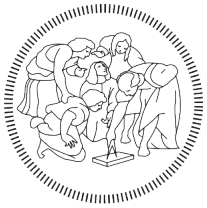


POLITECNICO
MILANO 1863

Ball and beam

Automation and control laboratory

C3R

Team members:

Ribolla Gabriele 10617369

A.Y. 2021/2022

Contents

Chapter 1

DC motor modeling

1.1 Description of the system

A metal ball is placed on a beam, where it can roll with one degree of freedom along the beam itself. One end of the beam is attached to a fixed hinge, while the other one is attached to a lever arm: on the other side the lever arm is linked to the servo gear that is free to rotate.

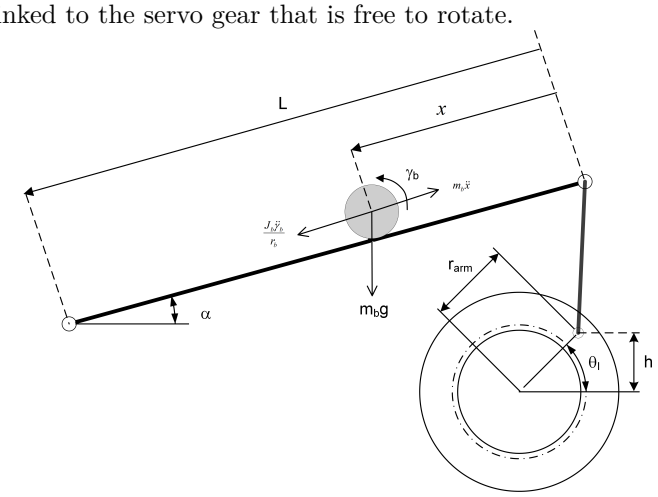


Figure 1.1: Scheme of the physical setup

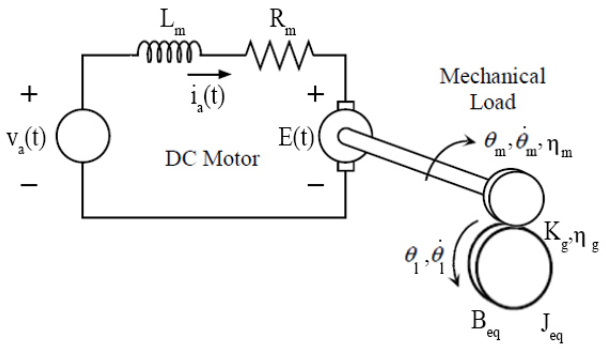


Figure 1.2: Electromechanical scheme

The permanent magnet DC motor has the role of moving the entire system. This motor has its own internal gearbox that drives external gears. An incremental encoder sensor is attached to the biggest gear (the one at which is linked the lever arm) and measures the rotation of that gear (angle θ_l).

1.2 DC motor subsystem parameters

The system parameters and their values were taken from the supplied datasheet.

Symbol	Description	Value
v_a	Motor armature input voltage	
i_a	Motor armature current	
ψ_a	Armature flux	
E	Back-emf	
T_m	Motor torque	
R_a	Motor armature resistance	2.6 Ω
L_a	Motor armature inductance	0.18 mH
k_c	Motor current-torque constant	$7.68x10^{-3}$ Nm/A
k_e	Motor back-emf constant	$7.68x10^{-3}$ V/(rad/s)
K_g	Total gear ratio	70
η_m	Motor efficiency	0.69
η_g	Gearbox efficiency	0.9
J_{eq}	Gear equivalent moment of inertia (without external load)	$2.087x10^{-3}$ kg m^2
B_{eq}	Gear equivalent viscous damping coefficient	0.015 Nm /(rad/s)
v_{a_max}	Maximum input voltage	±6 V
i_{a_max}	Maximum input current	±1 A
θ_{m_max}	Maximum motor speed	±628.3 rad/s

Table 1.1: System parameters value

1.3 DC motor subsystem control variables

The control variable for this subsystem is the input voltage v_a, while the output is the angle of the gear θ_l (which is equal to zero when the lever bar is perpendicular to the ground, as it is drawn on the hardware).
The sensor we can use to feedback our system during the control action is an incremental encoder and, considering as initial condition the position in which the screw of the lever bar is in contact with the angular potentiometer gear, the zero of the encoder is actually at about $\theta_l = -57°$.
To read correctly the data coming from the encoder, we multiply them to $\frac{2\pi}{4096}$ (since it measures 4096 counts per revolution) to obtain a measure in radians and then we add a bias of $-0,994838$ (corresponding to $-57°$ in radians).

1.4 DC motor subsystem equations

In order to describe the electromechanical model of the motor, we use the permanent magnet DC motor equation given by the literature:

$$v_a = R_a i_a + p\psi_a + E \tag{1.1}$$

$$E = k_e \dot{\theta}_m \tag{1.2}$$

$$p\psi_a = \frac{d\psi_a}{di_a}\frac{di_a}{dt} = L_a\frac{di_a}{dt} \tag{1.3}$$

$$T_m = k_c i_a \tag{1.4}$$

We substitute the results of the equations 1.2 and 1.3 in the equation 1.1 and we obtain:

$$v_a = R_a i_a + L_a\frac{di_a}{dt} + k_e\dot{\theta}_m$$

Knowing that $\dot{\theta}_m = -K_g\dot{\theta}_l$, we obtain:

$$v_a = R_a i_a + L_a\frac{di_a}{dt} - k_e K_g\dot{\theta}_l$$

that can be seen, through the Laplace transform, as:

$$v_a = R_a i_a + L_a i_a s - k_e K_g\theta_l s \tag{1.5}$$

Finally from equations 1.4 and 1.5 we get:

$$T_m = k_c\frac{v_a + k_e K_g\theta_l s}{R_a + L_a s} \tag{1.6}$$

If we consider the motor without the system attached (as in a load free test), we have just the external gear box linked to the internal gear box of the motor. From the datasheet we know the equivalent moment of inertia and the equivalent viscous damping coefficient of the external gear box, so we use these data to compute the torque equilibrium at the motor (considering also the motor and the gear efficiency and the gear ratio):

$$J_{eq}\ddot{\theta}_l + B_{eq}\dot{\theta}_l = -\eta_m\eta_g K_g T_m \tag{1.7}$$

Equation 1.7 can be seen, through Laplace transform, as:

$$J_{eq}\theta_l s^2 + B_{eq}\theta_l s = -\eta_m\eta_g K_g T_m \tag{1.8}$$

Now if we merge equation 1.6 and equation 1.8 we obtain:

$$J_{eq}\theta_l s^2 + B_{eq} = -\beta k_c\frac{v_a + k_e K_g\theta_l s}{R_a + L_a s}$$

where $\beta = \eta_m\eta_g K_g$.

We can now compute the transfer function of the entire system:

$$\frac{\theta_l}{v_a} = \frac{-\beta k_c}{s[L_a J_{eq}s^2 + (J_{eq}R_a + B_{eq}L_a)s + B_{eq}R_a + \beta k_c k_e K_g]} \tag{1.9}$$

which has static gain equal to -1.5281 and poles in $s = 0$, $s = -40$, and in $s = -14411$.

If we want to express the transfer function with respect to $\dot{\theta}_l$, instead of θ_l, we can simply remove the integrator present in the previous expression obtaining the following transfer function:

$$\frac{\dot{\theta}_l}{v_a} = \frac{-\beta k_c}{L_a J_{eq}s^2 + (J_{eq}R_a + B_{eq}L_a)s + B_{eq}R_a + \beta k_c k_e K_g} \tag{1.10}$$

This transfer function has the same static gain and the same poles of the previous one, without the pole in $s = 0$.

1.5 Model validation and parameters tuning

To validate our model we performed many load free tests giving to the motor a square wave input voltage with different amplitudes and frequencies. We compared the measured velocity, obtained through the derivative of the position measured by the encoder, and the one estimated by our model, because the position measured by the encoder drifts in time due to the gap present between the teeth of the gears and because of the friction. Through these experiments we noticed that the faster the motor rotates, the bigger was the difference between the estimated speed and the one obtained from the derivative of the encoder.

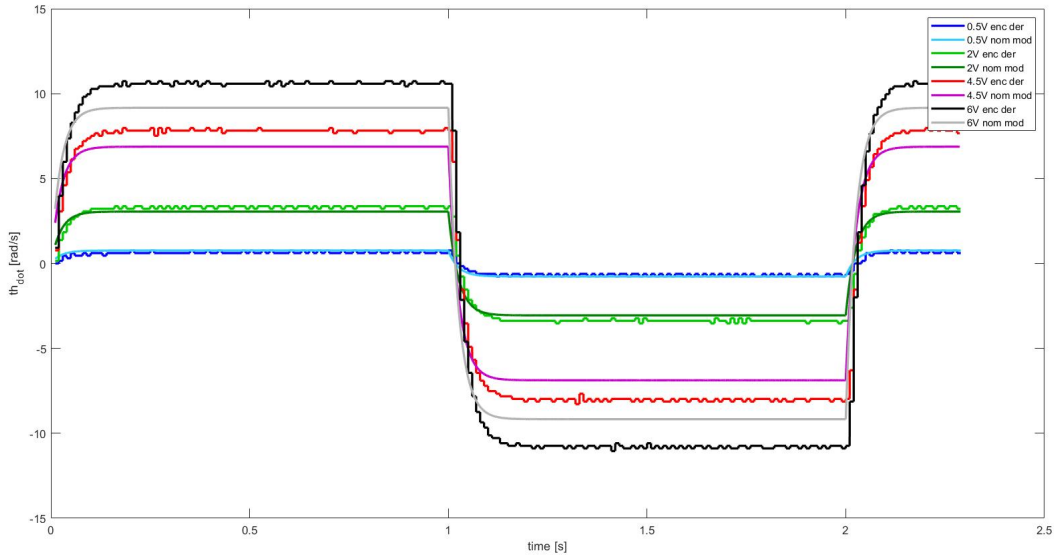


Figure 1.3: Velocities measured and estimated at different voltages (0.5 Hz), ignoring the very first values because the derivative of the encoder is not finite

To reduce the gap between our model speed and the measured one we tried to better estimate the equivalent viscous damping coefficient, assuming that all the other datasheet parameters were correct. We have thus computed the best B_{eq} value for all the voltages (and consequently for all the speeds reached by the motor with that input) and we have obtained this graph, which has been interpolated with a 5^{th} order function.

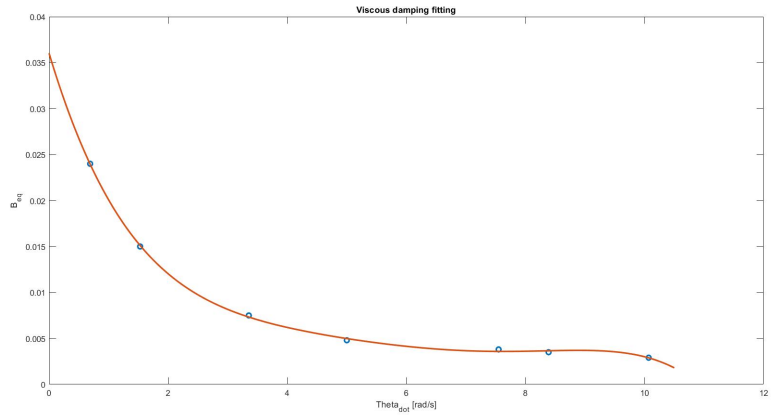


Figure 1.4: Best B_{eq} for different velocities

After this discovery we decided to use an average value for this parameter and we computed the weighted mean value, considering just the value of B_{eq} at speed greater than 2 rad/s in order to avoid taking into account also the static friction. So the new equivalent viscous damping coefficient value is equal to $B_{eq} = 0.004$.
With this new value, the estimated speed is much closer to the one measured.

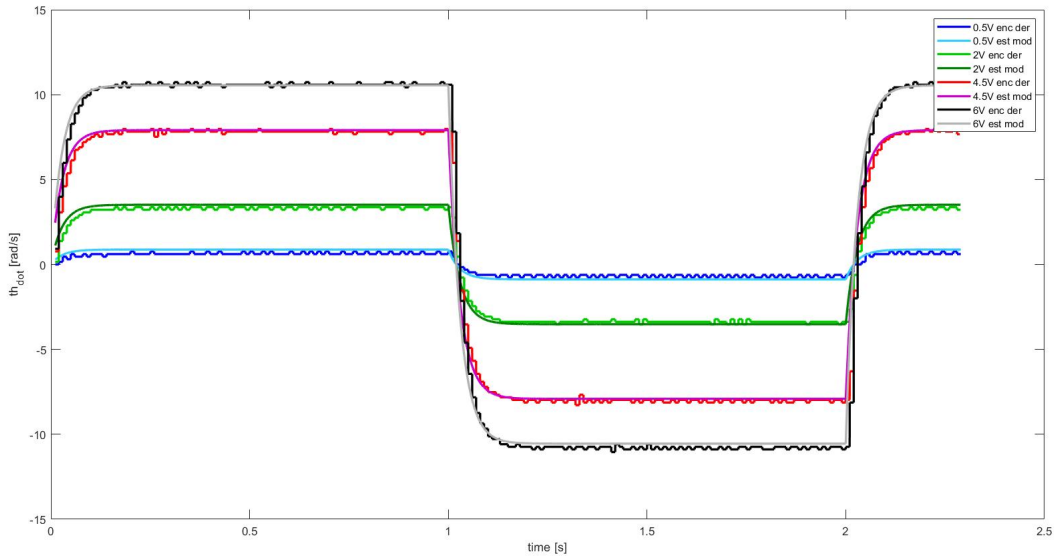


Figure 1.5: Velocities measured and estimated at different voltages (0.5 Hz), ignoring the very first values because the derivative of the encoder is not finite

The model transfer function, considering the new B_{eq} value, now becomes:

$$\frac{\theta_l}{v_a} = \frac{-\beta k_c}{s[L_a J_{eq} s^2 + (J_{eq} R_a + B_{eq\_est} L_a)s + B_{eq\_est} R_a + \beta k_c k_e K_g]} \tag{1.11}$$

with a static gain equal to -1.7582 and poles in $s = 0$, $s = -34.99$, and in $s = -14411$. This represents a minimum phase system, due to the fact that its static gain is bigger than 0, all poles have negative real part and we have no zeroes.

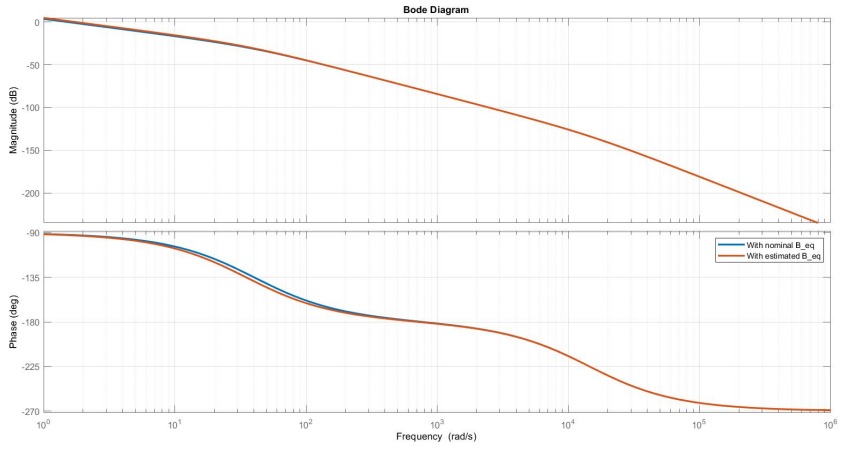


Figure 1.6: Bode plot of the transfer function (from (v_a to θ_l) with B_{eq} and with B_{eq_est}

As we can appreciate from the image above, the two transfer functions are essentially equal to each other.
The fact that we have a pole at very high frequency allows us to use for the motor control an approximated transfer function, considering only its dominant poles ($s = 0$, and $s = -34.99$):

$$\frac{\theta_l}{v_a} = \frac{-\beta k_c}{s(J_{eq} R_a s + B_{eq\_est} R_a + \beta k_c k_e K_g)} \tag{1.12}$$

which has static gain equal to -1.7582 and poles in $s = 0$, and $s = -34.99$.
As seen in the Bode plot below, the behaviour of the system represented by the two transfer functions (approximated and not) is very similar for the frequencies of our interest:

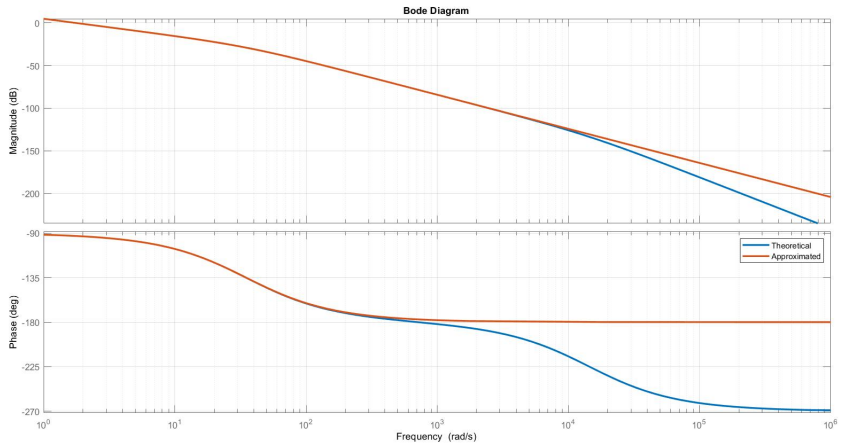


Figure 1.7: Bode plot of the transfer function (from v_a to θ_l) approximated and not

For this reason, it is reasonable to consider the approximated transfer function 1.12 as a good trade-off to develop θ_l position controllers for the DC motor.

