University of Padova

Department of Information Engineering

Master degree in Automation Engineering

Laboratory Report2: Digital position control of a DC servomotor

Group 2 Tuesday shift, components: Davide Baron, Alessio Bonetto, Marco Mustacchi, Luca Vittorio Piron

Academic Year 2021-2022

1 Introduction

1.1 Activity Goal

The goal of this laboratory activity is to design a digital position controller for the DC servomotor available in the laboratory. Two different design approaches are considered: in the design by emulation, the digital controller is obtained by discretization of a controller that is originally designed in the continuous-time domain; vice versa, in the direct digital design (or discrete design), the control design is performed directly in the discrete-time domain, using an exact discrete-time model of the plant to be controlled. A digital controller has some advantages with respect to a continuous-time controller, here are some examples:

- Digital signals are very resistant to noise. Noise interference with these signals hardly causes any distorsion in the original signal.
- Devices that usually process purely in analog tend to degrade or get damaged over time and needs to be constantly checked, or calibrated while in digital control systems, only the input sensor device and actuators usually have to be calibrated, making maintenance much easier.

However, most of the plants to be controlled are continuous-time system, so it is necessary to develop some techniques for `interfacing` the discrete-time controller with the continuous-time plant. A system where both continuous and discrete-time signals coexist is called sampled-data system (Figure 1).

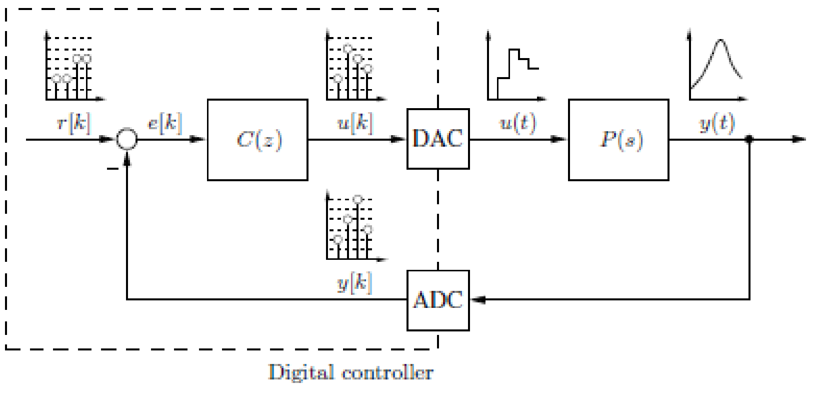


Figure 1: Sampled–data control system.

Figure 1: State space controller for nominal perfect tracking of constant set-points.

1.2 System and Model

The lumped-element diagram of the DC gearmotor is shown in Figure 2. All the symbols of the diagram are defined in the figure 2. For the numerical values consult the Appendix A.1.

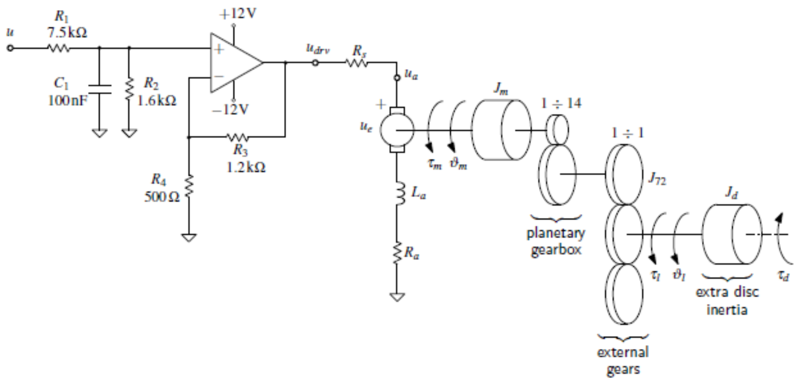


Figure 2: DC gearmotor: lumped-element diagram.

For our design process it is useful to compute the transfer function from the voltage driver input u (that will be provided by the controller through a DAC) to the load angle θ_l, which will be provided by an optical encoder attached to the central shaft of the gearbox. After some algebraic manipulations and simplifications the transfer function has been obtained:

$$P(s) \simeq \frac{1}{sN} \cdot \frac{k_t k_{drv}}{R_{eq}(J_{eq}s + B_{eq}) + k_t k_e} = \frac{1}{sN} \cdot \frac{k_m}{1 + sT_m} \tag{1}$$

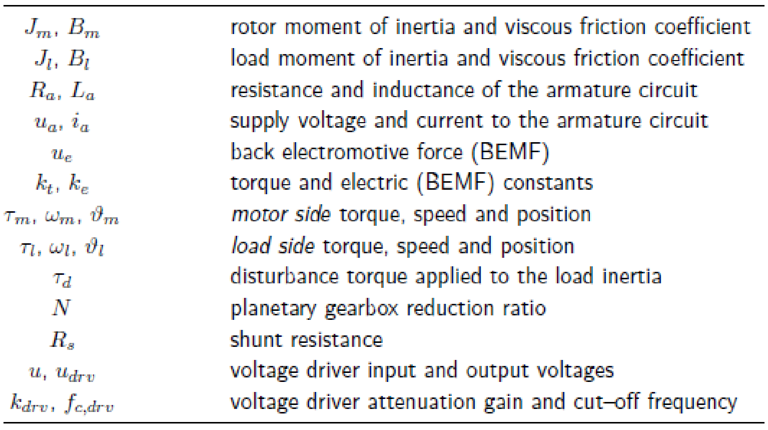
with:

$$k_m := \frac{k_{drv} k_t}{R_{eq} B_{eq} + k_t k_e} \quad T_m := \frac{R_{eq} J_{eq}}{R_{eq} B_{eq} + k_t k_e} \quad R_{eq} := R_a + R_s \tag{2}$$

The quantities:

$$B_{eq} = B_m + \frac{B_l}{N^2} = 1.2745 \cdot 10^{-6}\ Nms/rad$$
$$J_{eq} = J_m + \frac{J_l}{N^2} = 5.5567 \cdot 10^{-7}\ Nms^2/rad$$

were estimated in a previous laboratory activity together with the static friction value $\tau_{sf} =$ 0.0106Nm. Such quantities were used in the building of the accurate Simulink model (also called `nominal model` in this report) of the DC servomotor (see the Appendix A.1).



J_m, B_m	rotor moment of inertia and viscous friction coefficient
J_l, B_l	load moment of inertia and viscous friction coefficient
R_a, L_a	resistance and inductance of the armature circuit
u_a, i_a	supply voltage and current to the armature circuit
u_e	back electromotive force (BEMF)
k_t, k_e	torque and electric (BEMF) constants
τ_m, ω_m, ϑ_m	*motor side* torque, speed and position
τ_l, ω_l, ϑ_l	*load side* torque, speed and position
τ_d	disturbance torque applied to the load inertia
N	planetary gearbox reduction ratio
R_s	shunt resistance
u, u_{drv}	voltage driver input and output voltages
k_{drv}, $f_{c,drv}$	voltage driver attenuation gain and cut–off frequency

Figure 3: DC gearmotor with inertial load: symbols definitions.

2 Tasks, Methodologies and Results

In this experience, different design approaches will be tested and compared. The tests will be performed using three different sampling period: T=1ms, T=10ms and T=50ms. Since with almost all the methods tested, the closed-loop system result unstable, the figures obtained with T=50ms does not appear in order to not compromise the readability of the graphs. The omitted step responses can be checked in the Appendix A.2.

2.1 Design by emulation

This method consist in two principal steps:

- Design the continous-time controller $C_0(s)$ for the plant $P(s)$ considering the delay introduced by the ZOH, which is equal to $\frac{T}{2}$.
- Approximate $C_0(s)$ into a dicrete-time version $C(z)$.

The first step has already been done, in fact both the continous-time PID controller and the continous-time state space controller of the previous laboratory activities were designed on the DC servomotor accurate model (see the Appendix A.1), that already include a ZOH into the DAC block.
For the second step, different ways of approximately translating the continous-time controller into a discrete-time one are available. In particular the $Backward\ Euler$, $Forward\ Euler$, $Tustin$ and $Exact$ methods are kept into account. The main indea behind the design by emulation consists in approximating the relation $z = e^{sT}$ with a rational one. In the Handout 2, Table 1 you can see how each emulation method approximates such relation.
For a detailed explanation of the design by emulation method consult the Section 4.1 of the Handout 2.

2.1.1 Position PID-controller

The continou-time position PID controller for the DC servomotor, ha the following transfer function:

$$C_0(s) = K_P + \frac{K_I}{s} + K_D \frac{s}{T_L s + 1} \tag{3}$$

It was deigned using the Bode's design method, which is explained in the Section 3.2 of the Handout 0, using the plant transfer function (2) imposing M_p =10% and $t_{s,5\%}$ = 0.15s. This method return the following controller gains:

$$K_P = 8.6774 \quad K_I = 119.6293 \quad K_D = 0.1575 \quad T_L = 0.0148s \tag{4}$$

where T_L is the time constant of the real derivative block. Since these parameters do not satisfy the performance specifications, they need to be adjusted manually. The modified parameters are the following:

$$K_P = 15 \quad K_I = 15 \quad K_D = 0.22 \quad T_L = 0.0148s \tag{5}$$

Such parameters guarantee a satisfactory response to a 50 deg step reference, as you can see in the Table 1.

	Continous-time PID controller	
	Nominal model	Real motor
$t_{s,5\%}[ms]$	72	71.3
$M_p[\%]$	8	7.96

Table 1: Results obtained with the continuous-time PID controller. A green value means that such value meets the performance speci
cations.

