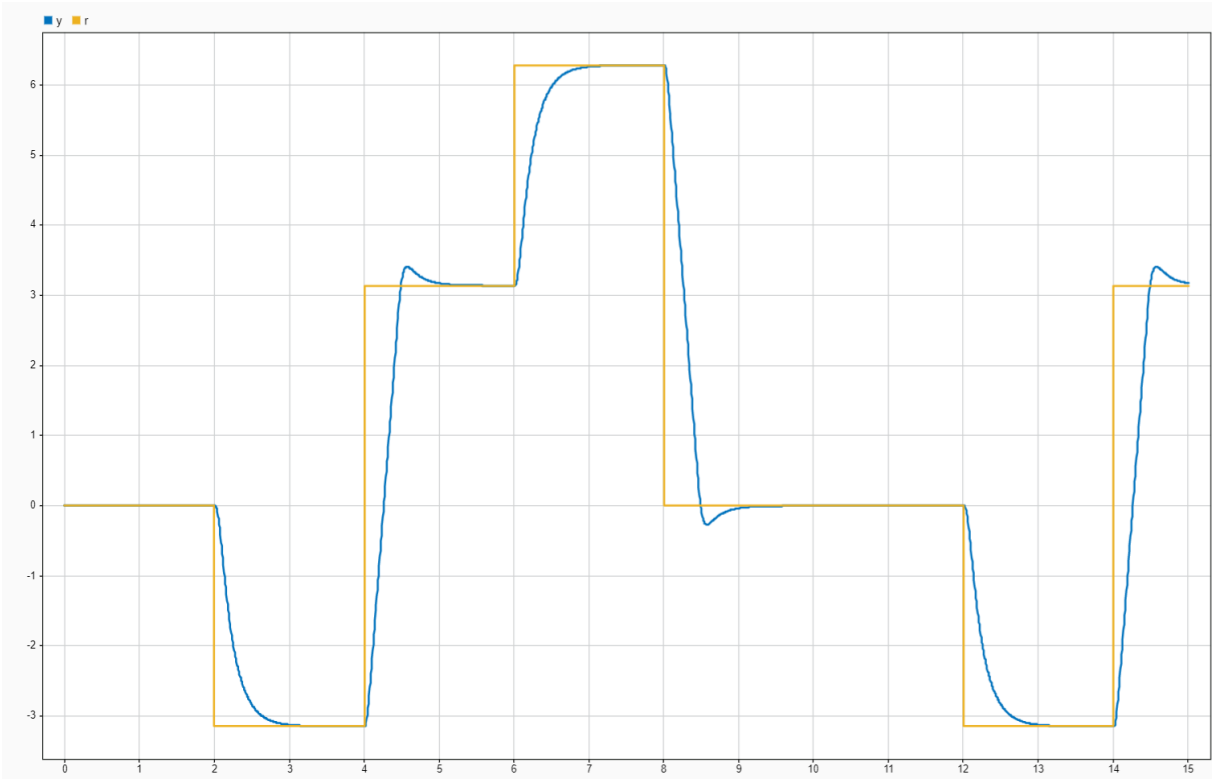

Figura 1: Simulazione MIL del controllo di posizione.