1.6 Frequency validation of the model

To have a frequency validation of our model we have also performed one test giving to the system a sinusoidal signal that started from 0.1 Hz and reached 4 Hz in 100 seconds. From the tests with different frequencies we discovered that it is better not to overcome frequency of 10 Hz of the input voltage to avoid stressing the mechanical system and that it is thus better to stay below 5/6 Hz during the longest experiment.
Since the system, after a certain amount of time, drifts because of friction and gaps between the gear teeth, we decided to use the speed acquired data.
The process we followed is the one here reported:

1. We have used the "System Identification Toolbox" provided by Matlab to identify the transfer function of the motor from the data. We asked the software to estimate a transfer function with 2 poles and no zeros, as the one from the voltage to speed $\dot{\theta}_l$ found previously.

2. We have plotted the Bode plot of that transfer function.

The results are shown below, together with the bode plot of the transfer function we have computed before in 1.10, but using B_{eq_est} instead of B_{eq}.

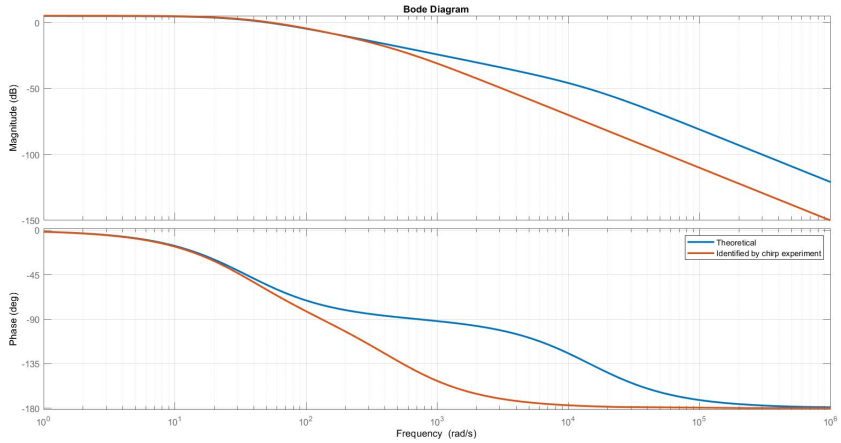


Figure 1.8: Bode plot of the computed transfer function between v_a and $\dot{\theta}_l$ and the one obtained with the chirp experiment

We can notice from the image above that, for low frequencies (as expected, since we developed an experiment that tested the system up to 4 Hz), our transfer function is very similar to the one identified through the experiment. For this reason, we can consider our system validated in the range of frequencies in which we are interested in and we can conclude that the transfer function expressed in 1.12 is a good estimation of the real system, and so we can use it to develop θ_l position controllers for the DC motor.

Obviously during the experiment with the ball and beam system attached to the motor, our model response will differ from the real one, but we will consider the load as a disturbance acting on the system as we will discuss in the following chapter.

Chapter 2

Ball and Beam mechanical system model

2.1 Ball and Beam subsystem parameters

The system parameters and their values were taken from the supplied datasheet.

Symbol	Description	Value
M_{beam}	Mass of ball beam module	0.65 Kg
L_{beam}	Beam length	42.55 cm
L_{lever}	Lever arm length	12.0 cm
r_{arm}	Distance from gear shaft to coupled joint	2.54 cm
	Support arm length	16.0 cm
r_b	Radius of the ball	1.27 cm
m_b	Mass of the ball	0.064 Kg
g	Gravitational acceleration	9.806 $\frac{m}{s^2}$

Table 2.1: System parameters value

Using the data from the table and thanks to literature, the following relations for the inertia moment can be obtained.

$$J_{beam} = \frac{1}{3} M_{beam} L_{beam}^2 \qquad J_{ball} = \frac{2}{5} m_b r_b^2$$

2.2 Ball and Beam subsystem control variables

According to figure 1.1, we consider:

- α: inclination of the beam with respect to the horizontal;
- $\mathbf{x}$: ball displacement with respect to the moving end of the beam.

In the real system there are two sensors that are able to provide us two different measurements: a potentiometer for the position of the ball $\mathbf{x}$ and an encoder for the angular position θ_l. These two sensors allow to control the system by acting on x and θ_l: for this reason the relationship between α and θ_l is needed. Considering figure 2.1, the kinematic closure can be computed.

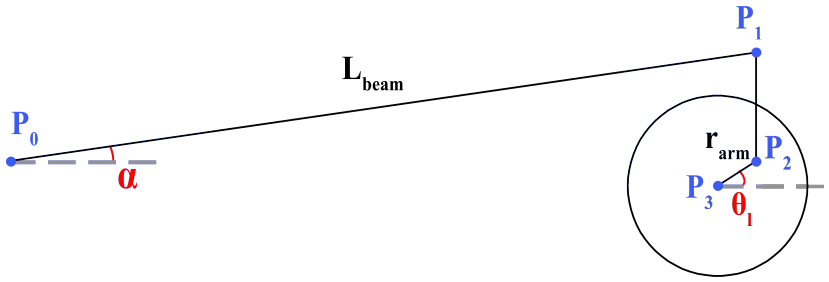


Figure 2.1: Schematics of the kinematic closure

Considering $P_0 = (0;0)$ as represented above, the relationship between P_0 and P_1 can be found from the previous kinematic closure and can be written as follows:

$$\begin{bmatrix} P_{1x} \\ P_{1y} \end{bmatrix} = \begin{bmatrix} L_{beam}\cos\alpha \\ L_{beam}\sin\alpha \end{bmatrix}$$

For θ_l values in the range of interest ($\pm 45°$), the angle between the lever arm and the horizontal can be considered constant and equal to 90°. For this reason, the following relation can be stated about the position of point P_1 and point P_2.

$$P_{1x} = P_{2x} \tag{2.1}$$

$$P_{1y} - P_{2y} = L_{lever}\sin(90°) \tag{2.2}$$

The relation between P_2 and P_3 obtained from the kinematic closure is:

$$\begin{bmatrix} P_{3x} \\ P_{3y} \end{bmatrix} = \begin{bmatrix} P_{2x} - r_{arm}\cos\theta_l \\ P_{2y} - r_{arm}\sin\theta_l \end{bmatrix}$$

Replacing 2.1 and 2.2 in the previous relations, it is found that:

$$\begin{bmatrix} P_{3x} \\ P_{3y} \end{bmatrix} = \begin{bmatrix} L_{beam}\cos\alpha - r_{arm}\cos\theta_l \\ L_{beam}\sin\alpha - L_{lever} - r_{arm}\sin\theta_l \end{bmatrix}$$

Evaluating the last expressions in a well-known point, such as α=0 and θ_l=0, it is obtained that:

$$\begin{bmatrix} P_{3x} \\ P_{3y} \end{bmatrix} = \begin{bmatrix} L_{beam} - r_{arm} \\ -L_{lever} \end{bmatrix}$$

Focusing on the second term:

$$-L_{lever} = L_{beam}\sin\alpha - L_{lever} - r_{arm}\sin\theta_l$$

$$L_{beam}\sin\alpha = r_{arm}\sin\theta_l$$

$$\sin\alpha = \frac{r_{arm}}{L_{beam}}\sin\theta_l \tag{2.3}$$

In equation 2.3 it is reported the useful relationship between the servo angle θ_l and the beam angle with respect to the horizontal α.

2.3 Ball and beam subsystem equations

In order to describe the mechanical model of the system, we assume to have pure rolling motion without slipping and we choose to compute the Lagrangian function of motion that characterizes the dynamical behaviour of the system.

$$L = T - U \tag{2.4}$$

The kinetic energy expression of the ball T_{ball} has three motion components: translation along the beam, rolling motion and rotation with respect to the fixed hinge of the beam. The kinetic energy expression of the beam T_{beam} is affected only by the rotation around the fixed hinge. Concerning potential energy components U_{ball} and U_{beam}, we consider the height as the distance with respect to the horizontal reference passing through the hinge. In this case each component has been studied separately and then everything was put together as in 2.4.

$$T_{ball} = \frac{1}{2} \cdot m_{ball} \cdot \dot{x}^2 + \frac{1}{2} \cdot J_{ball} \cdot \dot{\gamma_b}^2 + \frac{1}{2} \cdot m_{ball} \cdot (L_{beam} - x)^2 \cdot \dot{\alpha}^2$$

$$T_{beam} = \frac{1}{2} \cdot J_{beam} \cdot \dot{\alpha}^2$$

$$U_{ball} = m_{ball} \cdot g \cdot \sin\alpha \cdot (L_{beam} - x)$$

$$U_{beam} = m_{beam} \cdot g \cdot \sin\alpha \cdot \frac{L_{beam}}{2}$$

Starting from 2.4, we can compute:

$$\frac{d}{dt}(\frac{\partial L}{\partial \dot{q}}) - \frac{\partial L}{\partial q} = Q \tag{2.5}$$

Considering:

$$q = \begin{bmatrix} x \\ \alpha \end{bmatrix} \qquad \dot{q} = \begin{bmatrix} \dot{x} \\ \dot{\alpha} \end{bmatrix} \qquad Q = \begin{bmatrix} 0 \\ \tau \end{bmatrix}$$

Focusing on the first component (the ball displacement x), the first equation is obtained.

$$\ddot{x}\left(m_{ball} + \frac{J_{ball}}{r_{ball}^2}\right) + m_{ball}(L_{beam} - x)\dot{\alpha}^2 - m_{ball} g \sin\alpha = 0 \tag{2.6}$$

Focusing on the second component (the beam angle α), the second equation is obtained.

$$\left[m_{ball}(L_{beam} - x)^2 + J_{beam}\right]\ddot{\alpha} - 2m_{ball}(L_{beam} - x)\dot{x}\dot{\alpha} + \left[m_{ball}(L_{beam} - x) + m_{beam}\frac{L_{beam}}{2}\right] g\cos\alpha = \tau \tag{2.7}$$

2.4 Non-linear model

Considering the variables involved in the previous equation, it can be seen that in 2.7 the applied torque τ is an exogenous input of the system, but with the available setup it cannot be measured since there is not a dedicated sensor. Because of the fact that the goal is to control the ball position and to perform trajectory tracking, we decided to consider the torque in equation 2.7 as a static component that affects the DC motor output torque. We consider the mass of the beam as "concentrated" in its centre of gravity and, according to figure 1.1, we obtain:

$$\tau = m_{beam} g \frac{L_{beam}}{2} \sin(\frac{\pi}{2} - \alpha) + m_{ball} g (L_{beam} - x) \sin(\frac{\pi}{2} - \alpha) + m_{lever} g L_{beam} \sin(\frac{\pi}{2} - \alpha)$$

$$\tau = (\frac{m_{beam}}{2} L_{beam} + m_{lever} L_{beam} + m_{ball}(L_{beam} - x)) g \cos\alpha \tag{2.8}$$

This torque is transmitted to the gear, so we consider the power transmission:

$$\tau\dot{\alpha} = \tau_{\theta_l}\dot{\theta}_l \tag{2.9}$$

Knowing the relation 2.3, we can derive it obtaining:

$$\cos(\alpha)\dot{\alpha} = \frac{r_{arm}}{L_{beam}} \cos(\theta_l)\dot{\theta}_l$$

$$\dot{\alpha} = \frac{r_{arm}}{L_{beam}} \frac{\cos\theta_l}{\cos\alpha}\dot{\theta}_l \tag{2.10}$$

Substituting 2.8 and 2.10 in 2.9, we obtain:

$$\tau_{\theta_l} = \left(\left(\frac{m_{beam}}{2} + m_{lever}\right) L_{beam} + m_{ball}\left(L_{beam} - x\right)\right) \frac{r_{arm}}{L_{beam}} g \cos\theta_l \tag{2.11}$$

Due to the fact that the provided datasheet does not contain the values of m_{beam} and m_{lever}, we tried to estimate directly the value of the static torque that DC motor needs to overcome to start moving, performing an experiment.

- We removed the ball and we attached the beam to the DC motor, using a support that kept $\theta_l = 0$.
- We provided to the DC motor a ramp voltage input that started from 0V, in order to detect at which value the DC motor started to move.

With this experiment and knowing the transfer function of DC Motor, we were able to find the static torque needed for the system to start moving. We saw that the beam starts to move when input voltage was near 0.47V, which corresponds to a torque of about 0.06 Nm. Having estimated this torque, we were able to obtain a model of our system which is closer to reality even in load conditions: in later simulations, the difference between the estimated angle θ_l and the one of the real system was less than 1°. Thanks to this, we were able to consider the inertia of the beam as a disturbance, at least at the equilibrium point. From this point on we focused on 2.6 to obtain a model of the system. Substituting the expression of J_{ball} in 2.6, we obtained:

$$(m_{ball} + \frac{2}{5} m_{ball})\ddot{x} + m_{ball}(L_{beam} - x)\dot{\alpha}^2 - g m_{ball} \sin\alpha = 0$$

from which follows:

$$\ddot{x} = \frac{5}{7} g \sin\alpha - \frac{5}{7}(L_{beam} - x)\dot{\alpha}^2 \tag{2.12}$$

Considering $x_1 = x$ and $x_2 = \dot{x}$ as the states and taking into account the previous relation, we are able to find the state-space representation of the non-linear model:

$$\dot{x_1} = x_2 = f_1$$
$$\dot{x_2} = \frac{5}{7} g \sin\alpha - \frac{5}{7}(L_{beam} - x_1)\dot{\alpha}^2 = f_2$$

2.5 Linearization around equilibrium

In order to develop controllers for our model, we need to have a state-space representation of the linearized model: we choose to linearize the previously found non-linear model around an equilibrium point reported below, where x_0 represents a generic position of the ball along the beam:

$$\bar{\alpha} = 0 \qquad \dot{\bar{\alpha}} = 0 \qquad \bar{x} = x_0 \qquad \dot{\bar{x}} = 0$$

$$\delta x = \begin{bmatrix} \delta x_1 \\ \delta x_2 \end{bmatrix} = \begin{bmatrix} x_1 - \bar{x_1} \\ x_2 - \bar{x_2} \end{bmatrix} \qquad \delta \dot{x} = \begin{bmatrix} \delta \dot{x_1} \\ \delta \dot{x_2} \end{bmatrix} = \begin{bmatrix} \dot{x_1} - \dot{\bar{x}_1} \\ \dot{x_2} - \dot{\bar{x}_2} \end{bmatrix}$$

$$\delta\alpha = \alpha - \bar{\alpha}$$

Computing the partial derivatives that compose the matrixes A_{lin}, B_{lin} and C_{lin} ($D_{lin} = 0$, so it is a strictly proper system), we obtain:

$$\left.\frac{\partial f_1}{\partial x_1}\right|_{eq.} = 0 \qquad \left.\frac{\partial f_1}{\partial x_2}\right|_{eq.} = 1 \qquad \left.\frac{\partial f_1}{\partial \alpha}\right|_{eq.} = 0$$

$$\left.\frac{\partial f_2}{\partial x_1}\right|_{eq.} = 0 \qquad \left.\frac{\partial f_2}{\partial x_2}\right|_{eq.} = 0 \qquad \left.\frac{\partial f_2}{\partial \alpha}\right|_{eq.} = \frac{5}{7}g$$

Rearranging all the computed terms in a matrix representation:

$$\begin{bmatrix} \delta\dot{x_1} \\ \delta\dot{x_2} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} \delta x_1 \\ \delta x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{5}{7}g \end{bmatrix} \delta\alpha \tag{2.13}$$

$$\delta y = \begin{bmatrix} 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} \delta x_1 \\ \delta x_2 \end{bmatrix} \tag{2.14}$$

This state-space representation describes the model that is controlled by the input variable α that we cannot directly measure. Due to the fact that in linearization we consider small movements around the equilibrium, the angle α and θ_l are very small and thanks to this we can approximate $\sin\alpha \approx \alpha$ and $\sin\theta_l \approx \theta_l$. For this reason, we can re-write the expression 2.3 as follows:

$$\delta\alpha = \frac{r_{arm}}{L_{beam}}\delta\theta_l \tag{2.15}$$

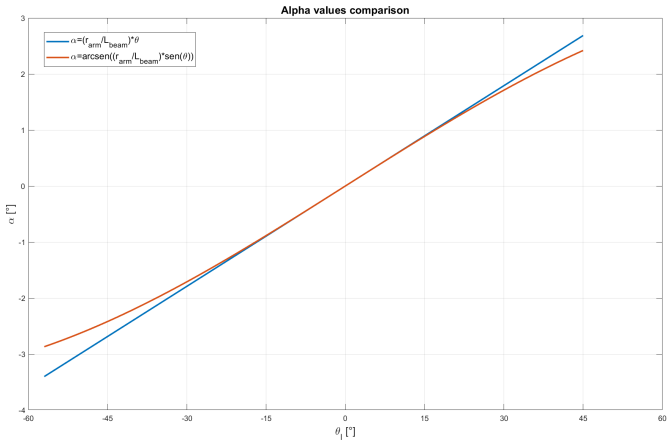


Figure 2.2: Differences between the relations 2.3 and 2.15

Computing the differences between the relations 2.3 and 2.15 in the range of interest, we obtained a difference of 0.27° when $\alpha = \pm 45°$, for which corresponds a height difference of 0.19mm. This small difference let us accept this tradeoff between the more complex model of the system and a simpler representation: for this reason we consider these two new values for θ_l and $\dot{\theta}_l$ at the equilibrium:

$$\bar{\theta}_l = 0 \qquad \dot{\bar{\theta}}_l = 0 \qquad \delta\theta_l = \theta_l - \bar{\theta}_l$$

Replacing the last expressions in 2.13, we obtain this final state-space representation:

$$\begin{bmatrix} \delta\dot{x_1} \\ \delta\dot{x_2} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} \delta x_1 \\ \delta x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{5}{7}\frac{r_{arm}}{L_{beam}}g \end{bmatrix} \delta\theta_l \tag{2.16}$$

The state-space representation presented above can be expressed with the matrices A, B, C, and D (with $C = \begin{bmatrix} 1 & 0 \end{bmatrix}$ and $D = 0$). If we check the reachability for the pair (A,B) and the observability for (A,C) we discover that the ball and beam subsystem is reachable and observable.