Backward Euler discretization method

The discrete-time controller obtained with this methos is:

$$C(z) = K_P + K_I \frac{Tz}{z-1} + K_D \frac{z-1}{(T_L+T)z - T_L} \tag{6}$$

considering the values in (5). The controller block scheme is reported in figure 4 and the step respsonse is showed in figure 5.

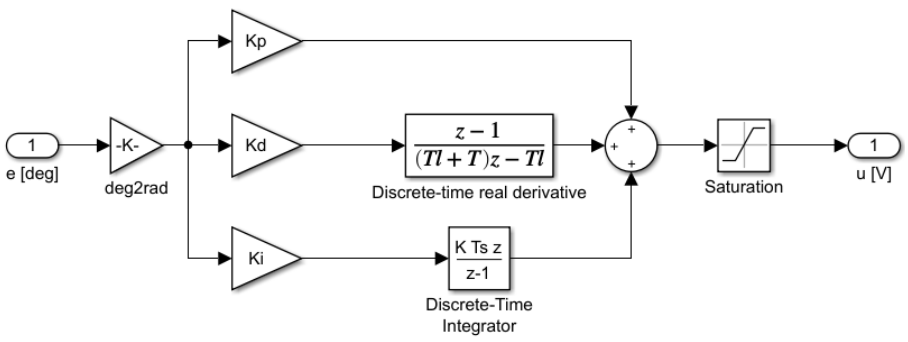
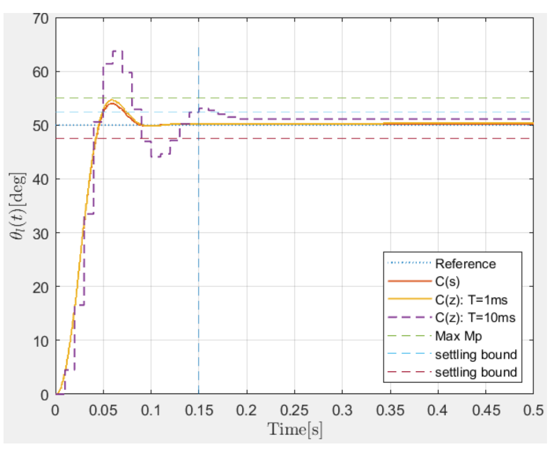


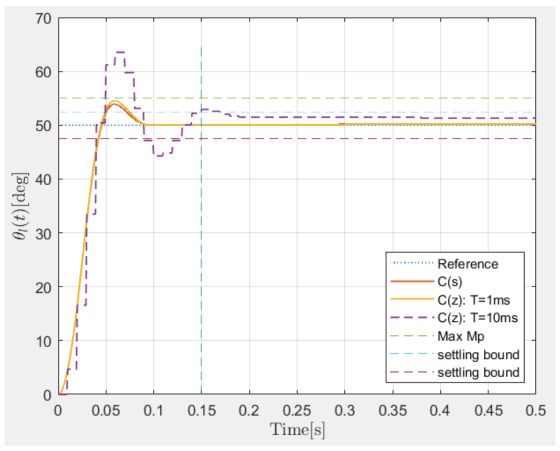
Figure 4: Discrete-time PID controller obtained with the backward Euler method.

Anti-windup

In order to reduce the overshoot in the step response, mitigating the effects of the actuator saturation, the integrator anti-windup mechanism has been implemented, for more details consult the Section 2.2 of the Handout 1. The test has been performed with 360deg step reference and a sampling period T=10ms. In the figure 6 you can see the new controller structure.



(a) Nominal model



(b) Real motor

Figure 5: Response of the closed-loop system to a 50 deg step, with a PID controller emulated with backward Euler and sampling period T = 1, 10 ms.

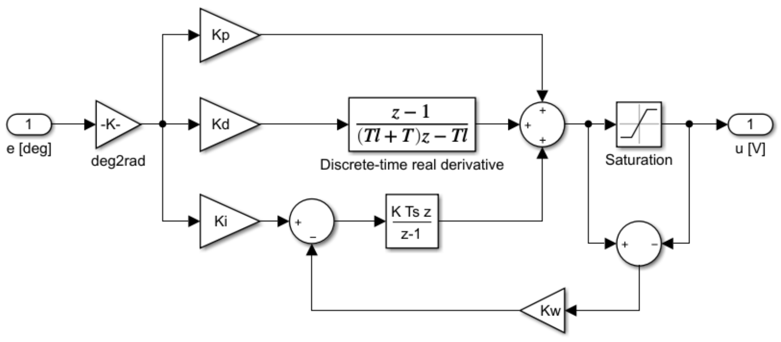
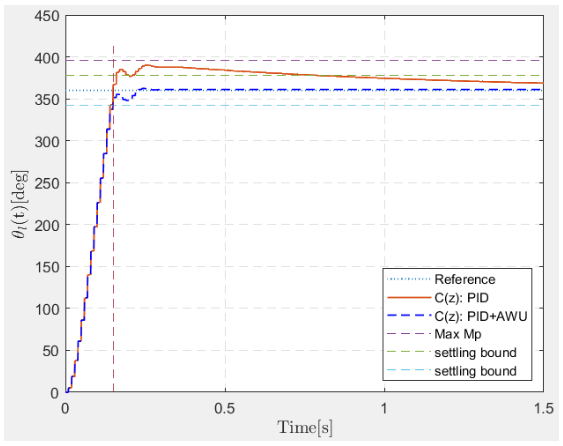
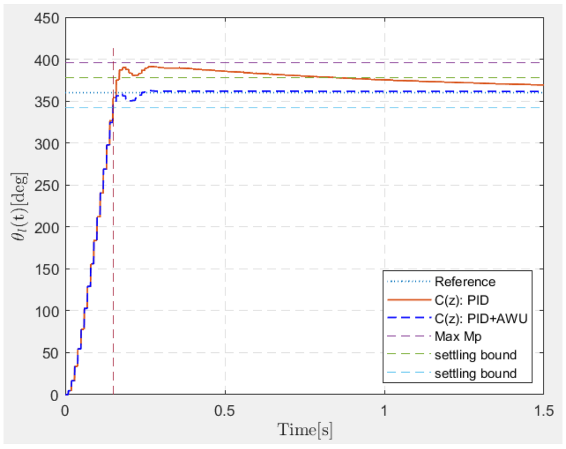


Figure 6: Discrete-time PID controller with the anti-windup mechanism obtained with the backward Euler discretization method.

The anti-windup gain has been found by a trial-and-error approach and a value that leads good results is $K_W = 1.1$. In the figure 7 and in the Table 2 the effectiveness of the anti-windup mechanism is shown.



(a) Nominal model



(b) Real motor

Figure 7: Response of the closed-loop system to a 360 deg step with and without the anti-windup mechanism applied on the backward Euler PID controller (the vertical line is the settling-time limit).

	Without anti-windup		With anti-windup	
	Nominal	Real	Nominal	Real
$t_{s,5\%}[ms]$	780	840	149.3	149.9
$M_p[\%]$	8.4	8.65	0.65	0.7

Table 2: Results obtained by implementing the anti-windup mechanism on the backward Euler PID controller with T=10 ms. A green value means that such value meets the performance specifications.

Feed-forward

A method used to reduce the controller's effort is to use a $feedforward$ compensation. This means use the knowledges of the plant, or the system to control, to design a nominal control that makes the output similar, even not exactly close, to the reference.
Now the feedback is used to compensate errors in the modelling procedure, or to increment the robustness of the control system.
The feed-forward scheme is showed in the figure 3 of the Section 2.1 of the Handout 1, allows a compensation for the inertia, static and viscous friction and back-electromotive force.
The gains of the feed-forward circuit are reported in the following:

$$K_{IN} = 0.053, \quad K_{FR} = 48.2217, \quad K_{BEMF} = 0.1798 \tag{7}$$

For the test, a periodic acceleration reference signal with main period defined as follows, has been considered:

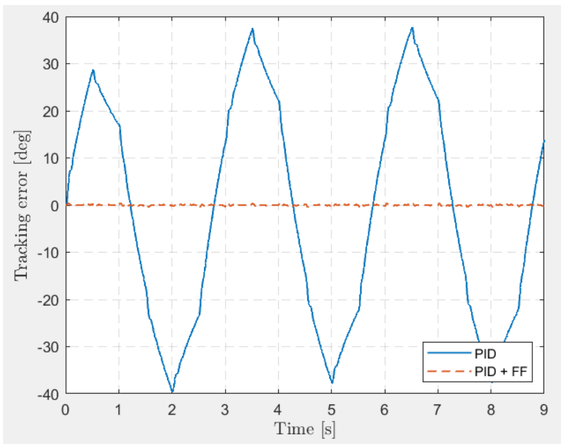
$$a_l^*(t) = \frac{d\omega_l^*(t)}{dt} = \begin{cases} 900rpm/s & \text{if 0s} \leqslant \text{t} < \text{0.5s} \\ 0rpm/s & \text{if 0.5s} \leqslant \text{t} < \text{1s} \\ -900rpm/s & \text{if 1s} \leqslant \text{t} < \text{2s} \\ 0rpm/s & \text{if 2s} \leqslant \text{t} < \text{2.5s} \\ -900rpm/s & \text{if 2.5s} \leqslant \text{t} < \text{3s} \end{cases} \tag{8}$$

which defines a trapezoidal speed profile. The speed and position references are obtained by integration of the acceleration profile, i.e.:

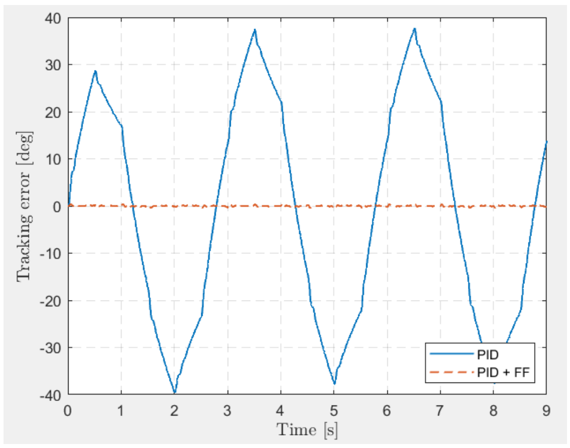
$$\omega_l^*(t) = \int_0^t a_l^*(\tau)d\tau \qquad \theta_l^*(t) = \int_0^t \omega_l^*(\tau)d\tau \tag{9}$$

The advantages brought by the direct chain compensation are clearly noticeable in the figure 8, where you can see that without feed-forward the absolute value of the tracking error exceeds 39deg, while with the feed-forward you have almost perfect tracking of the reference, with an absolute value of the error that does not exceed 0.36 deg.