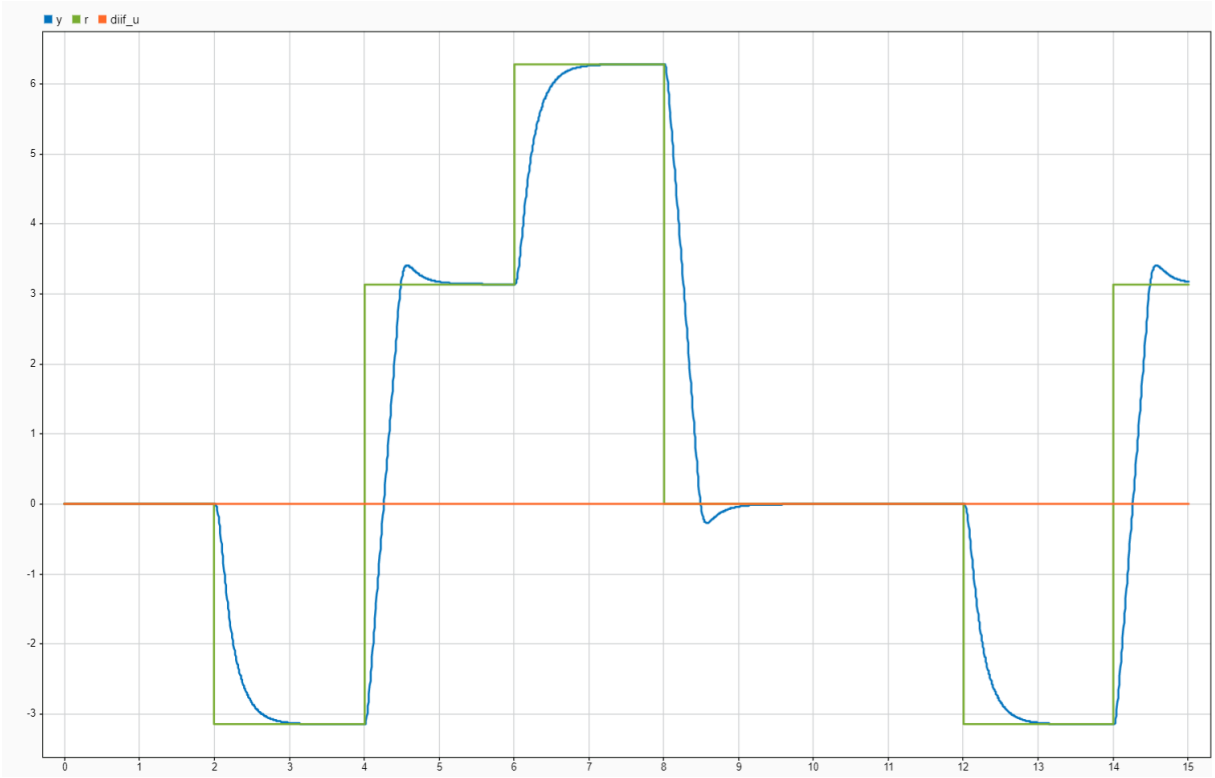

Figura 2: Simulazione SIL del controllo di posizione.

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: Tempi di esecuzione della simulazione SIL del controllo di posizione.