2.6 Transfer function

We consider the second equation of the linearized state-space representation 2.16.

$$\delta x_2 = \frac{5}{7}\frac{r_{arm}}{L_{beam}}g\delta\theta_l$$

Applying the Laplace transform, we obtain the system transfer function from input θ_l to output x_1.

$$G(s) = \frac{X_1(s)}{\Theta(s)} = \frac{5}{7}\frac{r_{arm}}{L_{beam}}g\frac{1}{s^2} \approx \frac{0.4181}{s^2} \tag{2.17}$$

What is obtained is the open-loop transfer function of the system: as shown in the Bode plots below, it corresponds to a double integrator, having a slope of -40dB/decade in the magnitude plot and -180° in the phase plot.

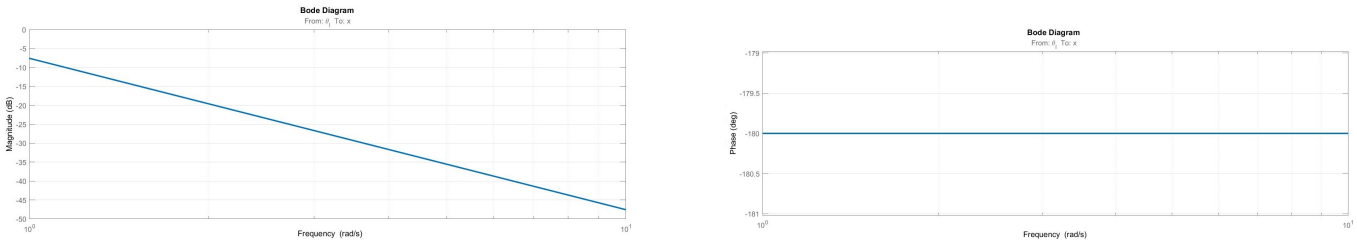


Figure 2.3: Bode plot of 2.17 transfer function

2.7 Open-loop response and model validation

Considering the linearized-system open-loop transfer function 2.17, its response to an unitary step has been used in order to describe the characteristics. It resulted in a typical response of an unstable system, due to the couple of poles in $s = 0$: it meant that, when designing the ball position controller, system stabilization had to be taken into account. Accordingly to datasheet, the potentiometer provides a measure that varies between $\pm 4.5V$, with 0V placed in the middle of the beam. For this reason, we include the following relation in simulink blocks, to set the reference of the real apparatus according to our reference previously defined:

$$S_{ensitivity} = \frac{L_{beam} - 2r_{ball}}{\Delta V} \approx \frac{0.4m}{9V} \approx 4.5\frac{mm}{V}$$

$$Gain_{Simulink} = -\frac{\frac{L_{beam}}{2} - r_{ball}}{S_{ensitivity}} \qquad Bias_{Simulink} = \frac{L_{beam}}{2} - r_{ball}$$

The open-loop response of the linearized model has been compared with the one given by the real system and the non-linearized one, implemented directly in Simulink using the equation 2.12 and the relation 2.3, in order to show the differences and to be able to validate our model. To compare the model responses to the real behavior, a step of $\frac{\pi}{4}$ after 5 seconds was used as input. In the figure below there are the responses of the non-linear and linearized system, together with the recorded data that came from the potentiometer of the real apparatus. It is clear that there is a delay of $0.5s$ in the response of the real system: this is probably caused by system dynamics which were neglected, such as friction between the ball and the beam, that is necessary for the assumption of pure rolling motion, that allows us to reach a good tradeoff between complexity and correct representation of the system. This is proven given an input of θ_l equal to a couple degrees: the ball in the linear and non-linear model starts to move, instead in the real system it stands still.

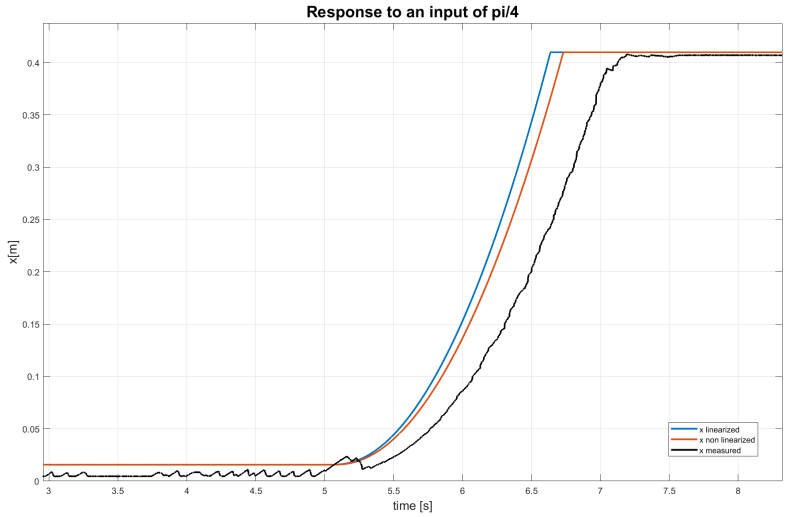
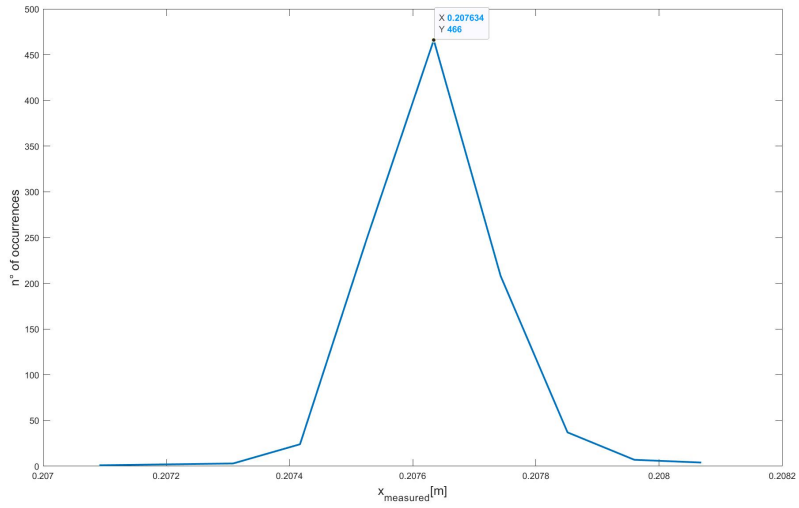


Figure 2.4: Open-loop responses comparison

Instead, the slope of the curve is pretty much the same for all the three curves represented in the image. The non linear model and the linearized one have a very little difference which increases when the angle θ_l increases as we have shown before. However, considering that the angle θ_l has a limited range and x is physically limited by the beam, we can conclude that we can rely on our linearized model for the implementation of ball position controllers, and it is reasonable to expect very close performances once we are in closed-loop.

Thanks to this experiment, we can draw other important conclusions. First, we know that the time the ball takes to travel the entire beam is ≈ 2 seconds, and this information is useful to set a target control settling time for controllers design phase. Secondly, we are able to understand that the potentiometer provides noisy data near the beginning and the end of the beam (accordingly to datasheet): for this reason we need to use only the central part of the beam as a reference during the controllers development. Due to the fact that datasheet does not provide information about the variance of the potentiometer measurements, we have tried to estimate it measuring the position of the ball in a point in the middle of the beam. Considering $1s$ of samples ($T_{sampling} = 0.002s$) we count the occurrences of each measurement. The shape of the obtained distribution can be reasonably approximated as a Gaussian with a variance equal to $\sigma^2 = 1 \cdot 10^{-8}$, and this is useful for reproducing it in simulated environment and for the observers design.



Chapter 3

DC motor control

3.1 DC Motor transfer function description

The control strategies chosen for the DC motor are based on the open-loop approximated transfer function obtained in chapter 1, that has been for simplicity changed of sign:

$$G_{speed}(s) \approx \frac{61.52}{s + 34.99} \qquad G_{pos}(s) \approx \frac{61.52}{s(s + 34.99)}$$

Thanks to this change of sign the motor controllers discussed in this chapter will have positive static gain. In order to compensate this change of sign, in the lab sessions, also the control voltage passed to the motor has been changed of sign (without it, a positive voltage provides a negative value for the speed).
Thanks to the position open-loop transfer function we can compute the closed-loop one:

$$F_{DC}(s) = \frac{G_{pos}(s)}{1 + G_{pos}(s)} = \frac{61.52}{s^2 + 34.99s + 61.52} \tag{3.1}$$

The resulting closed-loop poles are, respectively, in $s = -1.8567$ and $s = -33.133$, which are both stable poles, and accordingly with the following Bode plot, we have a frequency band included in $[0, \omega_c = 1.76\ \frac{rad}{s}]$.

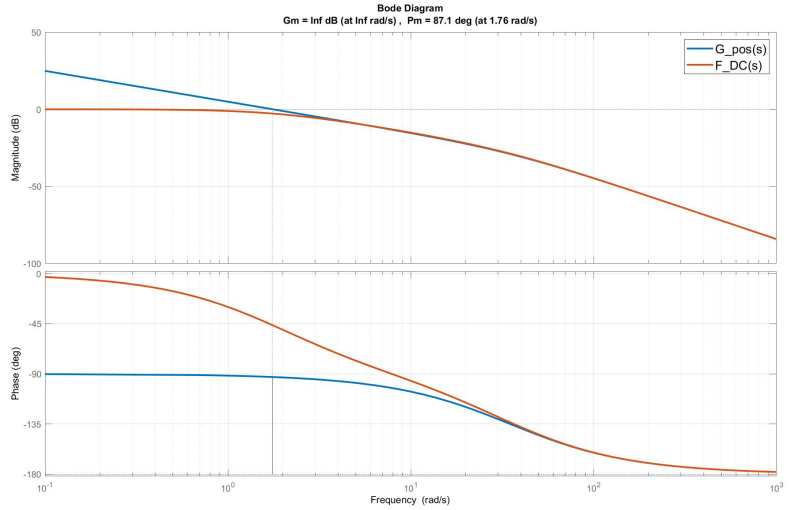


Figure 3.1: G_{pos} and F_{DC} Bode diagrams

Since the closed loop transfer function is a second order one, it can be re-written as:

$$F(s) = \frac{\omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2}$$

By comparing this last expression with 3.1, we obtain that $2\xi\omega_n = 34.99$ and $\omega_n^2 = 61.52$: from these values it can be seen that $\xi = 2.23 > 1$, so we can write 3.1 as:

$$F(s) = \frac{p_1 p_2}{(s + p_1)(s + p_2)}$$

where $p_1 = -1.8567$ and $p_2 = -33.133$. Since the two poles have a different magnitude ($p_1 << p_2$), we can compute the settling time of this transfer function as the one of a 1^{st} order system with the dominant pole in p_1. The obtained settling time of F_{DC} is then:

$$T_s = 5\tau_1 = 2.69s.$$

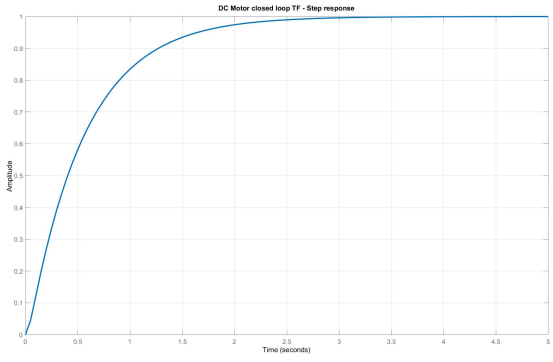


Figure 3.2: Closed loop unitary step response of G_{pos}

We tried several strategies to develop frequency-based controllers for θ (from now on for simplicity we will indicate "θ_l" with "θ") position: initially we start from literature and self-given specifications testing them in simulations, and then we test them on the real system both without and with the ball and beam load, evaluating performances and also taking into account voltage usage.

3.2 Proportional control

3.2.1 Controller development

The first kind of controller we have designed is the proportional one: it is one of the simplest controller and it is a feasible one for this system. The open-loop transfer function considered to design the proportional controller was G_{pos}. According to literature, however, to have a closed-loop system with a controller which guarantees static precision, we should design it with the following structure:

$$R_1(s) = \frac{\mu_R}{s^{g_r}} \tag{3.2}$$

When studying the static precision of a controller of this kind applied to our system, we have seen that, even with $g_r = 0$, static precision is guaranteed: in fact, we have $g_L = g_{G_{pos}} + g_r = 1$, which guarantees that $e_\infty = 0$. This also allows us to avoid using an integrator which will result in a too little phase margin by adding a pole in $s = 0$, in addition to the one already present in the open-loop transfer function in which we rely on in order to obtain a null steady state error. So, we have chosen to set $g_r = 0$ since precision is guaranteed and phase start from $-90°$. We are thus free to choose a proportional gain:

$$R_p = \mu_R \tag{3.3}$$

$$L(s) = R_p(s) \cdot G_{pos}(s)$$

The closed-loop transfer function is a second order transfer function and can be computed as:

$$F(s) = \frac{L(s)}{1 + L(s)} = \frac{61.52\mu_R}{s^2 + 34.99s + 61.52\mu_R} \tag{3.4}$$

In order to assign a value to the proportional gain μ_R we have set a design requirement on overshoot: we have decided to have a maximum overshoot of 2° when the system receives as reference a step of 45°, which is rounded to a 5% percentage overshoot. This corresponds to a damping value of:

$$\xi = \sqrt{\frac{\ln^2(\frac{\%overshoot}{100})}{\pi^2 + \ln^2(\frac{\%overshoot}{100})}} \approx 0.69$$

Thus, by comparing the denominator of this transfer function to the one of a generic second order transfer function, we have computed the value of the natural frequency and then of the proportional gain:

$$s^2 + 2\omega_n\xi s + \omega_n^2 = s^2 + 34.99s + 61.52\mu_R$$

$$\omega_n = \frac{34.99}{2\xi} \approx 25.355\frac{rad}{s} \qquad \omega_n^2 = 61.52\mu_R$$

$$\mu_R = 10.448 \approx 10.5 \qquad R_p = \mu_R = 10.5 \tag{3.5}$$

We can now implement the proportional controller as a gain that multiplies G_{pos}, computing the open-loop transfer function:

$$L(s) = \frac{646}{s(s + 34.99)} \tag{3.6}$$

Accordingly to our computations, we have obtained for this transfer function a phase margin of $\varphi_m = 64.5° < 75°$ (5% overshoot), a crossover frequency of $\omega_c = 16.7\frac{rad}{s}$ and a settling time of $T_s \approx 0.3s$. The closed-loop transfer function 3.4, when plugging in the gain value, results in:

$$F(s) = \frac{646}{s^2 + 34.99s + 646}$$

This gives us two complex conjugate poles in $s = -19.995 \pm j15.69$. $F(s)$ has a unitary gain and guarantees correct tracking of the reference signal for the frequencies included in [0, ω_c] and the attenuation of noises with frequencies higher than the crossover one.