Forward Euler discretization method



(a) Nominal model

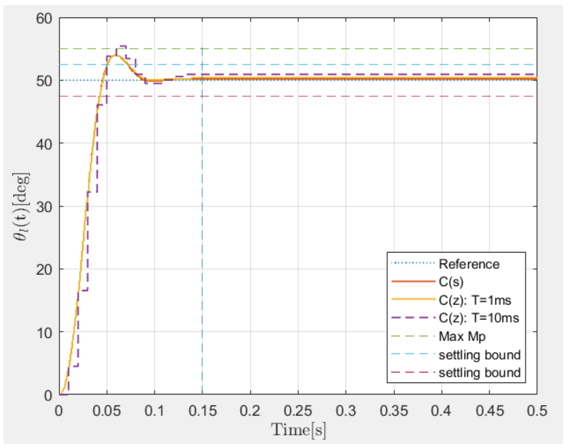


(b) Real motor

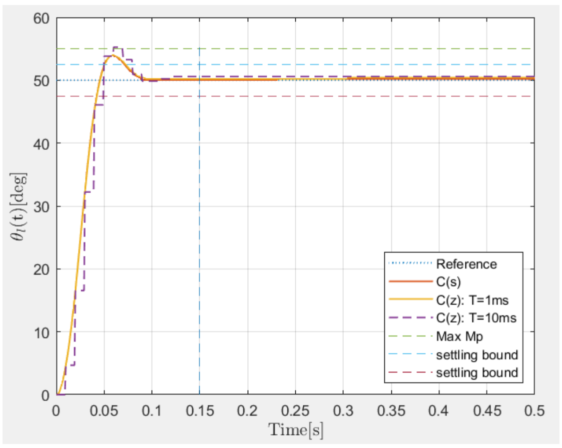
Figure 8: Tracking error with and without the feed-forward mechanism on the backward Euler PID controller with T=10 ms.

The controller structure is the same that the one in figure 4, while the transfer function of the discrete -time controller becomes:

$$C(z) = K_P + K_I \frac{T}{z-1} + K_D \frac{z-1}{T_L z + T - T_L)} \tag{10}$$



(a) Nominal model



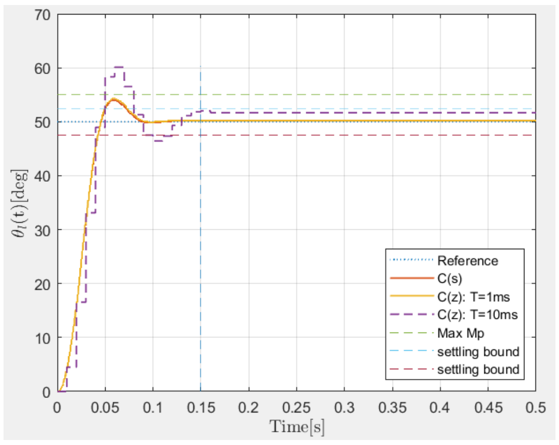
(b) Real motor

Figure 9: Response of the closed-loop system to a 50 deg step, with a PID controller emulated with forward Euler and sampling period T = 1, 10 ms.

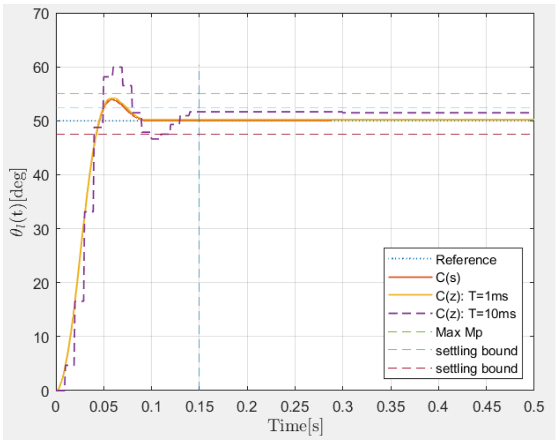
Tustin's discretization method

As in the previous approximation methods, the controller structure is the same that the one in figure 4, while the transfer function of the discrete -time controller becomes:

$$C(z) = K_P + K_I \frac{T(z+1)}{2(z-1)} + K_D \frac{z-1}{(T_L + T/2)z + T/2 - T_L)} \tag{11}$$



(a) Nominal model

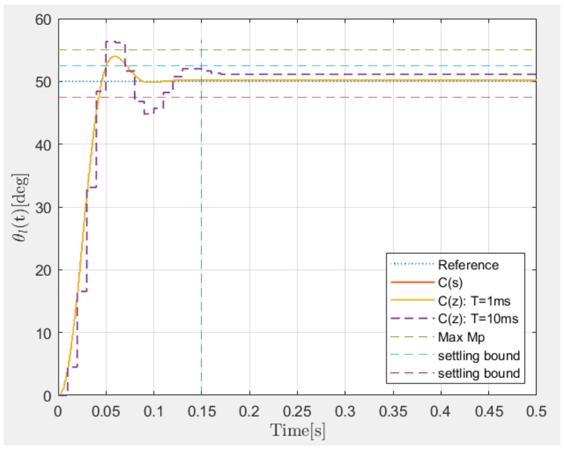


(b) Real motor

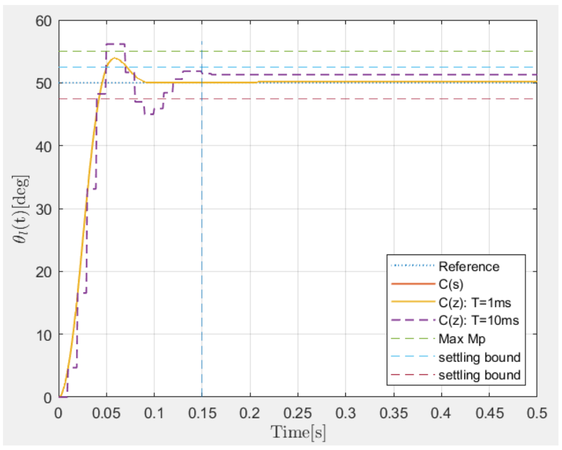
Figure 10: Response of the closed-loop system to a 50 deg step, with a PID controller emulated with Tustin and sampling period T = 1, 10 ms.

Exact discretization method

In this case, the controller transfer function was obtained from the continuous-time one using the c2d routine of the Control System Toolbox. The controller structure is the same used in the previous discretization methods.



(a) Nominal model



(b) Real motor

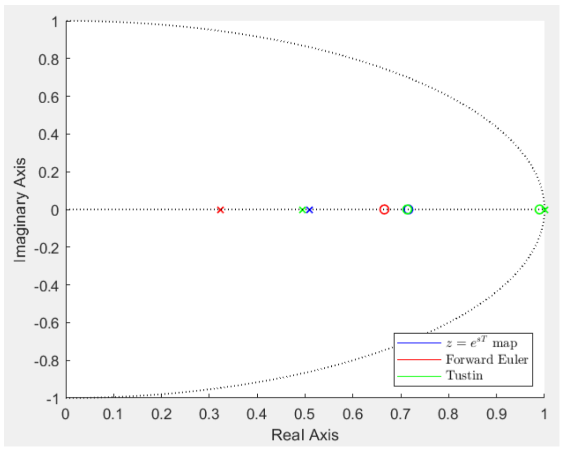
Figure 11: Response of the closed-loop system to a 50 deg step, with a PID controller obtained with exact discretization and sampling period T = 1, 10 ms.

PID design by emulation - Results and observations

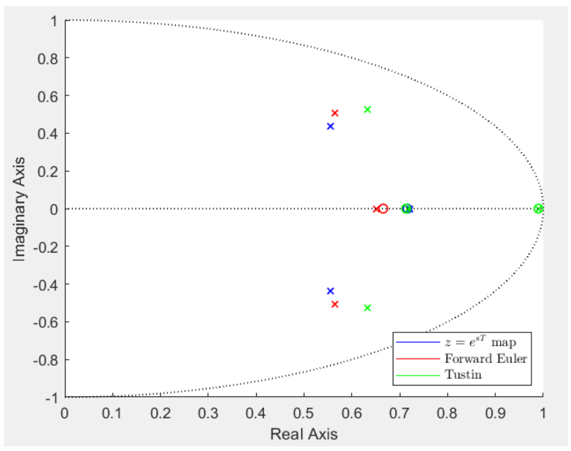
First of all it is important to observe that, with T=1ms, all the emulation methods allow to approximate the continous-time controller almost perfectly. This is what expected from the theory when the sampling period is chosen small enough. In Table 3, the numerical values relative to the performances obtained with all methods are reported.