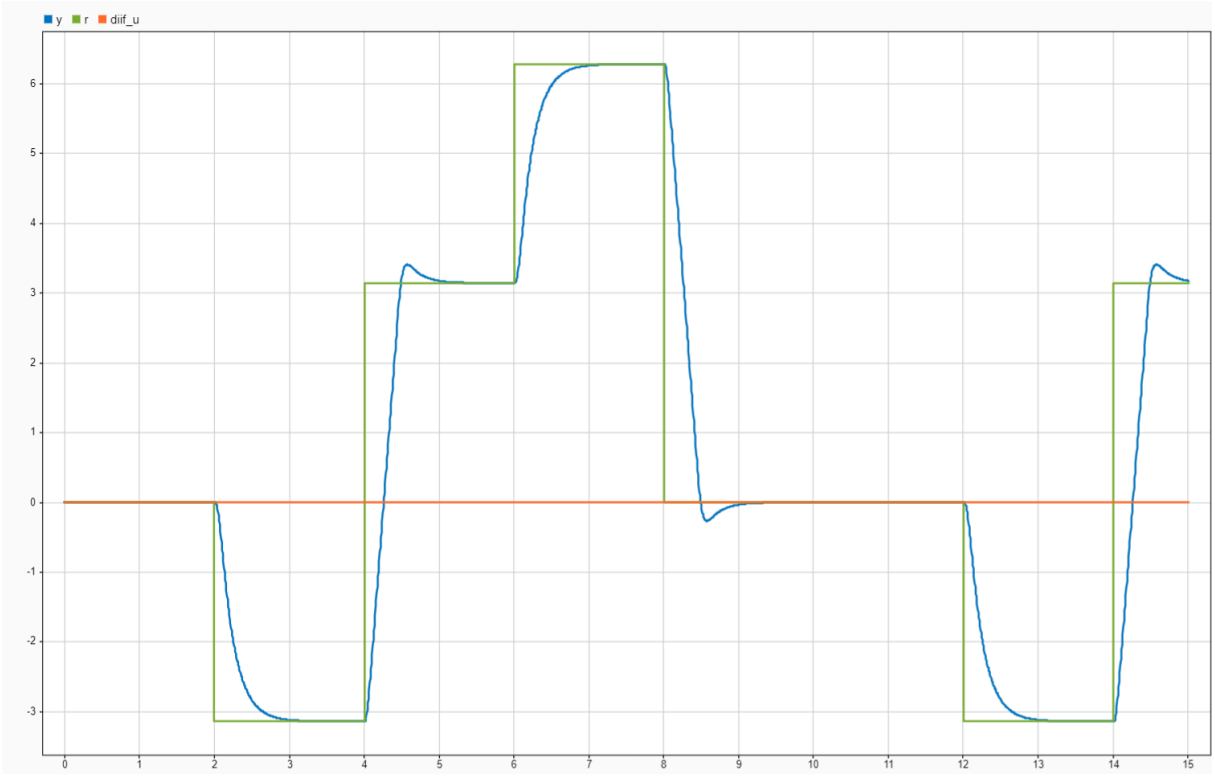

Figura 4: Simulazione PIL del controllo di posizione.

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: Tempi di esecuzione della simulazione PIL del controllo di posizione.

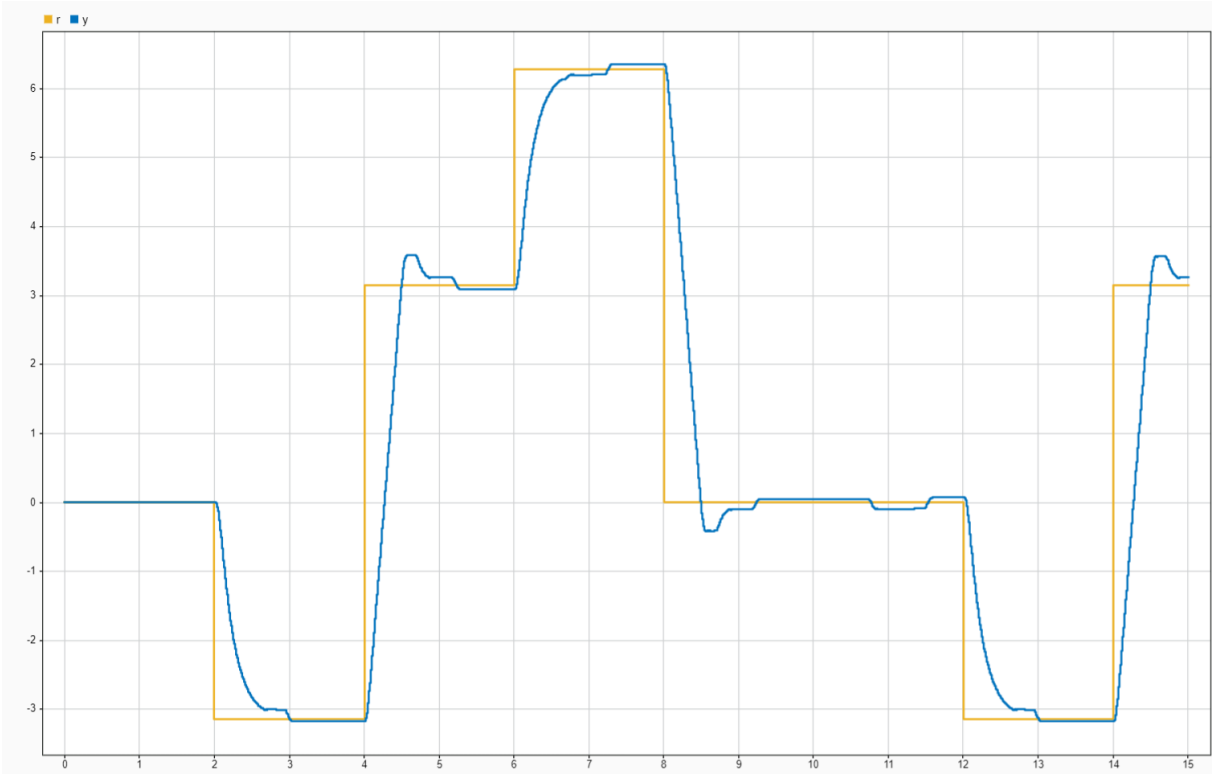

Figura 6: Esecuzione sul motore fisico del controllo di posizione, mediante auto generazione del codice.