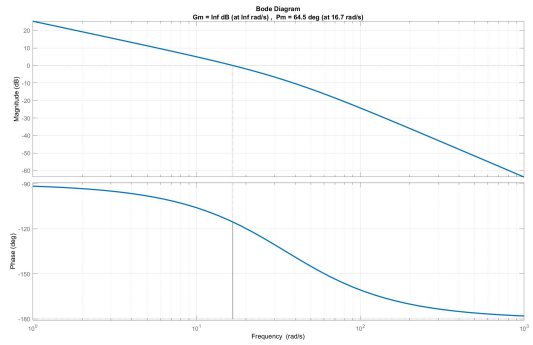


Figure 3.3: Bode plot of L(s)

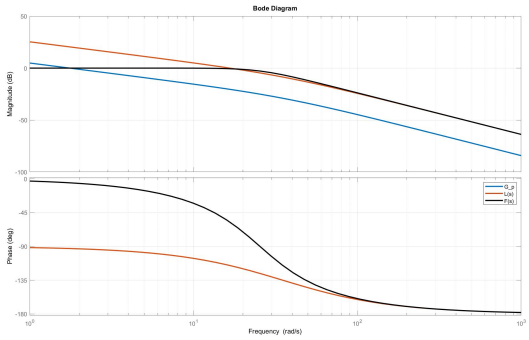


Figure 3.4: Frequency analysis of proportional controller

3.2.2 Controller implementation

Once we test this controller on the real system, we immediately notice that its behaviour is very similar to the simulated one: there is the specified overshoot and the response is near $0.3s$ as previously computed. The voltage usage does not present high frequency spikes. We validate the proportional controller giving as input a square reference for the angle that varies between $\pm\frac{\pi}{4}$, with an increasing frequency from 0.1 to 4Hz in 100s. As expected, it follows the reference in a good way until it reaches the crossover frequency. When the step duration is less than the rise time, it cannot reach the reference signal and for this reason the output starts to be attenuated.

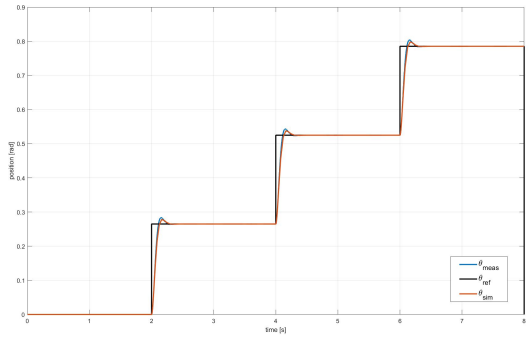


Figure 3.5: incremental $\frac{\pi}{12}$ steps responses

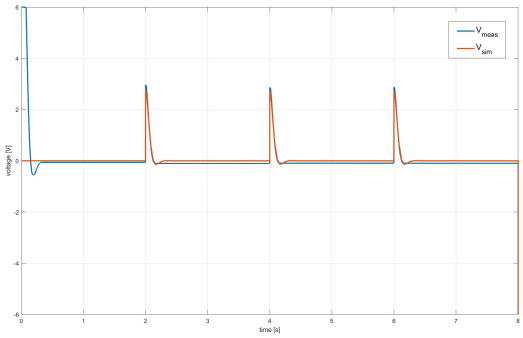


Figure 3.6: Voltage usage for the responses of previous fig.

When we attach the load to the motor, we can notice the presence of a steady state error proportional to the reference input amplitude. We try to increase and decrease several times the proportional gain, changing also the overshoot specifications, but this kind of steady state error still remains. This is probably caused by the pole in $s = 0$ of the system, which in reality does not act as the pole in the same location of a PI controller: behaviour in which instead we rely on. This is also due to the fact that the proportional gain was suited for the transfer function 3.6 and when there was a disturbance acting on the system (the ball and beam load), this controller was not enough robust. As previously discussed, we discard the possibility to insert an integral action to remove the steady state error because it affects the φ_m decreasing it in a critical way. For all these reasons, we can conclude that this controller is not the best choice we can take in order to control DC motor, due to its inaccuracy in the reference tracking once the load is attached.

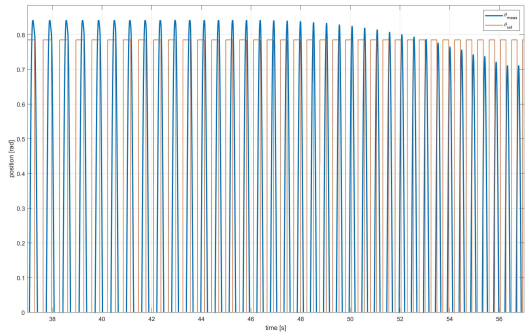


Figure 3.7: Detail of the frequency response of the proportional controller

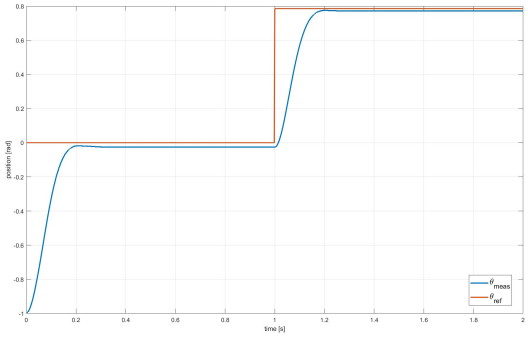


Figure 3.8: Example of the behaviour of the controller once the load is attached (data taken from another experiment)

3.3 Cascade control

Cascade control can be useful to exploit frequency decoupling: inner loop would act on motor speed, while the outer one would control position. In our case, since we have transfer function G_{speed}, we simply put an integrator after it to obtain the position transfer function: this approach also allows to have a simpler control problem due to the fact that from outer loop we can assume $F_{speed} = 1$ because it is studied for a frequency band bigger than the outer one. We also expected to have better performances than a simpler proportional control, without the presence of steady state error once the load is attached. The controller has been designed according to the following architecture:

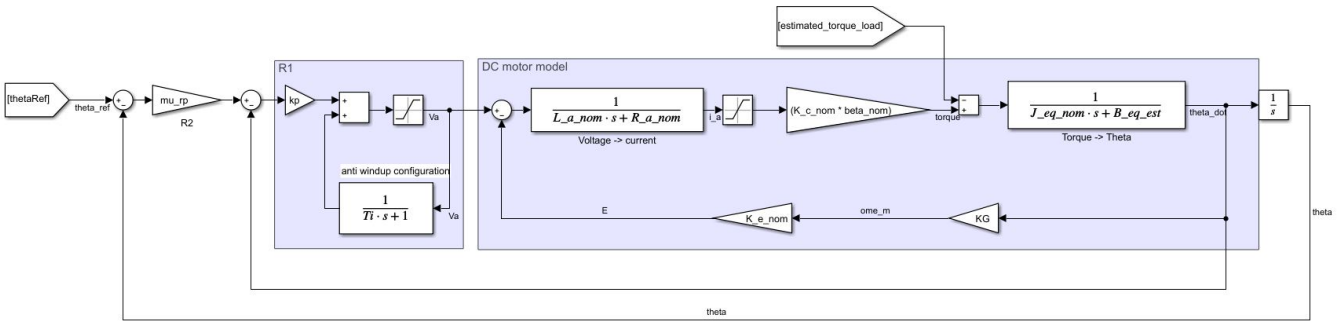


Figure 3.9: Block scheme of the cascade controller

3.3.1 Position controller P

In our case $R_1(s)$ will control speed, while $R_2(s)$ will control position. Starting from $R_2(s)$ in the outer loop, if we suppose to have an inner loop which is at least a decade faster than the outer one, we can consider F_{speed} closed-loop transfer function to be unitary if seen from the outside, and so we can design a controller for:

$$L_{pos}(s) = \frac{R_{pos}}{s} F_{speed} = \frac{R_{pos}}{s} \tag{3.7}$$

$$F_{pos}(s) = \frac{R_{pos}}{s + R_{pos}} \tag{3.8}$$

For the analogue motivations discussed before for proportional controller, we can assume $R_{pos} = \mu_R$ (so without adding any integral action) to obtain a transfer function with a shape similar to an integrator $\frac{\mu_R}{s}$. By doing so, the resulting closed-loop transfer function is a simple first order system as it can be easily shown in 3.8. In order to create a controller that can be comparable with the previous controller, we set the same specifications:

$$T_s = 0.3s \qquad \varphi_m \geq 75°$$

The second specification, in previous controller could not be reached due to ξ damping coefficient and the structure of the transfer function: in this controller, instead, since we have a first-order system, this specification is intrinsically reached thanks to the integral action ($\varphi_m = 90°$) and we are granted to have no overshoot nor oscillations. For what concerns the first specification:

$$\tau_p = \frac{T_s}{5} = 0.06 \qquad \omega_{cp} = \frac{1}{\tau_p} \approx 17 = \mu_r$$

So, the resulting proportional controller is:

$$R_2(s) = R_{pos} = 17 \tag{3.9}$$

and the resulting open-loop and closed-loop transfer function are:

$$L_{pos}(s) = \frac{17}{s} \tag{3.10}$$

$$F_{pos}(s) = \frac{17}{s + 17} \tag{3.11}$$

From the $L_{pos}(s)$ we obtain a phase margin of $\varphi_m = 90°$, a crossover frequency $\omega_{cp} = 17$ and a settling time of $T_s \approx 0.3s$, matching perfectly the requirements. 3.11 has a unitary gain and guarantees the correct tracking of the reference signal for the frequencies included in $[0, \omega_{cp}]$, and the attenuation of noises with frequencies higher than the crossover one.

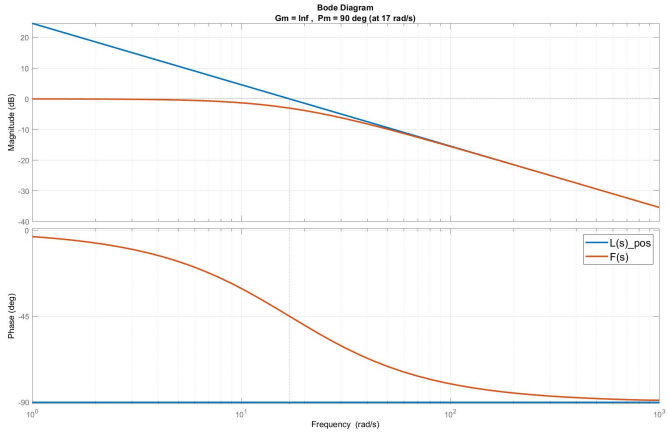


Figure 3.10: Frequency analysis of outer loop cascade controller

3.3.2 Speed controller PI

Now, considering the inner loop, we need to design $R_{speed}(s)$ based on G_{speed} transfer function. To match the constraints which simplified a lot the design of $R_{pos}(s)$, we set for the inner loop 10 times faster than the outer loop:

$$\omega_{cs} = 10 \cdot \omega_{cp} = 170 \frac{rad}{s}$$

A simple way to further improve the behavior of the system is to obtain a transfer function with a shape similar to an integrator, using the controller to do a pole-zero cancellation to obtain an open-loop transfer function like:

$$L_{speed}(s) = \frac{\mu_L}{s}$$

This, as previously discussed, will help to reduce steady state error and gives us a phase margin of $\varphi_m = 90°$. To obtain it we need to choose $R_{speed}(s)$ as follows:

$$R_{speed}(s) = (s + 34.99)$$

From literature, it is known that a controller with this shape cannot be realizable, but decoupling the study of the controller with a static and dynamic project, we were able to create a feasible controller.

$$R_1(s) = \frac{\mu_r}{s^{g_r}} \qquad R_2(s) = \frac{\prod_i (1 + sT_i)}{\prod_k (1 + s\tau_k)} \qquad R_{speed} = R_1 * R_2$$

To obtain this regulator we have designed separately its static and dynamic projects. For the static project we do not assign immediately a value to gain μ_r and we have chosen $g_r = 1$, in order to obtain a null steady state error for every gain value. The dynamic part has been chosen as $(s + 34.99)$ to perform a zero-pole cancellation, as discussed before. These choices lead to a feasible controller and the following open-loop transfer function, after the cancellations:

$$L_{speed}(s) = G_{speed} \cdot R_{speed} = \frac{61.52\mu_r}{s}$$

From the design specifications we know that we need to reach $\omega_{cs} = 170 \frac{rad}{s}$: by imposing it we get $\mu_r \approx 2.7629$. We can rewrite the controller $R_1(s)$ as:

$$R_1(s) = \frac{2.7629}{s} \cdot 34.99 \left(\frac{s}{34.99} + 1 \right) \tag{3.12}$$

In Matlab we implement this as a PI controller, since this structure allows to do so. Comparing equation 3.12 with the structure of a generic PI controller with an anti-windup configuration:

$$R_{PI}(s) = k_p + \frac{k_i}{s} = \frac{k_i(1 + sT_i)}{s} \qquad T_i = \frac{k_p}{k_i} \tag{3.13}$$

After the computations, we get $k_i = 96.6739$ and $k_p = 2.7629$. Plugging these values in 3.13 we can compute the inner open loop and closed-loop transfer functions using G_{speed}:

$$L_{speed}(s) = \frac{170}{s} \tag{3.14}$$

$$F_{speed}(s) = \frac{170}{s + 170} \tag{3.15}$$

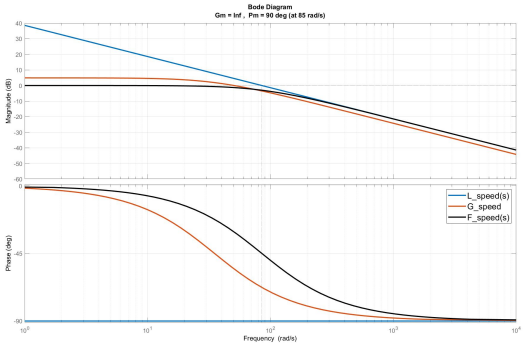


Figure 3.11: Frequency analysis of inner loop cascade controller

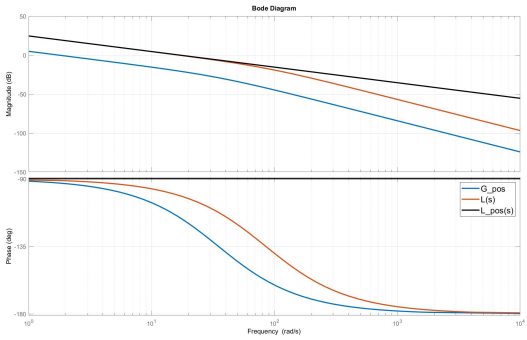


Figure 3.12: Bode plot of cascade controller, comparing $L(s)$ 3.16 and $L_{pos}(s)$ 3.10

Now, recalling the position controller 3.9, we can compute the overall open-loop and closed-loop transfer functions using 3.15:

$$L(s) = R_{pos}(s) \cdot F_{speed}(s) \cdot \frac{1}{s} = \frac{2890}{s^2 + 170s} \tag{3.16}$$

$$F(s) = \frac{L(s)}{1 + L(s)} = \frac{2890}{s^2 + 170s + 2890} \tag{3.17}$$

As seen in figure 3.12, by comparing transfer function $L(s)$ (the red one) with the one obtained in equation 3.10 (the black one) we can conclude that, in our range of interest, both the transfer functions behave in the same way, with the same gain and crossover frequency, so the same conclusion reached previously are valid also for this controller.

3.3.3 Overall controller implementation

Once the controller is implemented using Matlab and Simulink, the physical system behavior is not similar to the simulation one. Firstly, we have seen that this kind of controller follows the tracking in a better way with respect to proportional controller, because it does not present steady state error (also when load is attached) nor overshoots. But, as it can be seen in the pictures below, the voltage measurement has periodic spikes: this behaviour is caused by the fact that the reference speed is not measured by any sensor, but it is fed into the system by deriving the encoder measurements. This implies that we have a derivative action together with the controller we have designed, which was not present at all in simulation, due to the continuous time of the compiler. This kind of high frequency spikes are not acceptable because they do not allow us to perform frequency validation for this controller and because they can create troubles in reference tracking, in particular for the ball position reference tracking. In order to avoid them, we try to reduce the ratio between inner and outer loop: the voltage signal becomes "cleaner" but high frequency spikes are already present. Moreover, reducing too much the ratio between outer and inner loop (less than 4 to 6 times) could compromise the approximation of the inner closed loop function because it is not enough faster with respect to the outer one and for this reason cannot be seen as a unitary transfer function. For all these reasons, we can conclude that we cannot choose this controller in order to control DC motor, due to its voltage spikes.