It is interesting to notice, instead, the behavior of the emulation methods with T=50ms. In fact, the forward Euler method is the only one that almost meets the performance specification, performing even better than the Tustin'ss controller, which from the theory should guarantees the best poles/zeros mapping in the controller discretization, among the various proposed

methods.
In fact this is true, as shown in figure 12a it is possible to observe how the Tustin's method approximates better the relation $z = e^{sT}$, with respect to the forward Euler method. The problem stands in the fact that the forward Euler method leads to a better dominant poles mapping of the closed-loop system, leading to better performance with respect to the Tustin's method. This fact denotes a big drawback of the emulation method: the impossibility of exactly allocating the discrete-time closed-loop system eigenvalues.



(a) Discrete-time controller poles/zeros



(b) Discrete-time closed-loop system poles/zeros

Figure 12: Pole-zero map comparison with T = 10 ms.

	BE		FE		Tustin		Exact	
	Nom.	Real	Nom.	Real	Nom.	Real	Nom.	Real
	T = 1ms							
$t_{s,5\%}[ms]$	74	74.3	73	72.3	73	73.3	72	71.3
$M_p[\%]$	9.44	9.08	8.1	7.9	8.72	8.36	8	8
	T = 10ms							
$t_{s,5\%}[ms]$	169.9	169.1	79.9	79.3	119.9	110	110	109.6
$M_p[\%]$	27.44	27.08	10.88	10.52	20.24	19.88	12.68	12.32

Table 3: Design by emulation - PID results. A green value means that such value meets the performance specifications.

2.1.2 Position state-space controller

The aim of this assigment is to design a state-space controller that guarantees the perfect tracking of constant position set–points in the nominal case.
The continuous-time position state-space controller in figure 13 was obtained by using the pole allocation method explained in the Section 3.2 of the Handout 1, using the state-space reachable and observable realization of the DC gearmotor transfer function (2):

$$A = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{1}{T_m} \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ \frac{k_m}{NT_m} \end{bmatrix} \qquad C = \begin{bmatrix} 1 & 0 \end{bmatrix} \qquad D = 0 \tag{12}$$

In which the state is composed by the angular dispacement of the external gearbox and its rotation's speed: $x = [\theta_l \;\, \dot{\theta}_l]^T$.

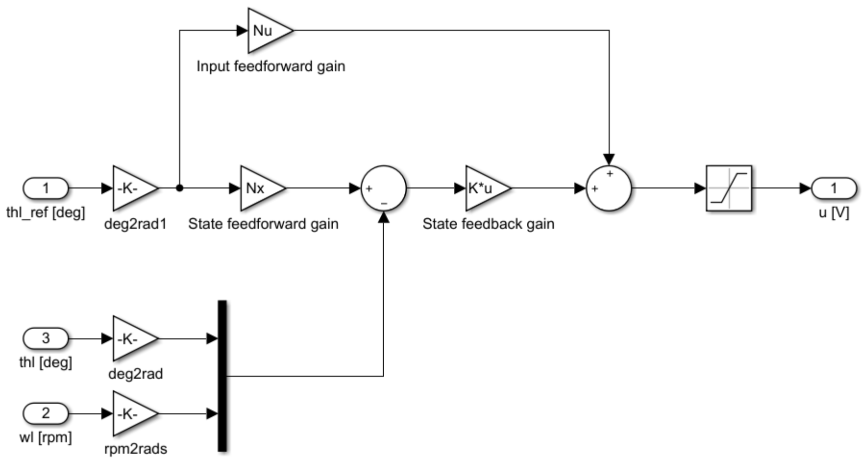


Figure 13: State space controller for nominal perfect tracking of constant set-points.

The state and input feedforward gains are obtained by following the Handout procedure:

$$\begin{bmatrix} Nx \\ Nu \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix}^{-1} \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ \hline 0 \end{bmatrix} \tag{13}$$

Then, the control law becomes:

$$u = N_u r - K(x - N_x r) = -Kx + (N_u + KN_x)r \tag{14}$$

The assignment requires the following transient performances:

- nominal perfecct steady state tracking of step position (load side) references
- step response (load side) with settling time $t_{s,5\%} \leqslant 0.15s$ and overshoot $M_p \leqslant 10\%$

Resorting to the dominant pole approximation approach explained in Sec 3.1 of Handout 0, the eigenvalues of A-BK should be placed in:

$$\lambda_{1,2} = -\delta\omega_n \pm j\omega_n\sqrt{1-\delta^2} = -20 \pm j27.2875 \tag{15}$$

Where δ and ω_n are obtained by solving the following equations:

$$\delta = \frac{log(1/M_p)}{\sqrt{\pi^2 + log^2(1/M_p)}} \qquad \omega_n = \frac{3}{\delta t_{s,5\%}} \tag{16}$$

This dominant poles' allocation correspond to the state feedback gain K = [6.0907 0.0212]. Since the eigevalues' allocation's choiche does not provide a satisfactory step response, because

of static friction, it was necessary to increase $\omega_n := |\lambda_{1,2}|$ from $\frac{3}{\delta t_{s,5\%}}$ to $\frac{4}{\delta t_{s,5\%}}$, where δ is defined in (16), in order to compensate the static friction.
The ne eigenvalues are:

$$\lambda_{1,2} = -26.667 \pm j36.383 \tag{17}$$

which imply the state feedback gain K = [19.7503   0.0907].
In figure 14 it is possible to see the impovements introduced by this change.

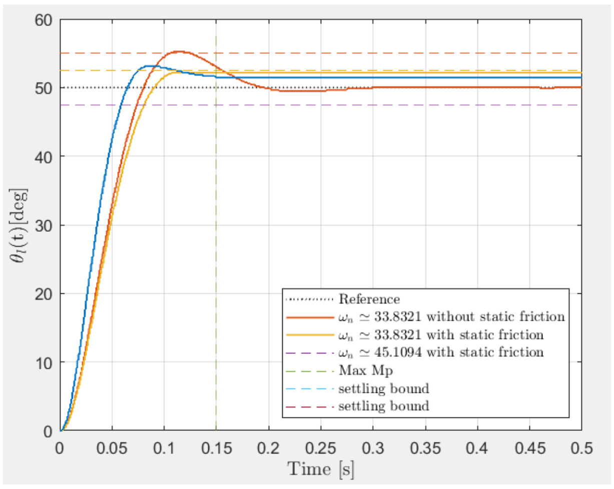


Figure 14: Static friction compensation by increasing eigenvalues frequency.

However, also increasing ω_n the static friction introduce a non-zero steady-state error. As known from the theory, only the model without static friction reaches the reference without steady-state error.

Continous-time reduced-order state observer

As said before, the plant state is represented by the couple $x = [\theta_l \ \dot{\theta}_l]^T$, but the two variables are not bot directly accessible. In order to get an estimate of all the unmeasurable state variables, and hence to allow a state feedback, a state observer has been introduced. Since the measure of the angular displacement, θ_l is already provided by the encoder, the only state variable that has to be estimated is ω_l, then is convenient to use a reduced-order observer.
Following the Section 5.1 of the Handout 2, I designed a continuous-time reduced-order state observer1, which is a dynamical system with the state-space representation reported in the Section 2.1 (equation (7) and (8)) of the Assignment 2. The continuous-time position state-space control loop is shown in figure 15.
Defining $\hat{x}$ the estimated state at the output of the observer, the control law becomes:

$$u = N_u r - K(\hat{x} - N_x r) = -K\hat{x} + (N_u + KN_x)r \tag{18}$$

In order to guarantee tath the observer dynamics is much faster than the closed loop dynamics, the obsver eigenvalues are placed such that it is 3 times faster than the closed loop ones, λ_o=-$3\omega_n$=-135.3284, which implies L=99.1063. The main idea is that the estimation error

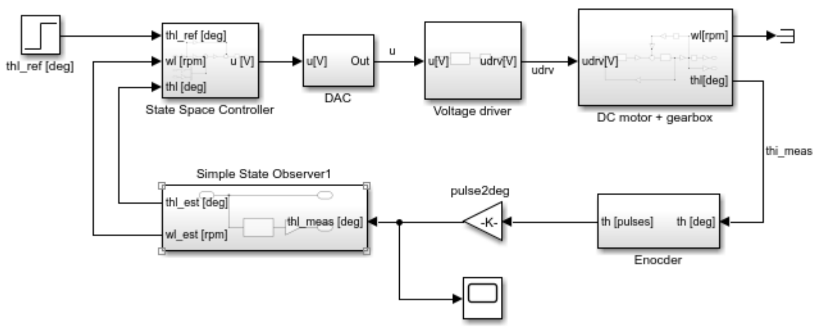
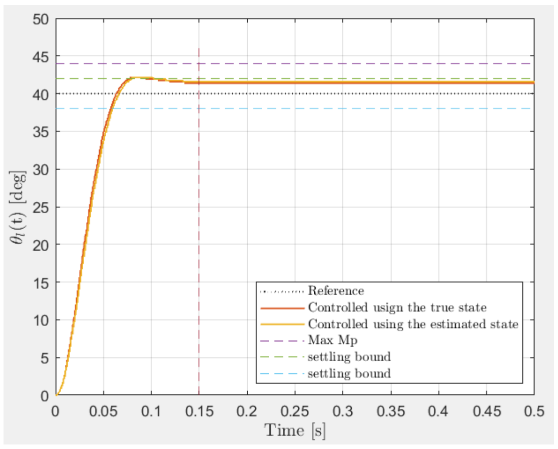
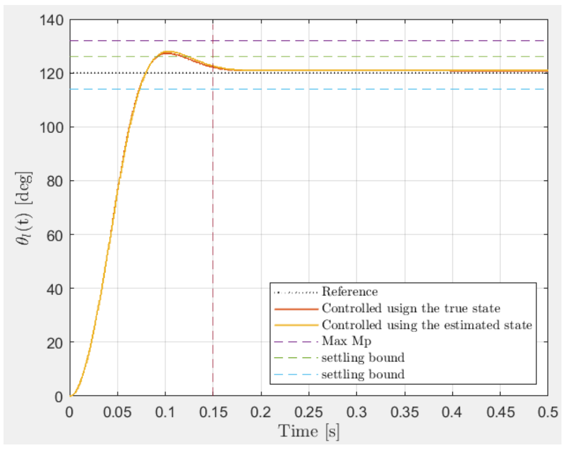


Figure 15: Position state-space control loop.

converges quickly to zero, then the controller is very similar to the one obtained using the true state, as can verify in the figure 16.