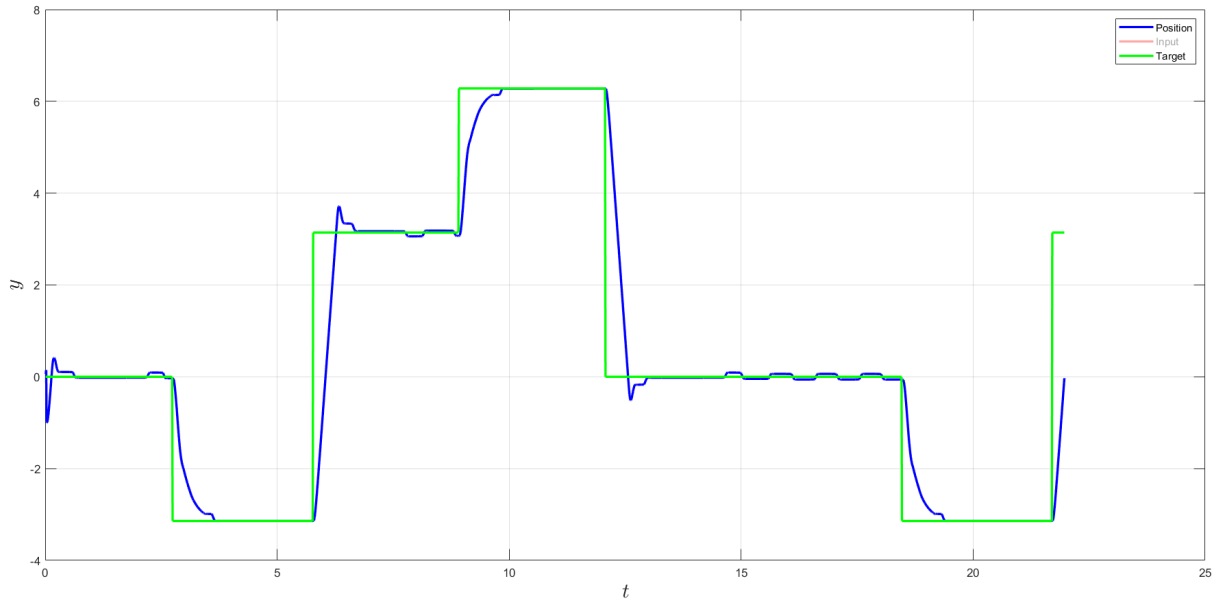

Figura 7: Esecuzione sul motore fisico del controllo di posizione, mediante la tecnica del direct coding.