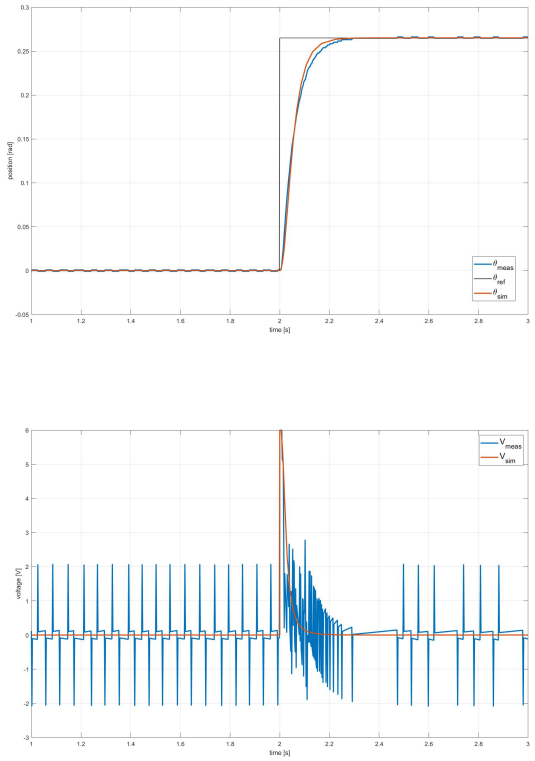


Figure 3.13: Comparison to response to a step of $\frac{\pi}{12}$ in simulation and in real system

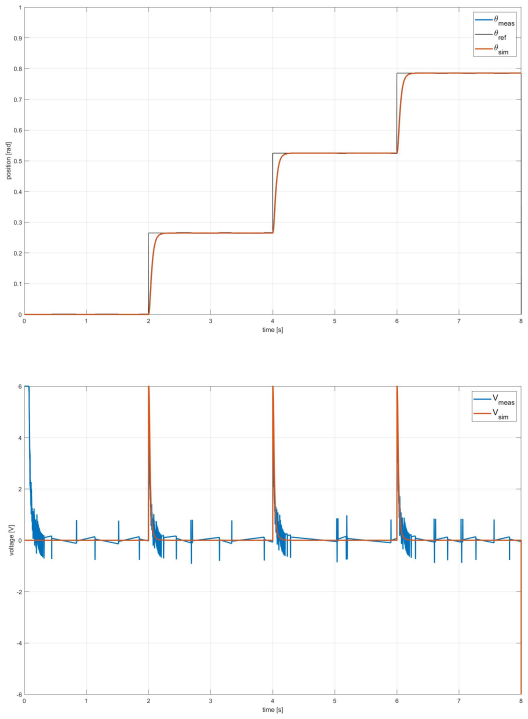


Figure 3.14: incremental $\frac{\pi}{12}$ steps responses: in this experiment, inner loop was set 5 times faster with respect to the outer loop in order to see the effects on voltage

3.4 Lead-lag control

Our third control strategy for motor controllers is a lead-lag compensator: this simple controller is designed in two parts using root locus. The first part, called "phase-lead", allows to place the roots wherever we want, dragging the root locus centroid and closed-loop poles to the left, thus adding stability. The second one, called phase lag, addresses the steady state error problem: however, since we do not have this problem in our system because it has a pole in the origin, a practical rule is to keep this part close to the imaginary axis, reducing moving roots. Our system can be represented in closed-loop by the transfer function 3.1, which can be rewritten as follows to highlight the closed-loop poles.

$$F(s) = \frac{61.53}{(s + 1.8568)(s + 33.136)}$$

The controller is designed with the following structure:

$$R_{LL}(s) = k \cdot \frac{s + \omega_{Zlead}}{s + \omega_{Plead}} \cdot \frac{s + \omega_{Zlag}}{s + \omega_{Plag}} \tag{3.18}$$

3.4.1 Phase lead compensator

As for the other controllers, we have defined a set of design specifications, accordingly with the ones chosen for others controllers:

$$e_{\text{inf}} = 0 \qquad \%overshoot = 0.05 \qquad T_s = 0.1s$$

In this case however we could not choose a settling time of $0.3s$ just as before because the centroid would not be moved enough: because of this, we have chosen $T_s = 0.1s$, which will lead to a much faster controller with respect to the previous two. From these specifications we can obtain the corresponding damping ratio ξ and natural frequency ω_n:

$$\xi = \sqrt{\frac{\ln^2 0.05}{\ln^2 0.05 + \pi^2}} = 0.69 < 1 \qquad \omega_n = \frac{4}{T_s \xi} = 57.971 \frac{rad}{s}$$

These requirements imply that closed-loop poles should be placed in $-\xi\omega_n \pm i\omega_n\sqrt{1-\xi^2} = -40 \pm i41.96$. Plotting the root locus for our system shows that, obviously, these requirements are not met yet. The centroid of the open-loop root locus of the DC motor is located in -17.495.

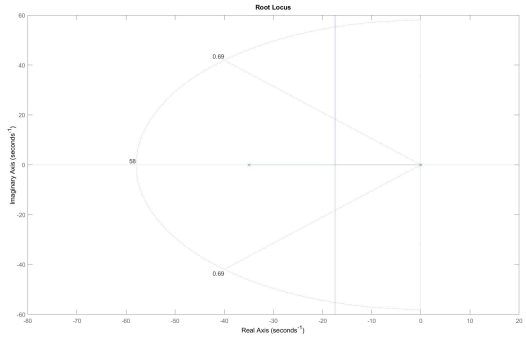


Figure 3.15: Root locus plot of the motor transfer function

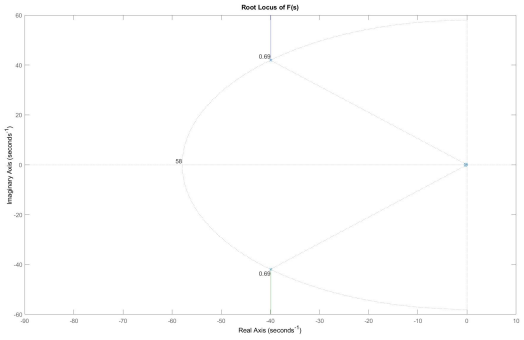


Figure 3.16: Desired plot for closed-loop transfer function

The shape of lead compensator is:

$$R_{Lead}(s) = k \cdot \frac{s + \omega_{Zlead}}{s + \omega_{Plead}} \tag{3.19}$$

Recalling the position transfer function G_{pos}, in order to further simplify the design, we choose to put a zero in the controller used for a pole-zero cancellation. This is done by choosing $\omega_{Zlead} = 34.99$: in this way we cancel out the system pole, having freedom to place a new pole in a desired position. To determine the location of the pole of the phase lead part, which must be $\omega_{Plead} > \omega_{Zlead}$ to respect the characteristics of this kind of controller, we use the Graphical Method and we have computed:

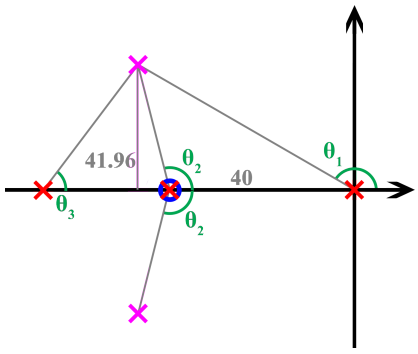


Figure 3.17: Qualitative plot of the root locus used as reference for next computations (open-loop poles are the red ones, closed-loop poles the pink ones)

$$\tan(180° - \theta_1) = \frac{41.96}{40} = 1.049 \qquad \theta_1 = 133.63°$$

$$\sum \theta_P - \sum \theta_Z = 180°$$

$$\theta_1 + \theta_2 + \theta_3 - \theta_2 = 180°$$

$$\theta_3 = 180° - \theta_1 = 46.37°$$

$$1.049 = tan\theta_3 = \frac{41.96}{\omega_{Plead} - 40}$$

$$\omega_{Plead} = 80\frac{rad}{s}$$

In order to have the pole above, the resulting transfer function must be:

$$L(s) = 61.52 \cdot k_{lead}\frac{1}{s(s+80)} \cdot \frac{\cancel{s + 34.99}}{\cancel{s + 34.99}}$$

$$F(s) = \frac{L(s)}{1 + L(s)} = \frac{61.52 \cdot k_{lead}}{s^2 + 80s + 61.52 \cdot k_{lead}}$$

By comparing $F(s)$ with a generic second order transfer function, we obtain that $\omega_n = 57.97 \frac{rad}{s}$ as said before, but most importantly we get:

$$k_{lead} = \frac{\omega_n^2}{61.52} \approx 54.62$$

Now that we have all the components, we can plug these values in the lead part of the controller transfer function 3.19.

$$R_{Lead}(s) = k_{lead} \cdot \frac{s + \omega_{Zlead}}{s + \omega_{Plead}} = 54.62 \frac{s + 34.99}{s + 80} \tag{3.20}$$

$$L(s) = \frac{3361}{s(s + 80)} \tag{3.21}$$

3.4.2 Phase lag compensator

Now we can focus on the other part of the regulator, the phase lag. We need to add this part because using only the lead one would mean having a steady state error: in this case, however, if we look at our system described by 3.21, we see that it is of type 1, so zero steady state error for a step input is already guaranteed. Having said that, since steps will not be the only input of our motor, we choose to put a design requirement on the steady state error with a ramp reference:

$$e_{\inf\, desired} = 0.01 \tag{3.22}$$

The actual steady state error for an input ramp is:

$$e_{\inf\, ramp} = \lim_{s \to 0} s \cdot \frac{1}{1 + L(s)} \cdot \frac{1}{s^2} = \frac{s + 80}{s^2 + 80s + 3361} \approx 0.0238 \tag{3.23}$$

The controller can be rewritten as:

$$R_{Lag}(s) = \frac{s + \omega_{Zlag}}{s + \omega_{Plag}} = \frac{\omega_{Zlag}}{\omega_{Plag}} \cdot \frac{\frac{s}{\omega_{Zlag}} + 1}{\frac{s}{\omega_{Plag}} + 1} = k_{lag} \cdot \frac{\frac{s}{\omega_{Zlag}} + 1}{\frac{s}{\omega_{Plag}} + 1}$$

$$L(s) = \frac{3361 \cdot k_{lag} \left(\frac{s}{\omega_{Zlag}} \right) + 1}{s(s + 80) \left(\frac{s}{\omega_{Plag}} + 1 \right)}$$

With this last one transfer function, using final value theorem for a unitary input ramp as before and imposing the design requirement on steady state error, we get that:

$$k_{lag} = 2.38 \qquad \omega_{Zlag} = 2.38 \cdot \omega_{Plag}$$

We have chosen to place the pole near the dominant one in $s = 0$ in order to neglect its effect: to have a good difference in magnitude with other poles and zeros we have chosen $\omega_{Plag} = 0.1$, which implies $\omega_{Zlag} = 0.238$. So, the shape of lag part of the controller is obtained by plugging these values in the corresponding part of 3.18.

$$R_{Lag}(s) = \frac{s + \omega_{Zlag}}{s + \omega_{Plag}} = \frac{s + 0.238}{s + 0.1} \tag{3.24}$$

Now that we have computed all the poles and zeroes, we are able to write the complete structure of lead-lag controller:

$$R_{LL}(s) = 54.62 \cdot \frac{(s + 34.99)(s + 0.238)}{(s + 80)(s + 0.1)} \tag{3.25}$$

The open-loop and closed-loop transfer functions are:

$$L_{LL}(s) = \frac{3360s + 799,9}{s^3 + 80.1s^2 + 8s} \tag{3.26}$$

$$F_{LL}(s) = \frac{3360s + 799,9}{s^3 + 80.1s^2 + 3368s + 799.9} \tag{3.27}$$

Accordingly with Matlab plots, with this controller we have obtained a phase margin of $\varphi_m = 64.4°$, the desired settling time of $T_s = 0.1s$, and a crossover frequency of $\omega_c \approx 38$ rad/s, that allows us to have a bigger frequency band with respect to previous controllers, with a good signal tracking.

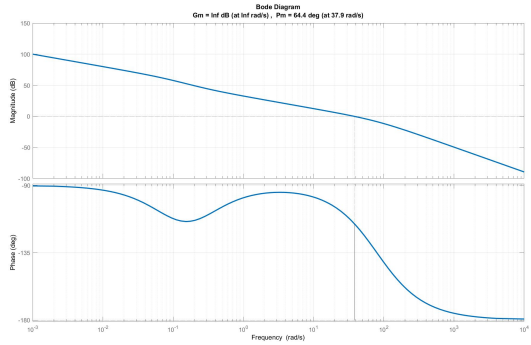


Figure 3.18: Bode plot of L(s)

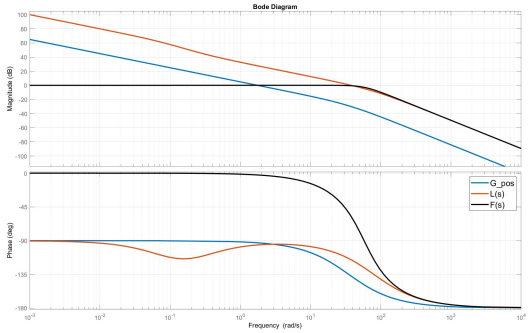


Figure 3.19: Frequency analysis of lead-lag controller

3.4.3 Overall controller implementation

Once the controller is implemented using Matlab and Simulink, the physical system behavior is very similar to the simulated one. We have seen that this kind of controller follows the tracking in a better way with respect to proportional controller, because it presents a smaller, even if present, steady state error (position signal bigger than reference) in the step following reference: moreover, as the specification suggest, the system is fast and tracks the reference in 0.1s. The voltage usage does not present any spikes, with a clean signal with every kind of step reference angle. We validate this controller giving as input a square reference for the angle that varies between $\pm\frac{\pi}{4}$, with an increasing frequency from 0.1 to 4Hz in 100s. As expected, it follows the reference in a good way until it reaches the crossover frequency, in fact when the step duration is less than the rise time, it cannot reach the reference signal and, for this reason, the output starts to be attenuated. When we attach the load to the motor, the reference tracking does not suffer the same problems of the proportional controller: the small steady state error previously mentioned disappears thanks to the load weight, guaranteeing a correct reference tracking for a step reference angle.

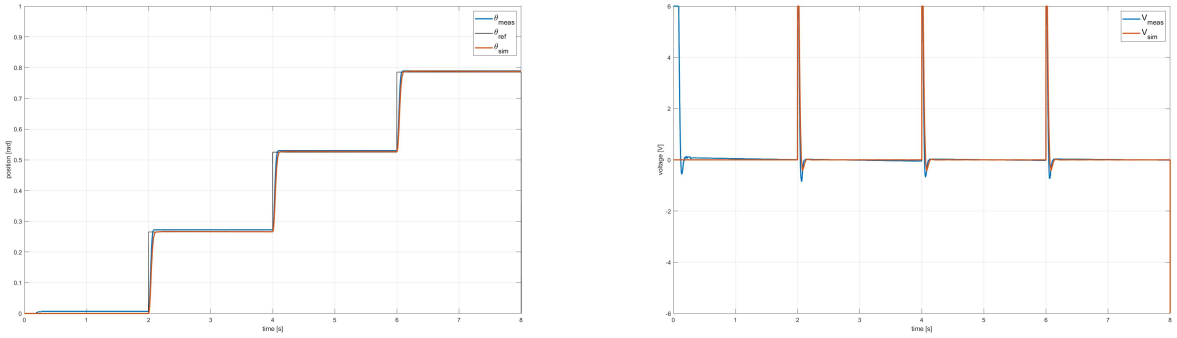


Figure 3.20: incremental $\frac{\pi}{12}$ steps responses

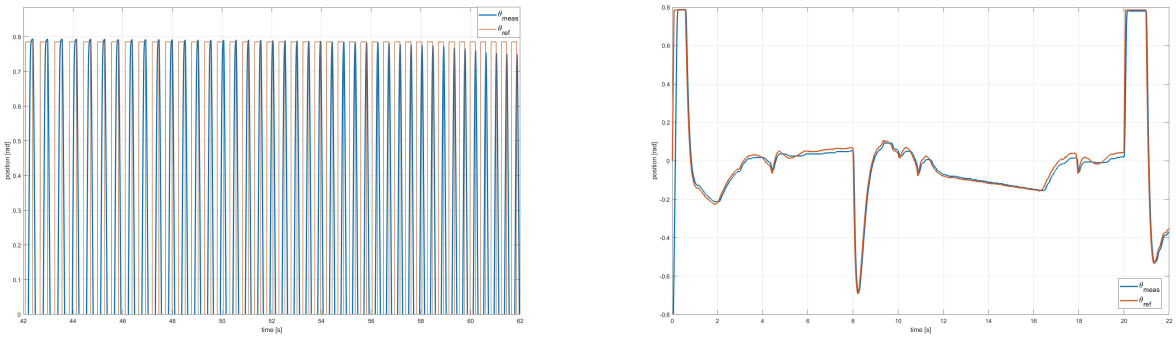
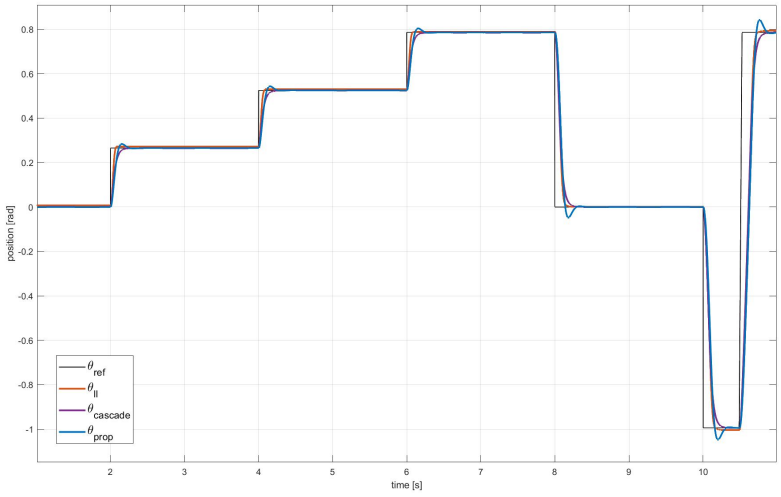


Figure 3.21: On the left: detail of the frequency response of the lead-lag controller. On the right: response of the controller during a ball trajectory tracking experiment (load is attached)

Also comparing the step response of this controller with the other two developed controllers, it is clear that this controller is the best one we have developed, due to the fact that it is the fastest, does not present steady state errors as the proportional one, and uses the voltage in a better way with respect to the cascade controller. In addition, its response with attached load is very satisfactory as it can be appreciated from figure 3.21 guaranteeing precision and fast responses. For all these reasons, we can conclude that this controller is the best choice we can take in order to control DC motor, and it will be the θ position controller that we will use during the ball position control.