(a) 40 deg step reference.



(b) 40 deg step reference.

Figure 16: Continuous-time reduced-order state observer performance. You can see that the results obtained by feeding the controller with the estimated state are very close to the results obtained by feeding the controller with the true state.

Integral action

In order to guarantee perfect ssteady state robust tracking of constant position reference and perfect rejection of contant load disturbances, like the static friction, an integral action has been introduced to the state-space controller in figure 13 .
The new state vector becomes $x_e = [x_I \;\; x]^T$, where x is the state vector of the previous assignment and x_I is the new added component.
Referring to the control law (18) the integral action can be included as follows:

$$u = -Kx + (N_u + KN_x)r - K_I \int_0^t [y(\tau) - r(\tau)]d\tau \tag{19}$$

which decribes the block scheme in figure 17.
in which the integral component try to compensate the in some way the constant friction of the model or the modelling errors.
In order to implement the integral action I followed the extended state-space pole allocation

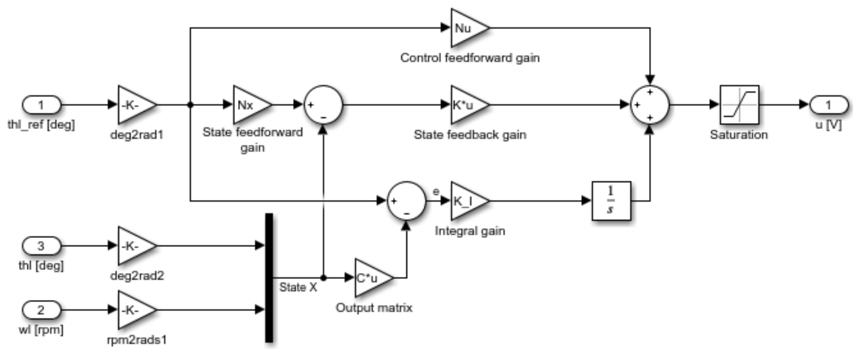


Figure 17: Continous-time state-space controller for nominal perfect tracking of constant set-points.

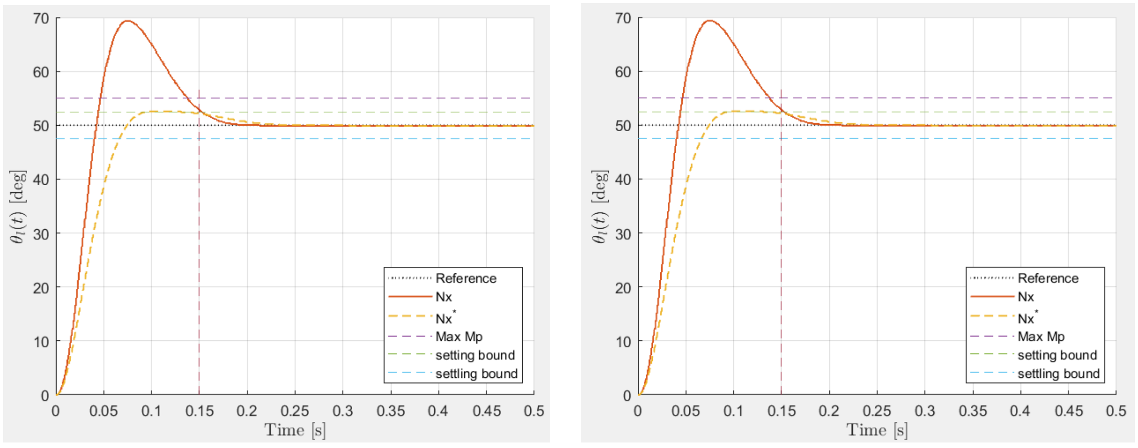
method explained in the Handout 1, placing the eigenvalues at:

$$\lambda_{1,2} = -26.667 \pm \jmath 36.383 \quad \lambda_3 = -26.667 \tag{20}$$

The integral gain and the state-feedback matrix obtained by the acker routine of the CST are:

$$K = [18.2641 \;\; 0.2315] \quad K_I = 286.6815 \tag{21}$$

The feedforward matrixes N_x and N_u calculated in the previous point should remain unchanged, but even so, you would have a very big overshoot that would ruin the performance specifications. The reason behind that behavior has to be researched in the position of the zeros in the closed loop transfer function, on which you do not have lot of control. A good solution is to decrease the value of N_x in order to decrease the overshoot's amplitude.
In figure 18, it is possible to observe a comparison between the 50degs step response with the original N_x and the adjusted one N_x^*=$N_x/2$, that has been choosen in order to satify the spacifications on M_p.



(a) Nominal model (b) Real motor

Figure 18: Response to 50deg step with N_x and N_x^*

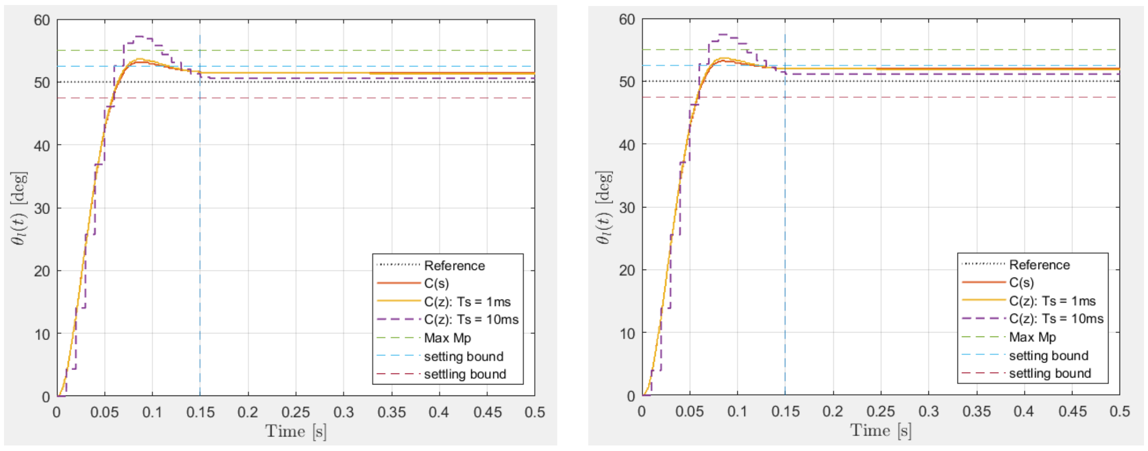
Note that reducing (modifying in general) N_x is not possible without the integral action, because in such a case only one value of N_u and N_x allows a perfect nominal tracking of step references.

	Continous-time state-space controller			
	Nominal tracking		Robust tracking	
	Nominal	Real	Nominal	Real
$t_{s,5\%}[ms]$	112	117	125	132
$M_p[\%]$	6.2	6.56	5.84	5.48
$e_{ss}[deg][\%]$	1.48	2.02	0	0

Table 4: Results obtained with the continuous-time state-space controller. A green value means that such value meets the performance specifications.

Forward Euler discretization

The discretization of the regulator simply consists of the discrete-time observer, combined, in case of the robust regulator, with the discretization of the integrator. For more details about the different discretization methods refer to the tables 1 (for the control law) and 3 (for the state-observer) of the Handout 2.



(a) Nominal model (b) Real motor

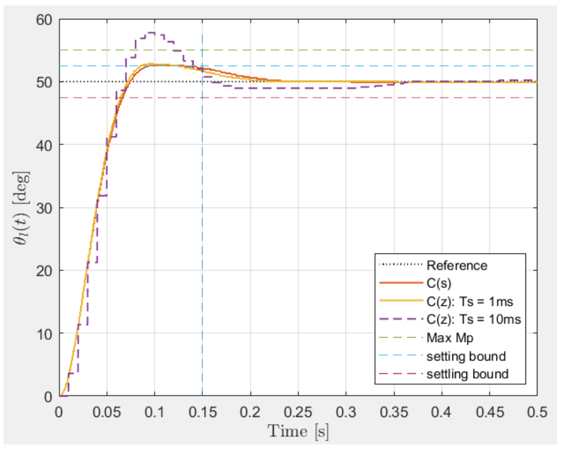
Figure 19: Forward Euler discretization, nominal tracking - T = 1 ms, 10 ms.