A.2 Controllo in velocità

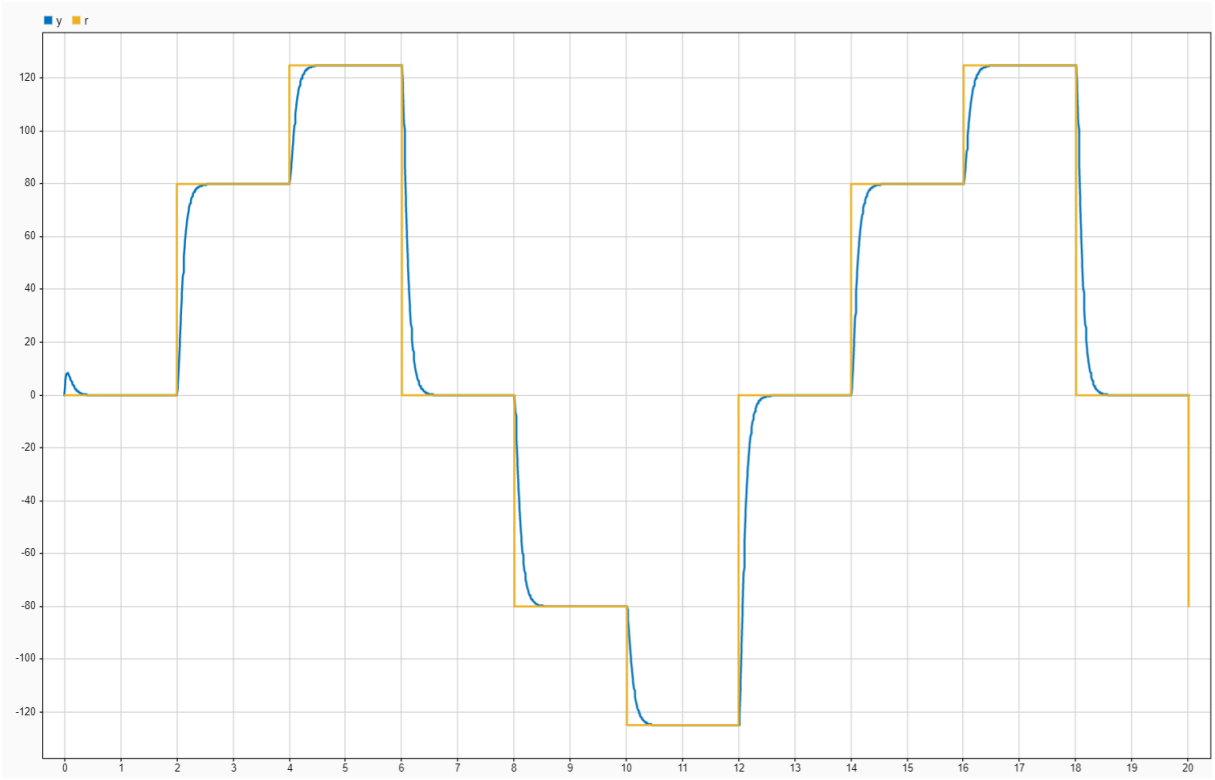

Figura 8: Simulazione MIL del controllo di velocità.

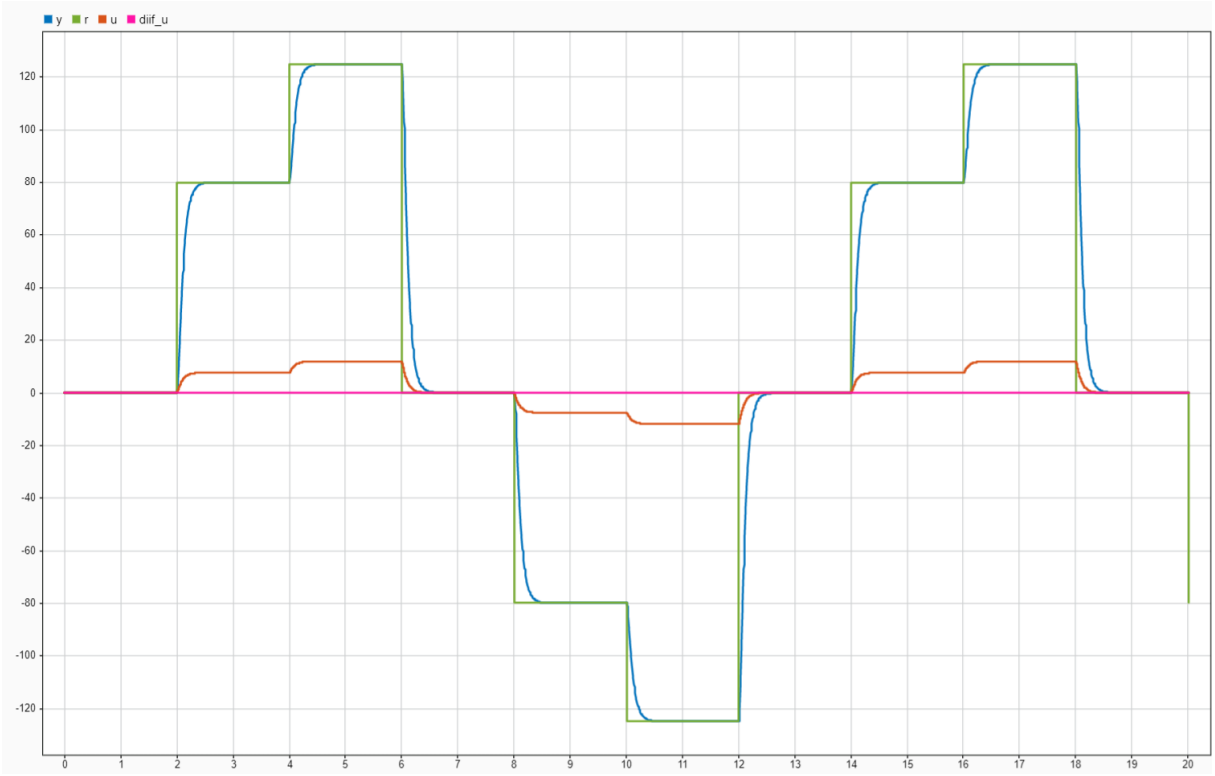

Figura 9: Simulazione SIL del controllo di velocità.

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: Tempi di esecuzione della simulazione SIL del controllo di velocità.