Chapter 4

Ball position control

We tried different strategies to control the ball position: a frequency-based controller (phase lead), two state-space controllers (pole placement and linear quadratic regulators), and one model predictive controller.

4.1 Phase Lead controller

4.1.1 Controller development

The lead compensator is designed to improve the frequency response of the ball and beam subsystem: since it is an unstable system, the first control goal is to stabilize it and, while doing it, meeting the set requirements. This traditional controller design is based on system transfer function 2.17, with θ control variable and position x as output. The general idea, just as for the lead-lag compensator designed for the motor, is to use root locus to choose the location of a pole and a zero which will give us the desired behavior of the system.

$$R_{PL}(s) = k \cdot \frac{s + \omega_{Zlead}}{s + \omega_{Plead}} \tag{4.1}$$

The requirements set for this control strategy are:

$$T_s = 2.5s \qquad \%overshoot = 0.02 \approx 0.8cm$$

These requirements resulted in the following values for damping ξ and natural frequency ω_n, accordingly with the formulas for second order systems.

$$\xi \approx 0.78 \qquad \omega_n = 2.05 \frac{rad}{s}$$

This means that the closed-loop poles must be in $-1.6 \pm i1.283$. Looking at the root locus plot from transfer function 4.1, since it has two poles in $s = 0$, we know it is impossible to obtain such closed loop poles with a simple proportional gain, because we cannot move the centroid placed in $x_A = 0$. In order to simplify the design and easily shift the root locus to the left, we opted for a pole-zero cancellation: the zero of the controller was chosen in $\omega_{Zlead} = 0.1$, near the origin, to cancel out one of the poles of the system transfer function. To choose the new pole location we use the graphical methods as made for the phase-lead part of the lead-lag controller designed for the DC motor, which are reported in chapter 3. we have obtained $\omega_{Plead} = 3.2$ rad/s and, substituting the pole and zero locations in the regulator and computing the closed-loop transfer function, we have obtained a gain $k = 10.05$. So, the phase-lead controller and the open-loop transfer function (recalling 2.17) are:

$$R_{PL}(s) = 10.05 \cdot \frac{s + 0.1}{s + 3.2} \qquad L_{PL}(s) = \frac{4.211s + 0.4211}{s^3 + 3.2s^2}$$

With the aid of Matlab we have obtained with this controller a phase margin of $\varphi_m = 64°$ and a crossover frequency of $\omega_c = 1.23$. The phase lag part was not developed at all because it would have been composed of a pole-zero couple

really close to zero, having almost no impact on the system.

Once we tried to simulate the system, we have noticed that the performances were not the ones expected: when the system received a step reference it did not reach the desired value, behaving like a much smaller step was fed into the system. This is due to the actuator saturation effect on the θ angle, which made it impossible for the system to behave correctly. Since this controller can be rewritten as a PID one, we can think of introducing an anti-windup configuration. It can be implemented directly with a controller of this kind, with an auxiliary controller which depends only on the regulator parameters.

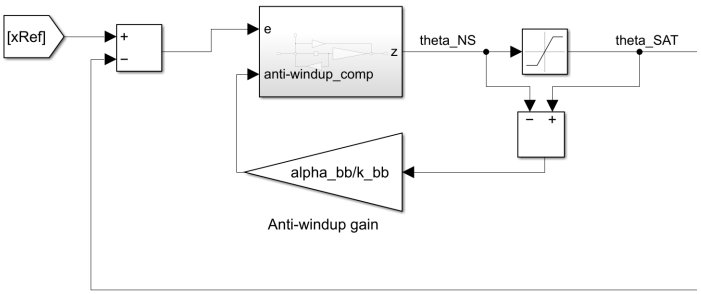


Figure 4.1: Anti-windup configuration implemented

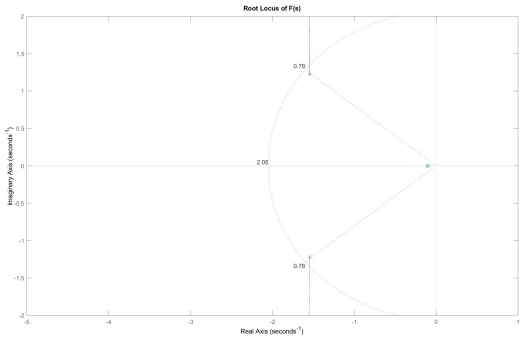


Figure 4.2: Root locus of closed-loop transfer function

To compute the anti-windup gain $m = \frac{\alpha}{K}$, which minimizes the 2-norm of the difference between the saturated and non-saturated control variable $\bar{e} = u - z$, we need to rewrite the controller transfer function as follows:

$$R_{PL}(s) = K \cdot \frac{1 + Ts}{1 + \alpha Ts}$$

$$K > 0 \qquad T > 0 \qquad 0 < \alpha < 1$$

For our regulator we have computed all the values above, obtaining the following results:

$$R_{PL} = 0.3147 \cdot \frac{1 + 10s}{1 + 0.313s}$$

$$m = \frac{\alpha}{K} = 0.0993 \tag{4.2}$$

4.1.2 Controller implementation

Once we implemented the anti-windup configuration in the control scheme, when simulating the complete system, the results were much better than before: as expected, the system should reach the reference value without any actuator saturation issue, with a settling that satisfies the specification. However, once we tested this controller on the real system, we saw that the behaviour was different with respect to the simulated one due to a big steady state error (up to 3cm) that can be clearly seen in the following plots in figure 4.3 (PL_1). The settling time meets the specification and the voltage usage is also acceptable (similar to the simulated one shown below), without the presence of high frequency spikes with high amplitude. After some attempts, we placed the pole in -4.5, a location far to -3.2, so without using the one suggested by the graphical method (obviously our Matlab scripts recompute 4.2): we obtained better performances, with a more aggressive control action, but steady-state error of around 2cm was still present. However, increasing the pole worsens the use of voltage that presents spikes with high amplitude, as shown below in figure 4.4. Moreover, repeating experiments with this controller several times, the system does not behave always in the same way (in terms of tracking and steady state error). This repeatability problem, added to the other problems already highlighted, allows us to understand that this kind of controller is not a good choice for the trajectory tracking of the ball.

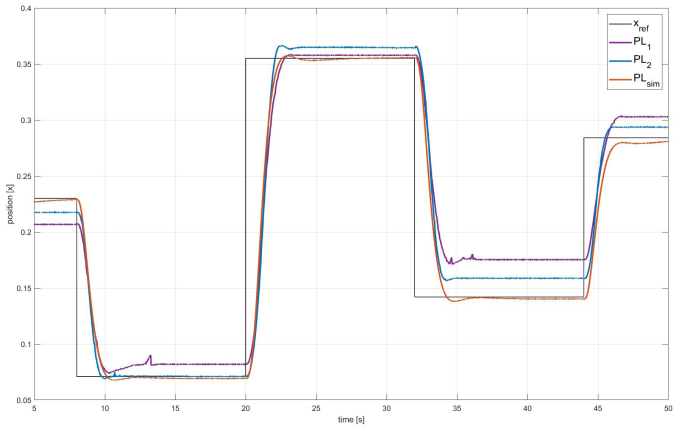


Figure 4.3: Comparison between the two phase-lead controller described above: PL_1 follows eq. 4.2, PL_2 forces the pole in -4.5

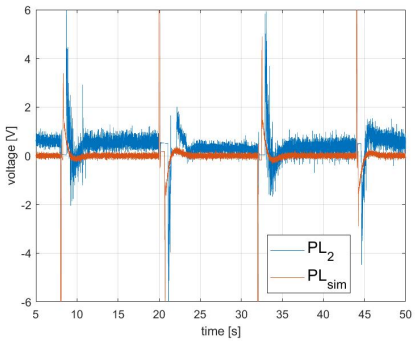


Figure 4.4: Comparison between voltage usage of PL_2 in real system and in simulation

4.2 Pole placement controller

One of the control strategies we chose for the ball position control is Pole placement. During the various laboratory sessions we tried different approaches involving this technique to improve more and more the performances in terms of settling time, overshoot and steady state error. All the pole placement strategies we tried use a full-order asymptotic state observer for both the position and the speed of the ball (the system is observable). The state observer we used is based on pole placement and the choice of the poles position we used, together with the explanation of the form of the observer, is treated at the end of the "Pole placement controller" discussion. We also tried a reduced-order observer, but since the position measurement coming from the potentiometer is very noisy, the system was affected by too much noise and we had a very unsuitable control action in terms of voltage (noisy and full of high spikes).

4.2.1 Pole placement without integral action

First of all, we tried a pole placement technique that simply involved our system, the observer, and the feedback gain generated by this control strategy. The control law that features this kind of controller is the following:

$$u(t) = -K \cdot x(t) + \overline{u}(t)$$

In order to compute the feedback gain K that characterizes the pole placement technique, knowing from chapter 2 that the system is reachable, we started to obtain all the necessary matrices "by hand" step by step, and at the end we verified that our results were equal to the ones we can obtain with the function "place" provided by Matlab. Since our results and the Matlab ones coincided, we used just the Matlab function in the following tests. The hardest part of pole placement technique is to chose where to place closed-loop poles. As suggested in our "Advanced and Multivariable Control" classes we tried to place one of the two poles at low frequency and the other one at a higher frequency to have a system that in closed-loop behaves like a first-order system. To choose the position of the slowest pole we decided a settling time equal to 2.5 seconds and we computed consequently the poles:

$$T_s = 2.5 \qquad p_1 = -\frac{5}{T_s} = -2 \qquad p_2 = 10 \cdot p_1 = -20$$

These poles generate a pole placement gain vector equal to: [95.667 52.6168]. Unfortunately with these poles and with this gain the system response was very unsuitable, in fact it was not like a first-order system response (the feedback gain was too high), and the steady-state error were so large that it was impossible to compute the settling-time. Plots of the response are present in figure 4.5 under the name "case 1".

The second experiment we made it has been that to place the poles in such a way that the system behaves like a second-order system. To do so, we set a desired settling time again equal to 2.5 seconds and a desired overshoot equal to 2%. Here you can find the poles position computation:

$$T_s = 2.5 \qquad OS\% = 0.02$$

$$\xi = \sqrt{\frac{\log(OS\%)^2}{\log(OS\%)^2+\pi^2}} = 0.7797 \qquad \omega_n = \frac{4}{T_s\xi} = 2.0521$$

$$p_1 = -\omega_n\xi + j\cdot\omega_n\sqrt{1-\xi^2}) = -1.6 + j\cdot 1.2849 \qquad p_2 = -\omega_n\xi - j\cdot\omega_n\sqrt{1-\xi^2}) = -1.6 - j\cdot 1.2849$$

These poles generate a pole placement gain vector equal to: [10.0712 7.6534]. Unfortunately, also in this case, the feedback gain was too much high and so, even if the performances improved, the steady state error was too large to consider acceptable this control strategy with these poles and this form (plots under the name "case 2" in figure 4.5). The last test we performed with pole placement technique without integrator was that in which we tried the best poles position (from the point of view of performances) we found through trial and error in simulation. These poles were:

$$p_1 = -0.65 \qquad p_2 = -0.65$$

(Actually p_2 was in - 0.65 + 0.00001 in order to be able to use the function "place")

These poles generates a feedback gain equal to [1.0105 3.1092] that allowed a better response (plots under the name "case 3" in figure 4.5) with respect to the first two cases. However, because of the absence of the integral action, the system response was not satisfying (as you can appreciate from the plots). In fact, to use a lower feedback gain generated a faster response but also a steady-state value higher than the reference, while an higher feedback gain caused a slower response and in most cases a steady-state value lower with respect to the reference.

Below you can find the plots of the experiment performed in laboratory sessions with the 3 cases:

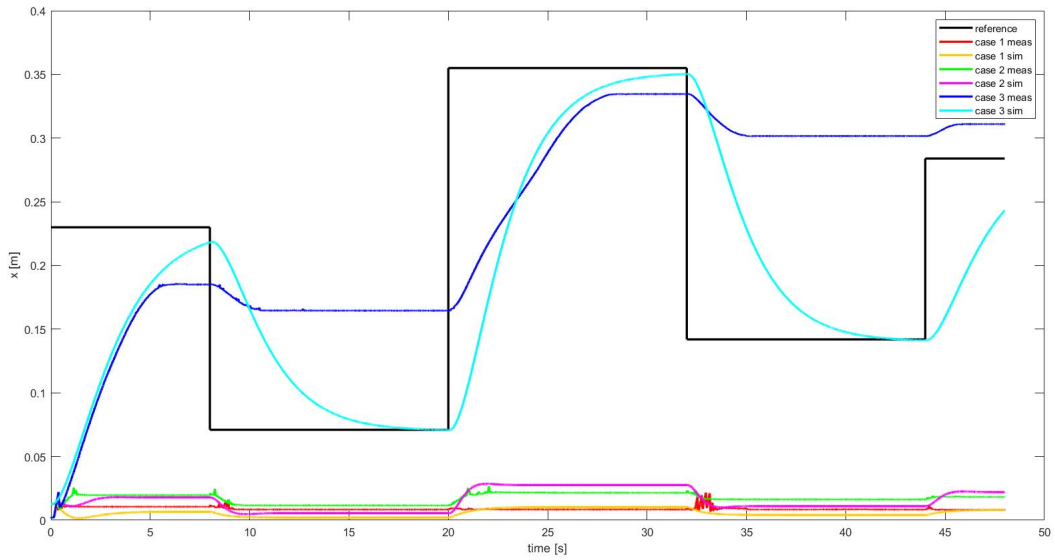


Figure 4.5: Responses of the system with different poles positions (measured and simulated)

4.2.2 Pole placement with integral action

Since the results obtained with the previous strategy were not satisfying, we decided to try to enlarge the system by adding an integrator in order to have a null steady-state error and to reduce the system settling time.

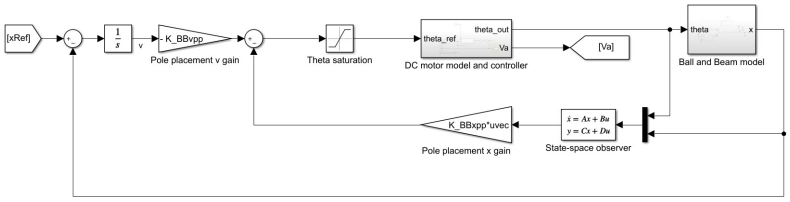


Figure 4.6: Control scheme of the Pole placement control with integral action

With this new control scheme we needed new state-space matrices that represented the system, so given the state-space representation of the ball and beam subsystem present in chapter 2, we defined:

$$\tilde{A} = \begin{bmatrix} A & \begin{matrix} 0 \\ 0 \end{matrix} \\ \begin{matrix} -C & 0 \end{matrix} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix} \qquad \tilde{B} = \begin{bmatrix} B \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{5 \cdot g \cdot r_{arm}}{7 \cdot L_{beam}} \\ 0 \end{bmatrix}$$

With these new matrices we checked the reachability of the enlarged system and we discovered that the system is still reachable. The observability has not to be checked again since we want to observe just the actual states of the system (as done previously), and not the state of the error (that is well known given that we know the reference and we measure the position). With the integral action added, 3 poles need to be placed and the control law becomes:

$$u(t) = -\begin{bmatrix} K_x & K_v \end{bmatrix} \cdot \begin{bmatrix} x(t) \\ v(t) \end{bmatrix} + \overline{u}(t)$$

so the gain vector K generated by the pole placement approach had to be split into two as follows:

$$K = \begin{bmatrix} k_1 & k_2 & k_3 \end{bmatrix} \qquad K_x = \begin{bmatrix} k_1 & k_2 \end{bmatrix} \quad K_v = \begin{bmatrix} k_3 \end{bmatrix}$$

Please notice that, as requested by the theory, the gain K_v is changed of sign in the control scheme (same thing for K_x but changed of sign through the negative feedback).