Backward Euler discretization

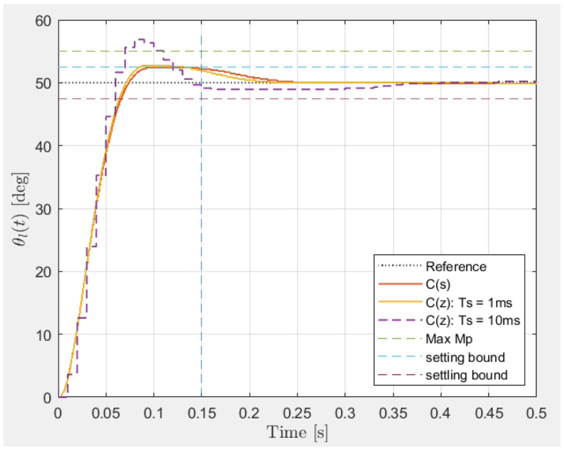
Always following the tables 1 and 3 of the Handout 2, the discrete-time state-space regulator has been obtained. Among all the design by emulation methods seen so far, the backward Euler method is one of the two methods that leads to the stability of the closed-loop system with T = 50 ms, but only without the integral action. With the integral action the response becomes in fact unstable, with large oscillations (see the Appendix A.2).

Tustin's discretization

Using the c2d routine of the CST for the state observer discretization and the Table 1 of the

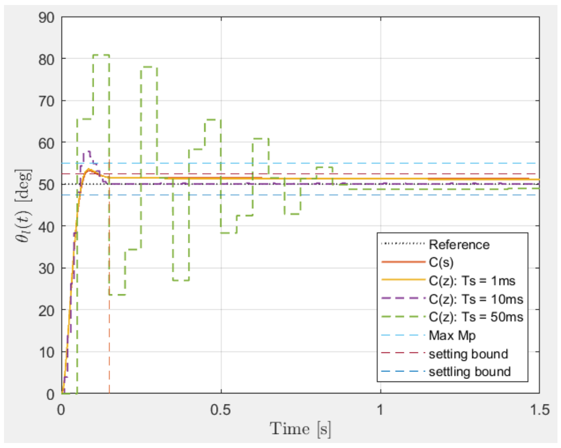


(a) Nominal model

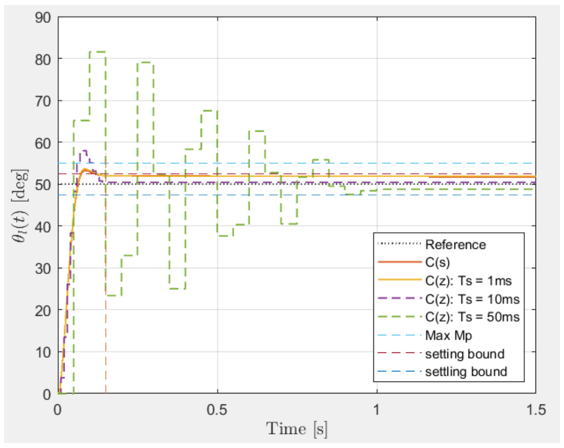


(b) Real motor

Figure 20: Forward Euler discretization, robust tracking - T = 1 ms, 10 ms.

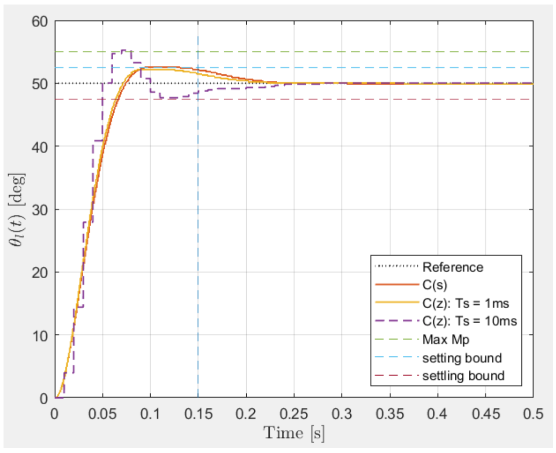


(a) Nominal model

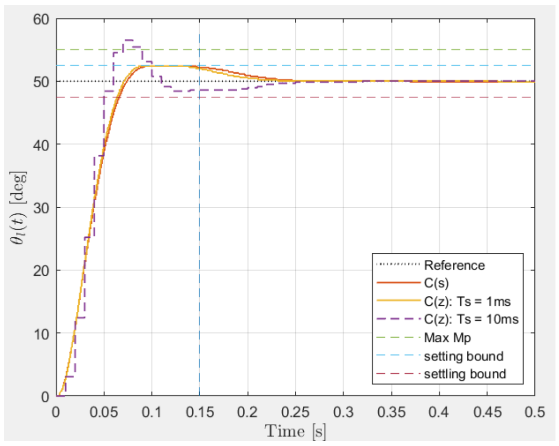


(b) Real motor

Figure 21: Backward Euler discretization, nominal tracking - T = 1 ms, 10 ms, 50 ms.



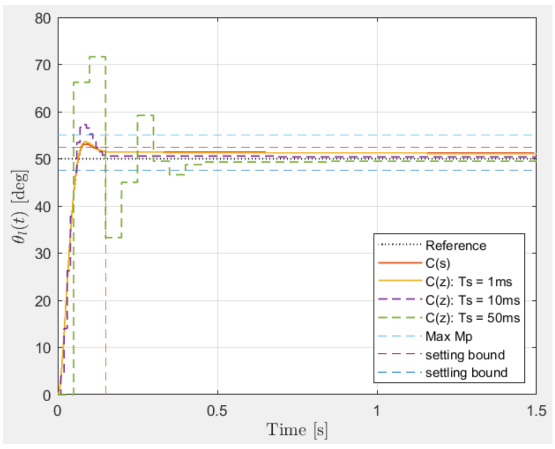
(a) Nominal model



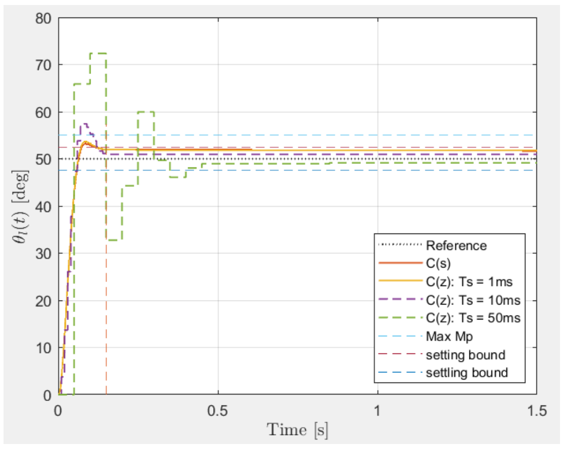
(b) Real motor

Figure 22: Backward Euler discretization, robust tracking - T = 1 ms, 10 ms.

Handout 2 for the control law discretization the discrete-time regulator has been obtained with the Tustin's discretization method. Also with this method closed-loop stability with T = 50 ms ha been obtained, but only without the integral action, like in the backward Euler method.

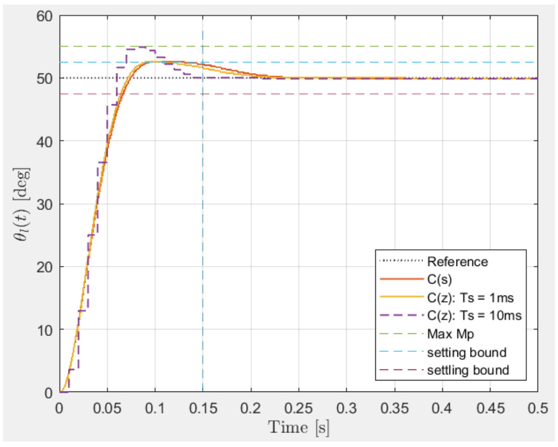


(a) Nominal model

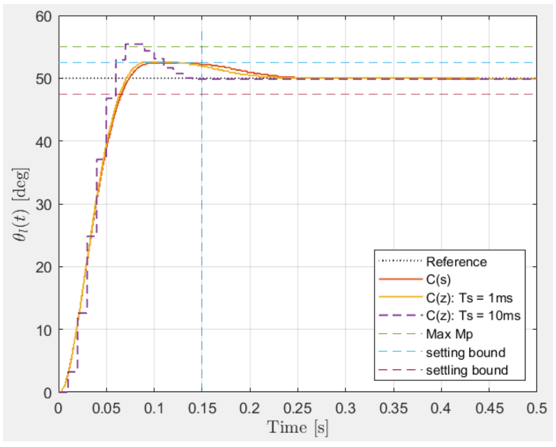


(b) Real motor

Figure 23: Tustin's discretization, nominal tracking - T = 1 ms, 10 ms, 50 ms.



(a) Nominal model



(b) Real motor

Figure 24: Tustin's discretization, robust tracking - T = 1 ms, 10 ms.

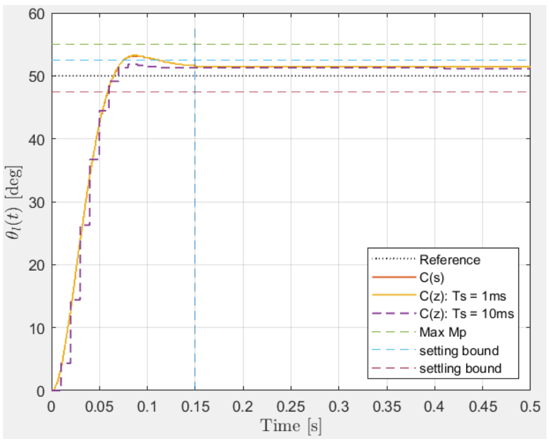
Exact discretization

For both the discretizations, the observer and the integrator, the c2d routine of the Control System Toolbox (for the integrator discretization I applied the c2d to the transfer function s^{-1}) has been used.