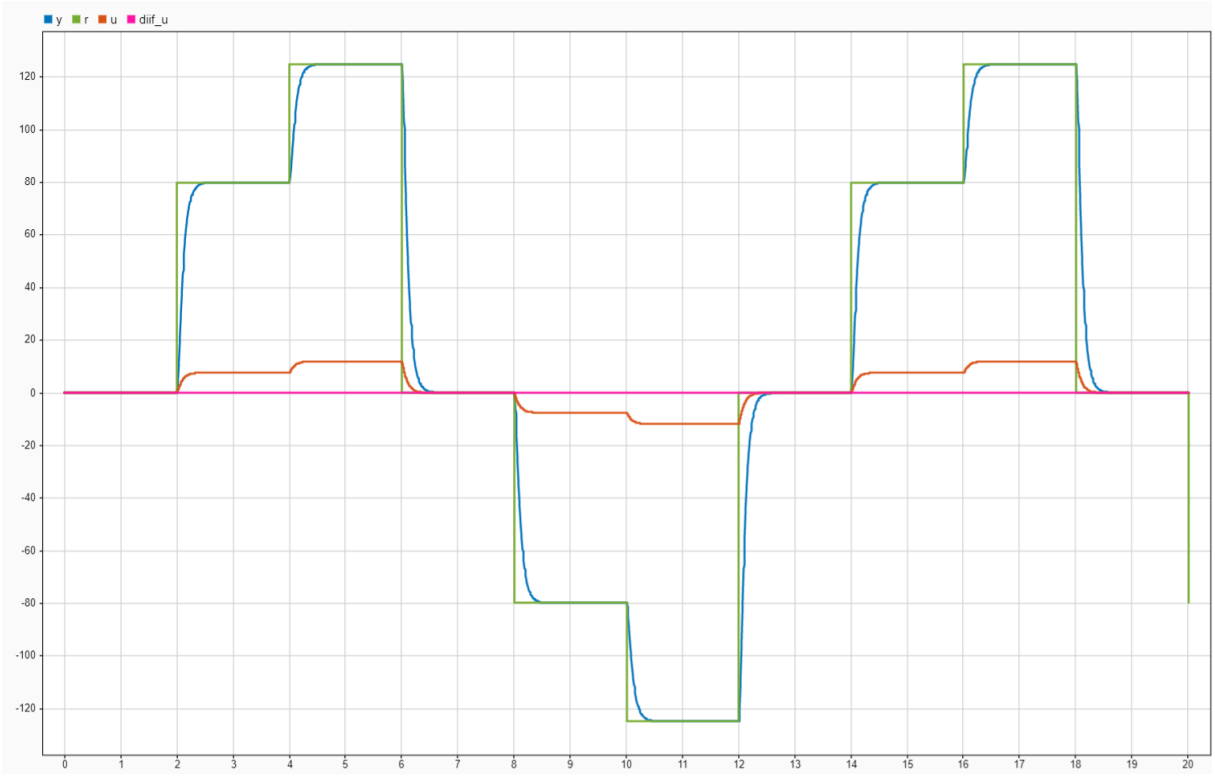

Figura 11: Simulazione PIL del controllo di velocità.

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: Tempi di esecuzione della simulazione PIL del controllo di velocità.