The first poles position we tried is based on the idea to make the system behave in closed-loop like a system of the second order and so, like in the case 2 of chapter 4.2.1 (with the same specifications), we used:

$$p_1 = -1.6 + j \cdot 1.2849 \qquad p_2 = -1.6 - j \cdot 1.2849$$

and $p_3 = 10 \cdot real(p_1)$ to place it at high frequency, generating a pole placement gain
$K = [135,3098 \quad 45.9201 \quad -205,6977]$. With these values, in the simulation, the system diverges after some reference steps, while in the real world the system does not diverge but the behaviour is not the desired one and the overshoot requirement is not satisfied (the ball hits continuously the end of the track). The oscillatory behaviour caused by the too big overshoot causes also the missed satisfaction of the request about the settling time. Plots of the response are present in figure 4.7 under the name "cc poles".

Since this response was not satisfactory we tried to place all the poles in the same place (that is because in simulation we saw that placing them like this made them easily tunable and above all it generated a very good system response). After some tests in simulation and in lab we discovered that the best position for those poles is in -2.3 (actually p_1 = - 2.3, p_2 = - 2.30001, and p_3 = - 2.30002 in order to be able to use the function "place"), in fact, with these poles, the pole placement gain is $K = [37.95364 \quad 16.5021 \quad -29.0969]$ and the response has (depending on the step we give as reference) these characteristics:

Performance indexes	**Minimum values**	**Maximum values**
Steady state error [mm]	$\cong 0.5$	$\cong 2$
Overshoot	0	$\cong 3\%$
Settling time [s]	$\cong 2.5$	$\cong 4$

Plots of the response are present in the figure below under the name "same position poles".

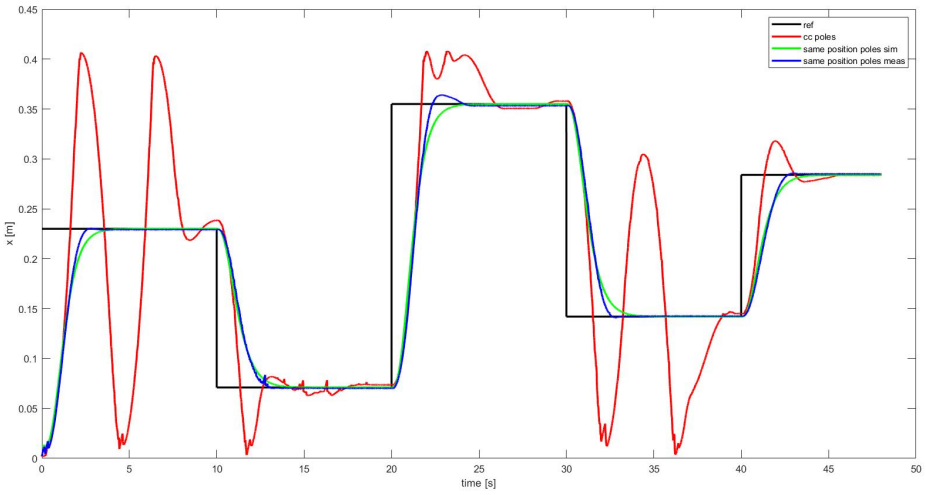


Figure 4.7: Responses of the system with different poles positions (measured only for the 'cc poles' case and measured and simulated for the other case)

4.2.3 Pole placement: observer

The observer has been used with the pole placement strategy because this is a state-space based control technique and we had just a noisy measure of the ball position (and not the measure of the speed).
This observer has been implemented in Simulink as a system in the state-space form with matrices:

$$A_{obs} = A - L \cdot C \qquad B_{obs} = \begin{bmatrix} B & L \end{bmatrix} \qquad C_{obs} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad D_{obs} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

A, B, C, D are the matrices of the state-space representation of the ball and beam subsystem, and L is vector generated by the observer pole placement, with the eigenvalues of $A - LC$ that determine the dynamic of the observer. From the theory we know that the dynamic of the observer must be faster than the dynamic of the system and that usually the poles of the observer are much faster than the closed-loop poles of the system to be controlled (usually they are 5, 10 times the poles of the system). The problem with too much fast poles for the observer is that the faster one pole is, the less it filters the measurement noise.
Having in mind all of these considerations, we tried several positions for the observer poles (whose results on the control voltage are shown in the figure below) until we discovered that to not include too much noise it is better to have observer poles that are twice the fastest closed-loop poles of the system, so:

$$l_1 = l_2 = 2 \cdot (-2.3) = -4.6$$

that generate a gain vector

$$L = \begin{bmatrix} 9.2 \\ 21.16 \end{bmatrix}$$

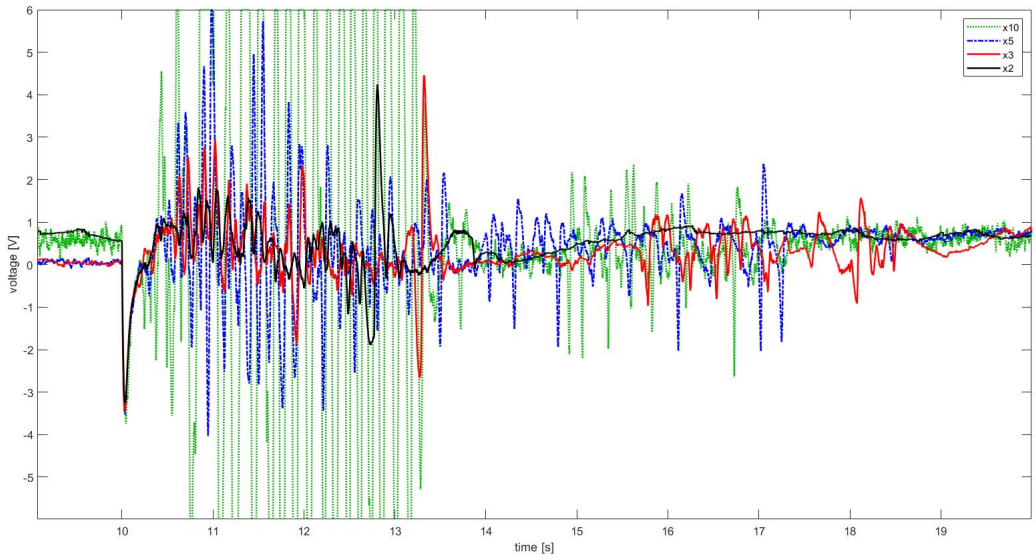


Figure 4.8: Section of the control variable V_a plots generated using different observer poles positions (system closed-loop pole multiplied by 10, 5, 3, and 2)

As we can appreciate from the figure above, with the right configuration of the state observer, the voltage usage of this controller is acceptable and does not present high frequency spikes.

4.3 LQ regulator

Another controller strategy (belonging to state-Space controllers class) used for the ball position control is the Linear Quadratic Regulator applied to the linearized system described in chapter 2. The control problem is transformed into an optimization problem for minimize the following cost function:

$$J(x_0, u, 0) = \int_0^\infty (x(t)'Qx(t) + u(t)'Ru(t))dt$$

with the diagonal matrix of states weight (the first for position, the second for the speed) $Q = Q' \geq 0$ ($Q = C_q'C_q$) and the matrix of input weight $R = R' > 0$ (1x1 matrix, having only 1 input). After the verification of Reachability for the pair (A, B) and Observability for (A, C_q) conditions (rank of the O_{bs} and R_{each} is equal to the number of states of the ball and beam system), we can compute the state-feedback control law:

$$u(t) = -(R^{-1}B'P)x(t) = -Kx(t) \tag{4.3}$$

where P is found by solving the stationary algebraic Riccati equation:

$$A'P + PA - (PB)R^{-1}(B'P) + Q = 0 \tag{4.4}$$

Thanks to this, we are guaranteed that the closed-loop system $\dot{x}(t) = (A - BK)x(t)$ is asymptotically stable. In order to properly feed the correct feedback states to the LQ_∞ gain and therefore tuning the weights of Q matrix, we need to know the values of measured states provided by sensor. As previously said, the only sensor in which we rely on is the potentiometer: for what concerns the states of ball speed, we initially try to obtain the position values at each sample time deriving the position state in order to know the speed value. However, once we connect the controller feeding the measures as described, the system does not work. This is due to the fact that the derived measurement of the speed was very noisy: even if we consider a very low value of the matrix Q component for the ball speed (in order to penalize the speed with respect to the position measurements) we cannot control the system. For this reason

we apply another strategy: we develop a Kalman Filter and we apply it to our LQ_∞ regulator in order to properly estimate both the position and more stable signal for the speed, in a similar way that happen for pole placement with observer.

4.4 LQ regulator with a Kalman filter

4.4.1 Controller development

The Linear Quadratic Gaussian controller is defined by a LQ_∞ regulator (described above) and a Kalman Filter, assuming the presence of white gaussian noises on the state dynamics and on the measurement (as previously verified). Considering the continuous time system:

$$x = Ax + Bu + w$$

$$y = Cx + Du + \nu$$

with known inputs u, process disturbances w (not measurable), and measurement noise ν, the Kalman filter provides optimal estimate $x_e(t)$ of $x(t)$ states through $y_e(t)$ by:

$$\begin{bmatrix} \dot{x}_e(t) \\ y_e(t) \end{bmatrix} = \begin{bmatrix} A_{kf} \\ C_{kf} \end{bmatrix} x_e(t) + \begin{bmatrix} B_{kf} \\ D_{kf} \end{bmatrix} u(t)$$

where:

$$A_{kf} = [A - L_{kf}C] \quad B_{kf} = [B - L_{kf}D, L_{kf}] \quad C_{kf} = I \quad D_{kf} = 0 \tag{4.5}$$

and A, B, C and D are the state-space matrices that represent our system.The process noise w and measurement noise ν are Gaussian white noises with variance matrices:

$$Q_{kf} = E[ww'] \quad R_{kf} = E[\nu\nu'] \tag{4.6}$$

that are assumed as diagonal matrices. For the measurements noise, we found the value $R_{kf} = 1 \cdot 10^{-8}$ (as reported at the end of chapter 2), for the process noise, there were found the same values for both the components $Q_{kf} = 1 \cdot 10^{-6}$ by tuning them in laboratory; so, we place more confidence on the sensor measurement (noise of measurement is smaller) having a fast observer. After verifying that the pair (A, B_q) was reachable (with $Q_{kf} = B_q B_q'$) and that the pair (A,C) was observable, we can conclude that the observer is asymptotically stable and the optimal the gain for a steady state Kalman filter is given by:

$$L_{kf} = P_{kf}C'R_{kf}^{-1} = \begin{bmatrix} 10.95 \\ 10.00 \end{bmatrix}$$

where P_{kf} is the unique positive definite solution of the stationary Riccati equation:

$$AP_{kf} + P_{kf}A_{kf}' - P_{kf}C'R_{kf}'CP_{kf} + Q_{kf} = 0$$

The comparison between KF estimation and real measurements will be treated in the next chapter.

4.4.2 Controller implementation

We have developed this controller in simulation, in which we start to tune the Q and R matrices of the LQ controller in order to understand the behaviour of the system and the correct order of magnitude of the weight matrices. After this initial process, we have continued our tests with the real system, trying to tune weights in order to reach the best result possible. We immediately noticed that the behaviour was different with respect to the simulated one in terms

of reference tracking. After taking different experiments, we have seen that a big steady state error not present in simulation was always present between the reference and the position of the controlled system; it can be decreased through the tuning of the parameters (we reached a minimum value of 1.5cm), without been able to remove it.

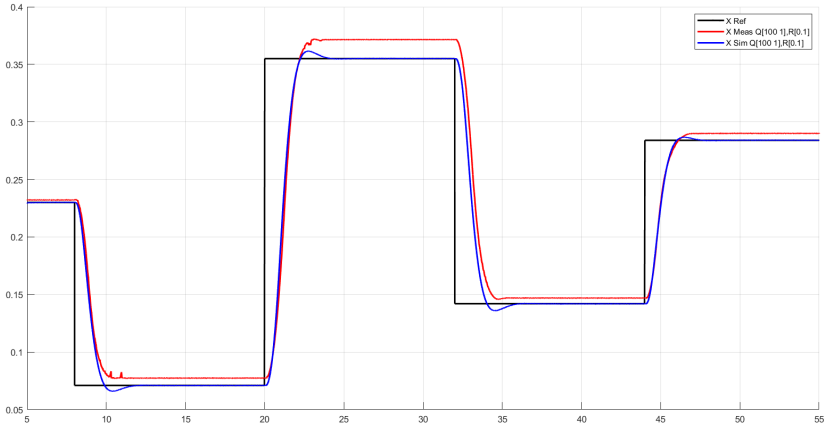


Figure 4.9: Comparison between simulated and real behaviour for LQG with $Q = [100, 1]$ and $R = [0.1]$

We consider this kind of error unacceptable, which was present because our linearized system is not a perfect realization of our real system (due to non-linearities, approximations,...). For this reason, we decide to add an integral action for the tracking error between the reference and the controlled variable, enlarging the system as done for pole placement controller.

4.4.3 Controlled based on enlarged system development

The structure of the controller is the same as the one with only 2 states previously described with the addition of an integrator and its gain. The considered enlarged system is the same of the pole placement one, with $\tilde{A}$ and $\tilde{B}$, and so it's still reachable. After having verified $O_{bs}(\tilde{A}, \tilde{C}_q)$ we understood that we have to obtain the estimation only for the real system states and not for the error one (well-know since we know the reference and the position measurement), and then Kalman Filter remains the same as the one of non-enlarged system. So, we can compute the gain K using 4.3, splitting it in the same way of pole placement with the enlarged system. This gain is affected by the weights of the matrices Q and R of LQR, with $Q = C_q' C_q$ that now is a 3-by-3 matrix, with the third value of the diagonal that represents the tracking error state.

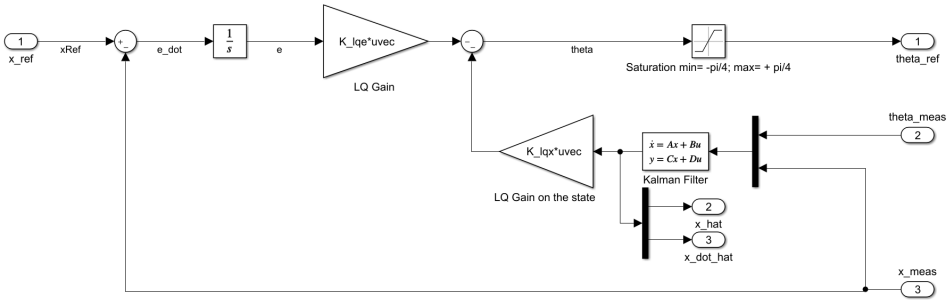


Figure 4.10: Simulink model of the LQG enlarged system

4.4.4 Controller implementation

We have developed this controller in simulation, where we started to tune the Q and R matrices of the LQ controller in order to understand the behaviour of the system and the correct order of magnitude of the weight matrices.

Considering diagonal values of $Q = [q_x, q_{\dot{x}}, q_e]$, the best simulations results were reached setting R=0.1, $q_{\dot{x}}$ 1 order of magnitude bigger than R, q_e 2 orders of magnitude bigger than R, and q_x the half of q_e, due to the different order of magnitude between the states and the control variable. After this initial process, we have continued our tests with the real system, trying to tune weights in order to reach the best result possible. The behaviour of the controller on the real system seems to be better with respect to the simulated one, with less overshoot and a lower settling time. From different experiments in which we tune the weights, we understand that:

- R component should be taken at least 1 order of magnitude lower than $q_{\dot{x}}$ and 3 order of magnitude lower that other two components of Q, in order to speed up the feedback control action (more aggressive) and provide a better tracking of the reference.
- $q_{\dot{x}}$ should be taken at least 2 order of magnitude lower than the other two components of Q, in order to speed up the system without having oscillations.
- if $q_x > q_e$, the control action is less aggressive and the settling time increases, with no presence of overshoot. If the difference becomes high (for example the doubles), the system do not correctly reach the reference.
- if $q_x < q_e$, the control action is more aggressive, this implies a bigger overshoot and the decreasing of the settling time, with some oscillations around the reference before the settling.
- if $q_x \approx q_e$, is a good trade-off between settling time and overshoot, due to the fact that q_e increases overshoot and q_x decreases overshoot reaching reference between 2.5 and 3 s.
- in general, the voltage usage is good, without high frequency spikes with high amplitudes.