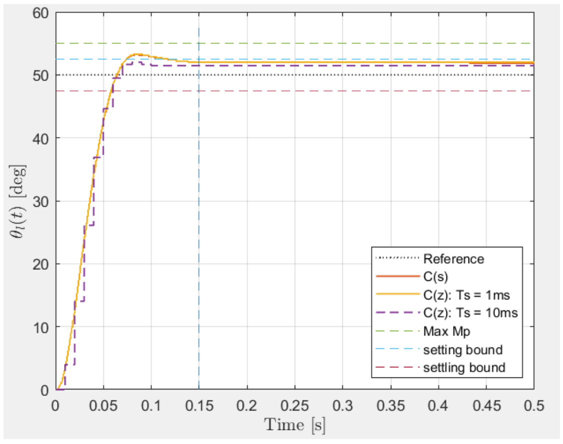
State-space design by emulation - Results and observations

As expected from the theory the only regulators which reach the zero steady state errror are those that are equipped with the integral controller, this is due to the static friction which is a constant load disturbance. So, the discussion below reguard only the integral controllers' performances. As in the PID case, with T=1ms all methods are acceptable and approximate well the continous-time controller. With T=10ms the best performances are reached by the discrete-time controller obtained by the Tustin's method, this is expected from the theory because this methodis the only one that maps the left-half s-plane into the entiere z-plane unit disk, so it should ensure the best emulation.

An interesting fact that confirm the instability in the nominal tracking case, with T=50ms,

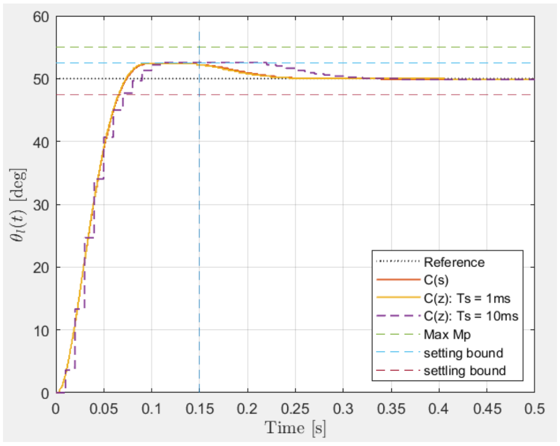


(a) Nominal model

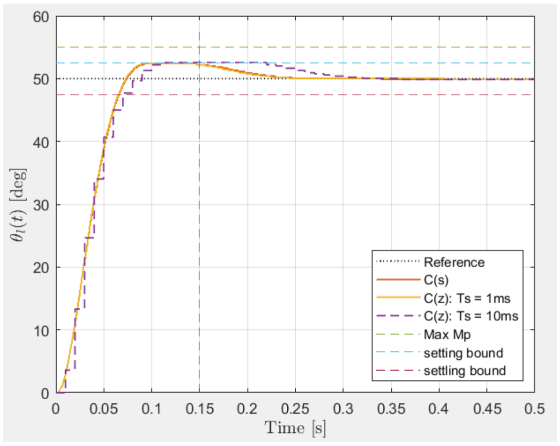


(b) Real motor

Figure 25: Exact discretization, nominal tracking - T = 1 ms, 10 ms.



(a) Nominal model



(b) Real motor

Figure 26: Exact discretization, robust tracking - T = 1 ms, 10 ms.

using the forward Euler dicretization, is the fact that the discrete-time observer eigenvalue has absolute value $\simeq$5.77, leading to an unstable closed-loop system.With backward Euler and Tustin this can not happen because both methods map stable poles inside the unit disk. However, even with the latter methods, instability of the closed-loop system may still occur as you saw in the PID case; in fact, also in this case, with the robust tracking controller and T = 50 ms none of the emulation methods lead to a stable closed-loop system. This fact will be better discussed at the end of this report.

2.2 Direct digital design

This method is divided in two major steps:

- Translate the continous-time model of the plant to be controlled into a dicrete-time one through the exact discretization method. For this step teh c2d routin of the Control System Toolbox has been used.
- Design, in the discrete-time domain, the discrete-time controller/regulator for the discrete-time model obtained in the previous point;

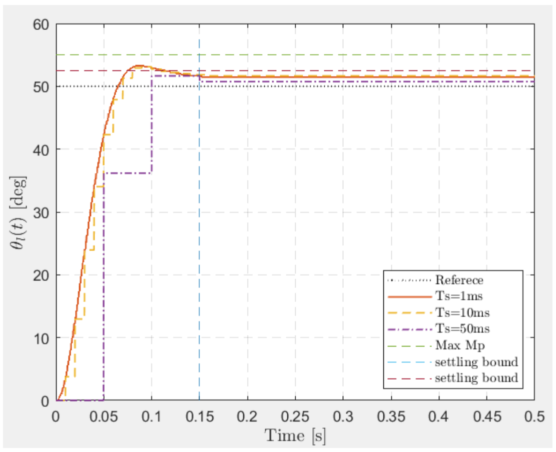
		FE		BE		Tustin		Exact	
		Nom.	Real	Nom.	Real	Nom.	Real	Nom.	Real
		T = 1ms							
Nominal	$t_{s,5\%}[ms]$	117.9	122.9	116	122	117	120.9	112	117.9
	$M_p[\%]$	7.28	7.28	7.28	7.28	7.28	7.28	6.56	6.56
	$e_{ss}[deg]$	1.3	1.84	1.3	1.84	1.12	1.84	1.48	2.02
Robust	$t_{s,5\%}[ms]$	125	132	64	63.9	106.9	124.9	129	65
	$M_p[\%]$	5.84	5.48	4.4	4.76	5.12	5.12	5.12	4.76
	$e_{ss}[deg]$	0	0	0	0	0	0	0	0
		T = 10ms							
Nominal	$t_{s,5\%}[ms]$	130	130	120	119.9	120	129.9	60	59.9
	$M_p[\%]$	14.48	14.84	15.56	15.92	14.48	14.84	3.68	4.04
	$e_{ss}[deg]$	0.58	1.12	0.04	0.4	0.4	0.94	1.12	1.48
Robust	$t_{s,5\%}[ms]$	140	120	89.9	99.9	110	110	219.9	210
	$M_p[\%]$	15.56	13.76	10.52	13.04	9.8	10.88	5.44	5.12
	$e_{ss}[deg]$	0	0	0	0	0	0	0	0
		T = 50ms							
Nominal	$t_{s,5\%}[ms]$	-	-	850	850	400	399.9	-	-
	$M_p[\%]$	-	-	61.64	63.08	43.28	44.72	-	-
	$e_{ss}[deg]$	-	-	1.04	1.22	0.5	0.86	-	-
Robust	$t_{s,5\%}[ms]$	-	-	-	-	-	-	-	-
	$M_p[\%]$	-	-	-	-	-	-	-	-
	$e_{ss}[deg]$	-	-	-	-	-	-	-	-

Table 5: Design by emulation - state-space results. A green value means that such value meets the performance specifications.

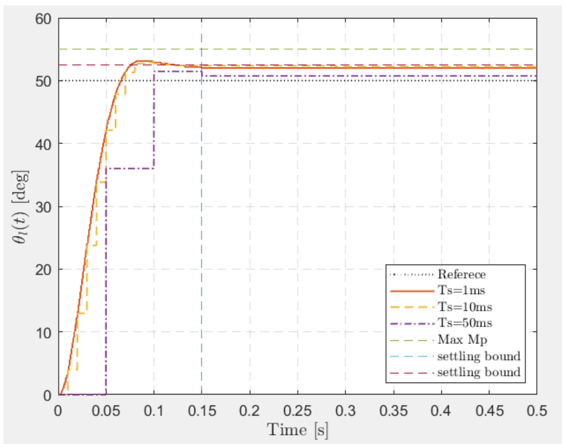
For the design of the discrete-time reduced-order state observer I followed the Section 5.2 of the Handout 2, while for the implementation of the integral action on this controller I followed the Point 4 of the Section 2.2 of the Assignment 2.
In order to obtain the reach the same performance specification of the continous-time controller, all the eigenvalues have been obtained frome the continous-time case eigenvalues by using the relation $z = e^{sT}$.
The feed-forward gains Nu and Nx were computed using the equation (19) of the Assignment 2, and they result unchanged with respect to the design by emulation part, for every choice of the sampling period. As for the state-space emulation methods, for the robust controller Nx has been multiplied by 0.5 in order to reduce the overshoots' amplitude.