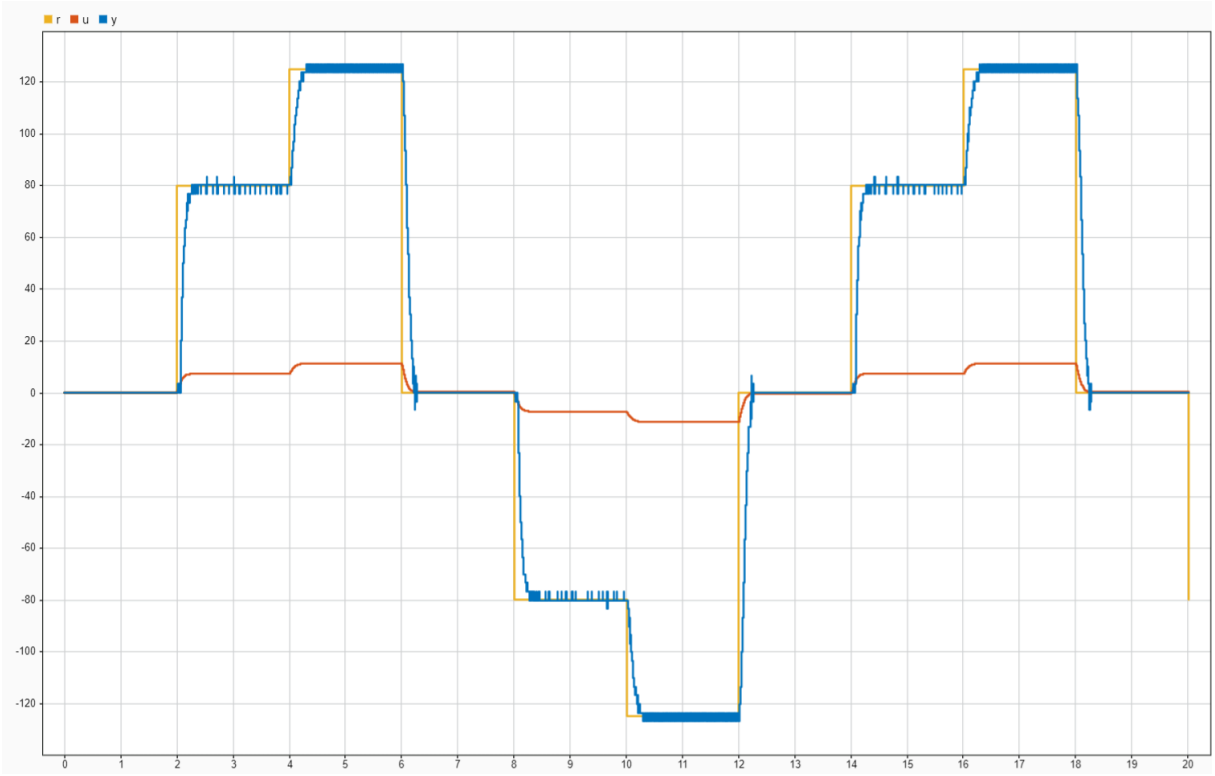

Figura 13: Esecuzione sul motore fisico del controllo di velocità, mediante auto generazione del codice.

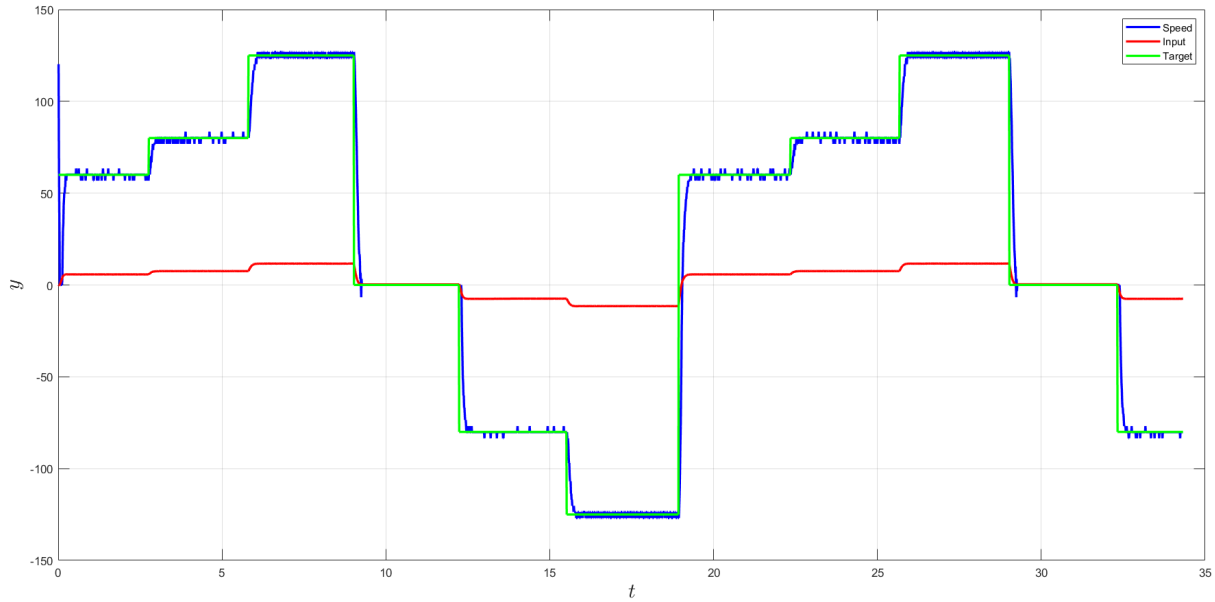

Figura 14: Esecuzione sul motore fisico del controllo di velocità, mediante la tecnica del direct coding.