To evaluate the best solution considering the values of the matrices Q and R, the controller was tested several times on steps with different amplitudes, analyzing data and evaluating performances for Settling time, Steady State error and Overshoot percentage. Looking at the following table 4.1 in which there are reported some of the best collected experiments, the better solution obtained has $Q = [170, 1, 200]$ $R = [0.1]$, that has a gain $K_{lqx} = [56.5509, 16.7482]$ $K_{lqe} = [-44.7214]$ (The steady state error was measured after 7 s from the given step reference, in order to consider a time bigger than the settling one). This controller is the one used also for trajectory tracking test, thanks to its great values obtained in each field of the evaluated performances.

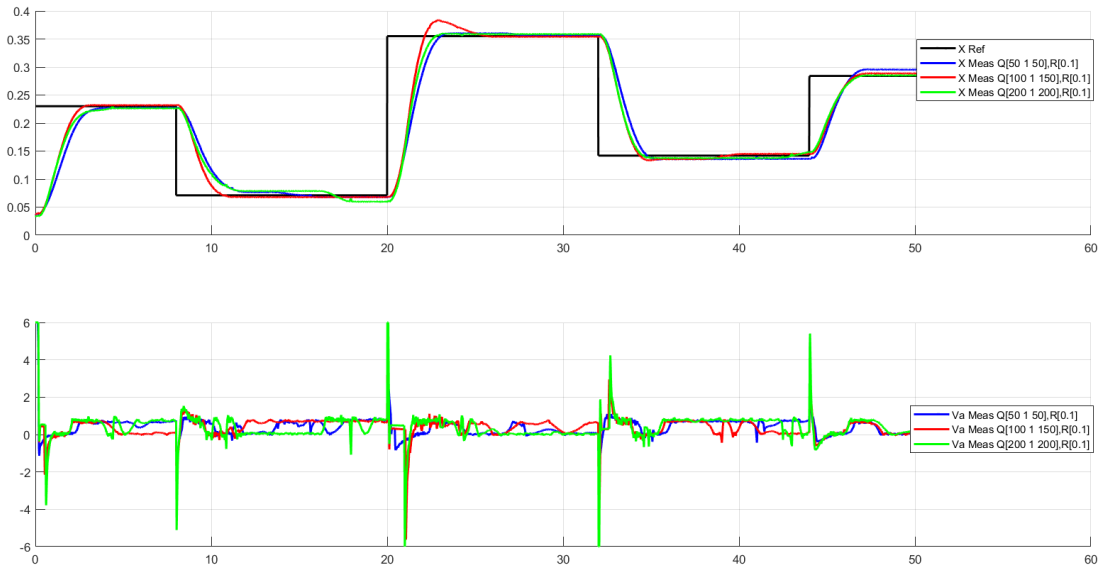


Figure 4.11: Comparison between LQG regulator with different Q matrix on enlarged system

Weighting matrices	T_s^{min} **[s]**	$T_s^{max}[s]$	$SS_e^{min}[mm]$	$SS_e^{max}[mm]$	$OV^{min}[\%]$	$OV^{max}[\%]$
$Q = [100, 1, 150] \quad R = [0.1]$	2.30	6.36	0.95	3.27	0.98	8.05
$Q = [150, 1, 150] \quad R = [0.1]$	2.47	3.56	1.18	3.63	0	6.08
$Q = [200, 1, 200] \quad R = [0.1]$	2.51	6.45	0.37	7.58	0	3.02
$Q = [170, 1, 200] \quad R = [0.1]$	2.19	3.46	0.74	3.54	0	5.71
$Q = [180, 1, 200] \quad R = [0.1]$	2.33	3.15	1.22	3.54	0	4.71
$Q = [50, 1, 50] \quad R = [0.1]$	2.81	12	0.53	11.11	0	5.22
$Q = [100, 1, 100] \quad R = [0.1]$	2.51	7	1.50	5.90	0	3.15

Table 4.1: Performance Indexes for different Q and R matrices

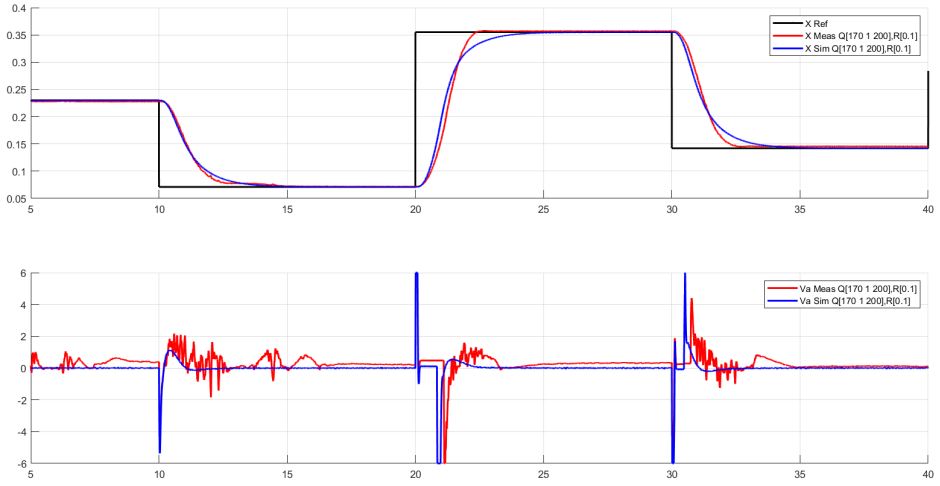
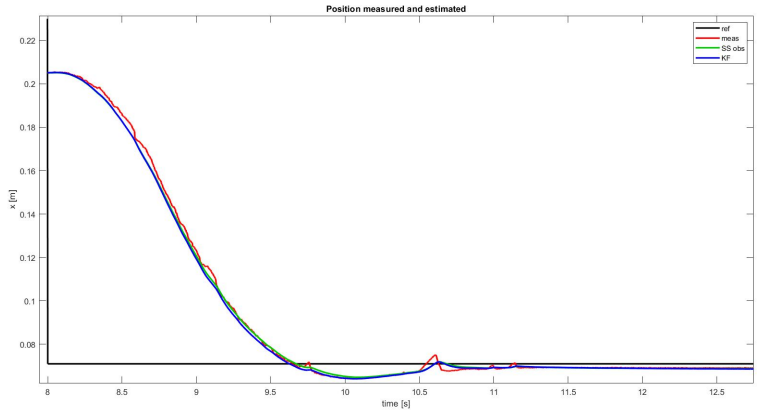


Figure 4.12: Comparison between simulated and real behaviour for LQG with matrices $Q = [170, 1, 200] \quad R = [0.1]$

4.5 Observers performances comparison

To be sure to have used a good estimation of the ball position and speed, we have performed an experiment in lab in which we have attached to the system controlled by the Phase Lead regulator the observers used with pole placement (state-space observer also using pole placements strategy) and with LQ regulators (Kalman Filter).
The developed observers show a good estimation of the state of the ball, and a smoother signal for the speed: both of the two estimations are satisfactory, and so we could rely on them to estimate system behaviour.



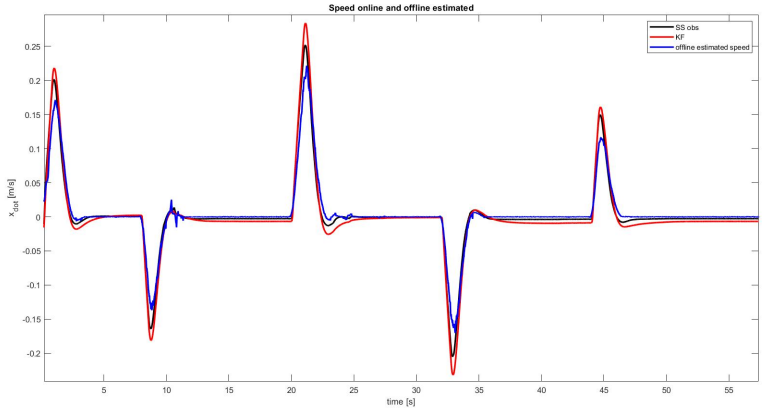


Figure 4.13: 1^{st} figure: position measured and estimated by the observers. 2^{nd} figure: speed estimated online by the observers and offline by non-causal technique.

The "offline estimated speed" plot has been obtained with a non-causal technique, based on this computation:

$$speed(i) = \frac{position(i+100) - position(i-100)}{0.002 \cdot 100 \cdot 2}$$

We chose to take the position of the 100^{th} sample before and of 100^{th} sample after the instant we was considering in order to have a trade-off between not filtering enough the signal and influencing the signal dynamics.

4.6 MPC

4.6.1 Controller development

We decide to implement an implicit Model Predictive Controller based on the State-Space representation of the linearized model 2.16, considering the same equilibrium point, in order to let the controller compute the future evolution of the controlled variables as function of the future evolution of the control inputs, solving classical finite Horizon optimal control problem. According with the dynamics of the system, $T_{rise} \approx 2s$ when we apply an angle of $\frac{\pi}{4}$ to the beam. In order to have a sufficient number of samples during the rising time, we decide to have a $\Delta T_{sample} = 0.05s$ (bigger with respect the one of ADC). This guarantees us to have a minimum number of 40 samples that are able to cover the rising time in an efficient way. For this reason, we decide to set the Predicted Horizon = 50 samples, in order to cover a total time that is bigger with respect to the rise time. Initially, we decide also to set the Control Horizon = 10 samples, in order to control only the 20% of the Predicted Horizon and not to overload system. However, once we have tested it in real apparatus, we have seen that better results were reached setting the Control Horizon equal to the Predicted Horizon, without having impact on computer perfomances. We decide to implement it in Simulink using the built-in Matlab MPC structure, in order to simplify the design phase of this advanced controller. The block already presents an internal Kalman filter used to observe non measurable states (the speed one), for this reason we were able to provide as input to the controller only the measured position and the reference relying on the KF estimation. The input sequence of MPC is computed minimizing the cost function

$$J(x(k), U(k), k) = (Y^{\circ}(k) - Y(k))'Q(Y^{\circ}(k) - Y(k)) + U'(k)RU(k)$$

under states, input, and output constraints: the constraints imposed to the manipulated variable θ and the state and output variable x of our MPC were:

$$-\frac{\pi}{4} \leq \theta \leq \frac{\pi}{4}$$

$$r_{ball} \leq x \leq L_{beam} - r_{ball}$$

This second constraint is also equal to the constraint on the position state; for what concerns the constraint on the reference signal, we choose an interval range lower than the one of the state variable, due to the fact that we would not control the ball position in the initial and final parts of the beam.

4.6.2 Controller implementation

We initially design and implement the controller in simulation and we set the weights of matrices Q and R tuning them by trial and error. From simulation, we understand that the weight of the output variable must be two order of magnitude bigger than the manipulated variable one, in order to speed up controller and obtaining a correct reference tracking. Once we tested the controller on the real system, we immediately noticed that controller was able to track steps of different amplitude along the beam in a settling time $\approx 3.5s$. However, also for this controller the behaviour is different with respect to the simulated one, as shown in the figure below (mpc_{sim} is the behaviour of mpc_1 in simulation). The voltage usage presents some steps related to the fact that our lead-lag controller action for the θ reference is very powerful with respect to the chosen ΔT_{sample}: in fact, reducing it, we obtain a cleaner signal but a slower response of the system, so we decide to accept these spikes. However, there was the presence of a high steady state error (up to 3 cm) that decreases with the increasing amplitude of the step.

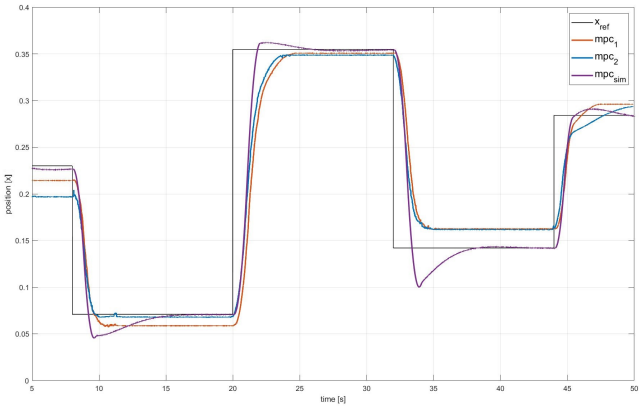


Figure 4.14: Comparison between mpc_1, mpc_2 and the simulated behaviour of mpc_1

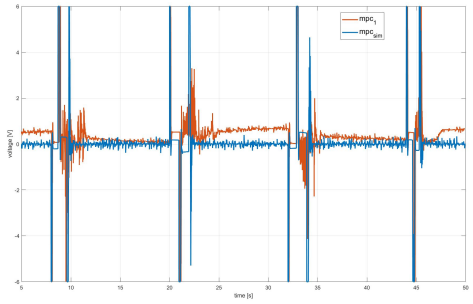


Figure 4.15: Comparison between voltage usage of mpc_1 in real system and in simulation

In order to reduce that, we have tried to improve performances of the MPC increasing the weight of the output measurement and so penalizing the manipulated variable and vice versa. We also set terminal weight and constraints for the last step in the prediction horizon, obtaining a better response without removing the steady state error. In the figure above, two of the best collected experimental data are shown, with the reference tracking of mpc_1 (Q Output Weight = 100, R Manipulated Variable Weight = 0.1, setting a terminal penalty) and mpc_2 (Q Output Weight = 50, R Manipulated Variable Weight = 0.1 without setting a terminal penalty). The presence of this steady state error is due to the fact that we consider the linearized model to develop the controller, and this does not represent at all the real system: in fact, as discussed also in previous controllers, we have not considered non linearities (as frictions,...) and so introducing modeling errors brings the computation of the steady state not to be correct, and steady state errors occurs. A possible solution to this problem would have been to add an integral action for the error in the state-space model of the plant, enlarging the system as done for other controllers. Unfortunately, due to time constraints of the laboratory sessions, we were not able to apply further modifications to this controller.

4.7 Final results comparison

At the end of all the experiments we performed the last tests to analyze how well our controllers behave in reaching different positions along the beam. As previously said, we decide to use the central part of the beam (between $\frac{L_{beam}}{7}$ and $\frac{6L_{beam}}{7}$) in order not to incur in noise problems.

To do these tests we used the best controllers (in terms of step response performances) we have developed: pole placement with integrator and LQ_∞ with Kalman Filter and integrator.

These controllers should track a reference composed by steps of different amplitude, some ramps and a low-frequency sinusoidal input (with a period of 40 s). Below you can appreciate the results:

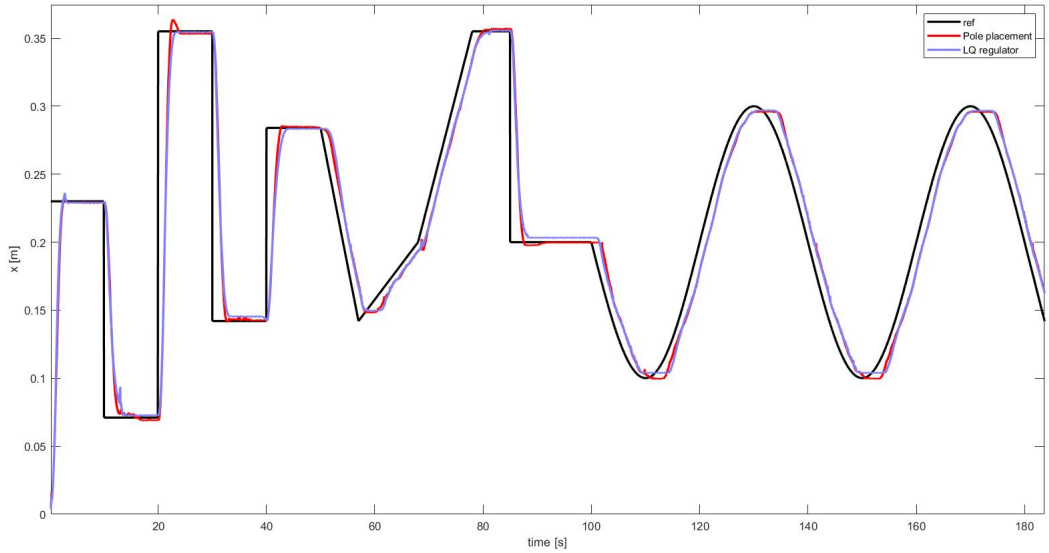


Figure 4.16: Trajectory tracking experiment performed using pole placement controller and LQ regulator

As you can notice from the figure above, the response of the system controlled by the two methods is almost the same and quite reactive and precise with both the controllers, presenting a delay of a couple of seconds in tracking the reference signal.

In general, Pole Placement controller seems to suffer from overshoot problems with steps of big amplitude, but its response to falling steps is quite better with respect to LQG, guaranteeing less steady state error. The settling time of both controllers is comparable (around $\approx 2.5s$ both for rising and falling steps); the voltage usage seems to be quite better in LQG controller, but it can be considered acceptable also for Pole Placement controller. Taking the same experiments different times, both the controllers behave always in the same good way, without presenting low repeatibility issues. We have noticed during the different tests that, probably due to a noisier measure given by the potentiometer, there are spikes (as you can appreciate in the image above near the first and second steps for LQ regulator) not always present in all the experiments.