2.3 Final observation and conclusions

This laboratory experience showed the fundamental role played by the sampling period in a sampled data control system, especially in the design by emulation methods, in which you saw that with big values of T you can occur in the instability of the closed-loop system even with discretization methods that guarantee the controller stability preservation (in the figure 29, this fact is observable). The reason is the impossibility of getting an exact discrete-time equivalent of the continuous-time controller/regulator: emulations methods work by approximating the

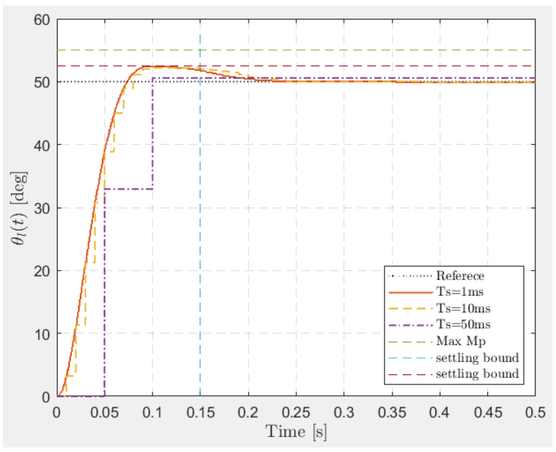


(a) Nominal model

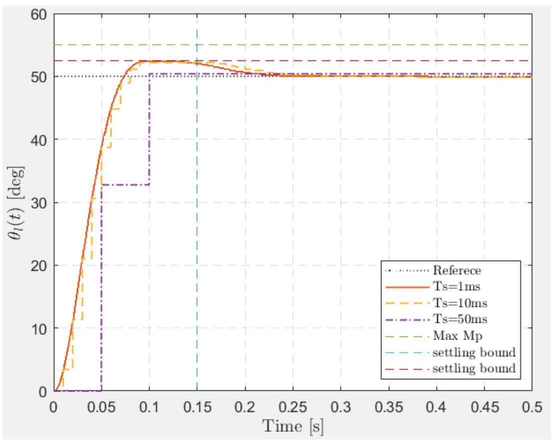


(b) Real motor

Figure 27: Direct digital design, nominal tracking - T = 1 ms; 10 ms; 50 ms.



(a) Nominal model



(b) Real motor

Figure 28: Direct digital design, robust tracking - T = 1 ms; 10 ms; 50 ms.

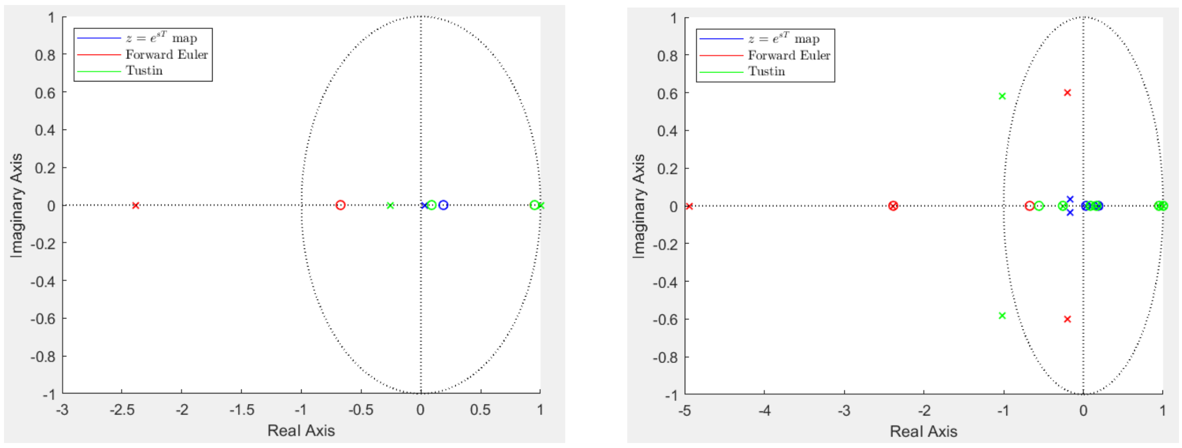
	Nominal tracking		Robust tracking	
	Nom.	Real	Nom.	Real
	T = 1ms			
$t_{s,5\%}[ms]$	114	121.9	65	65.9
$M_p[\%]$	6.56	6.2	4.76	4.76
$e_{ss}[deg]$	1.48	2.02	0	0
	T = 10ms			
$t_{s,5\%}[ms]$	110	120	70	69.9
$M_p[\%]$	6.2	5.84	4.4	4.4
$e_{ss}[deg]$	1.66	2.2	0	0
	T = 50ms			
$t_{s,5\%}[ms]$	100	99.9	100	100
$M_p[\%]$	3.32	2.96	1.16	0.8
$e_{ss}[deg]$	0.76	0.76	0	0

Table 6: Results obtained with the direct digital design. A green value means that such value meets the performance speci
cations.

relation $z = e^{sT}$ with a rational one, and such approximation becomesworse as T increases. That error in the emulation of the controller affects inevitably the placement accuracy of the closed-loop poles/eigenvalues, and none of the emulation methods tested in this experience can ensure stability of such poles/eigenvalues. That is why you may occur in the instability of the closed-loop system when T is big. But `big` in what sense? For a good emulation, said tr the rise-time of the closed-loop system, the rule of thumb is:

$$\frac{t_r}{1000} < T < \frac{t_r}{20}$$

which in our case, assuming $t_r = t_{s;5\%}$, says that you should not exceed T = 7.5 ms. Thus, it is not surprising that with T = 50 ms emulation methods work really bad.



(a) Discrete-time controller poles/zeros (b) Discrete-time closed-loop system poles/zeros

Figure 29: Pole-zero map comparison with T = 10 ms.

On the other hand, this laboratory experience showed the robustness of the direct digital design. The key factor in this method is the possibility of getting an exact discrete-time equivalent of the series ZOH → P(s) → A/D, because the input of P(s) is a piecewise constant signal obtained from the discrete-time controller output (so no loss of information occurs in the ZOH process). This allows you to have exact control on the discrete-time closed-loop system eigenvalues/poles (assuming that the plant to be controlled is reachable), independently on the value of T, ensuring stability of the closed-loop system in any case. In fact, this is the only design technique among all the ones tested in this laboratory experience that guarantees the system stability using the robust tracking controller with T = 50 ms. Furthermore, it is also the only design method among the ones tested in this experience that meets (by a wide margin) all performance specifications with T = 10 ms, as you can see in the Table 6. Thus, direct digital design allows you to use less computational power (lower clock speed) with respect to the design by emulation methods (⇒ possible cost reduction).
Anyway, also in the direct digital design the sampling period plays a fundamental role. In fact, a very important thing to keep in mind in the choice of the sampling period, regardless of the design method, is that between two consecutive samples the system works in open-loop mode, so there are not guarantees on what happens in between sampling times. For this reason,

regardless of the design method, the sampling time should never exceed $\frac{t_r}{10}$, as learned in the Digital Control course.

A Appendix

A.1 Nominal model of the DC servomotor

A.2 Unstable response

A.3 MATLAB code

A.3.1 Direct Digital Design

```matlab
%% Model
T1 = 0.001;
T2 = 0.01;
T3 = 0.05;
Ts=T1;

Req = mot.R + sens.curr.Rs;
km = drv.dcgain*mot.Kt/(Req*Beq_hat_final + mot.Kt*mot.Ke);
Tm = Req*Jeq_hat/(Req*Beq_hat_final+mot.Kt*mot.Ke);

A = [0,1;...
     0,-1/Tm];
B = [0;...
     km/gbox.N1/Tm];
C = [1,0];
D = 0;

%% Point 1 - Discrete using exact method the continuos-time state-space model
[Phi,Gamma,H,J] = function_discretizedStateSpace(A,B,C,D,'zoh',Ts);

%% Point 2 - Discrete time nominal state-space controller
ts5 = 0.15;
Mp = 0.1;
dampingFactor = log(1/Mp)/sqrt(pi^2+log(1/Mp)^2);
wn = 3/(ts5*dampingFactor);
z = -dampingFactor*wn + 1i*wn*sqrt(1-dampingFactor^2);
p = [z,conj(z)];
p = exp(p*Ts)

% Feedforward compensator in Discrete Time
sigma = [Phi-eye(2), Gamma; H, 0];
N = [0; 0; 0];
N = inv(sigma) * [0; 0; 1];
Nx = N(1:2)
Nu = N(3)

K = place(Phi,Gamma,p);

[Phi0,Gamma0,H0,J0] = function_Discrete_ReducedOrderStateObserver(Phi,Gamma,H,J,dampingFactor,wn,Ts)

Ts = 0.001;
[Phi,Gamma,H,J] = function_discretizedStateSpace(A,B,C,D,'zoh',Ts)

ts5 = 0.15;
Mp = 0.1;
dampingFactor = log(1/Mp)/sqrt(pi^2+log(1/Mp)^2);
wn = 3/(ts5*dampingFactor);
z = -dampingFactor*wn + 1i*wn*sqrt(1-dampingFactor^2);

sigma = real(z);
wd = imag(z);

p1 = [sigma + 1i*wd, sigma - 1i*wd, sigma]
p2 = [sigma, sigma, sigma]
p3 = [2*sigma + 1i*wd, 2*sigma - 1i*wd, 2*sigma]
p4 = [2*sigma + 1i*wd, 2*sigma - 1i*wd, 3*sigma]
p = p1
p = exp(p*Ts)

% Feedforward compensator in Discrete Time
sigma = [Phi-eye(2), Gamma; H, 0];
N = [0; 0; 0];
N = inv(sigma) * [0; 0; 1];
Nx = N(1:2)
Nu = N(3)

% Extended state-space model
Phi_e = [eye(size(H,1)),H;...
        zeros(size(Phi,1),size(H,1)),Phi];
Gamma_e = [zeros(size(H,1));...
             Gamma];

Ke = place(Phi_e,Gamma_e,p);

% Discrete-time Reduced-Order State Observer
[Phi0,Gamma0,H0,J0] = function_Discrete_ReducedOrderStateObserver(Phi,Gamma,H,J,dampingFactor,wn,Ts)
```